



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-1

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Introduction

Date of Lecture:

Topic of Lecture: The Way of Programming-What is programming

Introduction :

- Problems can be solved using computers.
- The problem to be solved is written in the form of programs. Computer programs are sequence of programs written to solve a task.
- The program is provided with inputs and it displays the outputs.
- Any computer program get the inputs, performs the process and provides the output to the user.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Computer Hardware Fundamentals
- Computer software Fundamentals
- Networking terminologies
- Basic programming

Detailed content of the Lecture:

PROBLEM SOLVING: Activity to formulate problems, think creatively about solutions, and express a solution clearly and accurately.

The way of programming is an excellent opportunity to practice problem solving skills.

WHAT IS PROGRAMMING:

Program - Sequence of instructions that specifies how to perform a computation. The computation may be mathematical symbolic computation.

Example:

```
# Python Program - Get String Input from User  
str = input("Enter any string: ")  
print(str)
```

Input - Data from the keyboard, a file, the network or some other device.

Example :>>> n=int(input("enter the number"))

Output - Display data on the screen, save it in a file, transferred over the network etc.

Example :>>> print('Hello, world!')

math - Basic mathematical operations like addition and multiplications.

Example: 1

```

# Store input numbers:
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')
# Add two numbers
sum = float(num1) + float(num2)
# Subtract two numbers
min = float(num1) - float(num2)
# Multiply two numbers
mul = float(num1) * float(num2)
# Divide two numbers
div = float(num1) / float(num2)
# Display the sum
print("The sum of {0} and {1} is {2}'.format(num1, num2, sum))
# Display the subtraction
print("The subtraction of {0} and {1} is {2}'.format(num1, num2, min))
# Display the multiplication
print("The multiplication of {0} and {1} is {2}'.format(num1, num2, mul))
# Display the division
print("The division of {0} and {1} is {2}'.format(num1, num2, div))

```

Example:2

```

# Python Program to find the area of triangle
a = float(input('Enter first side: '))
b = float(input('Enter second side: '))
c = float(input('Enter third side: '))
# calculate the semi-perimeter
s = (a + b + c) / 2
# calculate the area
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print("The area of the triangle is %0.2f %area)

```

Conditional execution - Check for certain conditional and run the appropriate code.

Example:1

```

# User enters the number
number =int(input("Enter number: "))
# checking the number
if number <0:print("The entered number is negative.")elif number >0:
print("The entered number is positive.")
elif number ==0:
print("Number is zero.")
else:
print("The input is not a number")

```

Example:2

```

# User enters the year
year = int(input("Enter Year: "))

# Leap Year Check
if year % 4 == 0 and year % 100 != 0:
    print(year, "is a Leap Year")
elif year % 100 == 0:
    print(year, "is not a Leap Year")

```

```
elif year % 400 ==0:
    print(year, "is a Leap Year")
else:
    print(year, "is not a Leap Year")
```

Example : 3

```
# taking user input
ch = input("Enter a character: ")
if((ch>='a' and ch<='z') or (ch>='A' and ch<='Z')):
    print(ch, "is an Alphabet")
else:
    print(ch, "is not an Alphabet")
```

Repetition - Perform some action repeatedly with some variation.

Example:2

```
str = input("Enter a string: ")
# counter variable to count the character in a string
counter =0
for s in str:
    counter = counter+1
print("Length of the input string is:", counter)
```

Example:1

```
# taking input from user
number = int(input("Enter any number: "))
# prime number is always greater than 1
if number > 1:
    for i in range(2, number):
        if (number % i) == 0:
            print(number, "is not a prime number")
            break
    else:
        print(number, "is a prime number")

# if the entered number is less than or equal to 1
# then it is not prime number
else:
    print(number, "is not a prime number")
```

Video Content / Details of website for further learning (if any):

[//www.youtube.com/watch?v=OV9WITd9a2U](https://www.youtube.com/watch?v=OV9WITd9a2U)

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016.
pp.1-2

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L-2

LECTURE HANDOUTS

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Introduction

Date of Lecture:

Topic of Lecture: Debugging

Introduction :

- A computer program must be typed , saved and debugged before execution.
- Debugging is the process of correcting the errors in the program.
- A program without any errors provides output on execution.
- Python programs are interpreted and executed.
- Syntax errors are corrected before executing the program

Prerequisite knowledge for Complete understanding and learning of Topic:

- System Software
- Application Software
- Translators
- Basic programming

Detailed content of the Lecture:

Debugging- Programming errors are called as bugs, and the process of tracking them down is called as debugging.

Example:1

Python program to add two numbers

```
num1 =15
num2 =12
# Adding two nos
sum=num1 +num2
# printing values
print("Sum of {0} and {1} is {2}".format(num1, num2, sum))
```

Example:2

```
# Python swap program
x = input('Enter value of x: ')
```

```
y = input('Enter value of y: ')
# create a temporary variable and swap the values
temp = x
x = y
y = temp
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))
```

Example:3

```
# Collect input from the user
kilometers = float(input('How many kilometers?: '))
# conversion factor
conv_fac = 0.621371
# calculate miles
miles = kilometers * conv_fac
print('%0.3f kilometers is equal to %0.3f miles' %(kilometers,miles))
```

Example 4:

```
#Collect input from the user
celsius = float(input('Enter temperature in Celsius: '))
# calculate temperature in Fahrenheit
fahrenheit = (celsius * 1.8) + 32
print('%0.1f Celsius is equal to %0.1f degree Fahrenheit'%(celsius,fahrenheit))
```

Example 5:

```
import calendar
# Enter the month and year
yy = int(input("Enter year: "))
mm = int(input("Enter month: "))
# display the calendar
print(calendar.month(yy,mm))
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=K2ah7wxHlZg>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.6

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-3

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Introduction

Date of Lecture:

Topic of Lecture: Formal and Natural Languages

Introduction :

- The languages are of two types Formal and Natural Languages.
- Natural languages evolve naturally and formal languages are languages designed by people to solve specific applications.
- Programming languages like C, C++, , Python are formal languages.

Prerequisite knowledge for Complete understanding and learning of Topic:

- System Software
- Application Software
- Translators
- Basic programming

Detailed content of the Lecture:

Natural Languages: Natural Languages are the languages spoken by the people such as English, Spanish etc, They are not designed by the people, the natural languages evolve naturally.

Formal languages: Languages designed by people for specific applications. Programming languages are formal languages that are designed to impress computations.

Eg: Notations used by mathematicians to denote relationships among numbers and symbols
Chemists the formal language to represent the chemical structure of mole

Formal language has strict syntax rules that govern the structure of statements. Syntax rules are different for token and structure. Tokens are the basic demands of the language such as words, members and chemical demands,

Examining the program and analyzing the syntactic structure is called are passing.

Natural language

- 1.Ambiguous
- 2.Have lots of redundancy
- 3.inaccurate

Formal language

- Unambiguous
- Less redundant and more concise
- Accurate

Natural Languages

Examples: Mathematical symbols, Chemical structures

Formal languages

Example:1

```
age = int(input("Enter your age? "))
if age >= 18:
    print("You are eligible to vote !!");
else:
    print("Sorry! you have to wait !!");
```

Example:2

```
num = int(input("enter the number?"))
if num % 2 == 0:
    print("Number is even...")
else:
    print("Number is odd...")
```

Example:3

```
a = int(input("Enter a? "))
b = int(input("Enter b? "))
c = int(input("Enter c? "))
if a > b and a > c:
    print("a is largest")
if b > a and b > c:
    print("b is largest")
if c > a and c > b:
    print("c is largest")
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=enG7xaK7PfA>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016.
pp.4-6

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-4

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Introduction

Date of Lecture:

Topic of Lecture: Python:Features

Introduction :

- Python is a high level object oriented programming languages.
- Python is used for wide variety of applications. The features of C, C++ and Java are available in Python. Python has standard library functions.
- It is a cross platform language.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Procedure Oriented Programming
- Object Oriented Programming
- Translators
- Basic programming

Detailed content of the Lecture:

1.Easy to learn and use

2. More understandable and readable, hence python in expressive languages.

3. Interpreted languages- An interpreter is a program that reads and executes code. This includes source code, pre-compiled code, and scripts. Common interpreters include Perl, Python, and Ruby interpreters, which execute Perl, Python, and Ruby code respectively.

4. Cross- platform languages- python runs on different platforms such as window, unix, linux etc,

5.Free and open sources- pthton languages and its sources code is freely available- The term "open source" refers in general to something that can be modified and shared because its design is publicly accessible. An open source programming language is thus one in which the source code to the language's compiler or interpreter is accessible for viewing, modifying and redistributing to the world.

6. object oriented languages-python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows the users to develop applications using an Object Oriented approach. In Python, classes and objects can be created easily.

Major principles of object-oriented programming system are given below.

- Object- An entity or thing is called as an object

- Class- collection of objects are called as class
- Method- methods are used to access the data members
- Inheritance- deriving new class from existing class is called as inheritance
- Polymorphism-ability of a message to be displayed in more than one form. is called as polymorphism
- Data Abstraction-Abstraction means displaying only essential information and hiding the details.
- Encapsulation- Encapsulation is defined as wrapping up of data and information under a single unit

7. Large standard library.

Python's standard library is very extensive, offering a wide range of facilities

8. GUI programming support.

9.Integrated- can be integrated with C, C++,JAVA etc..

10.Extensible

Video Content / Details of website for further learning (if any):

www.youtube.com/watch?v=enG7xaK7PfA

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016.
pp.3

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-5

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Introduction

Date of Lecture:

Topic of Lecture: Installation

Introduction :

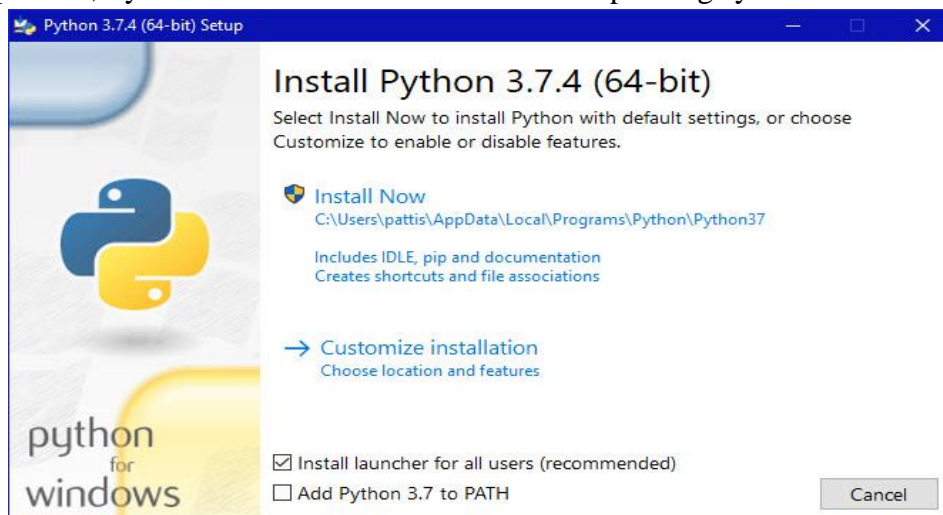
- Python is a open source software.
- The python version for installation is selected based upon the hardware configuration of the system. Latest version of python is 3.8.1.
- The user can install python according to their needs.
- Python installation steps are same for all versions.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Hardware configuration
- System Software
- Application Software
- Basic programming

Detailed content of the Lecture:

1. Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, Python 3.7.4 is installed on Windows operating system.



2. Ensure that the Install launcher for all users (recommended) and the Add Python 3.7 to PATH checkboxes at the bottom are checked.

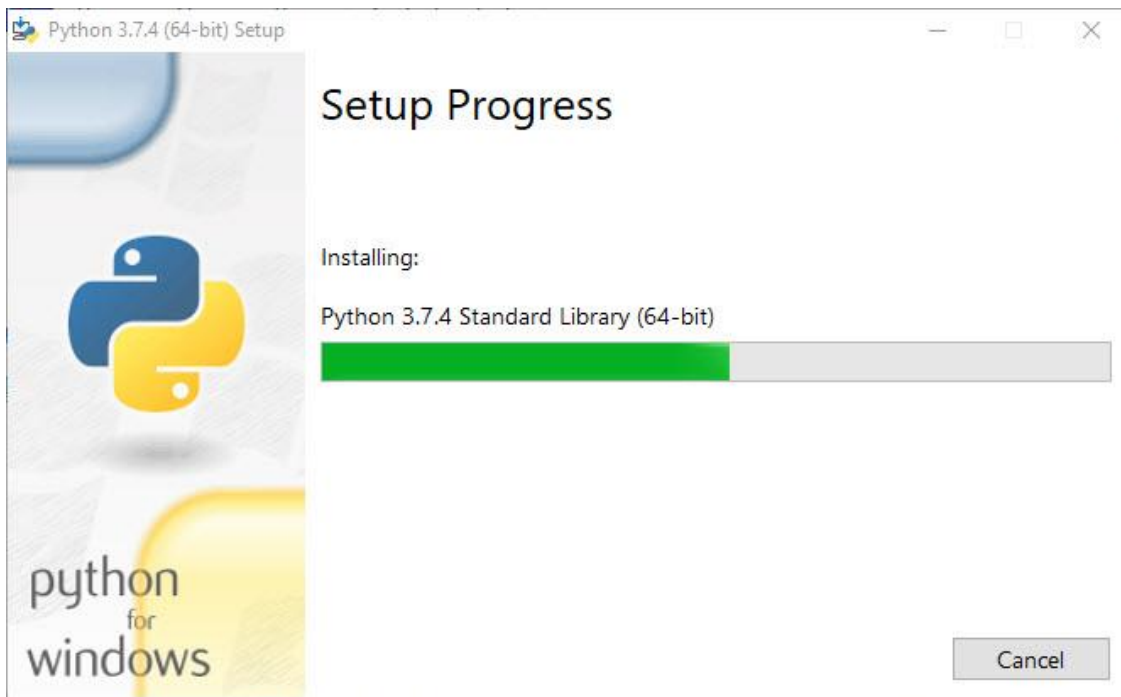
If the Python Installer finds an earlier version of Python installed on your computer, the Install Now message may instead appear as Upgrade Now (and the checkboxes will not appear).

3. Highlight the Install Now (or Upgrade Now) message, and then click it.

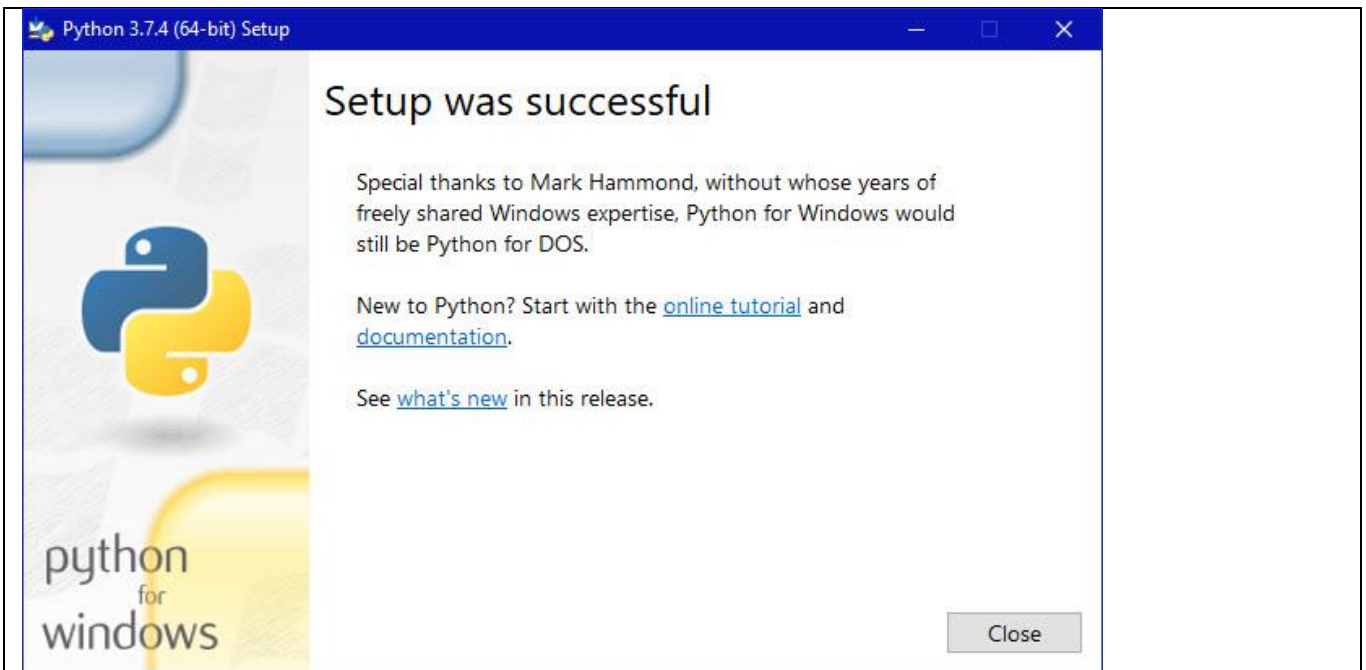
When run, a User Account Control pop-up window may appear on your screen. I could not capture its image, but it asks, Do you want to allow this app to make changes to your device.

4. Click the Yes button.

A new Python 3.7.4 (64-bit) Setup pop-up window will appear with a Setup Progress message and a progress bar.



During installation, it will show the various components it is installing and move the progress bar towards completion. A new Python 3.7.4 (64-bit) Setup pop-up window will appear with a Setup was successfully message.



5. Click the **Close** button.

Video Content / Details of website for further learning (if any):

//www.youtube.com/watch?v=K2ah7wxHlzg

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.4-6

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-6

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : I-Introduction Date of Lecture:

Topic of Lecture: Running Python programs, Modes Demo, Python modes Demo

Introduction :

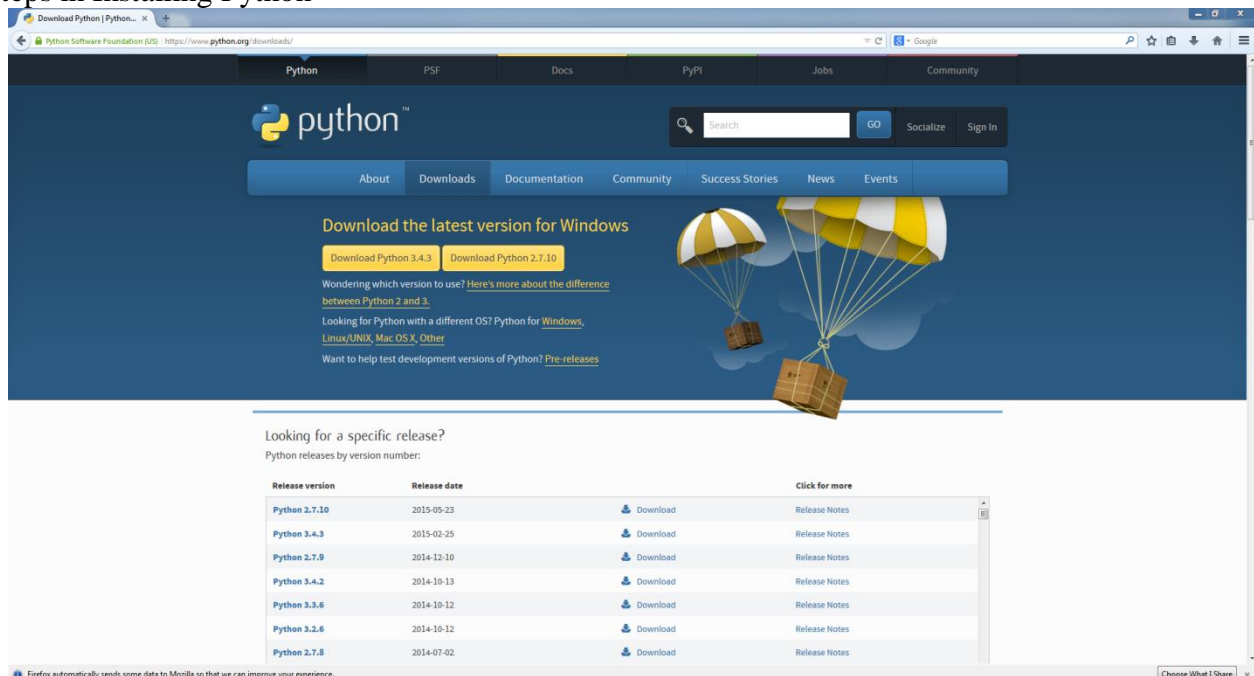
- Python runs on different platforms. Once the python programs are typed and saved they are executed.
- Python program upon execution provides the output.
- Short cut key for running python programs are F5. Python runs in two modes.

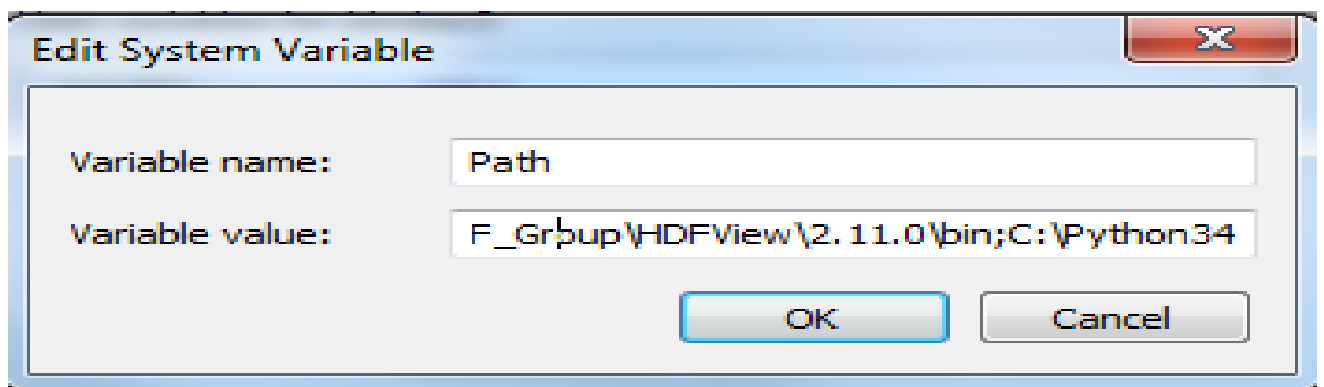
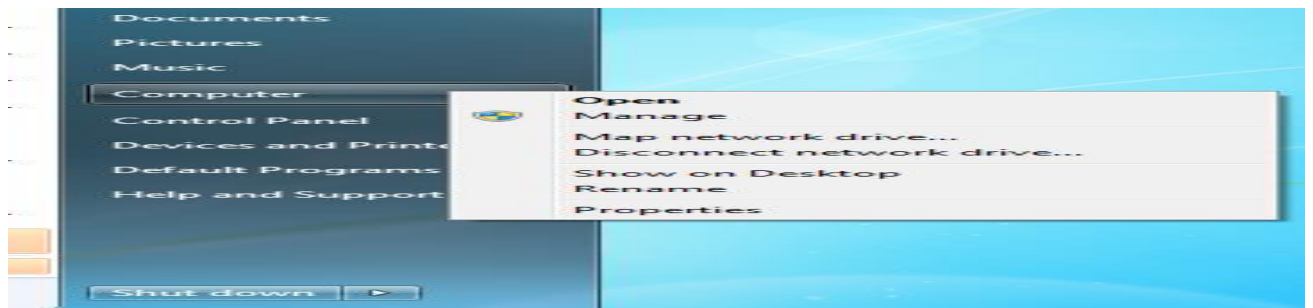
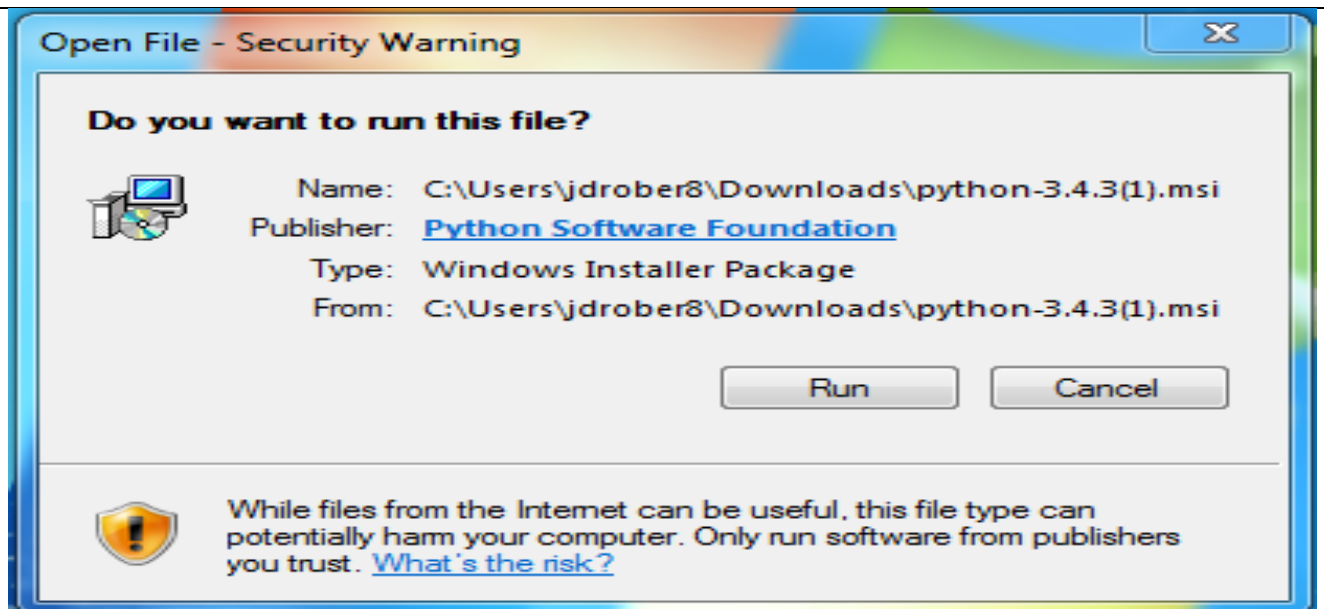
Prerequisite knowledge for Complete understanding and learning of Topic:

- Basic programming
- System Software
- Translators
- Shortcut keys

Detailed content of the Lecture:

Steps in Installing Python





```
cmd - Command Prompt - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\jdrober8>python
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=YyOCZeEtnGs>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.4-6

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-7

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: Values and types: int, float, boolean, variables

Introduction :

- A value is one of the basic things a program works with, like a letter or a number.
- Python Data Types. Data types are the classification or categorization of data items.
- Data types represent a kind of value which determines what operations can be performed on that data.
- Numeric, non-numeric and Boolean (true/false) **data** are the most used **data types**.
- **Python - Variable** Types - **Variables** are nothing but reserved memory locations to store values. This means that when you create a **variable**

Prerequisite knowledge for Complete understanding and learning of Topic:

- values and types
- Numbers
- Data types

Detailed content of the Lecture:

A value is one of the basic things a program works with, like a letter or a number.

Example: 2 is an integer, 42.0 is a floating-point number and 'Hello, World!' is a string.

The interpreter tells the type of value

Example:1

```
>>> type(2)
<class 'int'>
>>> type(42.0)
<class 'float'>
>>> type('Hello, World!')
<class 'str'>
```

Example:2

```
>>> type('2')
<class 'str'>
>>> type('42.0')
<class 'str'>
```

They're strings.

Value:

Value can be any letter ,number or string.

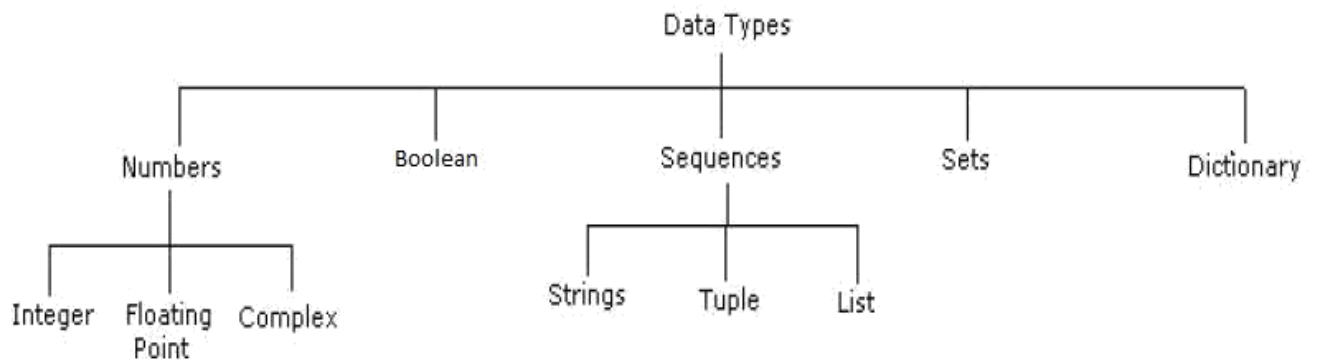
Eg, Values are 2, 42.0, and 'Hello, World!'. (These values belong to different data types.)

Data type:

Every value in Python has a data type.

It is a set of values, and the allowable operations on those values.

Python has four standard data types:



Numbers:

- ❖ Number data type stores **Numerical Values**.
- ❖ This data type is immutable [i.e. values/items cannot be changed].
- ❖ Python supports integers, floating point numbers and complex numbers. They are defined as,

Integers	Long	Float	Complex
- They are often called just integers or int . - They are positive or negative whole numbers with no decimal point.	- They are long integers. - They can also be represented in <u>octal</u> and hexadecimal representation.	- They are written with a decimal point dividing the integer and the fractional parts.	- They are of the form a + bj , where a and b are floats and <u>j</u> represents the square root of -1 (which is an imaginary number). - The real part of the number is a, and the imaginary part is b.
Eg, 56	Eg, 5692431L	Eg, 56.778	Eg, square root of -1 is a complex number

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

Important Books/Journals for further learning including the page nos.:

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-8

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: String and list, expressions, statements

Introduction :

- String(Sequence of Character) literals in python are surrounded by either single quotation marks, or double quotation marks.
- A **list** is a data structure in **Python** that is a mutable, or changeable, ordered sequence of elements.
- An **expression** is an instruction that combines values and operators and always evaluates down to a single value.
- A **Python expression** can be defined as any element in our program that evaluates to some value.
- This type of command where a value is assigned to a variable is called a **Python Statement**

Prerequisite knowledge for Complete understanding and learning of Topic:

- String
- List
- Expression
- Statements

Detailed content of the Lecture:

String Operations

In general, mathematical operations cannot be performed on strings, even if the strings look like numbers, so the following are illegal:

```
'2'-1' 'eggs'/'easy' 'third'*'a charm'
```

But there are two exceptions, + and *.

The + operator performs string concatenation, which means it joins the strings by linking them end-to-end. For example:

```
>>> first = 'throat'
```

```
>>> second = 'warbler'
```

```
>>> first + second throatwarbler
```

The * operator also works on strings; it performs repetition. For example, 'Spam'*3 is 'SpamSpamSpam'. If one of the values is a string, the other has to be an integer.

The syntax for accessing the elements of a list is the same as for accessing the characters of a string—the bracket operator. The expression inside the brackets specifies the index. The indices start at 0:

```
>>> cheeses[0] 'Cheddar'
```

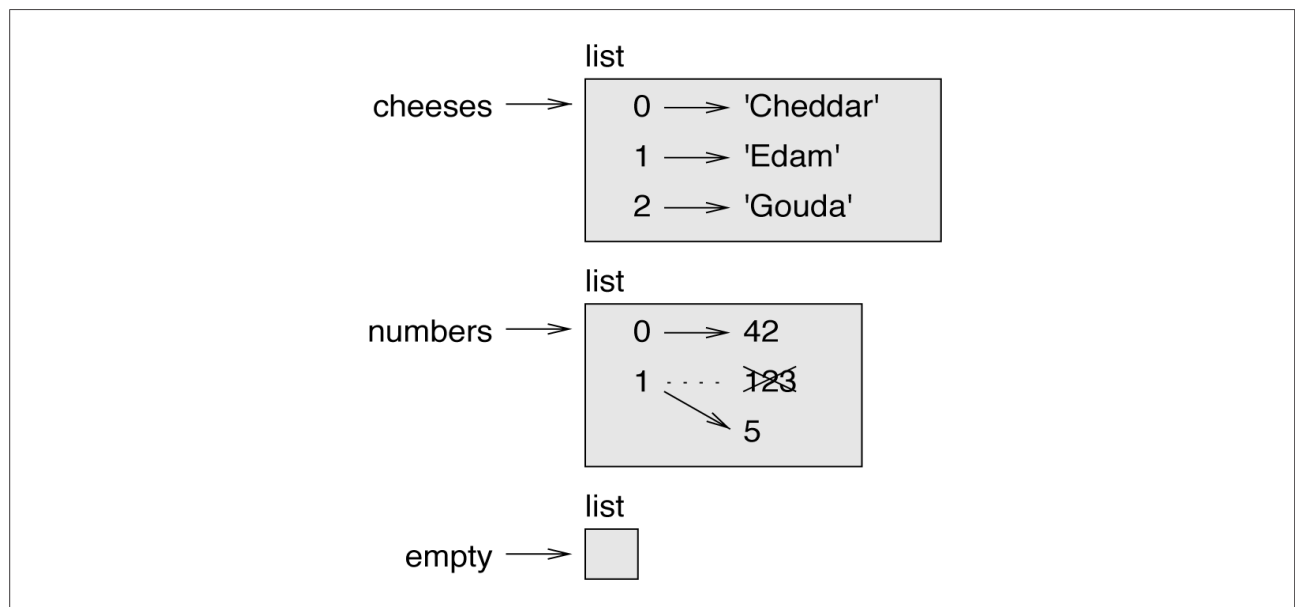
Unlike strings, lists are mutable. When the bracket operator appears on the left side of an assignment, it identifies the element of the list that will be assigned:

```
>>> numbers = [42, 123]
```

```
>>> numbers[1] = 5
```

```
>>> numbers [42, 5]
```

The one-eth element of numbers, which used to be 123, is now 5. The figure shows the state diagram for cheeses, numbers and empty.



Lists are represented by boxes with the word “list” outside and the elements of the list inside. cheeses refers to a list with three elements indexed 0, 1 and 2. numbers contains two elements; the diagram shows that the value of the second element has been reassigned from 123 to 5. empty refers to a list with no elements.

Expressions and Statements

An expression is a combination of values, variables, and operators. A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions:

Example

```
>>> 42
```

```
42
```

```
>>> n 17
```

```
>>> n + 25 42
```

When an expression is typed at the prompt, the interpreter evaluates it, which means that it finds the value of the expression. In this example, n has the value 17 and n + 25 has the value 42.

A statement is a unit of code that has an effect, like creating a variable or displaying a value.

```
>>> n = 17
```

```
>>> print(n)
```

17

The first line is an assignment statement that gives a value to n. The second line is a print statement that displays the value of n.

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

Important Books/Journals for further learning including the page nos.:

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011,
Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-8

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: Tuple Assignment

Introduction :

- A **tuple** is a sequence of immutable **Python** objects
- Assignment of Tuple at Various types of values.
- The left side is a tuple of variables; the right side is a tuple of expressions.
- Each value is assigned to its respective variable. All the expressions on the right side are
- Evaluated before any of the assignments.
- Basic of Tuple Operation Indexing ,slicing, concatenation, Reptition

Prerequisite knowledge for Complete understanding and learning of Topic:

- Tuple
- Assignment of Tuple
- Operation of tuple
- Tuple Expression

Detailed content of the Lecture:

Tuples:

- i) A tuple is a sequence of immutable Python objects.
- ii) Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- iii) Creating a tuple is as simple as putting different comma-separated values.

A tuple is a sequence of values. The values can be any type, and they are indexed by integers, so in that respect tuples are a lot like lists. The important difference is that tuples are immutable.

Syntactically, a tuple is a comma-separated list of values:

```
>>> t = 'a', 'b', 'c', 'd', 'e'
```

Although it is not necessary, it is common to enclose tuples in parentheses:

```
>>> t = ('a', 'b', 'c', 'd', 'e')
```

To create a tuple with a single element, you have to include a final comma:

```
>>> t1 = 'a',  
>>> type(t1)  
<class 'tuple'>
```

A value in parentheses is not a tuple:

```
>>> t2 = ('a')  
>>> type(t2)  
<class 'str'>
```

Another way to create a tuple is the built-in function `tuple`. With no argument, it creates an empty tuple:

```
>>> t = tuple()
>>> t ()
```

If the argument is a sequence (string, list or tuple), the result is a tuple with the elements of the sequence:

```
>>> t = tuple('lupins')
>>> t
('l', 'u', 'p', 'i', 'n', 's')
```

Because `tuple` is the name of a built-in function, you should avoid using it as a variable name.

Most list operators also work on tuples. The bracket operator indexes an element:

```
>>> t = ('a', 'b', 'c', 'd', 'e')
>>> t[0]
'a'
```

And the slice operator selects a range of elements:

```
>>> t[1:3] ('b', 'c')
```

But if you try to modify one of the elements of the tuple, you get an error:

```
>>> t[0] = 'A'
TypeError: object doesn't support item assignment
```

Because tuples are immutable, you can't modify the elements. But you can replace one tuple with another:

```
>>> t = ('A',) + t[1:]
>>> t
('A', 'b', 'c', 'd', 'e')
```

Tuple Assignment

It is often useful to swap the values of two variables. With conventional assignments, you have to use a temporary variable. For example, to swap `a` and `b`:

```
>>> temp = a
>>> a = b
>>> b = temp
```

This solution is cumbersome; tuple assignment is more elegant:

```
>>> a, b = b, a
```

The left side is a tuple of variables; the right side is a tuple of expressions. Each value is assigned to its respective variable. All the expressions on the right side are evaluated before any of the assignments.

The number of variables on the left and the number of values on the right have to be the same:

```
>>> a, b = 1, 2, 3
ValueError: too many values to unpack
```

More generally, the right side can be any kind of sequence (string, list or tuple). For example, to split an email address into a user name and a domain, you could write:

```
>>> addr = 'monty@python.org'
>>> uname, domain = addr.split('@')
```

The return value from `split` is a list with two elements; the first element is assigned to `uname`, the second to `domain`:

```
>>> uname 'monty'
>>> domain 'python.org'
```

Basic Tuples Operations

Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.

In fact, tuples respond to all of the general sequence operations we used on strings in the prior chapter –

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!') * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Indexing, Slicing, and Matrixes

Because tuples are sequences, indexing and slicing work the same way for tuples as they do for strings. Assuming following input –

L = ('spam', 'Spam', 'SPAM!')

Python Expression	Results	Description
L[2]	'SPAM!'	Offsets start at zero
L[-2]	'Spam'	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

Important Books/Journals for further learning including the page nos.:

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: precedence of operators, comments

Introduction :

- Basic **Operators in Python**. Arithmetic operators: Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division
- **Python** has well-defined rules for specifying the order in which the **operators** in an expression are evaluated when the expression has several **operators**.
- **Precedence** rules can be overridden by explicit parentheses.
- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.and to prevent execution when testing code

Prerequisite knowledge for Complete understanding and learning of Topic:

- Operators
- Precedence of operators
- Basic operation of Precedence of operators
- Comments

Detailed content of the Lecture:

Order of Operations

- When an expression contains more than one operator, the order of evaluation depends on the order of operations. For mathematical operators, Python follows mathematical convention. The acronym PEMDAS is a useful way to remember the rules:
- Parentheses have the highest precedence and can be used to force an expression to evaluate in the order you want. Since expressions in parentheses are evaluated first, $2 * (3-1)$ is 4, and $(1+1)**(5-2)$ is 8. You can also use parentheses to make an expression easier to read, as in $(\text{minute} * 100) / 60$, even if it doesn't change the result.
- Exponentiation has the next highest precedence, so $1 + 2**3$ is 9, not 27, and 2
- $* 3**2$ is 18, not 36.
- Multiplication and Division have higher precedence than Addition and Subtraction. So $2*3-1$ is 5, not 4, and $6+4/2$ is 8, not 5.
- Operators with the same precedence are evaluated from left to right (except exponentiation). So in the expression $\text{degrees} / 2 * \pi$, the division happens first and the result is multiplied by pi. To divide by 2π , you can use parentheses or write $\text{degrees} / 2 / \pi$.
- I don't work very hard to remember the precedence of operators. If I can't tell by looking at the expression, I use parentheses to make it obvious.

Operators are the constructs which can manipulate the value of operands.

Consider the expression $4 + 5 = 9$. Here, 4 and 5 are called operands and + is called operator.

- Python Operators Precedence
- The following table lists all operators from highest precedence to lowest.
- [[Show Example](#)]

Sr.No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>><< Right and left bitwise shift
6	& Bitwise 'AND'
7	^ Bitwise exclusive 'OR' and regular 'OR'
8	<= <>= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators

- **Comments**
- As programs get bigger and more complicated, they get more difficult to read. Formal languages are dense, and it is often difficult to look at a piece of code and figure out what it is doing, or why.
- For this reason, it is a good idea to add notes to your programs to explain in natural language what the program is doing. These notes are called comments, and they start with the # symbol:

- `# compute the percentage of the hour that has elapsed percentage = (minute * 100) / 60`
- In this case, the comment appears on a line by itself. You can also put comments at the end of a line:
- `percentage = (minute * 100) / 60 # percentage of an hour`
- Everything from the `#` to the end of the line is ignored—it has no effect on the execution of the program.
- Comments are most useful when they document non-obvious features of the code. It is reasonable to assume that the reader can figure out what the code does; it is more useful to explain why.
- This comment is redundant with the code and useless:
- `v = 5 # assign 5 to v`
- This comment contains useful information that is not in the code:
- `v = 5 # velocity in meters/second.`
- Good variable names can reduce the need for comments, but long names can make complex expressions hard to read, so there is a trade-off.

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

Important Books/Journals for further learning including the page nos.:

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011,
Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: Conditionals: Boolean values and operators

Introduction :

- The decision, in most cases, depends on the value of variables or arithmetic expressions are evaluated using the Boolean True or False values.
- The instructions for decision making are called conditional statements, executed under the given conditions.
- In many cases there are two code parts: One which will be executed, if the condition is True, and another one, if it is False.
- In other words, a branch determines which of two (or even more) program parts (alternatives) will be executed depending on one (or more) conditions.
- Conditional statements and branches belong to the control structures of programming languages, because with their help a program can react to different states that result from inputs and calculations.

Prerequisite knowledge for Complete understanding and learning of Topic:

- values and types
- precedence of operators
- Flowchart

Detailed content of the Lecture:

BOOLEAN VALUES:

Boolean:

- Boolean data type have two values. They are 0 and 1.
- 0 represents False
- 1 represents True
- True and False are keyword.
-

Example:

```
>>> 3==5
False
>>> 6==6
True
>>> True+True
2
>>> False+True
1
>>> False*True
```

OPERATORS:

- □ Operators are the constructs which can manipulate the value of operands.
- □ Consider the expression $4 + 5 = 9$. Here, 4 and 5 are called operands and + is called operator.
-

Types of Operators:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

Arithmetic operators:

They are used to perform mathematical operations like addition, subtraction, multiplication etc.

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a = 10, b = 20$ $a + b = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator	$a * b = 200$
/ Division	Divides left hand operand by right hand operand	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 0$
** Exponent	Performs exponential (power) calculation on operators	$a ** b = 10$ to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed	$5 // 2 = 2$

Comparison (Relational) Operators:

- Comparison operators are used to compare values.
- It either returns True or False according to the condition.

Operator	Description	Example $a=10, b=20$
==	If the values of two operands are equal, then the condition becomes true.	$(a == b)$ is not true.
!=	If values of two operands are not equal, then condition becomes true.	$(a != b)$ is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	$(a > b)$ is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	$(a < b)$ is true.

>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Assignment Operators

Assignment operators are used in Python to assign values to variables.

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation operators and assign value to the left operand on	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Logical Operators:

Logical operators are and, or, not operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Bitwise Operators:

Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
	Bitwise OR	x y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

Membership Operators:

- Evaluates to find a value or a variable is in the specified sequence of string, list, tuple, dictionary or not.
- To check particular element is available in the list or not

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Example:

```
x=[5,3,6,4,1]
>>>5 in x
True
>>>5 not in x
False
```

Identity Operators:

They are used to check if two values (or variables) are located on the same part of the memory.

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Example x = 5 y = 5 a = 'Hello' b = 'Hello'

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
2. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: Conditional (if), Alternative (if-else)

Introduction :

- In many cases there are two code parts: One which will be executed, if the condition is True, and another one, if it is False.
- In other words, a branch determines which of two (or even more) program parts (alternatives) will be executed depending on one (or more) conditions.
- Conditional statements and branches belong to the control structures of programming languages, because with their help a program can react to different states that result from inputs and calculations.

Prerequisite knowledge for Complete understanding and learning of Topic:

- values and types
- precedence of operators
- Boolean values and operators
- Flowchart

Detailed content of the Lecture:

Conditional (if):

Conditional (if) is used to test a condition, if the condition is true the statements inside if will be executed.

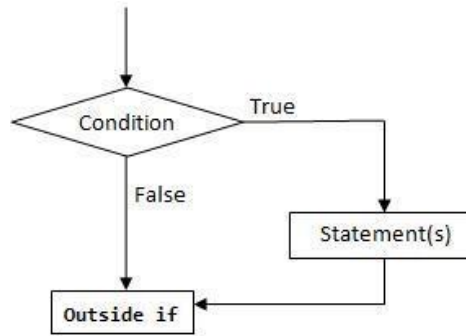
syntax:

```
if(condition 1):  
    Statement 1
```

Example:

1. Program to provide flat rs 500, if the purchase amount is greater than 2000.
2. Program to provide bonus mark if the category is sports.

Flowchart:



Program to provide flat rs 500, if the purchase amount is greater than 2000.	output
<pre>purchase=eval(input("enter your purchase amount")) if(purchase>=2000): purchase=purchase-500 print("amount to pay",purchase)</pre>	<pre>enter your purchase amount 2500 amount to pay2000</pre>
Program to provide bonus mark if the category issports	output
<pre>m=eval(input("enter ur mark out of 100")) c=input("enter urcategory G/S") if(c=="S"): m=m+5 print("mark is",m)</pre>	<pre>enter ur mark out of 100 85 enter urcategory G/S S mark is 90</pre>

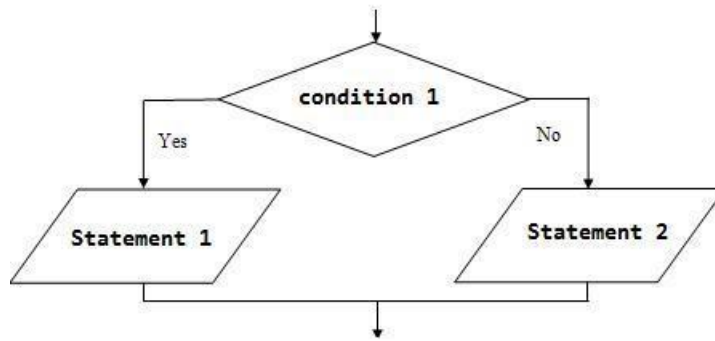
Alternative execution (if-else)

In the alternative the condition must be true or false. In this **else** statement can be combined with **if** statement. The **else** statement contains the block of code that executes when the condition is false. If the condition is true statements inside the if get executed otherwise else part gets executed. The alternatives are called branches, because they are branches in the flow of execution.

syntax:

```
if(condition 1):
    Statement 1
else:
    Statement 2
```

Flowchart:



Examples:

- odd or even number
- positive or negative number
- leap year or not
- greatest of two numbers
- eligibility for voting

Odd or even number	Output
<pre>n=eval(input("enter a number")) if(n%2==0):print("even number")else: print("odd number")</pre>	<pre>enter a number4 even number</pre>
positive or negative number	Output
<pre>n=eval(input("enter a number")) if(n>=0): print("positive number") else: print("negative number")</pre>	<pre>enter a number8 positive number</pre>
leap year or not	Output
<pre>y=eval(input("enter a yaer")) if(y%4==0): print("leap year") else: print("not leap year")</pre>	<pre>enter a yaer2000 leap year</pre>
greatest of two numbers	Output
<pre>a=eval(input("enter a value:")) b=eval(input("enter b value:")) if(a>b):print("greatest:",a) else: print("greatest:",b)</pre>	<pre>enter a value:4 enter b value:7 greatest: 7</pre>
eligibility for voting	Output
<pre>age=eval(input("enter ur age:"))</pre>	<pre>enter ur age:78</pre>

<pre>if(age>=18): print("you are eligible for vote") else: print("you are eligible for vote")</pre>	you are eligible for vote
Video Content / Details of website for further learning (if any): https://www.w3schools.com/python/	
Important Books/Journals for further learning including the page nos.: Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 41,42,25	

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



L-9

LECTURE HANDOUTS

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: chained conditional (if-elif-else);

Introduction :

- In many cases there are two code parts: One which will be executed, if the condition is True, and another one, if it is False.
- In other words, a branch determines which of two (or even more) program parts (alternatives) will be executed depending on one (or more) conditions.
- Conditional statements and branches belong to the control structures of programming languages, because with their help a program can react to different states that result from inputs and calculations.
- If the condition1 is False, it checks the condition2 of the elif block. If all the conditions are False, then the else part is executed.

Prerequisite knowledge for Complete understanding and learning of Topic:

- values and types
- precedence of operators
- Boolean values and operators, If condition
- Flowchart

Detailed content of the Lecture:

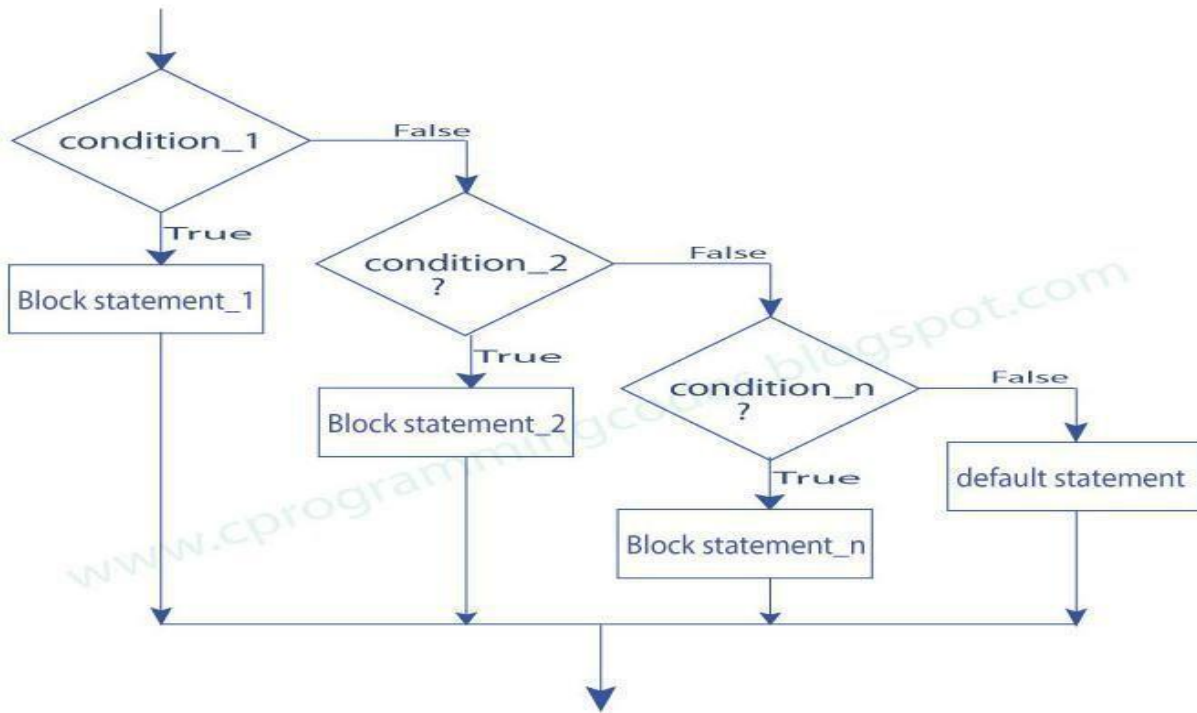
Chained conditionals(if-elif-else)

- The elif is short for else if. This is used to check more than one condition. If the condition1 is False, it checks the condition2 of the elif block. If all the conditions are False, then the else part is executed. Among the several if...elif...else part, only one part is executed according to the condition
- The if block can have only one else block. But it can have multiple elif blocks. The way to express a computation like that is a chained condition

syntax:

```
if(condition 1):
    statement 1
elif(condition 2):
    statement 2
elif(condition 3):
    statement 3
else:
    default statement
```

Flowchart:



Example:

- student mark system
- traffic light system
- compare two numbers
- roots of quadratic equation

student mark system	Output
<pre> mark=eval(input("enter ur mark:")) if(mark>=90): print("grade:S") elif(mark>=80): print("grade:A") elif(mark>=70): print("grade:B") elif(mark>=50): print("grade:C") else: print("fail") </pre>	<pre> enter ur mark:78 grade:B </pre>
traffic light system	Output
<pre> colour=input("enter colour of light:") if(colour=="green"): print("GO") elif(colour=="yellow"): print("GET READY") else: </pre>	<pre> enter colour of light:green GO </pre>

<code>print("STOP")</code>		
compare two numbers	Output	
<code>x=eval(input("enter x value:")) y=eval(input("enter y value:")) if(x == y): print("x and y are equal")elif(x < y): print("x is less than y")else: print("x is greater than y")</code>	enter x value:5 enter y value:7 x is less than y	
Roots of quadratic equation	output	
<code>a=eval(input("enter a value:")) b=eval(input("enter b value:")) c=eval(input("enter c value:")) d=(b*b-4*a*c)if(d==0): print("same and real roots")elif(d>0): print("diffrent real roots")else:print("imaginagry roots")</code>	enter a value:1 enter b value:0 enter c value:0 same and real roots	
Video Content / Details of website for further learning (if any): https://www.w3schools.com/python/ https://www.tutorialspoint.com/python/index.htm		
Important Books/Journals for further learning including the page nos.: Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 92-94		

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: Iteration: state, while, for

Introduction :

- Iteration statements or loop statements allow us to execute a block of statements as long as the condition is true.
- Loops statements are used when we need to run same code again and again, each time with a different value

Prerequisite knowledge for Complete understanding and learning of Topic:

(Max. Four important topics)

- values and types
- precedence of operators
- Boolean values and operators
- Flowchart

ITERATION/CONTROL STATEMENTS/LOOPS:

- state
- while
- for
- break
- continue
- pass

State:

Transition from one process to another process under specified condition with in a time is called state.

While loop:

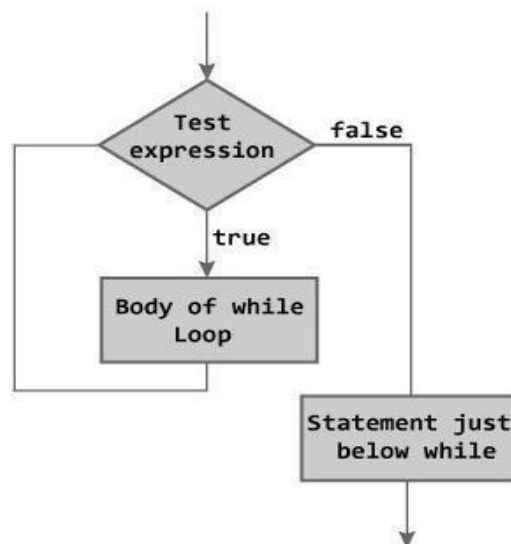
- While loop statement in Python is used to repeatedly executes set of statement as long as a given condition is true.□
- In while loop, test expression is checked first. The body of the loop is entered only if the test_expression is True.
- After one iteration, the test expression is checked again. This process continues until the test_expression evaluates to False.
- In Python, the body of the while loop is determined through indentation.
- The statements inside the while starts with indentation and the first unindented line marks the end.

Syntax:

For loop:

```
initial value  
while(condition):  
    body of while loop  
increment
```

Flowchart:



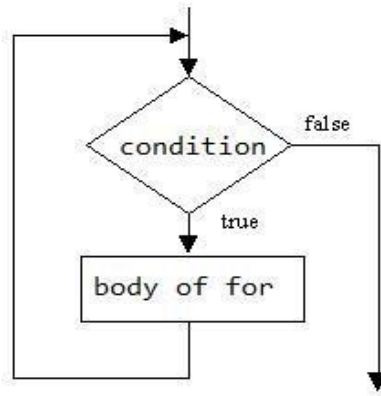
for in range:

- We can generate a sequence of numbers using range() function.
 - range(10) will generate numbers from 0 to 9 (10 numbers).
- In range function have to define the start, stop and step size as range(start,stop,step size). Step Size defaults to 1 if not provided.

Syntax

```
for i in range(start,stop,steps):  
    body of for loop
```

Flowchart:



For in sequence

- The for loop in Python is used to iterate over a sequence (list, tuple, string).
- Iterating over a sequence is called traversal. Loop continues until we reach the last element in the sequence.
- The body of for loop is separated from the rest of the code using indentation

```
for i in sequence:
    print(i)
```

Sequence can be a list, strings or tuples

s.no	sequences	example	output
1.	For loop in string	for i in "Ramu": print(i)	A M U
2.	For loop in list	for i in [2,3,5,6,9]: print(i)	2 3 5 6 9
3.	For loop in tuple	for i in (2,3,1): print(i)	2 3 1

Examples:

- print nos divisible by 5 not by 10:
- Program to print fibonacci series.
- Program to find factors of a given number
- check the given number is perfect number or not
- check the no is prime or not
- Print first n prime numbers
- Program to print prime numbers in range

check a number is perfect number or not	Output
n=eval(input("enter a number:"))	enter a number:6

<pre> sum=0 for i in range(1,n,1): if(n%i==0): sum=sum+i if(sum==n):print("the number is perfect number")else: print("the number is not perfect number") </pre>	<p>the number is perfect number</p>
Program to print first n prime numbers	Output
<pre> number=int(input("enter no of prime numbers to be displayed:")) count=1 n=2 while(count<=number): for i in range(2,n): if(n%i==0): break else: print(n) count=count+1 n=n+1 </pre>	<p>enter no of prime numbers to be displayed:5</p> <p>2</p> <p>3</p> <p>5</p> <p>7</p> <p>11</p>
<p>Video Content / Details of website for further learning (if any): https://www.w3schools.com/python/ https://www.tutorialspoint.com/python/index.htm</p>	
<p>Important Books/Journals for further learning including the page nos.: Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 63,64</p>	

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-9

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : I-Variables,Expressions,Conditionals Date of Lecture:

Topic of Lecture: break, continue, pass

Introduction :

- Break statements can alter the flow of a loop.It terminates the currentloop and executes the remaining statement outside the loop
- CONTINUE -It terminates the current iteration and transfer the control to the next iteration in the loop.
- PASS - It is used when a statement isrequired syntactically but you don't want any code toexecute.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Looping statements
- Conditional statements
- Flowchart

Break

- Break statements can alter the flow of a loop. It terminates the current loop and executes the remaining statement outside the loop
- If the loop has else statement, that will also gets terminated and come out of the loop completely.

Syntax:

```
while (test Expression):
```

```
    // codes
```

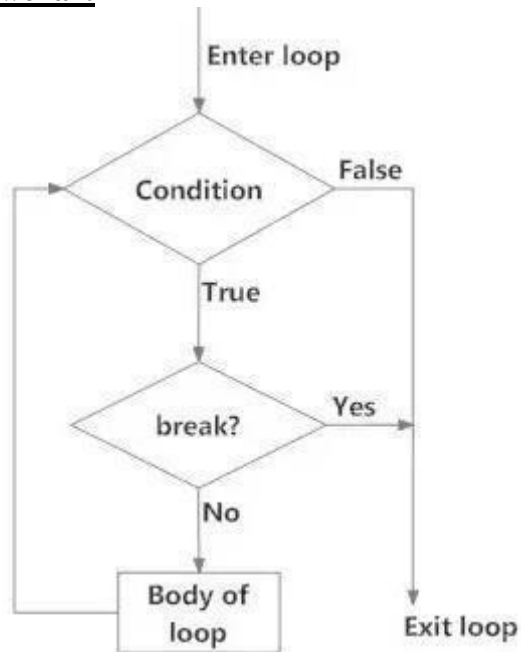
```
    if (condition for break):
```

```
        break
```

```
    // codes
```



Flowchart



example	Output
<pre>for i in "welcome": if(i=="c"): break print(i)</pre>	w e l

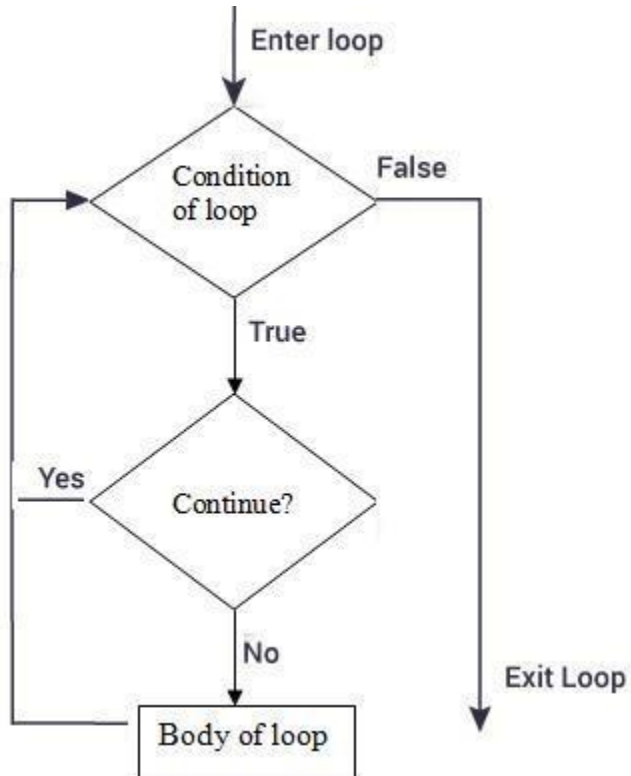
CONTINUE

It terminates the current iteration and transfer the control to the next iteration in the loop.

Syntax:Continue

```
→ while (test Expression):  
    // codes  
    if (condition for continue):  
        continue  
    // codes
```

Flowchart



Example:	Output
<pre> for i in "welcome": if(i=="c"): continue print(i) </pre>	<pre> w e l o m e </pre>

PASS

- It is used when a statement is required syntactically but you don't want any code to execute.
- It is a null statement, nothing happens when it is executed.

Syntax:

Example	Output
<pre> for i in "welcome": if(i=="c"): pass print(i) </pre>	<pre> w e l c o m e </pre>

Difference between break and continue

<u>break</u>	<u>continue</u>
It terminates the current loop and executes the remaining statement outside the loop.	It terminates the current iteration and transfer the control to the next iteration in the loop.
syntax: break	syntax: continue
for i in "welcome": if(i=="c"): break print(i)	for i in "welcome": if(i=="c"): continue print(i)
w e l	w e l o m e

Video Content / Details of website for further learning (if any):

<https://www.w3schools.com/python/>

<https://www.tutorialspoint.com/python/index.htm>

Important Books/Journals for further learning including the page nos.:

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011,
Page no 27,29

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-10

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II -Functions, Strings Date of Lecture:

Topic of Lecture: Functions,Function definition and use

Introduction :

- Functions are the most important aspect of an application.
- A function is a set of statements that take inputs, do some specific computation and produces output.
- Python provides built-in functions like print(), etc.
- Users can also create their own functions. These functions are called user-defined functions.
- The function is also known as procedure or subroutine in other programming languages.

Prerequisite knowledge for Complete understanding and learning of Topic:

- values and types
- Numbers
- Data types

Detailed content of the Lecture:

A function can be defined as the organized block of reusable code which can be called whenever required.

Python allows dividing a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the python program.

Function Definition

In python, **def** keyword is used to define the function. The syntax to define a function in python is given below.

Syntax

```
def my_function():  
    function-suite  
    return <expression>
```

The function block is started with the colon (:) and all the same level block statements remain at the same indentation.A function can accept any number of parameters that must be the same in the definition and function calling.

In python, a function must be defined before the function calling otherwise the python interpreter gives an error. Once the function is defined, the function can be called from another function or from the python

prompt. Function is called by the function name followed by the parentheses.

A simple function that prints the message "Hello Word" is given below.

Example

```
def hello_world():  
    print("hello world")
```

Output

```
hello_world()
```

Uses of Functions in Python

- By using functions, user's can avoid rewriting same logic/code again and again in a program.
- User's can call python functions any number of times in a program and from any place in a program.
- user can track a large python program easily when it is divided into multiple functions.
- Reusability is the main achievement of python function.

Video Content / Details of website for further learning (if any):

www.youtube.com/watch?v=6yrsX752CWk

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-11

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II- Functions, Strings Date of Lecture:

Topic of Lecture: Flow of Execution

Introduction :

- Execution always begins at the first statement of the program.
- Statements are executed one at a time, in order from top to bottom.
- Function definitions do not alter the flow of execution of the program
- Statements inside the function are not executed until the function is called
- The interpreter executes operations and functions in the order that it encounters them

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function definition
- Function Call
- Passing parameters to functions

Detailed content of the Lecture:

- When working with functions it is really important to know the order in which statements are executed. This is called the flow of execution.
- Execution always begins at the first statement of the program. Statements are executed one at a time, in order, from top to bottom.
- Function definitions do not alter the flow of execution of the program, but the statements inside the function are not executed until the function is called.
- Function calls are like a detour in the flow of execution. Instead of going to the next statement, the flow jumps to the first line of the called function, executes all the statements there, and then comes back to pick up where it left off.

Example 1:

```
def func (name):  
    print("Hi ",name);  
func("Ram")
```

Output: Hi Ram

Example 2:

```
def hello():  
    print("hello world")  
python()
```

```
def python():  
    print("testing main")
```

```
if __name__ == "__main__":  
    hello()
```

Output:

```
hello world  
testing main
```

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=fAw8pM_dQP4

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers,
2016 Page no 3-4
Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd,
2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-12

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II-Functions, Strings Date of Lecture:

Topic of Lecture: Fruitful functions: return values, parameters

Introduction :

- A **function** is a named sequence of statements that performs a computation
- The argument is a value or variable that are passed into the function as input to the function
- Function "takes" an argument and "returns" a result. The result is called the **return value**
- The function that returns a value is called as fruitful function

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function definition
- Function Call
- Passing parameters to functions

Detailed content of the Lecture:

A function that returns a value is called fruitful function. Fruitful functions still allow the user to provide information (arguments). However there is now an additional piece of data that is returned from the function.

Example 1:

```
Root=sqrt(25)
```

Example 2:

```
def add():  
a=10 b=20  
c=a+b return c  
c=add() print(c)
```

i. Return values

return keywords are used to return the values from the function.

Example 2:

```
def simple_interest(p,n,r):  
return (p*n*r)/100  
print("Simple Interest: ",simple_interest(n=10,r=10,p=1900))
```

ii. Parameters

Parameters are the variables which used in the function definition. Parameters are inputs to functions. Parameter receives the input from the function call.

It is possible to define more than one parameter in the function definition.

Types of parameters/Arguments:

1. Required/Positional parameters
2. Keyword parameters
3. Default parameters
4. Variable length parameters

1. Required/ Positional Parameter:

The number of parameter in the function definition should match exactly with number of arguments in the function call.

Example

```
def student( name,roll):  
    student("George",98)
```

Output:

```
George98
```

2. Keyword parameter:

When a function is called with some values, these values get assigned to the parameter according to their position.

When the function is called with keyword parameter, the order of the arguments can be changed.

Example:

```
def student(name,roll, mark):  
    print(name,roll, mark)  
    student(90,102, "bala")
```

Output: 90102bala

3. Default Parameter

Python allows function parameter to have default values; if the function is called without the argument, the argument gets its default value in function definition.

Example

```
def student( name,age=17):  
    print(name,age)  
    student("Kumar"):  
    student("Ajay"):
```

Output:

```
Kumar 17
```

```
Ajay 17
```

Video Content / Details of website for further learning (if any):

www.youtube.com/watch?v=Mtav39AKIUY

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4

Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-13

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : II- Functions, Strings Date of Lecture:

Topic of Lecture: Local And Global Scope

Introduction :

- A variable is only available from inside the region it is created. This is called scope
- The scope of a variable in **python** is that part of the code where it is visible.
- Variables can be declared locally or globally
- Variables defined inside a function or class, are local. Only the function or class can see the variable
- A variable at the top level of script or module is called as global variable

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function definition
- Function Call
- Passing parameters to functions
- Variable declarations

Detailed content of the Lecture:

The scopes of the variables depend upon the location where the variable is being declared. The variable declared in one part of the program may not be accessible to the other parts.

In python, the variables are defined with the two types of scopes.

1. Global variables
2. Local variables

The variable defined outside any function is known to have a global scope whereas the variable defined inside a function is known to have a local scope.

Example 1:

```
def print_message():  
    message = "hello !! I am going to print a message."  
    print(message)  
print_message()
```

Output:

hello !! I am going to print a message.

'message' is a local variable defined inside the function print_message().

Example 2:

```
def calculate(a):  
    sum = sum + arg  
    print("The sum is",sum)  
sum=0  
calculate(10)  
print("Value of sum outside the function:",sum)
```

Output:

The sum is 10
Value of sum outside the function: 0

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=NSbOtYzIQI0>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-14

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II-Functions, Strings Date of Lecture:

Topic of Lecture: Function composition, recursion

Introduction :

- A function can be defined as the organized block of reusable code which can be called whenever required.
- Function defined within another function is called as function composition
- In Python, a function can call other functions.
- It is even possible for the function to call itself.
- These type of construct are termed as recursive functions

Prerequisite knowledge for Complete understanding and learning of Topic:

- Function definition
- Function Call
- Passing parameters to functions
- Variable declarations

Detailed content of the Lecture:

A function can be called within another function. The ability to build functions by using other functions is called **Functioncomposition**.

Example

Output

```
defsum(a,b):
sum=a+b
avg(sum)
defavg(sum):
avg=sum/2returnavg
a=eval(input("entera:"))
b=eval(input("enterb:")) sum=sum(a,b)
avg=avg(sum)
print("theavgis",avg)
```

```
entera:4
enterb:8
the avgis6.0
```

v. Recursion

When a function call itself is known as recursion. Recursive function is called by some external code. If the base condition is met then the program does something meaningful and exits. Otherwise, function does some required processing and then calls itself to continue recursion.

Factorialofn

Output

```
deffact(n):
enterno. tofindfact:5
```



```
if(n==1):
    return 1
else:
    return n*fact(n-1)
n=eval(input("enter no. to find fact:"))
fact=fact(n)
print("Factis",fact)
```

Sum of numbers

```
def sum(n):
    if(n==1):
        return 1
    else:
        return n*sum(n-1)
n=eval(input("enter no. to find sum:"))
sum=sum(n)
print("Factis",sum)
```

Output

```
enter no. to find sum: 10
Factis 55
```

Video Content / Details of website for further learning (if any):

www.youtube.com/watch?v=ap8YSOIXWME

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-15

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II-Functions, Strings Date of Lecture:

Topic of Lecture: Strings: string slices, immutability

Introduction :

- Strings are amongst the most popular types in Python.
- Strings can be created simply by enclosing characters in quotes.
- Python treats single quotes the same as double quotes.
- Creating strings is as simple as assigning a value to a variable.
- Characters in the strings can be accessed using index values

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- String declaration
- Accessing strings

Detailed content of the Lecture:

A string is a sequence of characters.

Example

Strings can be enclosed within single or double or triple quotes

```
my_string='Hello'  
my_string="Hello"  
my_string="Hello"
```

i. String slices

Individual characters can be accessed from the string using indexing and a range of characters can be accessed using slicing. Index starts from 0. Trying to access a character out of index range will raise an `IndexError`. The index must be an integer. Use of float or other data types as index, will result into `TypeError`.

Example:

```
str='programiz'
```

```
print('str=',str)
```

```
#first character
```

```
print('str[0]=' ,str[0])
```

```
#last character
```

```
print('str[-1]=' ,str[-1])
```

#slicing2ndto5thcharacter

```
print('str[1:5]=' ,str[1:5])
```

#slicing6thto2ndlastcharacter

```
print('str[5:-2]=' ,str[5:-2])
```

slice()Parameters

slice()mainlytakes threeparameters

start-startingintegerwheretheslicingoftheobjectstarts

stop-integeruntilwhichtheslicingtakes place.Theslicingstops atindex**stop-1**.

step-integervaluewhichdetermines theincrementbetweeneachindexforslicing. Ifasingleparameteris passed,**startandstep**aresettoNone.

Returnvalue fromslice()

slice()returns asliceobjectusedtosliceasequenceinthegivenindices.

Example:

```
Pystring="python"
```

```
# contains indices (0, 1, 2)
```

```
# i.e. P, y and t
```

```
sObject = slice(3)
```

```
print(pyString[sObject])
```

Output:

```
pyt
```

Example:

```
yh
```

```
# contains indices (1, 3)
```

```
# i.e. y and h
```

```
sObject = slice(1, 5, 2)
```

```
print(pyString[sObject])
```

ii. String immutability

Strings are **immutable**, which means you cannot change an existing string. The best is to create a new string that is a variation on the original.

Example

```
greeting = "Hello, world!"
```

```
newGreeting = 'J' + greeting[1:]
```

```
print(newGreeting)
```

```
print(greeting)
```

Video Content / Details of website for further learning (if any):

www.youtube.com/watch?v=LTw5-5tx5wg

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
2. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd,

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-16

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : II-Functions, Strings Date of Lecture:

Topic of Lecture: Strings: String functions and methods string module

Introduction :

- Python has quite a few methods that string objects can call to perform frequently occurring task(related to string).
- The String class need to be imported to use the appropriate string functions
- String module is collection of string constants, functions and class
- String functions returns values
- String classes can be used to define own formats for the strings

Prerequisite knowledge for Complete understanding and learning of Topic:

- Strings
- String declaration
- Accessing strings

Detailed content of the Lecture:

String Functions and Methods

Strings are also objects. Each string instance has its own attributes and methods. The most important attribute of the string is the collection of characters. There are a wide variety of methods.

Method	Parameters	Description
upper	none	Returns a string in all uppercase
lower	none	Returns a string in all lowercase
capitalize	none	Returns a string with first character capitalized, the rest lower
strip	none	Returns a string with the leading and trailing whitespace removed
lstrip	none	Returns a string with the leading whitespace removed
rstrip	none	Returns a string with the trailing whitespace removed
count	item	Returns the number of occurrences of item
replace	old, new	Replaces all occurrences of old substring with new
center	width	Returns a string centered in a field of width spaces

ljust	width	Returns a string left justified in a field of width spaces
rjust	width	Returns a string right justified in a field of width spaces
find	item	Returns the leftmost index where the substring item is found, or -1 if not found
rfind	item	Returns the rightmost index where the substring item is found, or -1 if not found

Example

```
ss = "Hello, World"
print(ss.upper())
tt = ss.lower()
print(tt)
```

Output

HELLO, WORLD

hello, world

Example

```
ss = " Hello, World "
els = ss.count("l")
print(els)
print("***" + ss.strip() + "***")
print("***" + ss.lstrip() + "***")
print("***" + ss.rstrip() + "***")
news = ss.replace("o", "***")
print(news)
```

Output

3

Hello, World

***Hello, World ***

*** Hello, World***

Hell***, W***rld

String Module

It's a built-in module and we have to import it before using any of its constants and classes.

a. String Module Constants

```
import string
# string module constants
print(string.ascii_letters)
print(string.ascii_lowercase)
print(string.ascii_uppercase)
print(string.digits)
print(string.hexdigits)
print(string.whitespace) # '\t\n\r\x0b\x0c'
print(string.punctuation)
```

Output

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
0123456789abcdefghijklmnopqrstuvwxyz
!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
```

b. String capwords() function

Python string module contains a single utility function – `capwords(s, sep=None)`. This function split the specified string into words using `str.split()`. Then it capitalizes each word using `str.capitalize()` function. Finally, it joins the capitalized words using `str.join()`.

If the optional argument `sep` is not provided or `None`, then leading and trailing whitespaces are removed and words are separated with single whitespace. If it's provided then the separator is used to split and join the words.

Example

```
s = ' Welcome TO \n\n JournalDev '
print(string.capwords(s))
```

Output

```
Welcome To Journaldev
```

c. Python String Module Classes

Python string module contains two classes – `Formatter` and `Template`.

Formatter

It behaves exactly same as `str.format()` function. This class becomes useful when own format string syntax need to be defined.

Example

```
from string import Formatter
formatter = Formatter()
print(formatter.format('{website}', website='JournalDev'))
```

```
print(formatter.format('{} {website}', 'Welcome to', website='JournalDev'))
# format() behaves in similar manner
print('{} {website}'.format('Welcome to', website='JournalDev'))
```

Output

Welcome to JournalDev
Welcome to JournalDev

d. Template

This class is used to create a string template for simpler string substitutions as described in [PEP 292](#). It's useful in implementing internationalization (i18n) in an application where there is no need of complex formatting rules.

Example

```
from string import Template
t = Template('$name is the $title of $company')
s = t.substitute(name='Pankaj', title='Founder', company='JournalDev.')
print(s)
```

Output

Pankaj is the founder of JournalDev

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=QGLNQwfTO2w>

Important Books/Journals for further learning including the page nos.:

3. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
4. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-17

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : II-Functions, Strings Date of Lecture:

Topic of Lecture: List as Arrays

Introduction :

- List is a collection of items of different data types
- Arrays are collection of items of same data types
- Arrays are imported from numpy package
- Elements in the list are accessed sequentially
- Elements in the array can be using index positions

Prerequisite knowledge for Complete understanding and learning of Topic:

- Arrays
- Lists
- Numpy Package

Detailed content of the Lecture:

List can be treated as array

```
a =[1,3.5,"Hello"]
```

```
import array as arr
a = arr.array('i',[2,4,6,8])

print("First element:", a[0])
print("Second element:", a[1])
print("Last element:", a[-1])
```

```
import array as arr
numbers_list =[2,5,62,5,42,52,48,5]
numbers_array = arr.array('i', numbers_list)
print(numbers_array[2:5])# 3rd to 5th
print(numbers_array[:-5])# beginning to 4th
print(numbers_array[5:])# 6th to end
print(numbers_array[:])# beginning to end
```

Output

```
array('i', [62, 5, 42])  
array('i', [2, 5, 62])  
array('i', [52, 48, 5])  
array('i', [2, 5, 62, 5, 42, 52, 48, 5])
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=2vmvtxHVPJI>

Important Books/Journals for further learning including the page nos.:

5. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
6. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-18

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : II-Functions, Strings

Date of Lecture:

Topic of Lecture: Examples for Lists as Arrays

Introduction :

- List is a collection of items of different datatypes
- Arrays are collection of items of same datatypes
- Arrays are imported from numpy package
- Elements in the list are accessed sequentially
- Elements in the array can be using index positions

Prerequisite knowledge for Complete understanding and learning of Topic:

- Arrays
- Lists
- Numpy Package

Detailed content of the Lecture: Examples

```
public class ListOfArrayExample {
public static void main(String[] args) {
// create a list of arrays
List<Integer[]> numbers = new ArrayList<Integer[]>();
// create integer arrays
Integer[] arrOne = {1,2,3,4};
Integer[] arrTwo = {5,6,7,8};
// add to list
numbers.add(arrOne);
numbers.add(arrTwo);
// iterate over list
for (Integer[] array : numbers) {
System.out.println(Arrays.toString(array));
}
}
}
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=2vmvtxHVPJI>

Important Books/Journals for further learning including the page nos.:

7. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no 3-4
8. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, Network Theory Ltd, 2011, Page no 83-85

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-19

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists, Tuples, Dictionaries Date of Lecture:

Topic of Lecture: List Operations, List Slices

Introduction :

- List is an ordered sequence of items.
- Values in the list are called elements/ items.
- It can be written as a list of comma-separated items(values)between **square brackets[]**.
- Items in the lists can be of different data types.
- Items in the list are indexed.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings
- Operations on Strings
- Arrays

Detailed content of the Lecture:

Operations on List

- Indexing –index values are used to locate the elements in the list
- Slicing- used to extract a portion of the list
- Concatenation-joins two list
- Repetition- contents of the lists are repeated specified number of times

List methods are insertion, updation and deletion

Creating a list	<pre>>>>list1=["python", 7.79, 101, "hello"] >>>list2=["god",6.78,9]</pre>	Creating the list with elements of different data types.
Indexing	<pre>>>>print(list1[0]) python >>> list1[2] 101</pre>	<ul style="list-style-type: none"> ❖ Accessing the item in the position 0 ❖ Accessing the item in the position 2
Slicing(ending position -1) Slice operator is used to extract part of a string, or some part of a list Python	<pre>>>> print(list1[1:3]) [7.79, 101] >>>print(list1[1:]) [7.79, 101, 'hello']</pre>	<ul style="list-style-type: none"> - Displaying items from 1st till 2nd. - Displaying items from 1st position till last.
Concatenation	<pre>>>>print(list1+list2) ['python', 7.79, 101, 'hello', 'god',</pre>	-Adding and printing the items of two lists.
	<pre>6.78, 9]</pre>	
Repetition	<pre>>>> list2*3 ['god', 6.78, 9, 'god', 6.78, 9, 'god', 6.78, 9]</pre>	Creates new strings, concatenating multiple copies of the same string
Updating the list	<pre>>>> list1[2]=45 >>>print(list1) ['python', 7.79, 45, 'hello']</pre>	Updating the list using index value
Inserting an element	<pre>>>> list1.insert(2,"program") >>> print(list1) ['python', 7.79, 'program', 45, 'hello']</pre>	Inserting an element in 2 nd position
Removing an element	<pre>>>> list1.remove(45) >>> print(list1) ['python', 7.79, 'program', 'hello']</pre>	Removing an element by giving the element directly
Video Content / Details of website for further learning (if any):		
<ol style="list-style-type: none"> 1. https://www.programiz.com/python-programming/list 2. https://www.geeksforgeeks.org/list-methods-python/ 		
Important Books/Journals for further learning including the page nos.:		
1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.91-93.		

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-20

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit : III-Lists, Tuples, Dictionaries Date of Lecture:

Topic of Lecture: List Methods, List Loop

Introduction :

- List is python data type
- It is one among the sequence data type
- List is collection of data items of different data types
- Operations performed on the list are indexing, slicing concatenation and repetition.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

List methods are

- Updating the List
- Inserting an element into the List
- Removing an element from the List

Python provides methods that operate on lists. For example, append adds a new element to the end of a list:

```
>>> t = ['a', 'b', 'c']
>>> t.append('d')
>>> print t
['a', 'b', 'c', 'd']
```

extend takes a list as an argument and appends all of the elements:

```
>>> t1 = ['a', 'b', 'c']
>>> t2 = ['d', 'e']
>>> t1.extend(t2)
>>> print t1
['a', 'b', 'c', 'd', 'e']
```

This example leaves t2 unmodified.

sort arranges the elements of the list from low to high:

```
>>> t = ['d', 'c', 'e', 'b', 'a']
```

```
>>> t.sort()
```

```
>>> print t
```

```
['a', 'b', 'c', 'd', 'e']
```

List methods are all void; they modify the list and return None

List Loop

The most common way to traverse the elements of a list is with a for loop. The syntax is the same as for strings:

Example 1

```
for cheese in cheeses:  
    print cheese
```

Example 2

```
for i in range(len(numbers)):  
    numbers[i] = numbers[i] * 2
```

This loop traverses the list and updates each element. **len** returns the number of elements in the list. **range** returns a list of indices from 0 to $n-1$, where n is the length of the list. Each time through the loop i gets the index of the next element. The assignment statement in the body uses i to read the old value of the element and to assign the new value.

A for loop over an empty list never executes the body:

```
for x in []:  
    print "This never happens."
```

Although a list can contain another list, the nested list still counts as a single element. The length of this list is four:

```
['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]
```

Video Content / Details of website for further learning (if any):

1. <https://www.programiz.com/python-programming/list>
2. <https://www.geeksforgeeks.org/list-methods-python/>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.91-92.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-21

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaries Date of Lecture:

Topic of Lecture: List Mutability, Aliasing, Cloning List, List Parameters

Introduction :

- Lists are mutable ie. Data items of the lists can be changed.
- Referring to a List with more than one name is called as List aliasing.
- Creating a copy of a List is called as Cloning.
- Cloning of a List can be done in several ways.
- List can be passed as Parameters to functions

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

Mutability

Lists are mutable. When the bracket operator appears on the left side of an assignment, it identifies the element of the list that will be assigned.

```
>>> numbers = [17, 123]
```

```
>>> numbers[1] = 5
```

```
>>> print numbers
```

```
[17, 5]
```

The one-eth element of numbers, which used to be 123, is now 5.

Aliasing

If a refers to an object and if b = a, then both variables refer to the same object:

```
>>> a = [1, 2, 3]
```

```
>>> b = a
```

```
>>> b is a
```

```
True
```

The association of a variable with an object is called a **reference**. In this example, there are two references to the same object.

An object with more than one reference has more than one name, so we say that the object is **aliased**.

If the aliased object is mutable, changes made with one alias affect the other:

```
>>> b[0] = 17
>>> print a
[17, 2, 3]
```

Although this behavior can be useful, it is error-prone. In general, it is safer to avoid aliasing when you are working with mutable objects.

For immutable objects like strings, aliasing is not as much of a problem. In this example:

```
a = 'banana'
b = 'banana'
```

It almost never makes a difference whether a and b refer to the same string or not.

List cloning

Cloning can be done using various methods such as

- Slicing
- Extend()
- Copy()
- List comprehension
- Append()

The above methods vary in execution time.

List parameters

When you pass a list to a function, the function gets a reference to the list. If the function modifies a list parameter, the caller sees the change. For example, `delete_head` removes the first element from a list:

```
def delete_head(t):
    del t[0]
```

Here's how it is used:

```
>>> letters = ['a', 'b', 'c']
>>> delete_head(letters)
>>> print letters
['b', 'c']
```

Video Content / Details of website for further learning (if any):

<https://www.geeksforgeeks.org/python-cloning-copying-list>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.90,96-97.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-22

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaries Date of Lecture:

Topic of Lecture: Tuple Assignment, Tuple as return value

Introduction :

- A tuple is a sequence of values.
- The values can be any type, and they are indexed by integers, so in that respect tuples are a lot like lists.
- The important difference is that tuples are immutable.
- Tuple can be of variable length.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

Tuple Assignment

It is often useful to swap the values of two variables. For example, to swap a and b:

```
>>> temp = a
>>> a = b
>>> b = temp
```

This solution is cumbersome; **tuple assignment** is more elegant:

```
>>> a, b = b, a
```

The left side is a tuple of variables; the right side is a tuple of expressions. Each value is assigned to its respective variable. All the expressions on the right side are evaluated before any of the assignments.

The number of variables on the left and the number of values on the right have to be the same:

```
>>> a, b = 1, 2, 3
ValueError: too many values to unpack
```

More generally, the right side can be any kind of sequence (string, list or tuple). For example, to split an email address into a user name and a domain, you could write:

```
>>> addr = 'monty@python.org'
>>> uname, domain = addr.split('@')
```

The return value from split is a list with two elements; the first element is assigned to uname, the second to domain.

```
>>> print uname
monty
>>> print domain
python.org
```

Video Content / Details of website for further learning (if any):

<https://realpython.com/lessons/tuple-assignment-packing-unpacking/>

<https://runestone.academy/runestone/books/published/thinkcspy/Lists/TuplesasReturnValues.html>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.116-117.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-23

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaries Date of Lecture:

Topic of Lecture: Example for tuples

Introduction :

- A tuple is a sequence of values.
- The values can be any type, and they are indexed by integers, so in that respect tuples are a lot like lists.
- The important difference is that tuples are immutable.
- Tuple can be of variable length.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

Tuples as return values

A function can only return one value, but if the value is a tuple, the effect is the same as returning multiple values. For example, if you want to divide two integers and compute the quotient and remainder, it is inefficient to compute x/y and then $x\%y$. It is better to compute them both at the same time.

The built-in function `divmod` takes two arguments and returns a tuple of two values, the quotient and remainder. You can store the result as a tuple:

```
>>> t = divmod(7, 3)
>>> print t
(2, 1)
```

Or use tuple assignment to store the elements separately:

```
>>> quot, rem = divmod(7, 3)
>>> print quot
2
>>> print rem
1
```

Here is an example of a function that returns a tuple:

```
def min_max(t):  
    return min(t), max(t)
```

max and min are built-in functions that find the largest and smallest elements of a sequence. min_max computes both and returns a tuple of two values.

Video Content / Details of website for further learning (if any):

<https://realpython.com/lessons/tuple-assignment-packing-unpacking/>

<https://runestone.academy/runestone/books/published/thinkcspy/Lists/TuplesasReturnValues.html>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.116-117.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-24

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaryes Date of Lecture:

Topic of Lecture: Dictionary Operations

Introduction :

- Lists are ordered sets of objects,where as **dictionaries are unordered sets.**
- Dictionary is related by using **curly brackets** i.e. { }
- Dictionaries **are accessed via keys** and not via their position.
- A dictionary is an associative array(also known as hashes).Any key of the dictionary is associated(or mapped)to a value.
- The values of a dictionary can be any Python data type. So dictionaries are **Unordered key-value-pairs**(The association of a key and a value is called a key-value pair)

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

- Dictionaries don't support the sequence operation of the sequence data types like strings, tuples and lists.

Creating a dictionary	<pre>>>> food = {"ham": "yes", "egg": "yes", "rate": 450 } >>> print(food) {'rate': 450, 'egg': 'yes', 'ham': 'yes'}</pre>	Creating the dictionary with elements of different data types.
Indexing	<pre>>>>> print(food["rate"]) 450</pre>	Accessing the item with keys.
Slicing(ending position -1)	<pre>>>> print(t[1:3]) (7.79, 101)</pre>	Displaying items from 1st till 2nd.

If you try to access a key which doesn't exist, you will get an error message:

```
>>> words = {"house": "Haus", "cat": "Katze"}
>>> words["car"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'car'
```

Data type	Compile time	Run time
int	a=10	a=int(input("enter a"))
float	a=10.5	a=float(input("enter a"))
string	a="panimalar"	a=input("enter a string")
list	a=[20,30,40,50]	a=list(input("enter a list"))
tuple	a=(20,30,40,50)	a=tuple(input("enter a tuple"))

Video Content / Details of website for further learning (if any):

<https://www.programiz.com/python-programming/dictionary>

https://www.w3schools.com/python/python_dictionaries.asp

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.103-107.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-25

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaries Date of Lecture:

Topic of Lecture: Dictionary Methods

Topic of Lecture: Examples for Dictionary Methods

Introduction :

- Lists are ordered sets of objects, where as **dictionaries are unordered sets.**
- Dictionary is created by using **curly brackets**,i.e. { }
- Dictionaries **are accessed via keys** and not via their position.
- A dictionary is an associative array(also known as hashes).Any key of the dictionary is associated(or mapped)to a value.
- The values of a dictionary can be any Python data type. So dictionaries are **Unordered key-value-pairs**(The association of a key and a value is called a key-value pair)

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

The function dict creates a new dictionary with no items. Because dict is the name of a built-in function, hence variable name can be avoided.

```
>>> eng2sp = dict()
>>> print eng2sp{ }
```

The squiggly-brackets, {}, represent an empty dictionary. To add items to the dictionary, you can use square brackets:

```
>>> eng2sp['one'] = 'uno'
```

This line creates an item that maps from the key 'one' to the value 'uno'. If we print the dictionary again, we see a key-value pair with a colon between the key and value:

```
>>> print eng2sp
```

```
{'one': 'uno'}
```

This output format is also an input format. For example, you can create a new dictionary with three items:

```
>>> eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

```
>>> print eng2sp
```

```
{'one': 'uno', 'three': 'tres', 'two': 'dos'}
```

The order of the key-value pairs is not the same. In fact, if you type the same example on your computer, you might get a different result. In general, the order of items in a dictionary is unpredictable.

But that's not a problem because the elements of a dictionary are never indexed with integer indices. Instead, you use the keys to look up the corresponding values:

```
>>> print eng2sp['two']
```

```
'dos'
```

The key 'two' always maps to the value 'dos' so the order of the items doesn't matter.

If the key isn't in the dictionary, you get an exception:

```
>>> print eng2sp['four']
```

```
KeyError: 'four'
```

The len function works on dictionaries; it returns the number of key-value pairs:

```
>>> len(eng2sp)
```

```
3
```

The in operator works on dictionaries; it tells you whether something appears as a *key* in the dictionary (appearing as a value is not good enough).

```
>>> 'one' in eng2sp
```

```
True
```

```
>>> 'uno' in eng2sp
```

```
False
```

To see whether something appears as a value in a dictionary, you can use the method values, which returns the values as a list, and then use the in operator:

```
>>> vals = eng2sp.values()
```

```
>>> 'uno' in vals
```

```
True
```

Video Content / Details of website for further learning (if any):

<https://www.programiz.com/python-programming/dictionary>

https://www.w3schools.com/python/python_dictionaries.asp

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-26

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III- Lists,Tuples,Dictionaryes Date of Lecture:

Topic of Lecture: Advanced list Processing

Introduction :

- List is collection of data items of different data types.
- List operations are indexing, slicing, repetition etc.
- Lists are mutable hence supports addition, deletion and updation of data items.
- Advanced List operations are List comprehension and Nested Lists.
- Data items within Nested Lists are accessed using multiple indices.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

List Comprehension:

- List comprehensions provide a concise way to apply operations on a list.
- It creates a new list in which each element is the result of applying a given operation in a list.
- It consists of brackets containing an expression followed by a “for” clause, then a list.
- The list comprehension always returns a result list.

Syntax

list=[expression for item in list if conditional]

List Comprehension	Output
>>>L=[x**2 for x in range(0,5)] >>>print(L)	[0, 1, 4, 9, 16]
>>>[x for x in range(1,10) if x%2==0]	[2, 4, 6, 8]
>>>[x for x in 'Python Programming' if x in ['a','e','i','o','u']]	['o', 'o', 'a', 'i']
>>>mixed=[1,2,"a",3,4.2] >>> [x**2 for x in mixed if type(x)==int]	[1, 4, 9]
>>>[x+3 for x in [1,2,3]]	[4, 5, 6]
>>> [x*x for x in range(5)]	[0, 1, 4, 9, 16]
>>> num=[-1,2,-3,4,-5,6,-7] >>> [x for x in num if x>=0]	[2, 4, 6]
>>> str=["this","is","an","example"] >>> element=[word[0] for word in str] >>> print(element)	['t', 'i', 'a', 'e']

Nested list:

List inside another list is called nested list.

Example:

```
>>> a=[56,34,5,[34,57]]
```

```
>>> a[0]
```

```
56
```

```
>>> a[3]
```

```
[34, 57]
```

```
>>> a[3][0]
```

```
34
```

```
>>> a[3][1]
```

```
57
```

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Advanced-list-processing---Python_35943/

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.89-95.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-27

CSE

I / II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : III-Lists,Tuples,Dictionaries Date of Lecture:

Topic of Lecture: List Comprehension

Introduction :

- List is collection of data items of different data types.
- Advanced List operations are List comprehension and Nested Lists.
- List comprehension is a method of creating a new list.
- There are various ways for creating a new list like cloning.
- List comprehension is the simplest way of creating a new list.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data Types
- Strings and Operations on Strings
- List Operations
- Arrays

Detailed content of the Lecture:

List comprehension is an elegant way to define and create list in Python. These lists have often the qualities of sets, but are not in all cases sets.

List comprehension is a complete substitute for the lambda function as well as the functions map(), filter() and reduce().

Example 1:

```
>>> Celsius = [39.2, 36.5, 37.3, 37.8]
>>> Fahrenheit = [ ((float(9)/5)*x + 32) for x in Celsius ]
>>> print Fahrenheit
[102.56, 97.70000000000003, 99.14000000000001, 100.03999999999999]
>>>
```

Example 2:

Cross product of two sets

```
>>> colours = [ "red", "green", "yellow", "blue" ]
>>> things = [ "house", "car", "tree" ]
>>> coloured_things = [ (x,y) for x in colours for y in things ]
>>> print coloured_things
[('red', 'house'), ('red', 'car'), ('red', 'tree'), ('green', 'house'), ('green', 'car'), ('green', 'tree'), ('yellow', 'house'), ('yellow', 'car'), ('yellow', 'tree'), ('blue', 'house'), ('blue', 'car'), ('blue', 'tree')]
>>>
```

Generator comprehensions were introduced with Python 2.6. They are simply a generator expression with a parenthesis - round brackets - around it. Otherwise, the syntax and the way of working is like list comprehension, but a generator comprehension returns a generator instead of a list.

```
>>> x = (x **2 for x in range(20))
>>> print(x)
at 0xb7307aa4>
>>> x = list(x)
>>> print(x)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361]
```

Video Content / Details of website for further learning (if any):

https://www.python-course.eu/list_comprehension.php

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey Think Python: How to Think Like a Computer Scientist O'Reilly Publishers 2016. pp.93.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-28

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Files and exception: text files

Introduction :

- Files and Exception are the most important aspect of an application in python.
- Python has several functions for creating, reading, updating, and deleting files.
- File handling is an important part of any web application
- Users can also create their own Files. To store data and retrieve data.
- There are different types of files such as binary files and text files

Prerequisite knowledge for Complete understanding and learning of Topic:

- Files
- File Open
- File Close

Detailed content of the Lecture:

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.

Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

There are two types of Files they are

- Binary files
- Text files

File operation and file functions

A file is one, which will enable the user to read, write and store a group of related data, without losing them even if the program is over. To perform these functions there are several basic file operations as,

- Naming a file - It is conventional, and convenient to name files in relation to the data stored.
- Opening a file
- Reading data from the file
- Writing data to the file and
- Closing a file

open()

Before going to do any file operation, the concerned file should be opened.

Syntax:

file_object=open("file_name", "file_mode")

The **open()** function creates a file object. Finally it should be given, as to what purpose the file is being used. It is also called the file mode.

file_mode	Description
R	Opens a file for reading purpose and this is the default file opening mode/
W	Opens a file for writing purpose
A	Opens a file for appending data to it.
r+	Existing file is opened to the beginning for both reading and writing
w+	Opens a file for reading and writing purpose. The file pointer is positioned at the beginning of the file.
a+	Opens a file for reading and writing purpose. The file pointer is positioned at the end of the file
Rb	Opens a file for reading in binary format and this is the default file opening mode.
Ab	Opens a file for appending data in binary format
Wb	Opens a file for writing in binary format
rb+	Opens a file for reading and writing purpose in a binary format and the file pointer is positioned at the beginning of the file
wb+	Opens a file for reading and writing purpose in a binary format. If the file already exists it overwrites the file. If the file does not exist it will create a new one for both reading and writing.
ab+	Opens a file for appending and reading data in binary format. Here the file pointer is positioned at end if the file already exists. It will create a new one if the file does not exists.

For example, a simple open() function call is as follows.

```
out=open("abc.dat", "w")
```

close()

This function closes a file that was opened by a call **open()**.

The general form of the function call to **close()** is

```
close(file-object)
```

(e.g.)

```
p1=open("abc.dat", "w");
```

```
p2=open("def.dat", "r");
```

```
_____  
_____
```

```
close(p1); close(p2);
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=ixEeeNjjOJ0>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no .137-138.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-29

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit :IV- Files,Modules,Packages Date of Lecture:

Topic of Lecture: Reading files, Writing files

Introduction :

- Retrieve the data from Files using read()
- Store the data to files using write ().
- Basic of all input/output functions in file handling
- Users can also create their own Files to store data and retrieve data.
- File need to be opened before read or Write operation and finally closed.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Read File
- Write File

Detailed content of the Lecture:

read()andwrite()

These two functions are considered as most basic of all input/output functions in file handling. The write() function writes characters to a disk file that was previously opened for writing,through the use of the function **open()**.Similarly read() function is used to read characters from a file opened in read mode by **open()**.The general format for read() methodis:

```
file_object.read(no_of_bytes_to_be_read)
```

The write() method is used to write data to a file. This method requires only one parameter, that must be a string and writes that string into a file. The general for write() method is:

```
file_object.write(string)
```

Text file –example:

```
F1=open("abc.txt","x")
out=open("abc.dat","w") str=
input("Enter string : ")
out.write(str)
out.close()
out=open("abc.dat","r")
str=out.read() print("File
contains") print(str)
out.close()
```

Output

Enter string : Welcome to Python file handling

File contains

Welcome to Python filehandling

Read Only Parts of the File

By default the `read()` method returns the whole text, but you can also specify how many characters you want to return:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

You can return one line by using the `readline()` method:

Example

Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

Write ()

To write to an existing file, you must add a parameter to the `open()` function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Example

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

#open and read the file after the appending:

```
f = open("demofile2.txt", "r")
print(f.read())
```

Example

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

#open and read the file after the appending:

```
f = open("demofile3.txt", "r")
print(f.read())
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=vRLOZSjwbS0>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no. 291-294.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-30

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Format Operators

Introduction :

- Command Line arguments are passed to the program using \$python command
- Used to send the input directly to the program
- sys.argv is imported to access the command line arguments
- The number of arguments can be counted and displayed
- Any number of arguments can be passed as command line arguments

Prerequisite knowledge for Complete understanding and learning of Topic:

- Command line arguments
- sys.srgv

Detailed content of the Lecture:

Command line arguments:

Input can be directly sent as an argument. When the program is running under command prompt then inputs can be passed directly in the command and can be fetched using "sys" module.

Important steps to be followed:

- Import the module 'sys'.
- Use sys.argv for getting the list of command line arguments.
- Use len (sys.argv) for getting total number of arguments.

Example program:

```
import sys
noargs=len(sys.argv)
print ("Number of arguments :%d" %noargs)
arguments= str(sys.argv)
print ("Arguments are : %s" %arguments)
```

Output

```
C:\Python27>python cmd1.py one two
Number of arguments :3
```

```
Arguments are : ['cmd1.py', 'one', 'two']
```

Python provides a **getopt** module that helps you parse command-line options and arguments.

```
$ python test.py arg1 arg2 arg3
```

The Python **sys** module provides access to any command-line arguments via the **sys.argv**. This serves two purposes –

- `sys.argv` is the list of command-line arguments.
- `len(sys.argv)` is the number of command-line arguments.

Here `sys.argv[0]` is the program ie. script name.

Example

Consider the following script `test.py` –

```
#!/usr/bin/python
import sys
print'Number of arguments:', len(sys.argv),'arguments.'
print'Argument List:', str(sys.argv)
```

Now run above script as follows –

```
$ python test.py arg1 arg2 arg3
```

This produce following result –

```
Number of arguments: 4 arguments.
Argument List: ['test.py', 'arg1', 'arg2', 'arg3']
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=d3uv23jvp4w>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 . Pg-No. 138-139.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-30

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Command Line Arguments

Introduction :

- Command Line arguments are passed to the program using \$python command
- Used to send the input directly to the program
- sys.argv is imported to access the command line arguments
- The number of arguments can be counted and displayed
- Any number of arguments can be passed as command line arguments

Prerequisite knowledge for Complete understanding and learning of Topic:

- Command line arguments
- sys.srgv

Detailed content of the Lecture:

Command line arguments:

Input can be directly sent as an argument. When the program is running under command prompt then inputs can be passed directly in the command and can be fetched using "sys" module.

Important steps to be followed:

- Import the module 'sys'.
- Use sys.argv for getting the list of command line arguments.
- Use len (sys.argv) for getting total number of arguments.

Example program:

```
import sys
noargs=len(sys.argv)
print ("Number of arguments :%d" %noargs)
arguments= str(sys.argv)
print ("Arguments are : %s" %arguments)
```

Output

```
C:\Python27>python cmd1.py one two
Number of arguments :3
Arguments are : ['cmd1.py', 'one', 'two']
```

Python provides a **getopt** module that helps you parse command-line options and arguments.

```
$ python test.py arg1 arg2 arg3
```

The Python **sys** module provides access to any command-line arguments via the **sys.argv**. This serves two purposes –

- `sys.argv` is the list of command-line arguments.
- `len(sys.argv)` is the number of command-line arguments.

Here `sys.argv[0]` is the program ie. script name.

Example

Consider the following script `test.py` –

```
#!/usr/bin/python
import sys
print'Number of arguments:', len(sys.argv),'arguments.'
print'Argument List:', str(sys.argv)
```

Now run above script as follows –

```
$ python test.py arg1 arg2 arg3
```

This produce following result –

```
Number of arguments: 4 arguments.
Argument List: ['test.py', 'arg1', 'arg2', 'arg3']
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=d3uv23jvp4w>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 . Pg-No. 138-139.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-31

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Errors and exceptions

Introduction :

- Errors to refer as bugs.
- Syntax errors are found at the time of compilation
- Run time errors are called as Exceptions
- There are various types of Exceptions
- Exceptions are handled using try and exception block

Prerequisite knowledge for Complete understanding and learning of Topic:

- Errors
- Types of Errors
- Exception
- Structure of exception

Detailed content of the Lecture:

Errors and Exceptions:

Errors – referred as bugs in the program.Errors occurs maximum by the fault of the programmer.

Debugging – Process of finding and correcting errors.

Two types of errors.:

• **Syntax errors**

python interpreter find the syntax error when it executes the coding. Once find the error, it displays the error by stopping the execution.

Common occurring syntax errors are

- Putting a keyword at wrongplace
- Misspelling thekeyword
- Incorrectindentation
- Forgetting symbols like comma, brackets, quotes (“ or ‘)
- Emptyblock

•**Run timeerrors**

if a program is free of syntax errors then it runs by the interpreter and the errors occurs during the run time ofthe program due to logical mistake is called runtimeerrors.

Examples:

- Trying to access a file that doesn'texists
- Performing the operations like division byzero

➤ Using an identifier which is not defined

```
(a,b)=(6,0)
try:# simple use of try-except block for handling errors
    g = a/b
except ZeroDivisionError:
print("This is a DIVIDED BY ZERO error")
```

Exceptions:

An exception is an event, which occurs during the execution of the program that disrupts the normal flow of the program.

When the program raises an exception, then python must handle the exception otherwise it terminates and quits

Exceptions handling in Python are very similar to Java. The code, which harbours the risk of an exception, is embedded in a try block. But whereas in Java exceptions are caught by catch clauses, we have statements introduced by an "except" keyword in Python. It's possible to create "custom-made" exceptions: With the raise statement it's possible to force a specified exception to occur.

Let's look at a simple example. Assuming we want to ask the user to enter an integer number. If we use a input(), the input will be a string, which we have to cast into an integer. If the input has not been a valid integer, we will generate (raise) a ValueError. We show this in the following interactive session:

```
>>> n = int(input("Please enter a number: "))
Please enter a number: 23.5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module> ValueError: invalid literal for int() with base 10: '23.5'
```

Structure of blocks of exceptions

try:

write the suspicious code here

except exception1:

If exception1 occurs then this block will be executed

except exception2:

If exception2 occurs then this block will be executed.

..

else:

If there is no exception then this code will be executed

Example program:

try:

```
n=int(input("enter a value"))
```

```
except:
```

```
print("you didn't enter the integer input")
```

else:

```
print("value entered correctly and stored")
```

output:

enter a value:5

value entered correctly and stored

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=O2Nk3JFZE58>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers,
2016 .Pg.No.1401-41

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-32

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Types of Exceptions

Introduction :

- Disrupts the normal flow of the program.
- Handling mechanism are try, except and else blocks.
- Python must handle the exception otherwise it terminates and quits
- The code that generates exception are given within try block
- The exception are handling using exception block

Prerequisite knowledge for Complete understanding and learning of Topic:

- try
- exception
- else

Detailed content of the Lecture:

Definition

- An exception is an event, which occurs during the execution of the program that disrupts the normal flow of the program.
- When the program raises an exception, then python must handle the exception otherwise it terminates and quits
- The handling mechanism is done by
- try, except and else blocks
- try block – suspicious code (code that makes exception) placed here
- except block – code that handles the exception placed here and gets executed during exception
- else block – code that is to be executed if no exception is placed here for normal execution

Structure of blocks of exceptions

try:

write the suspicious code here except exception1:

If exception1 occurs then this block will be executed except exception2:

If exception2 occurs then this block will be executed

```
.  
..  
else:
```

If there is no exception then this code will be executed

Example program:

```
try:  
n=int(input("enter a value"))  
print("you didn't enter the integer input")  
else:  
print("value entered correctly and stored")
```

```
output:  
enter a value:5  
value entered correctly and stored
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=Ia1i5EIGp9k>

Important Books/Journals for further learning including the page nos.:

1. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 .pg.No:67-71.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-33

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit :IV- Files,Modules,Packages Date of Lecture:

Topic of Lecture: Handling Exceptions

Introduction :

- Modules are collection of classes and functions
- The necessary modules are imported and used in the program
- Modules Logically arrange related code.
- Makes the code easier to understand and use.
- Python has many useful functions and resources in modules.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Import modules
- Bulit-in function
- Classes

Detailed content of the Lecture:

Syntax:

- import module
- from module import function
- from module import *
- When import statement is encountered by the interpreter, the corresponding module is imported.
- **import statement**
- Example:
 - import math
 - Print(math.sqrt(25))
- from module import function
- It allows us to import specific attributes from a module into the current namespace.
- Example:
 - from math import sqrt

– print sqrt(25)

- from module import *
- It allows us to import all names from a module into current namespace

- Example:

– from math import *

– >>> print(sqrt(25))

Writing own modules

```
def add(a,b):
    print("result of addition is ",a+b)
    return
def sub(a,b):
    print("result of subtraction is ",a-b)
    return
def mul(a,b):
    print("result of multiplication is ",a*b)
    return
```

Save the above program as cal.py. Now cal.py file can be imported as a module in another python source file and its functions can be called from the new file as below

```
>>> import cal
>>> cal.add(3,4)
result of addition is 7
```

Locating modules

- When you import a module, the Python interpreter searches for the module in the following sequences:
 - The current directory.
 - If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
 - If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.
- The module search path is stored in the system module sys as the sys.path variable.
- The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=f26nAmfJggw>

Important Books/Journals for further learning including the page nos.:

1.Allen B. Downey,Think Python: How to Think Like a Computer Scientist,O'Reilly Publishers, 2016.Pg.No:143-144.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-34

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Modules, Packages

Introduction :

- Large number of modules are organized into packages
- Similar modules are placed in one package
- Different modules are placed in different packages
- Package is a hierarchical file directory structure
- Packages are imported and used in the program

Prerequisite knowledge for Complete understanding and learning of Topic:

- Similar Modules
- Different Modules

Detailed content of the Lecture:

Steps to create package

- Create a directory and give it your package's name.
- Put modules in it.
- Create a `__init__.py` file in the directory
 - The `__init__.py` file is necessary because with this file, Python will know that this directory is a Python package directory other than an ordinary directory.
 - One can import necessary modules in this python file

Example package creation cal.py

```
def add(a,b):
    print("result of addition is ",a+b)
    return
def sub(a,b):
    print("result of subtraction is ",a-b)
    return
def mul(a,b):
    print("result of multiplication is ",a*b)
    return
__init__.py
```

```
from cal import add
from cal import sub
from cal import mul
from week import day1
from week import day2
from week import day3
from week import day4
from week import day5
from week import day6
from week import day7
```

Week.py

```
def day1():
    print("sunday")
def day2():
    print("monday")
def day3():
    print("tuesday")
def day4():
    print("wednesday")
def day5():
    print("thursday")
def day6():
    print("friday")
def day7():
    print("saturday")
```

Main program – sample.py

```
import mypack
mypack.day1()
mypack.add(2,3)
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=qmsTqQbcBNM>

Important Books/Journals for further learning including the page nos.:

2. Guido van Rossum and Fred L. Drake Jr, An Introduction to Python, 2011.Pg.No:55.

Network Theory Ltd,

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-35

CSE

I/II

Course Name with Code :21GES08&Python Programming

Course Faculty :S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Programming with Packages

Introduction :

- Packages are a way of structuring many packages and modules.
- Packages help in a well-organized hierarchy of data set, making the directories and modules easy to access.
- Packages help us in storing other sub-packages and modules, so that it can be used by the user when necessary.
- `__init__.py` implies subpackages within package .
- `__init__.py` file can be left blank or can be coded with the initialization code for the package

Prerequisite knowledge for Complete understanding and learning of Topic:

- Objects
- Classes
- Modules

Detailed content of the Lecture:

Package is basically a directory with Python files and a file with the name `__init__.py`. This means that every directory inside of the Python path, which contains a file named `__init__.py`, will be treated as a package by Python. It's possible to put several modules into a Package.v

Example:

```
def bar():
    print("Hello, function 'bar' from module 'a' calling")
def foo():
    print("Hello, function 'foo' from module 'b' calling")
>>> from simple_package import a, b
>>> a.bar()
Hello, function 'bar' from module 'a' calling
>>> b.foo()
Hello, function 'foo' from module 'b' calling
>>>
```

```
>>> import simple_package
>>>
>>> simple_package.a.bar()
Hello, function 'bar' from module 'a' calling
>>>
>>> simple_package.b.foo()
Hello, function 'foo' from module 'b' calling
```

Video Content / Details of website for further learning (if any):

https://www.python-course.eu/python3_packages.php

<https://packaging.python.org/overview/>

Important Books/Journals for further learning including the page nos.:

-

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-36

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :IV-Files,Modules,Packages Date of Lecture:

Topic of Lecture: Packages with examples

Introduction :

- Retrieve the data from Files using read()
- Store the data to files using write ().
- Basic of all input/output functions in file handling
- Users can also create their own Files to store data and retrieve data.
- File need to be opened before read or Write operation and finally closed.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Read File
- Write File

Detailed content of the Lecture:

read()andwrite()

These two functions are considered as most basic of all input/output functions in file handling. The write() function writes characters to a disk file that was previously opened for writing,through the use of the function **open()**.Similarly read() function is used to read characters from a file opened in read mode by **open()**.The general format for read() methodis:

```
file_object.read(no_of_bytes_to_be_read)
```

The write() method is used to write data to a file. This method requires only one parameter, that must be a string and writes that string into a file. The general for write() method is:

```
file_object.write(string)
```

Text file –example:

```
F1=open("abc.txt","x")
out=open("abc.dat","w") str=
input("Enter string : ")
out.write(str)
out.close()
out=open("abc.dat","r")
str=out.read() print("File
contains") print(str)
out.close()
```

Output

Enter string : Welcome to Python file handling

File contains

Welcome to Python filehandling

Read Only Parts of the File

By default the `read()` method returns the whole text, but you can also specify how many characters you want to return:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

You can return one line by using the `readline()` method:

Example

Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

Write ()

To write to an existing file, you must add a parameter to the `open()` function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Example

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

#open and read the file after the appending:

```
f = open("demofile2.txt", "r")
print(f.read())
```

Example

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

#open and read the file after the appending:

```
f = open("demofile3.txt", "r")
print(f.read())
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=vRLOZSjwbS0>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers,
2016 Page no. 291-294.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-37

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :V- Tensorflow,Keras Date of Lecture:

Topic of Lecture: Introduction To Tensorflow

Introduction :

- TensorFlow is **an open source library for numerical computation and large-scale machine learning**.
- TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor.

Prerequisite knowledge for Complete understanding and learning of Topic:

- TensorFlow works with Python 2.7 and Python 3.3+.
- Import the Fashion MNIST dataset.
- Train and evaluate your model.
- Add TensorFlow Serving distribution URI as a package source:
- Install TensorFlow Serving.

Detailed content of the Lecture:

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.

Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

The goal of this article is to define and solve practical use cases with TensorFlow. To do so, we will solve. An optimization problem

- A linear regression problem, where we will adjust a regression line to a dataset
- And we will end solving the "Hello World" of Deep Learning classification projects with the MNIST Dataset.

PROGRAM:

```
import numpy as np
import tensorflow as tfx = tf.Variable(initial_value=tf.random_uniform([1], 34, 35),name='x')
y = tf.Variable(initial_value=tf.random_uniform([1], 0., 50.), name='y')# Loss function
s = tf.add(tf.add(632.0, tf.multiply(8.0, y)), tf.divide(2400.0, y), 's')opt =
tf.train.GradientDescentOptimizer(0.05)
train = opt.minimize(s)sess = tf.Session()init = tf.initialize_all_variables()
sess.run(init)old_solution = 0
tolerance = 1e-4
```

```
for step in range(500):
    sess.run(train)
    solution = sess.run(y)
    if np.abs(solution - old_solution) < tolerance:
        print("The solution is y = {}".format(old_solution))
        break

    old_solution = solution
    if step % 10 == 0:
        print(step, "y = " + str(old_solution), "s = " + str(sess.run(s)))
```

TensorFlow :

- Developed by Google Brain Team
- Written in C++, Python, and CUDA

TensorFlow Applications:

- Face detection in electronic devices.
- Machine language translation through apps such as Google Translate.
- Fraud detection in the banking and financial sectors.
- Object detections on videos.

The APIs in languages other than Python are not yet covered by the API stability promises.

Python

JavaScript

C++

Java

Video Content / Details of website for further learning (if any):

https://www.youtube.com/results?search_query=videos+for+tensorflow

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no .137-138.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-38

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :V- Tensorflow,Keras Date of Lecture:

Topic of Lecture: Tensor flow Graphs

Introduction :

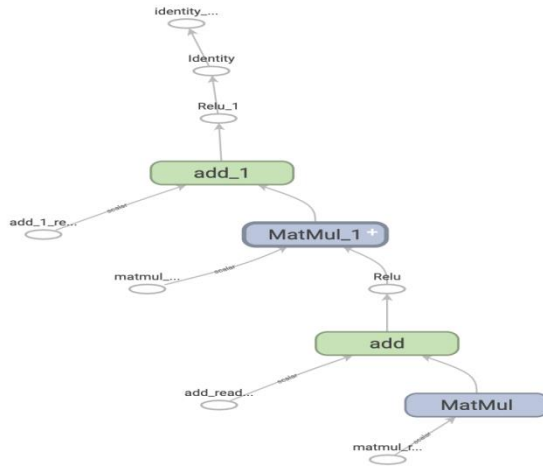
- "A computational graph (or graph in short) is a series of TensorFlow operations arranged into a graph of nodes".
- Basically, it means a graph is just an arrangement of nodes that represent the operations in your model.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Read File
- Write File

Detailed content of the Lecture:

- In the previous three guides, you ran TensorFlow **eagerly**. This means TensorFlow operations are executed by Python, operation by operation, and returning results back to Python.
- While eager execution has several unique advantages, graph execution enables portability outside Python and tends to offer better performance.
- Graph execution means that tensor computations are executed as a *TensorFlow graph*, sometimes referred to as a [tf.Graph](#) or simply a "graph".
- "Graphs are data structures that contain a set of [tf.Operation](#) objects, which represent units of computation; and [tf.Tensor](#) objects, which represent the units of data that flow between operations. They are defined in a [tf.Graph](#) context.
- Since these graphs are data structures, they can be saved, run, and restored all without the original Python code. This is what a TensorFlow graph representing a two-layer neural network looks like when visualized in TensorBoard.



Taking advantage of graphs

Define a Python function.

```
def a_regular_function(x, y, b):
    x = tf.matmul(x, y)
    x = x + b
    return x
```

```
# `a_function_that_uses_a_graph` is a TensorFlow `Function`.
a_function_that_uses_a_graph = tf.function(a_regular_function)
```

Make some tensors.

```
x1 = tf.constant([[1.0, 2.0]])
y1 = tf.constant([[2.0], [3.0]])
b1 = tf.constant(4.0)
```

```
orig_value = a_regular_function(x1, y1, b1).numpy()
# Call a `Function` like a Python function.
tf_function_value = a_function_that_uses_a_graph(x1, y1, b1).numpy()
assert(orig_value == tf_function_value)
```

Traffic Prediction, rumor and fake news detection, modeling disease spread, physics simulations, and understanding why molecules smell.

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=vRLOZSjwbS0>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no. 291-294.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-39

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : V- Tensorflow,Keras

Date of Lecture:

Topic of Lecture: Variables, Placeholders

Introduction :

- A TensorFlow **variable** is the recommended way to represent shared, persistent state your program manipulates. This guide covers how to create, update, and manage instances of [tf.Variable](#) in TensorFlow.
- Use [variables](#) to hold and update parameters. Variables are in-memory buffers containing tensors. They must be explicitly initialized and can be saved to disk during and after training. You can later restore saved values to exercise or analyse the model.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Many different ways to use TensorFlow.
- Be mindful of the differences between TensorFlow versions.
- TensorFlow doesn't abstract as many of the hard parts of programming as most APIs do.
- You still need to (mostly) understand ML.
- Gathering and preparing training data

Detailed content of the Lecture:

- - A placeholder is **simply a variable that we will assign data to at a later date.**
 - It allows us to create our operations and build our computation graph, without needing the data

Create two variables.

```
weights = tf.Variable(tf.random_normal([784, 200], stddev=0.35),  
                      name="weights")
```

```
biases = tf.Variable(tf.zeros([200]), name="biases")
```

...

Add an op to initialize the variables.

```
init_op = tf.initialize_all_variables()
```

Later, when launching the model

```
with tf.Session() as sess:
```

```
    # Run the init operation.
```

```
    sess.run(init_op)
```

...

```
    # Use the model
```

Create some variables.

```
v1 = tf.Variable(..., name="v1")
```

```
v2 = tf.Variable(..., name="v2")
```

```
...
# Add an op to initialize the variables.
init_op = tf.initialize_all_variables()

# Add ops to save and restore all the variables.
saver = tf.train.Saver()

# Later, launch the model, initialize the variables, do some work, save the
# variables to disk.
with tf.Session() as sess:
    sess.run(init_op)
    # Do some work with the model.
    ..
    # Save the variables to disk.
    save_path = saver.save(sess, "/tmp/model.ckpt")
    print "Model saved in file: ", save_path
```

Video Content / Details of website for further learning (if any):

<https://github.com/tensorflow/tensorflow>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 Page no. 291-294.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-40

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty :S.Suvitha

Unit :V- Tensorflow,Keras Date of Lecture:

Topic of Lecture: Download Tensorflow

Introduction :

- TensorFlow is an **open source library and can be download and used it for free**. In this article, we will see how to install TensorFlow on a Windows machine.
- Click on Install on top navigation bar of Tensorflow website
- Before proceeding we need to get python environment.
- go to python section and install python environment to work
- Python environment can be **downloaded from python.org**
-

Prerequisite knowledge for Complete understanding and learning of Topic:

- GPU drivers
- CUDA Toolkit: CUDA 9.0.
- NCCL 2.2 (optional)
- cuDNN SDK (7.2 or higher)
- TensorRT for improved latency and throughput.

Detailed content of the Lecture:

- On Windows, TensorFlow can be installed via either "pip" or "anaconda"
- Python comes with the **pip** package manager, so if you have already installed Python, then you should have **pip** as well.
- The package can install TensorFlow together with its dependencies.

The TensorFlow Python API supports Python 2.7 and Python 3.3+.

The GPU version works best with Cuda Toolkit 7.5 and cuDNN v5. Other versions are supported (Cuda toolkit >= 7.0 and cuDNN >= v3) only when installing from sources. Please see [Cuda installation](#) for details. For Mac OS X, please see [Setup GPU for Mac](#).

\$ python

```

...
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print(sess.run(a + b))
42

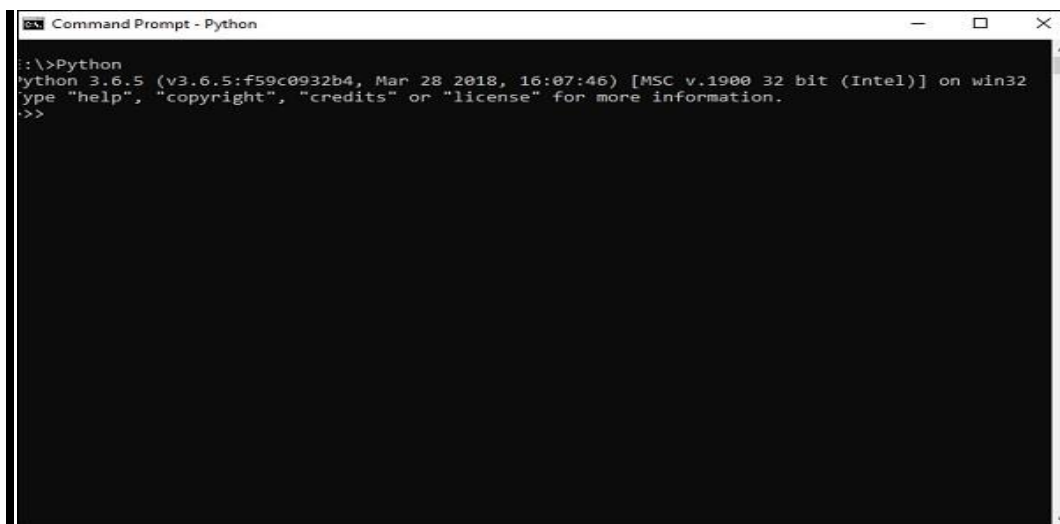
```

```
>>>
/usr/local/lib/python2.7/dist-packages/tensorflow
/usr/local/lib/python2.7/site-packages/tensorflow
# Using 'python -m' to find the program in the python search path:
$ python -m tensorflow.models.image.mnist.convolutional
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
...etc.
```

To install TensorFlow, it is important to have “Python” installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation.

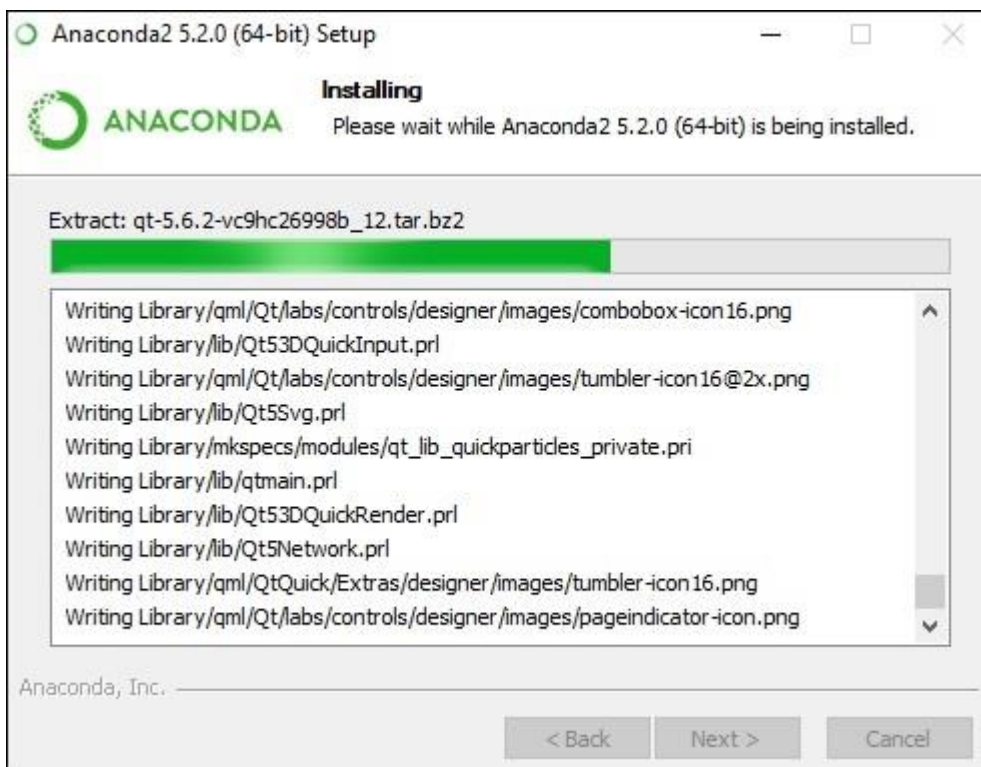
Consider the following steps to install TensorFlow in Windows operating system.

Step 1 – Verify the python version being installed.



```
Command Prompt - Python
.\>Python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
```

Step 2 – A user can pick up any mechanism to install TensorFlow in the system. We recommend “pip” and “Anaconda”. Pip is a command used for executing and installing modules in Python.



Step 3 – Execute the following command to initialize the installation of TensorFlow –

conda create --name tensorflow python = 3.5

```
Command Prompt - conda create --name tensorflow python=3.5
vc-14                | h0510ff6_3      | 3 KB
wincertstore-0.2     | py35hfbbdb8_0   | 13 KB
wheel-0.31.1         | py35_0          | 81 KB
certifi-2018.4.16    | py35_0          | 143 KB
python-3.5.5         | h0c2934d_2      | 18.2 MB
-----
Total:                |                  | 20.8 MB

The following NEW packages will be INSTALLED:

certifi: 2018.4.16-py35_0
pip: 10.0.1-py35_0
python: 3.5.5-h0c2934d_2
setuptools: 39.2.0-py35_0
vc: 14-h0510ff6_3
vs2015_runtime: 14.0.25123-3
wheel: 0.31.1-py35_0
wincertstore: 0.2-py35hfbbdb8_0

Proceed ([y/n])? y

Downloading and Extracting Packages
pip-10.0.1 | 1.8 MB | ##### | 100%
setuptools-39.2.0 | 593 KB | ##### | 100%
vc-14 | 3 KB | ##### | 100%
wincertstore-0.2 | 13 KB | ##### | 100%
wheel-0.31.1 | 81 KB | ##### | 100%
certifi-2018.4.16 | 143 KB | ##### | 100%
python-3.5.5 | 18.2 MB | ##### | 70%
```

The code for first program implementation is mentioned below –

```
>>> activate tensorflow
>>> python (activating python shell)
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, Tensorflow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=d3uv23jvp4w>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 . Pg-No.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-41

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : V- Tensorflow,Keras

Date of Lecture:

Topic of Lecture: Install Tensorflow

Introduction :

- Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK).
- Theano is a python library used for fast numerical computation tasks.
- TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models.
- TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Any kind of OS (Windows, Linux or Mac)
- Python version 3.5 or higher.

Detailed content of the Lecture:

Python Keras is python based neural network library so python must be installed on your machine.

If python is properly installed on your machine, then open your terminal and type python, you could see the response similar as specified below, Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license" for more information.

>>> As of now the latest version is '3.7.2'. If Python is not installed, then visit the official python link - <https://www.python.org/> and download the latest version based on your OS and install it immediately on your system.

pip install TensorFlow

Once we execute keras, we could see the configuration file is located at your home directory inside and go to .keras/keras.json.

keras.json

```
{
  "image_data_format": "channels_last",
  "epsilon": 1e-07, "floatx": "float32", "backend": "tensorflow"
}
```

Here,

- **image_data_format** represent the data format.
- **epsilon** represents numeric constant. It is used to avoid **DivideByZero** error.
- **floatx** represent the default data type **float32**. You can also change it to **float16** or **float64** using **set_floatx()** method.
- **image_data_format** represent the data format.

Suppose, if the file is not created then move to the location and create using the below steps –

```
> cd home
> mkdir .keras
> vi keras.json
```

backend module

backend module is used for keras backend operations. By default, keras runs on top of TensorFlow backend. If you want, you can switch to other backends like Theano or CNTK. Default backend configuration is defined inside your root directory under .keras/keras.json file.

Keras *backend* module can be imported using below code

```
>>> from keras import backend as k
```

If we are using default backend *TensorFlow*, then the below function returns *TensorFlow* based information as specified below –

```
>>> k.backend()
'tensorflow'
>>> k.epsilon()
1e-07
>>> model.summary() Model: "sequential_10"
```

Layer (type)	Output Shape	Param
dense_13 (Dense)	(None, 32)	288
dense_14 (Dense)	(None, 64)	2112
dense_15 (Dense)	(None, 8)	520

Total params: 2,920
Trainable params: 2,920
Non-trainable params: 0

```
>>>
```

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/deep_learning_and_neural_networks_python_keras/index.asp

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 . Pg-No.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-42

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : V- Tensorflow,Keras Date of Lecture:

Topic of Lecture: Keras

Introduction :

- A Keras layer requires shape of the input (input_shape) to understand the structure of the input data, *initializer* to set the weight for each input and finally activators to transform the output to make it non-linear.
- In between, constraints restricts and specify the range in which the weight of input data to be generated and regularizer will try to optimize the layer (and the model) by dynamically applying the penalties on the weights during optimization process.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Any kind of OS (Windows, Linux or Mac)
- Python version 3.5 or higher.

Detailed content of the Lecture:

Keras modules contains pre-defined classes, functions and variables which are useful for deep learning algorithm.

- Initializers: Provides a list of initializers function. We can learn it in details in Keras layer chapter. during model creation phase of machine learning.
- Regularizers: Provides a list of regularizers function. We can learn it in details in Keras Layers chapter.
- Constraints: Provides a list of constraints function. We can learn it in details in Keras Layers chapter.

To summarise, Keras layer requires below minimum details to create a complete layer. Shape of the input data

- Number of neurons / units in the layer
- Initializers
- Regularizers
- Constraints
- Activations

```
from keras.models import Sequential from keras.layers import Activation, Dense from keras import initializers from keras import regularizers from keras import constraints
```

```
model = Sequential()
```

```
model.add(Dense(32, input_shape=(16,), kernel_initializer='he_uniform', kernel_regularizer=None, kernel_constraint='MaxNorm', activation='relu')) model.add(Dense(16, activation='relu')) model.add(Dense(8))
```

```

>>> from keras.models import Sequential
>>> from keras.layers import Activation, Dense
>>> model = Sequential()
>>> layer_1 = Dense(16, input_shape=(8,))
>>> model.add(layer_1)
>>> layer_1.input_shape (None, 8)
>>> layer_1.output_shape (None, 16)
>>>

```

VarianceScaling

It finds the *stddev* value for normal distribution using below formula and then find the weights using normal distribution,

$stddev = \sqrt{\text{scale} / n}$

where **n** represent,

- number of input units for mode = fan_in
- number of out units for mode = fan_out
- average number of input and output units for mode = fan_avg

Similarly, it finds the *limit* for uniform distribution using below formula and then find the weights using uniform distribution,

$limit = \sqrt{3 * \text{scale} / n}$

placeholder

It is used instantiates a placeholder tensor. Simple placeholder to hold 3-D shape is shown below –

```

>>> data = k.placeholder(shape = (1,3,3))
>>> data
<tf.Tensor 'Placeholder_9:0' shape = (1, 3, 3) dtype = float32>

```

If you use `int_shape()`, it will show the shape.

```

>>> k.int_shape(data) (1, 3, 3)

```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=O2Nk3JFZE58>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 .Pg.No.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-43

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : V- Tensorflow, Keras

Date of Lecture:

Topic of Lecture: Introduction to Keras

Introduction :

- Keras is an open source deep learning framework for python. It has been developed by an artificial intelligence researcher at Google named **Francois Chollet**
- Leading organizations like Google, Square, Netflix, Huawei and Uber are currently using Keras. This tutorial walks through the installation of Keras, basics of deep learning, Keras models, Keras layers, Keras modules and finally conclude with some real-time applications
- In between, constraints restricts and specify the range in which the weight of input data to be generated and regularizer will try to optimize the layer (and the model) by dynamically applying the penalties on the weights during optimization process

Prerequisite knowledge for Complete understanding and learning of Topic:

- Any kind of OS (Windows, Linux or Mac)
- Python version 3.5 or higher.

Detailed content of the Lecture:

Keras modules contains pre-defined classes, functions and variables which are useful for deep learning algorithm.

- **Initializers:** Provides a list of initializers function. We can learn it in details in Keras layer chapter. during model creation phase of machine learning.
- **Regularizers:** Provides a list of regularizers function. We can learn it in details in Keras Layers chapter.
- **Constraints:** Provides a list of constraints function. We can learn it in details in Keras Layers chapter.
To summarise, Keras layer requires below minimum details to create a complete layer. Shape of the input data
 - Number of neurons / units in the layer
 - Initializers
 - Regularizers
 - Constraints
 - Activations

Theano is an open source deep learning library that allows you to evaluate multi-dimensional arrays effectively. We can easily install using the below command –

```
pip install theano
```

By default, keras uses TensorFlow backend. If you want to change backend configuration from

TensorFlow to Theano, just change the backend = theano in keras.json file. It is described below –

keras.json

```
{
  "image_data_format": "channels_last",
  "epsilon": 1e-07,
  "floatx": "float32",
  "backend": "theano"
}
```

Now save your file, restart your terminal and start keras, your backend will be changed.

```
>>> import keras as k
using theano backend.
```

- number of input units for mode = fan_in
- number of out units for mode = fan_out
- average number of input and output units for mode = fan_avg

Similarly, it finds the *limit* for uniform distribution using below formula and then find the weights using uniform distribution,

```
limit = sqrt(3 * scale / n)
```

placeholder

It is used instantiates a placeholder tensor. Simple placeholder to hold 3-D shape is shown below –

```
>>> data = k.placeholder(shape = (1,3,3))
>>> data
<tf.Tensor 'Placeholder_9:0' shape = (1, 3, 3) dtype = float32>
```

If you use int_shape(), it will show the shape.

```
>>> k.int_shape(data) (1, 3, 3)
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=O2Nk3JFZE58>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 .Pg.No.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-44

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit : V- Tensorflow, Keras

Date of Lecture:

Topic of Lecture: Keras Layers

Introduction :

- Layers are the basic building blocks of neural networks in Keras. A layer consists of a tensor-in tensor-out computation function (the layer's `call` method) and some state, held in TensorFlow variables
- In between, constraints restricts and specify the range in which the weight of input data to be generated and regularizer will try to optimize the layer (and the model) by dynamically applying the penalties on the weights during optimization process.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Any kind of OS (Windows, Linux or Mac)
- Python version 3.5 or higher.

Detailed content of the Lecture:

Keras layers contains pre-defined classes, functions and variables which are useful for deep learning algorithm.

Keras allows to create our own customized layer. Once a new layer is created, it can be used in any model without any restriction. Let us learn how to create new layer in this chapter.

Keras provides a base **layer** class, `Layer` which can sub-classed to create our own customized layer. Let us create a simple layer which will find weight based on normal distribution and then do the basic computation of finding the summation of the product of input and its weight during training.

```
>>> model = Sequential()
>>> layer_1 = Dense(16, input_shape=(8,))
>>> model.add(layer_1)
>>> layer_1.input_shape (None, 8)
>>> layer_1.output_shape (None, 16)
>>>
```

VarianceScaling

It finds the *stddev* value for normal distribution using below formula and then find the weights using normal distribution,

$stddev = \sqrt{scale / n}$

where **n** represent,

- number of input units for mode = `fan_in`
- number of out units for mode = `fan_out`

- average number of input and output units for mode = fan_avg

Similarly, it finds the *limit* for uniform distribution using below formula and then find the weights using uniform distribution,

limit = sqrt(3 * scale / n)

- **Line 1** defines **compute_output_shape** method with one argument **input_shape**
- **Line 2** computes the output shape using shape of input data and output dimension set while initializing the layer.

Implementing the **build**, **call** and **compute_output_shape** completes the creating a customized layer. The final and complete code is as follows

```
from keras import backend as K from keras.layers import Layer
class MyCustomLayer(Layer):
    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        super(MyCustomLayer, self).__init__(**kwargs)
    def build(self, input_shape): self.kernel =
        self.add_weight(name = 'kernel',
            shape = (input_shape[1], self.output_dim),
            initializer = 'normal', trainable = True)
        super(MyCustomLayer, self).build(input_shape) #
        Be sure to call this at the end
    def call(self, input_data): return K.dot(input_data, self.kernel)
    def compute_output_shape(self, input_shape): return (input_shape[0], self.output_dim)
```

Using our customized layer

Let us create a simple model using our customized layer as specified below –

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(MyCustomLayer(32, input_shape = (16,)))
model.add(Dense(8, activation = 'softmax')) model.summary()
```

Here,

- Our **MyCustomLayer** is added to the model using 32 units and **(16,)** as input shape

Running the application will print the model summary as below –

```
Model: "sequential_1"
Layer (type) Output Shape Param
#=====
my_custom_layer_1 (MyCustomL (None, 32) 512
dense_1 (Dense) (None, 8) 264
=====
Total params: 776
Trainable params: 776
Non-trainable params: 0
```

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=O2Nk3JFZE58>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 .Pg.No.

Course Faculty

Verified by HOD



MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu



LECTURE HANDOUTS

L-45

CSE

I/II

Course Name with Code : 21GES08&Python Programming

Course Faculty : S.Suvitha

Unit :V- Tensorflow,Keras Date of Lecture:

Topic of Lecture: Modules

Introduction :

- Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK).
- Theano is a python library used for fast numerical computation tasks.
- TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models.
- TensorFlow is very flexible and the primary benefit is distributed computing.
- CNTK is deep learning framework developed by Microsoft.
- It uses libraries such as Python, C#, C++ or standalone machine learning toolkits.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Any kind of OS (Windows, Linux or Mac)
- Python version 3.5 or higher.

Detailed content of the Lecture:

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features: Consistent, simple and extensible API.

- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.
- `shape=(16), kernel_initializer='he_uniform', kernel_regularizer=None, kernel_constraint='MaxNorm', activation='relu')) model.add(Dense(16, activation='relu')) model.add(Dense(8))`

Arguments

- **line_length**: Total length of printed lines (e.g. set this to adapt the display to different terminal window sizes).
- **positions**: Relative or absolute positions of log elements in each line. If not provided, defaults to `[.33, .55, .67, 1.]`.
- **print_fn**: Print function to use. Defaults to `print`. It will be called on each line of the summary. You can set it to a custom function in order to capture the string summary.
- **expand_nested**: Whether to expand the nested models. If not provided, defaults to `False`.
- **show_trainable**: Whether to show if a layer is trainable. If not provided, defaults to `False`.

formula and then find the weights using uniform distribution,

`limit = sqrt(3 * scale / n)`

Retrieves a layer based on either its name (unique) or index.

If **name** and **index** are both provided, **index** will take precedence. Indices are based on order of horizontal graph traversal (bottom-up).

Arguments

- **name**: String, name of layer.
- **index**: Integer, index of layer.

Video Content / Details of website for further learning (if any):

<https://www.youtube.com/watch?v=O2Nk3JFZE58>

Important Books/Journals for further learning including the page nos.:

Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Publishers, 2016 .Pg.No.

Course Faculty

Verified by HOD