P P SAVANI UNIVERSITY

School of Engineering

PROGRAMMING WITH PYTHON

Barkha Wadhvani
Assistant Professor
P.P. Savani University
School Of Engineering

P P SAVANI UNIVERSITY

School of Engineering

REFERENCE BOOKS

Title	Author/s	Publication
Python Programming: A modular approach	Sheetal Taneja, Naveen Kumar	Pearson
Think Python: How to Think Like a Computer Scientist	Allen Downey	Green Tea Press
Python Cookbook	David Ascher, Alex Martelli Oreilly	O Reilly Media

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION

Python is

- Interpreted,
- Object oriented ,
- Dynamic semantics,
- High Level General purpose programming Language.

3

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION

- Python is general purpose programming language that used to develop ,
  - Web Pages
  - Machine Learning based Application
  - Artificial Intelligent based Application

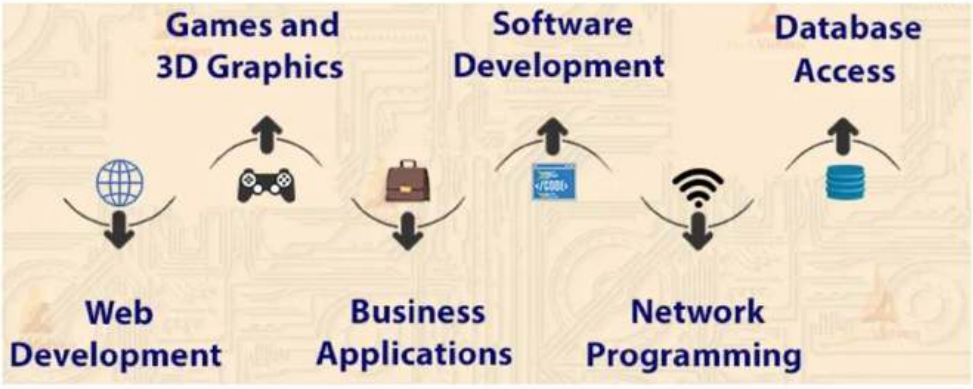
  etc.

4

P P SAVANI UNIVERSITY

School of Engineering

APPLICATION OF PYTHON



5

P P SAVANI UNIVERSITY

School of Engineering

APPLICATIONS OF PYTHON

❑ **Web Development:**

Python is used for **web development and internet applications**. Web framework like **Django and Flask** are one of the most popular frameworks. They allow you to write server-side code in Python language.

❑ **Scientific and Computational Applications**:

The **higher speeds, productivity and availability of tools**, such as Scientific Python and Numeric Python, have resulted in Python becoming an integral part of applications involved in computation and processing of scientific data. 3D modeling software, such as FreeCAD, and finite element method software, such as Abaqus, are coded in Python.

6

P P SAVANI UNIVERSITY

School of Engineering

APPLICATIONS OF PYTHON

- **ERP Progress:** Python is being applied for developing business software which **solves enterprise-level issues.** Successful ERP like Odoo and Tryton is enduring small and important businesses in managing their entire administration and stock index.

- **Game Development:** Programmers can develop games applying python even though most favoured structure for game development is Unity, python does have **PyGame, PyKyra structures for game-development** with Python. Users can get a mixture of 3D-rendering libraries to generate 3D games.

7

P P SAVANI UNIVERSITY

School of Engineering

APPLICATIONS OF PYTHON

- **GUI-Based Desktop Applications:**

  Python has simple syntax, modular architecture, rich text processing tools and the ability to work on multiple operating systems.

  There are various **GUI toolkits like wxPython, PyQt or PyGtk** available which help developers create highly functional Graphical User Interface (GUI).

- **Image Processing and Graphic Design Applications:**

  Python has been used to **make 2D imaging software such as Inkscape, GIMP, Paint Shop Pro and Scribus.** Further, 3D animation packages, like Blender, 3ds Max, Cinema 4D, Houdini, Lightwave and Maya, also use Python in variable proportions.

8

P P SAVANI UNIVERSITY

School of Engineering

APPLICATIONS OF PYTHON

- **Science and Numeric Purposes:** Python is widely used by most of the **Data Scientists** because of its library compilation which is devised for **statistical and numerical analysis**
  - **SciPy** : it is a collection of packages for mathematics, science, and engineering.
  - **Pandas** : A package extensively used for data analysis and modelling
  - **Ipython** : A powerful shell designed for simple editing and recording of work concourses and supports visualization and parallel computing.
  - **NumPy** : helps in dealing with complex numerical calculations.

9

P P SAVANI UNIVERSITY

School of Engineering

WHERE PYTHON STANDS?

10

P P SAVANI UNIVERSITY

School of Engineering

HISTORY

- Python was developed by **Guido Van Rossum in late 1980s.**
- The **first version** of Python was released in **the year 1991**.
- In December of 1989, van Rossum started the implementation of Python and finally published the code of version 0.9.0 in 1991
- Van Rossum was inspired by the ABC programming language that led to the development of Python.
- The name python was adopted from his favourite BBC TV show **"Monty Python's Flying Circus"**.

12

P P SAVANI UNIVERSITY

School of Engineering

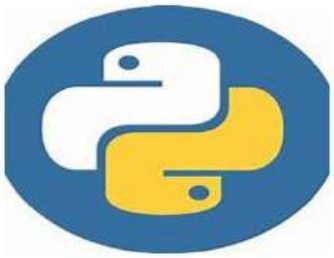
HISTORY

13

P P SAVANI UNIVERSITY

School of Engineering

PYTHON LOGO

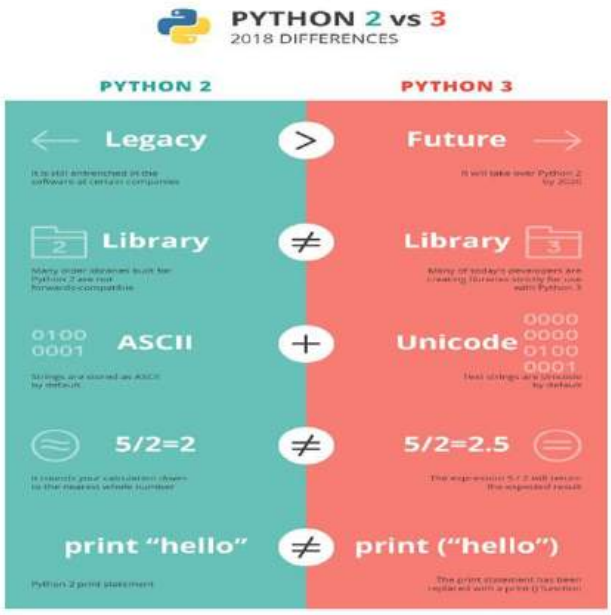


- Python Logo contains two cartoon snakes lying side by side facing up and down.

14

P P SAVANI UNIVERSITY

School of Engineering



PYTHON 2.X PYTHON 3.X

```
>>> print "Hello World!"
Hello World!
>>> print 3/2
1
>>> variable = 123456789
>>> print (type(variable))
<type 'int'>
```

```
>>> print ("Hello World!")
Hello World!
>>> print (3/2)
1.5
>>> variable = 123456789
>>> print (type(variable))
<class 'int'>
```

15

P P SAVANI UNIVERSITY

School of Engineering

ASCII VERSUS UNICODE

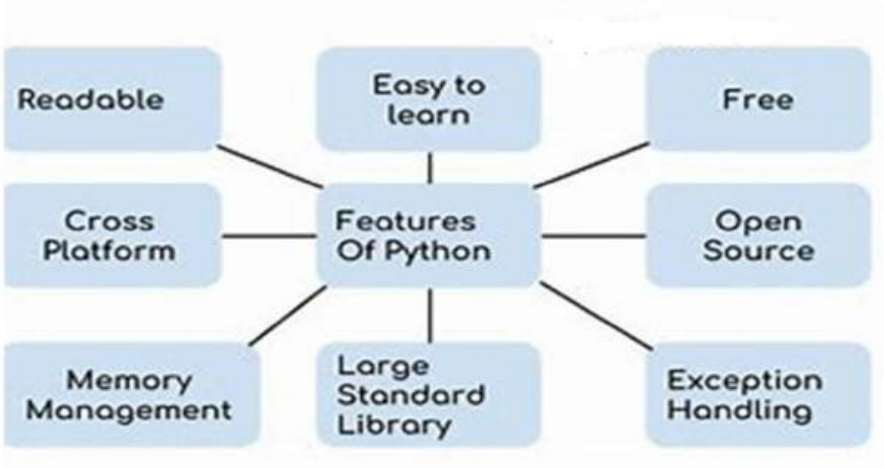
ASCII	UNICODE
A character encoding standard for electronic communication	A computing industry standard for consistent encoding, representation, and handling of text expressed in most of the world's writing systems
Stands for American Standard Code for Information Interchange	Stands for Universal Character Set
Supports 128 characters	Supports a wide range of characters
Uses 7 bits to represent a character	Uses 8bit, 16bit or 32bit depending on the encoding type
Requires less space	Requires more speace

16

P P SAVANI UNIVERSITY

School of Engineering

FEATURES OF PYTHON



17

P P SAVANI UNIVERSITY

School of Engineering

FEATURES OF PYTHON

- **Simple**
  - Python is a simple and minimalistic language in nature.
  - Reading a **good** python program should be like reading English.
  - Its Pseudo-code nature allows one to concentrate on the problem rather than the language.
- **Easy to Learn**
- **Free & Open source**
  - Freely distributed and Open source.
  - Maintained by the Python community.
- **High Level Language –memory management**

18

P P SAVANI UNIVERSITY

School of Engineering

FEATURES OF PYTHON

- **Interpreted**
  - You run the program straight from the source code.
  - You can just copy over your code to another system and it will auto-magically work! *with python platform.
- **Object-Oriented**
  - Simple and additionally supports procedural programming.
- **Extensible – easily import other code**
- **Embeddable –easily place your code in non-python programs**
- **Extensive libraries**
  - (i.e. reg. expressions, doc generation, CGI, ftp, web browsers, ZIP, WAV, cryptography, etc...) (wxPython, Twisted, Python Imaging library)

19

P P SAVANI UNIVERSITY

School of Engineering

PROS OF PYTHON

- Less Coding
- Simple and easy (So that we can focus on solutions)
- Free and Open Source
- Embeddable and Extensible (can embedded with other language)
- Interpreted (Management of memory and CPU is efficient)
- Extensive Libraries
- Object Oriented (replicate real-world problems)
- Portable
- IOT Opportunity

20

P P SAVANI UNIVERSITY

School of Engineering

CONS OF PYTHON

- Speed Limitations (because of Interpreted language)
- Problems with Threading (one sequence of bytecode instructions (thread) to be executed at a time.)
- Not Native To Mobile application development (rarely used for mobile application development)
- Design Restrictions (because of more runtime errors and dynamic semantics)
- High Memory consumption (Memory consumption is high due to the flexibility of datatypes.)
- Underdeveloped Database Access Layers (compared to JDBC and ODBC)
- Difficult to Test

21

P P SAVANI UNIVERSITY

School of Engineering

WHO USED PYTHON?

22

P P SAVANI UNIVERSITY

School of Engineering

PYTHON INTERFACES

- IDLE (Integrated Development and Learning Environment)
- Python Shell
- PyCharm
- Google Collaboratory
- Visual Studio
- Spyder
- Anaconda
- Jupyter Notebook

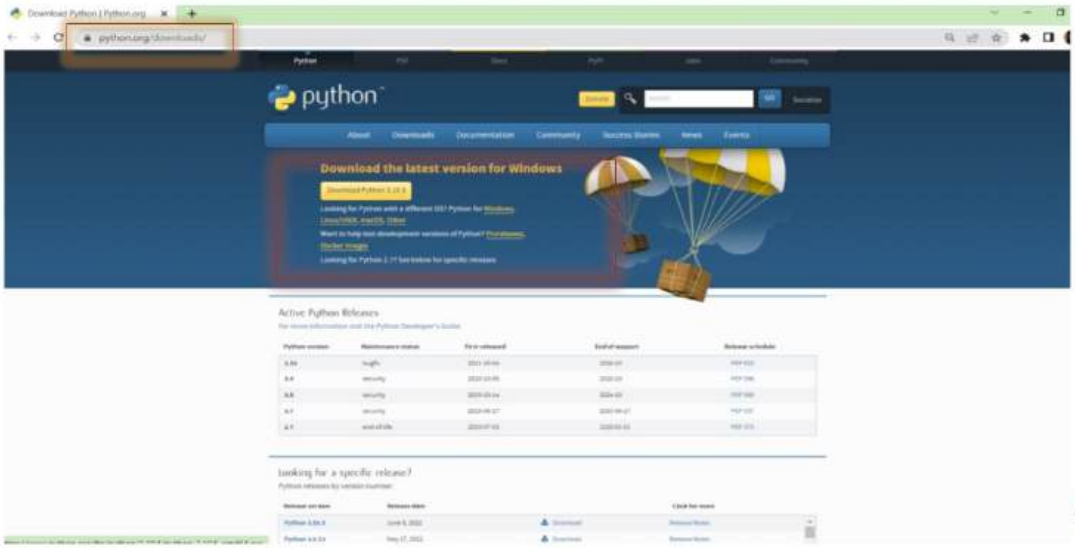
And many more.

23

P P SAVANI UNIVERSITY

School of Engineering

PYTHON DOWNLOAD



24

P P SAVANI UNIVERSITY

School of Engineering

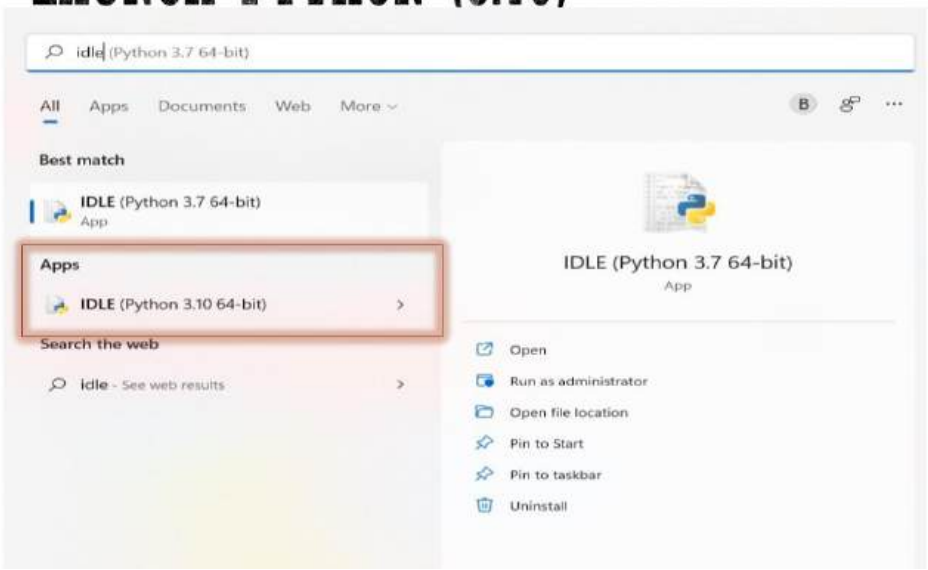
PYTHON SETUP

25

P P SAVANI UNIVERSITY

School of Engineering

LAUNCH PYTHON (3.10)

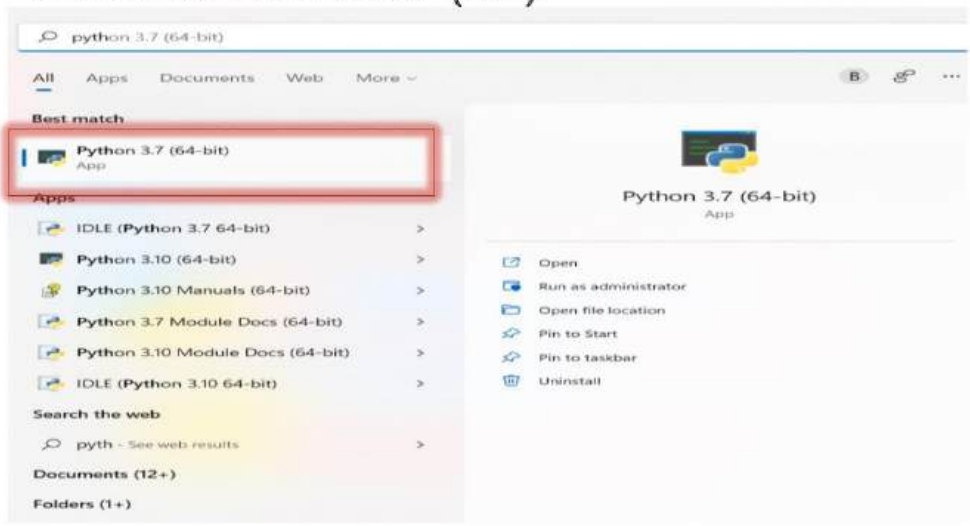


26

P P SAVANI UNIVERSITY

School of Engineering

LAUNCH PYTHON (3.7)

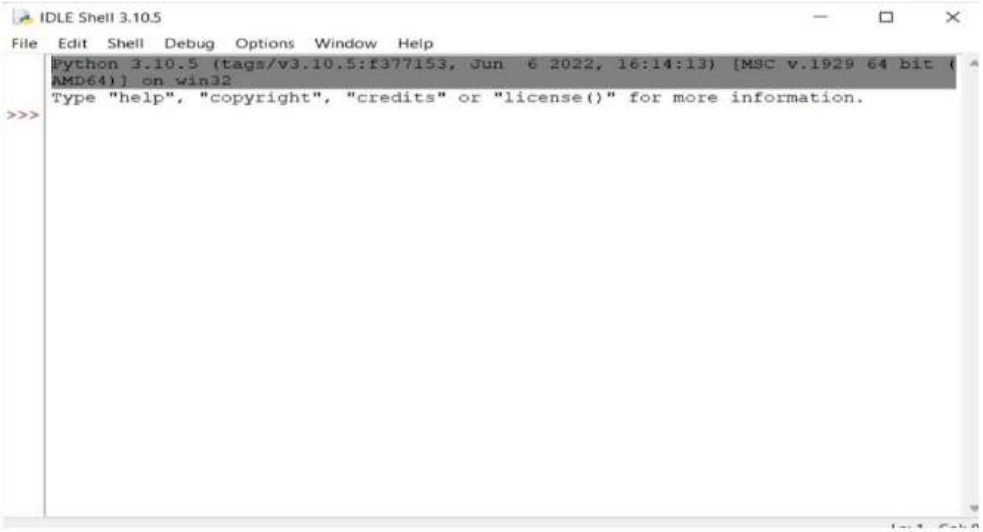


27

P P SAVANI UNIVERSITY

School of Engineering

PYTHON IDLE SHELL



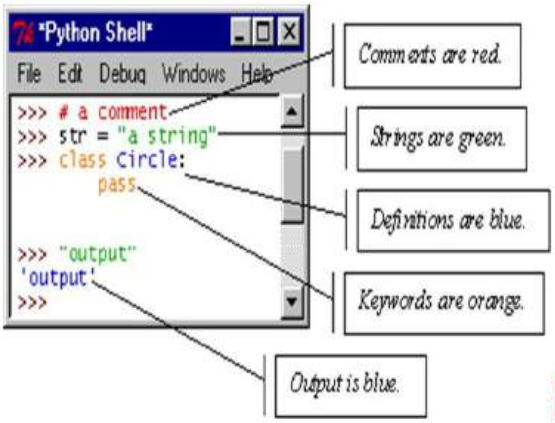
28

P P SAVANI UNIVERSITY

School of Engineering

IDLE DEVELOPMENT ENVIRONMENT

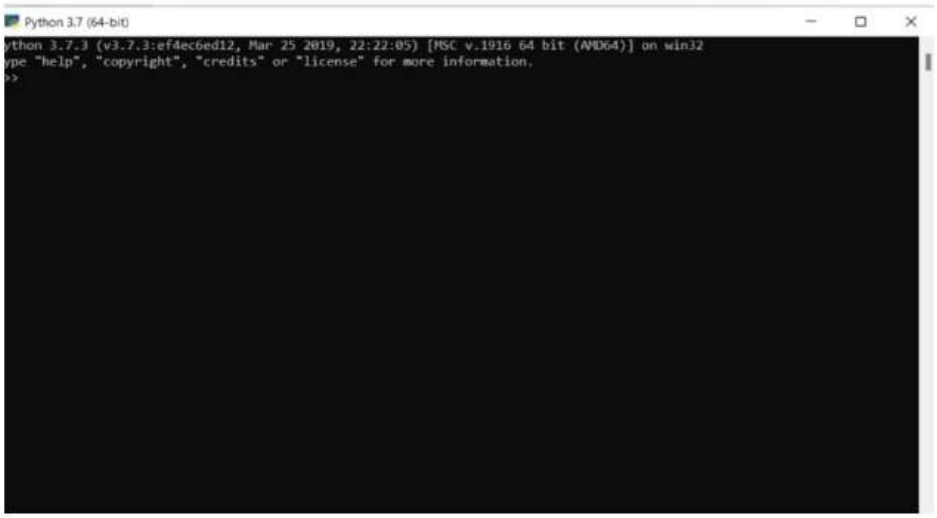
- IDLE helps you program in Python by:
  - ✓color-coding your program code
  - ✓debugging
  - ✓auto-indent
  - ✓interactive shell



P P SAVANI UNIVERSITY

School of Engineering

PYTHON COMMAND PROMPT ENVIRONMENT



30

P P SAVANI UNIVERSITY

School of Engineering

OPEN PYTHON IN COMMAND PROMPT

- Command : **python or py in command prompt**

31

P P SAVANI UNIVERSITY

School of Engineering

PRINT()

Print ()

- It is used to print Message on the Screen, , or other standard output device.
- The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax

print(object(s), sep=separator, end=end, file=file, flush=flush)

32

P P SAVANI UNIVERSITY

School of Engineering

PRINT() PARAMETERS

Parameter	Description
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='separaor'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='end'	Optional. Specify what to print at the end. Default is '\n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

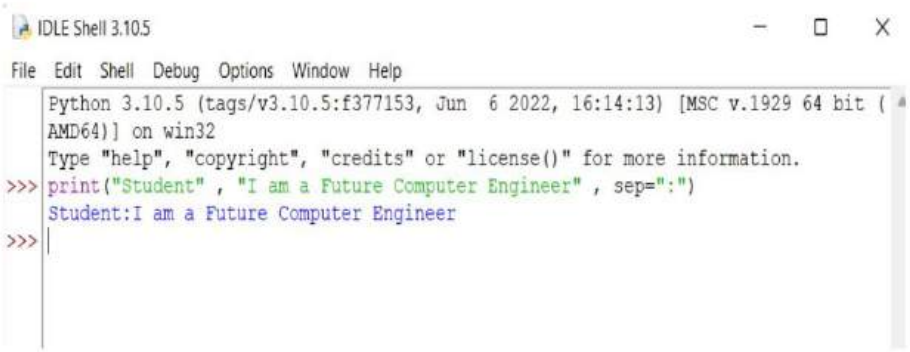
33

P P SAVANI UNIVERSITY

School of Engineering

PRINT() EXAMPLE

print("Student" , "I am a Future Computer Engineer" , sep=":")



34

P P SAVANI UNIVERSITY

School of Engineering

WORKING WITH SCRIPT

- For efficient working we will be using the script mode.
- By writing script , we can write multiple lines code into a file which can be executed later.

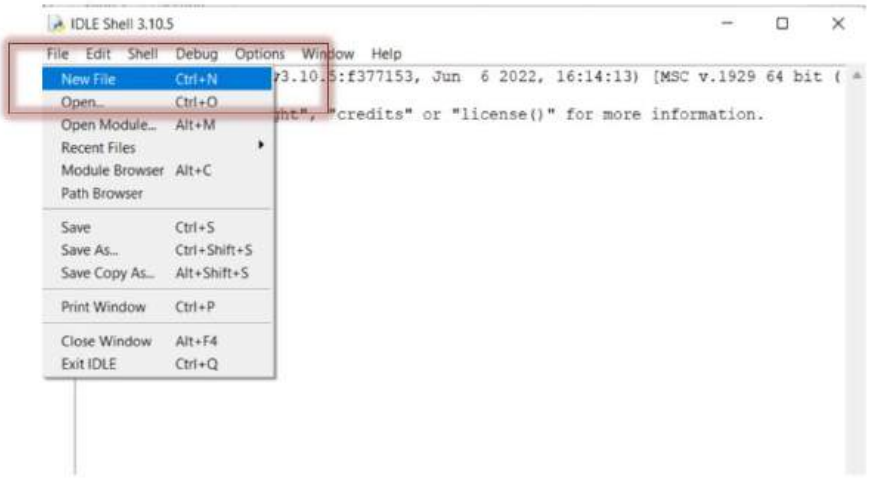
- **How to create Script**
  - ✓Open editor like notepad, create a file named and save it with **.py** extension, which stands for **"Python".**
  - ✓Write your code in file.
  - ✓Open the File in IDLE.
  - ✓Run the File from IDLE.

35

P P SAVANI UNIVERSITY

School of Engineering

CREATE NEW SCRIPT

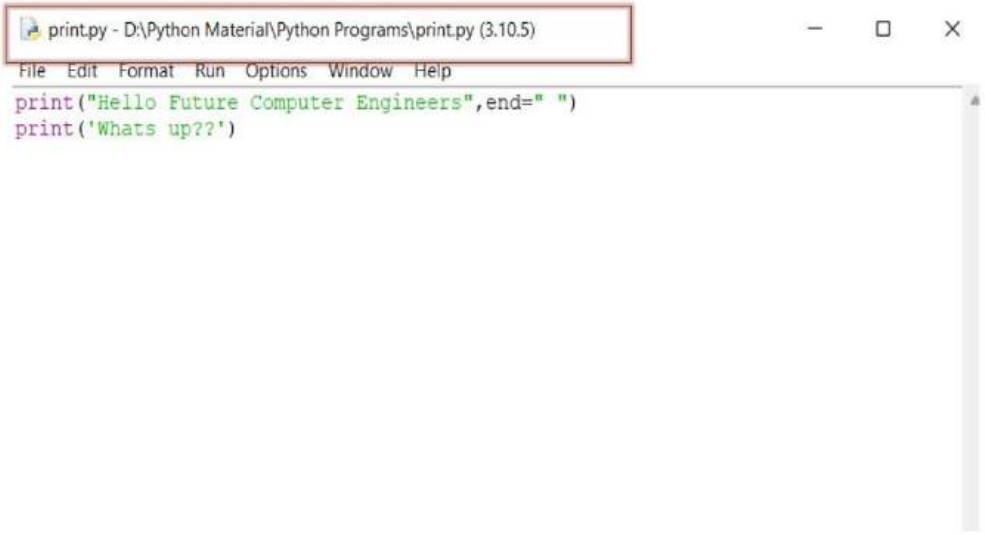


36

P P SAVANI UNIVERSITY

School of Engineering

PRINT DEMO SCRIPT

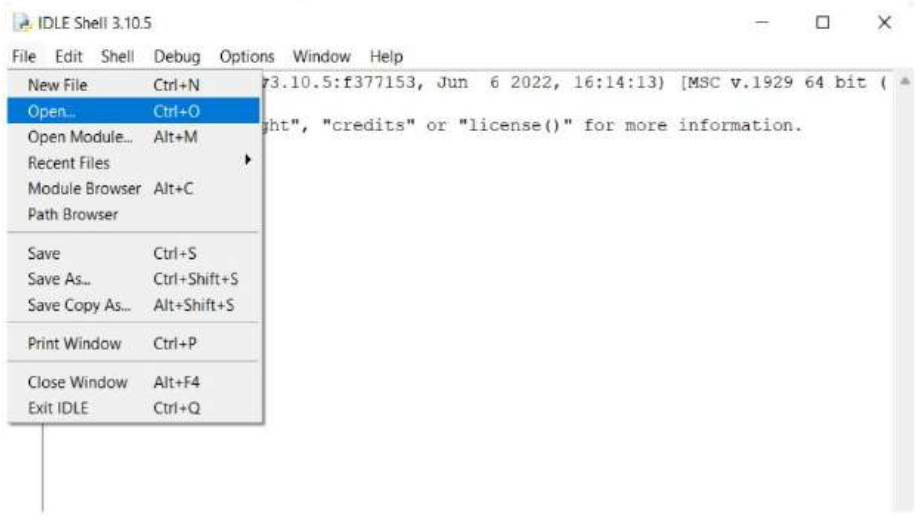


37

P P SAVANI UNIVERSITY

School of Engineering

OPEN THE SCRIPT

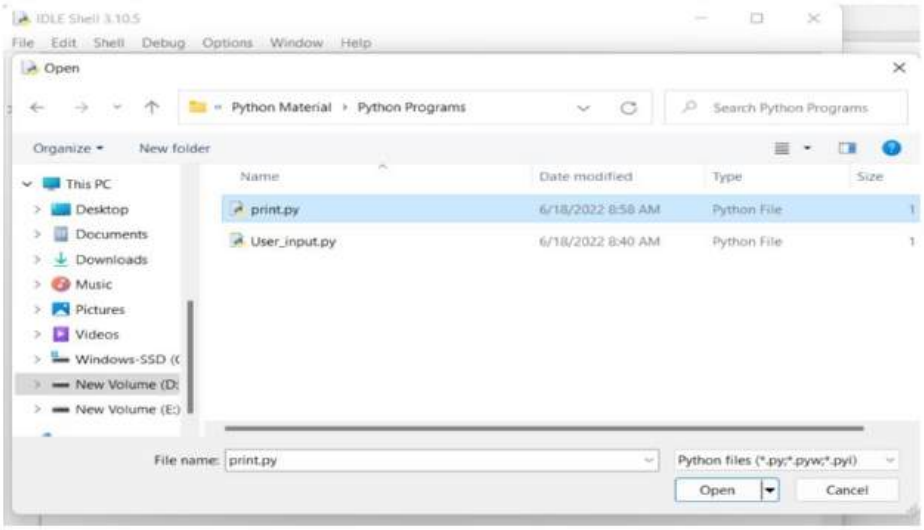


38

P P SAVANI UNIVERSITY

School of Engineering

SELECT THE SCRIPT TO EXECUTE

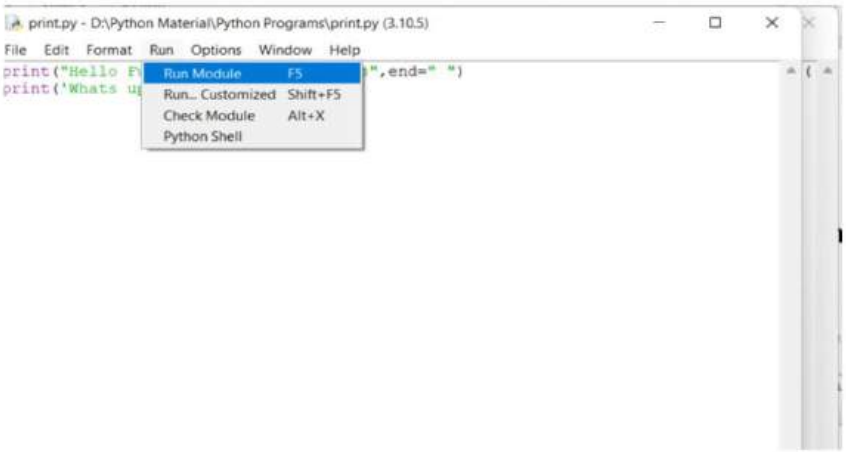


39

P P SAVANI UNIVERSITY

School of Engineering

RUN THE SCRIPT



40

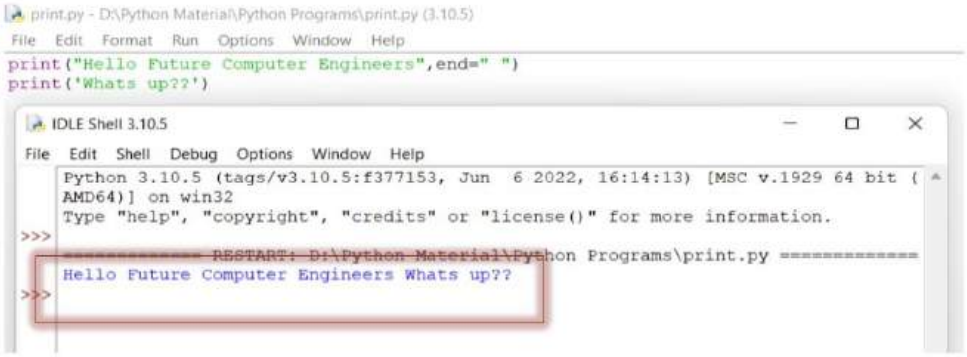
P P SAVANI UNIVERSITY

School of Engineering

SCRIPT EXECUTION OUTPUT

print('Hello Future Computer Engineers' , end=" ")

print('Whats up??')



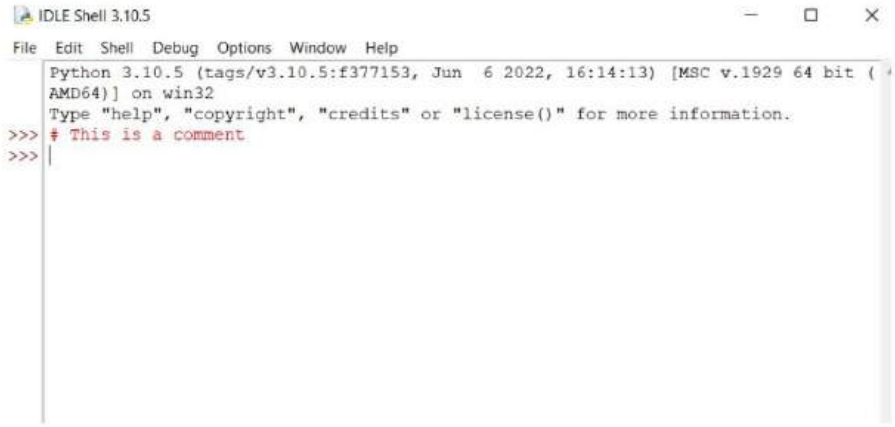
41

P P SAVANI UNIVERSITY

School of Engineering

COMMENT IN PYTHON

- Comment is used for Documentation purpose and can not be executed.
- **# symbol** represents Single and Multi Line Comment.



42

P P SAVANI UNIVERSITY

School of Engineering

VARIABLE

Variable

- Variable is the container of a value. Variable names are case sensitive.
- You can re-declare the variable even after you have declared it once.
- Python is dynamic typed language so no need to declare the variable

Syntax:

variable_name = variable_values

Examples

Sample=5

Sample="Hello Engineers"

Sample

43

P P SAVANI UNIVERSITY

School of Engineering

VARIABLE NAMING

Variable names in Python can be

- ❑ any length
- ❑ can consist of uppercase and lowercase letters (A-Z, a-z),
- ❑digits (0-9),
- ❑and the underscore character (_).

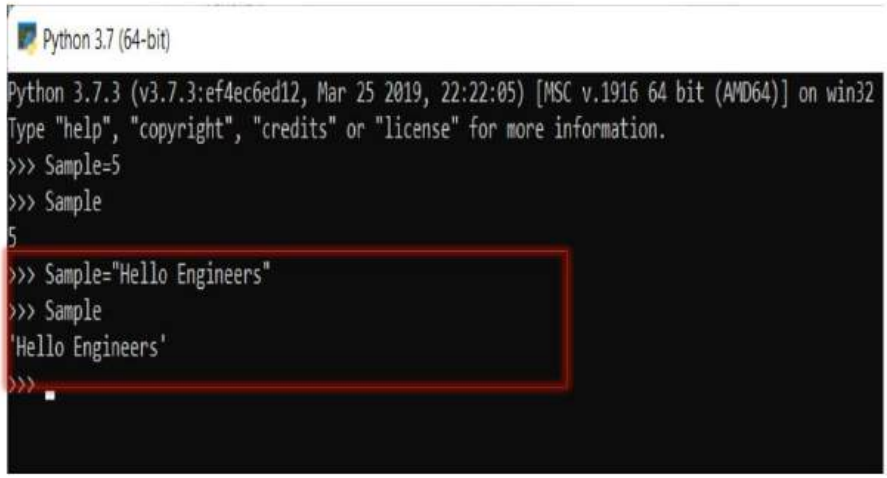
An additional restriction is that, although a variable name can contain digits, the first character of a variable name cannot be a digit.

44

P P SAVANI UNIVERSITY

School of Engineering

VARIABLE CREATION AND DISPLAY



45

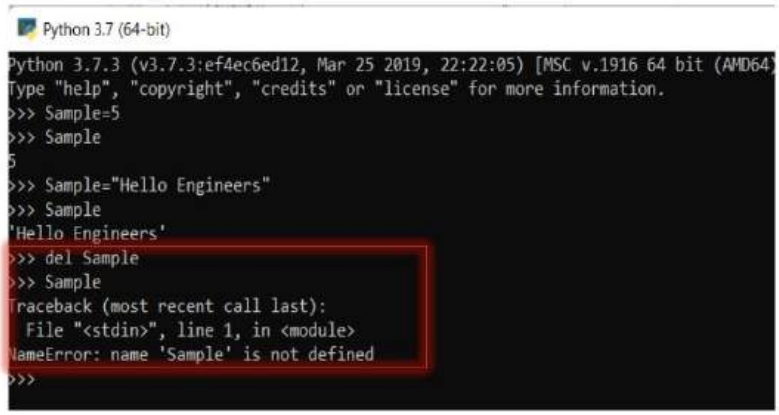
P P SAVANI UNIVERSITY

School of Engineering

DELETE VARIABLE

Syntax:

del Sample



46

P P SAVANI UNIVERSITY

School of Engineering

PRINT VARIABLE

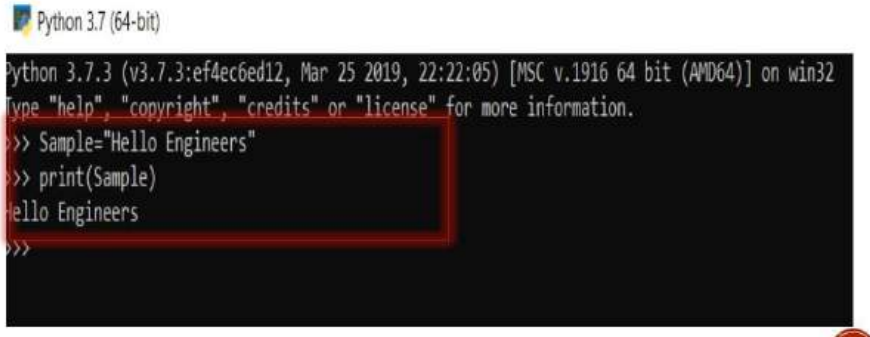
Syntax:

Print(Variable Name)

Example

Print(Sample)

Sample



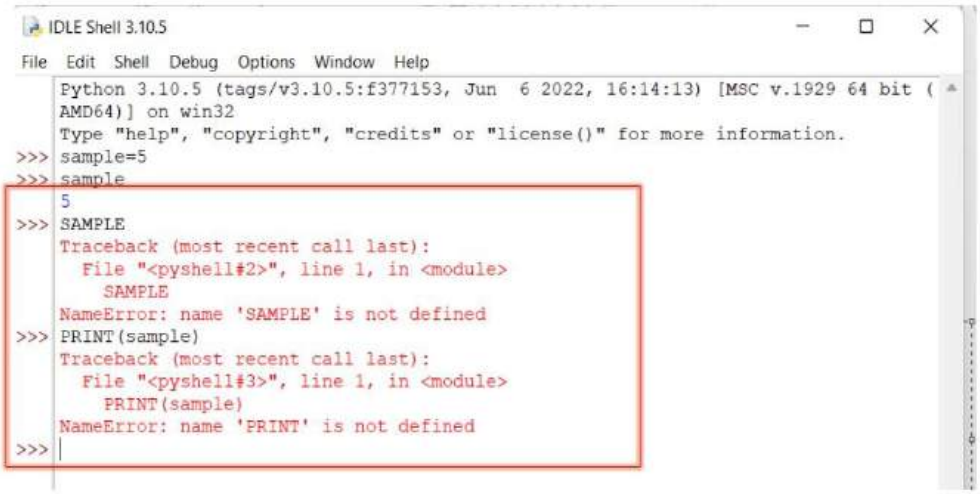
47

P P SAVANI UNIVERSITY

School of Engineering

PYTHON IS CASE SENSITIVE LANGUAGE

It's the differentiation between lower- and uppercase letters.



48

P P SAVANI UNIVERSITY

School of Engineering

WORK WITH VARIABLES

❑ **Assign multiple values to multiple variables in one line**

```
a,b,c=11,22,33
print(a,b,c)
```
→ **Output : 11,22,33**

```
a,b,c = 11,22
```
→ **Output : Not enough values**

```
a,b = 11,22,33
```
→ **Output : Too many values**

```
a,b,_ =11,22,33
print(a,b)
```
→ **Output : 11,22**

49

P P SAVANI UNIVERSITY

School of Engineering

WORK WITH VARIABLES

50

P P SAVANI UNIVERSITY

School of Engineering

WORK WITH VARIABLES

- **Assign a single value to several variables simultaneously**

```
x=y=z=11
print(x,y,z)
```

→ **Output : 11,11,11**

Change value of Y

```
y = 22
print(x,y,z)
```

→ **Output : 11,22,11**

51

P P SAVANI UNIVERSITY

School of Engineering

WORK WITH VARIABLES

52

P P SAVANI UNIVERSITY

School of Engineering

CLEAR SCREEN

```
Import os
os.system('cls')
```

53

P P SAVANI UNIVERSITY

School of Engineering

USER INPUT

For input, User will ask input from User .

Syntax

VariableName=input("Prompt Message")

Example

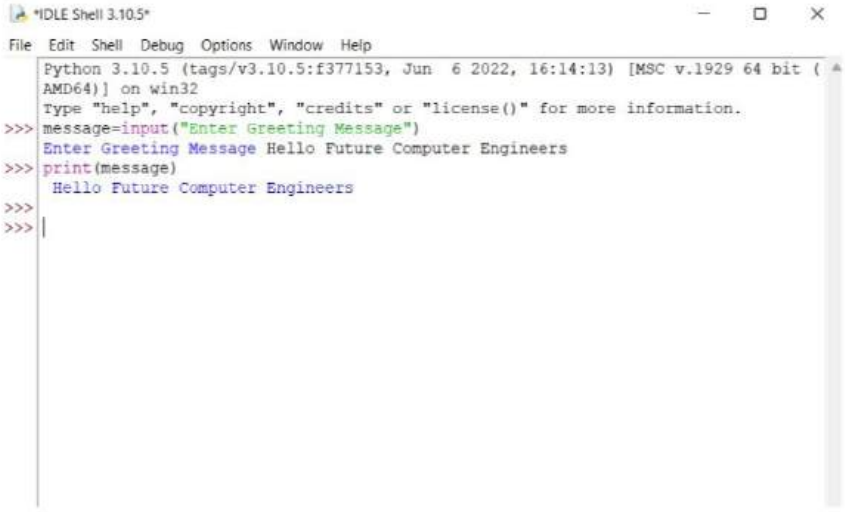
```
Message=input("Enter greeting Message")
Print(Message)
Print("Entereded Message is " + Message)
```

54

P P SAVANI UNIVERSITY

School of Engineering

USER INPUT



55

P P SAVANI UNIVERSITY

School of Engineering

TYPE()

Type() returns the class type of the argument(object) passed as a parameter.

Syntax:

Type(Variable name)

Example

Sample=5

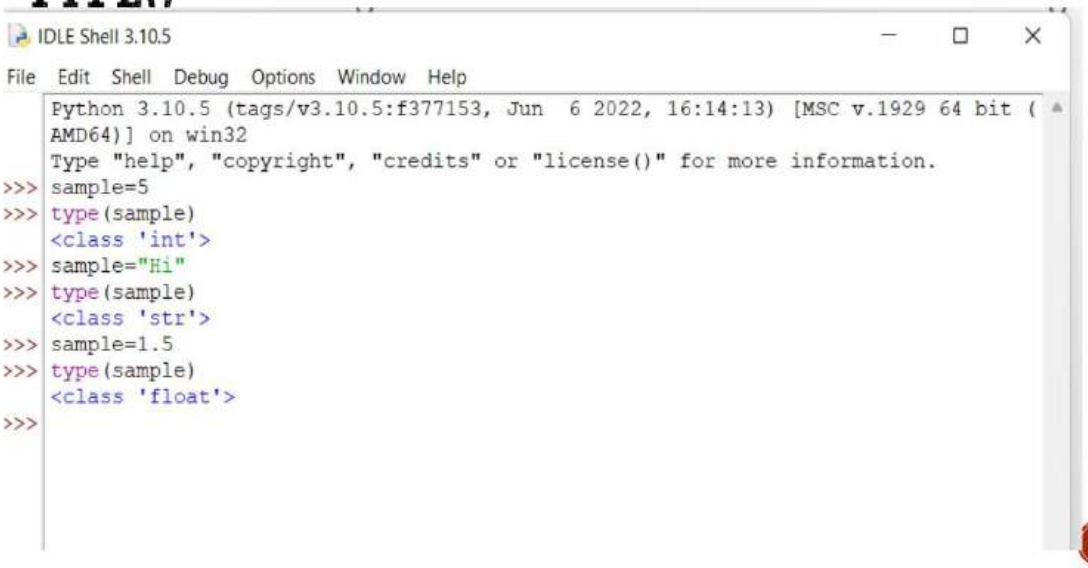
Type(Sample) ——→ **Output : 5**

56

P P SAVANI UNIVERSITY

School of Engineering

TYPE()



57

P P SAVANI UNIVERSITY

School of Engineering

ID()

- id() function returns the "identity" of the object.
- The Id of an object is generally the memory location of the object.

Syntax:

id(Variable name)

Example

```
Sample=5
id(Sample)
```

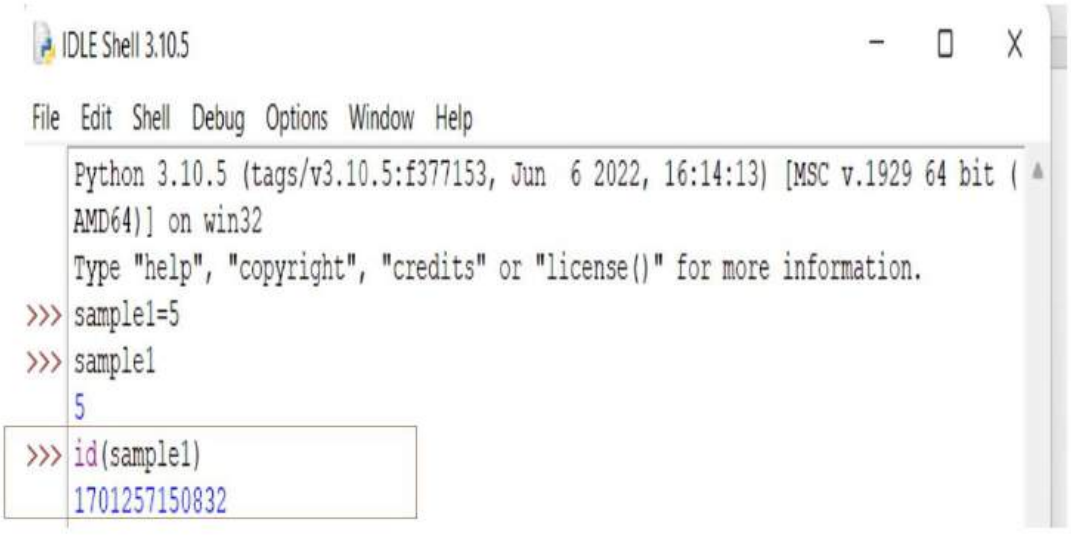
→ **Output : Memory location of value 5**

58

P P SAVANI UNIVERSITY

School of Engineering

ID()



59

P P SAVANI UNIVERSITY

School of Engineering

DATA TYPES

Datatype represents the type of data to be store in a Variable.

Below are Types of Datatype:

- None
- Numeric
- List
- Tuple
- Set
- String
- Range
- Dictionary

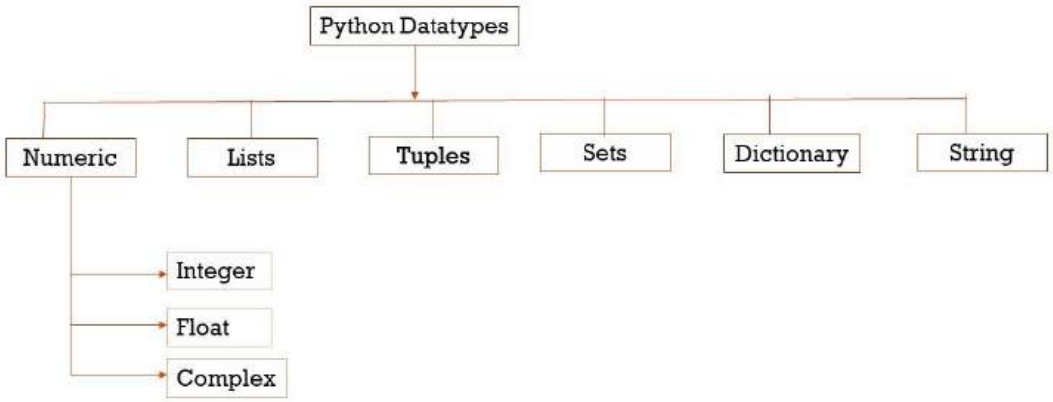
60

P P SAVANI UNIVERSITY

School of Engineering

DATA TYPES

Datatype represents the type of data to be store in a Variable.



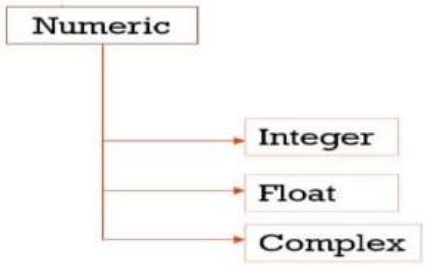
61

P P SAVANI UNIVERSITY

School of Engineering

NUMERIC DATA TYPE

- Numeric datatypes are used to store numerical values in the variables.
- Numeric datatype is **immutable.**

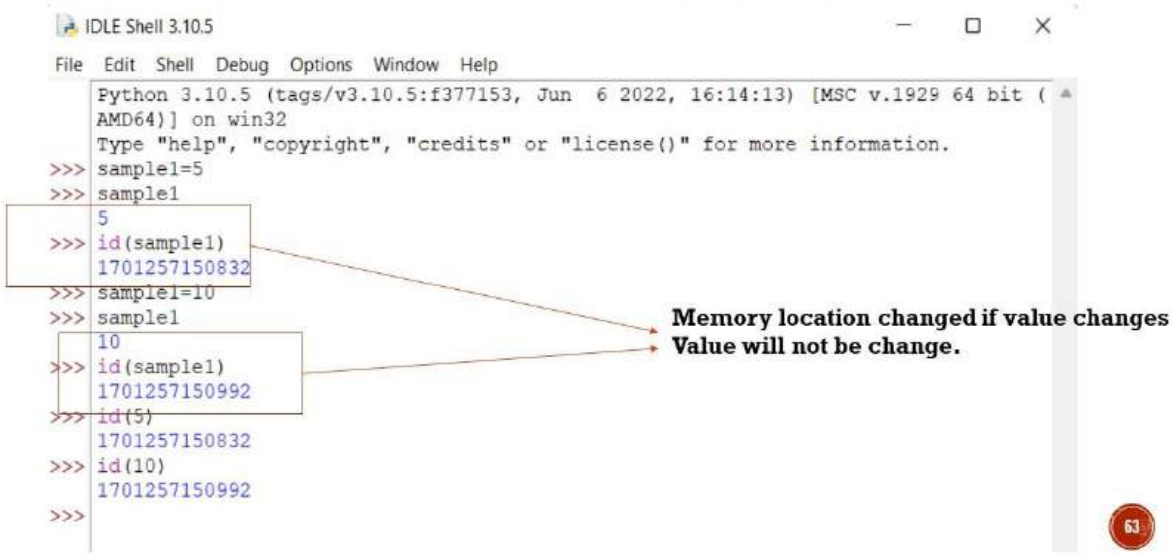


62

P P SAVANI UNIVERSITY

School of Engineering

NUMERIC DATATYPE IS IMMUTABLE



63

P P SAVANI UNIVERSITY

School of Engineering

NUMERIC DATA TYPE

Integer

- It includes **whole numbers, Positive numbers and Negative Numbers.** e.g., 5, 0,-33

Float

- It includes **decimal point numbers**. e.g., 23.987666, 1.5

Complex

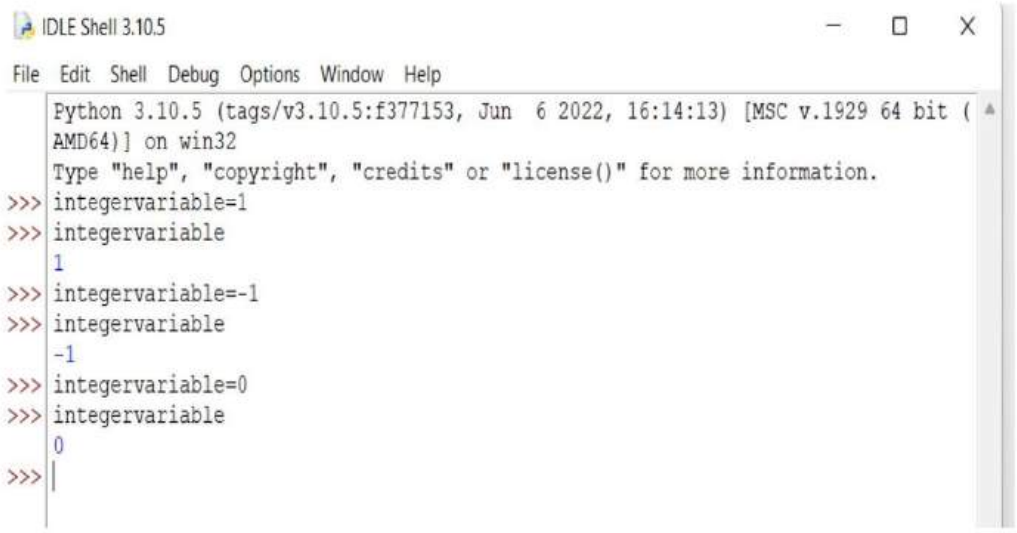
- It is a **combination of real number and imaginary number.**
- E.g., 4+3j

64

P P SAVANI UNIVERSITY

School of Engineering

INTEGER

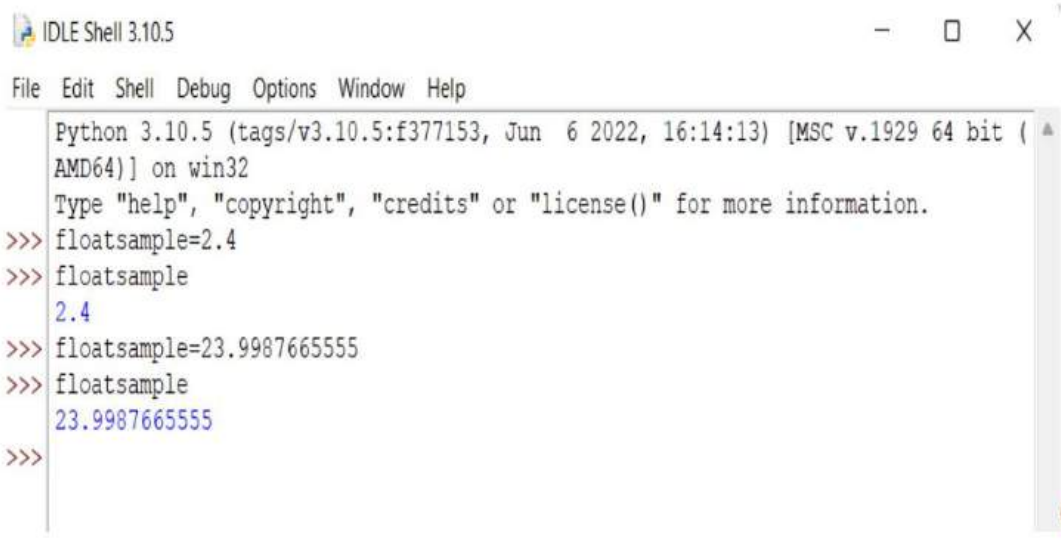


65

P P SAVANI UNIVERSITY

School of Engineering

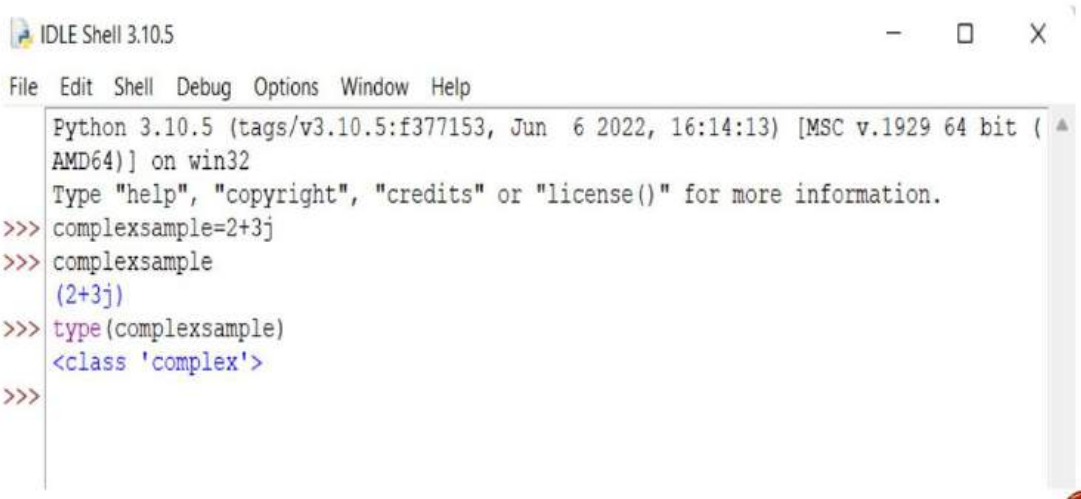
FLOAT



P P SAVANI UNIVERSITY

School of Engineering

COMPLEX



P P SAVANI UNIVERSITY

School of Engineering

LIST

- Lists are used to store multiple items (collection of data) in a single variable.
- List items are ordered, changeable, and allow duplicate values.
- List can **have heterogenous data types in them.**
- List items are indexed, the first item has index [0], the second item has index [1] etc.
- Lists are **mutable**.
- Lists are created using **square brackets**.

68

P P SAVANI UNIVERSITY

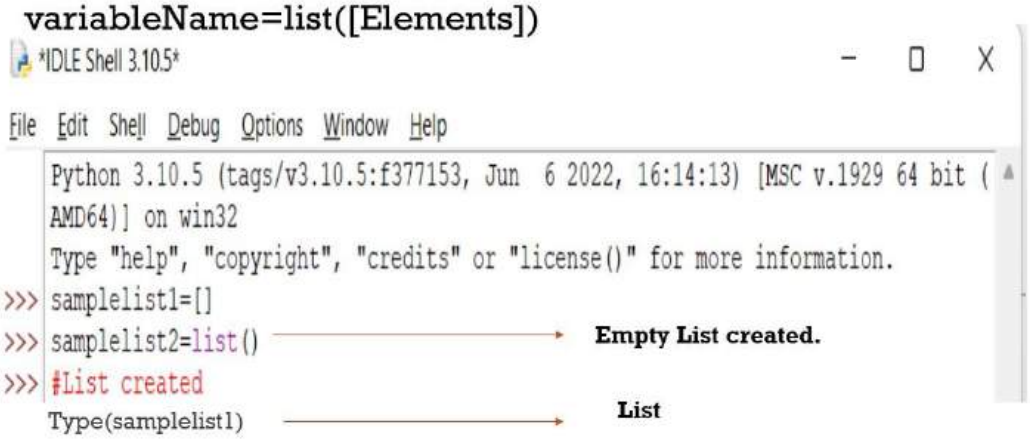
School of Engineering

LIST

Syntax

variableName=[Elements]

variableName=list([Elements])



69

P P SAVANI UNIVERSITY

School of Engineering

LIST

Example

names = [“Yuvraj", “Yath", “Yatri"]

Numbers = [11,22,33,44]

MixedList = [11, ”computer”, 11.5]

70

P P SAVANI UNIVERSITY

School of Engineering

LIST DEMO

List created with different type of elements.

71

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING LIST ELEMENTS BY USING INDEX

- List Index starts from zero.
- -1 represents last Element of List.

Syntax:

listName[index]

72

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING LIST ELEMENTS BY USING INDEX

Example

samplelist = [11, "Yatri","Yuvraj", "Yatharth",15.5]

- **Access first Element of a List**

Print(samplelist[0])

- **Access Last Element of a List (negative sign index used to access the list from backwards)**

Print(samplelist[-1])

73

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING LIST ELEMENTS BY SLICING

List elements can be accessed by providing start and stop value.

Syntax:

ListName[startIndex:EndIndex:skip index]

74

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING LIST ELEMENTS BY SLICING

Example

- **Access all list elements (from list start to end)**

Print(samplelist[:])

- **Access specific range of Element in List** (0 to 3 Index elements access)

Print(samplelist[0:3])

- **Access specific range of Element in List by jumping 2 elements instead of default value 1**

Print(samplelist[0:3:**2**])

75

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING LIST ELEMENTS (USING INDEXING AND SLICING)

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> #Access List Elements
>>> list=[11,"computer",11.5,33]
>>> list
[11, 'computer', 11.5, 33]
>>> print(list[:])
[11, 'computer', 11.5, 33]
>>> print(list[0])
11
>>> print(list[0:2])
[11, 'computer']
>>> print(list[::-1])
[33, 11.5, 'computer', 11]
>>> print(list[0:3:2])
[11, 11.5]
>>>
```

Access all Element (start from 0 and end with length of lis

Access particular Element

Access Elements from 0 to 2 Index [Slicing]

Access Elements in reverse order
(-1 represents the Last Element)

Access Elements from 0 to 3 & jumping 2 elements instead of 1

76

P P SAVANI UNIVERSITY

School of Engineering

MUTABLE LIST DEMO

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> samplelist=[11,22,"Yuvraj","Yatharth",15.5]
>>> samplelist
[11, 22, 'Yuvraj', 'Yatharth', 15.5]
>>> print(samplelist[1])
22
>>> samplelist[1]="Yatri"
>>> print(samplelist)
[11, 'Yatri', 'Yuvraj', 'Yatharth', 15.5]
>>> #List are mutable
>>> type(samplelist)
<class 'list'>
```

Element updated

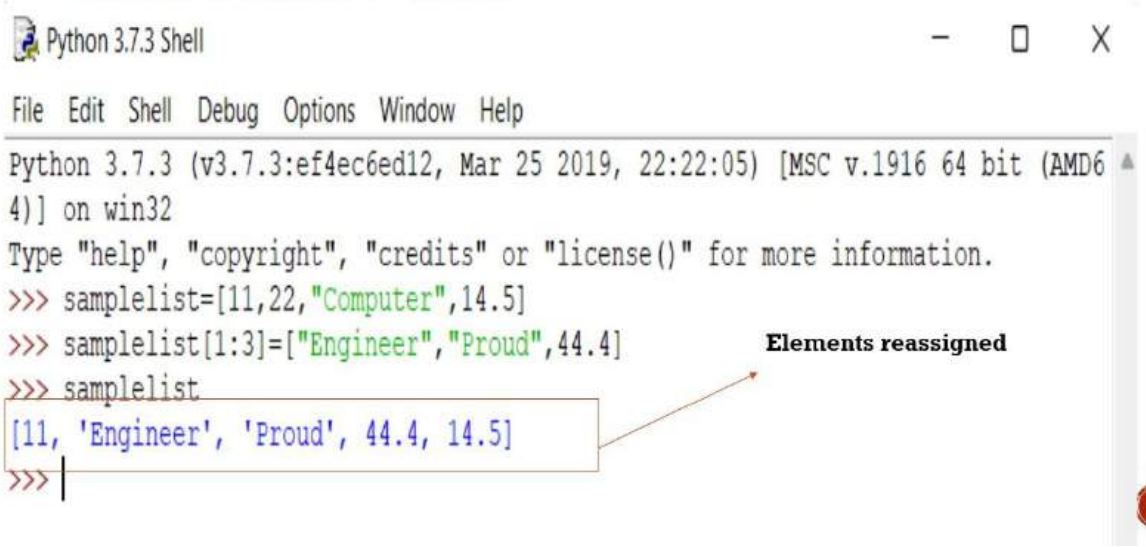
Datatype of created List is List

77

P P SAVANI UNIVERSITY

School of Engineering

REASSIGNING LIST



78

P P SAVANI UNIVERSITY

School of Engineering

NESTED LIST

- If list is nested within another list then inner list can be accessed by using multiple indexes.

Example

sample_mixed_list = ["Computer Engineering", [11,22,33], True]

- **Access Elements from Inner list [11,22,33] at index 1.**

sample_mixed_list[1][2] # Inner List at index 1 and in that Element at index 2 → 33

- **Access Element from Inner String Computer Engineering String**

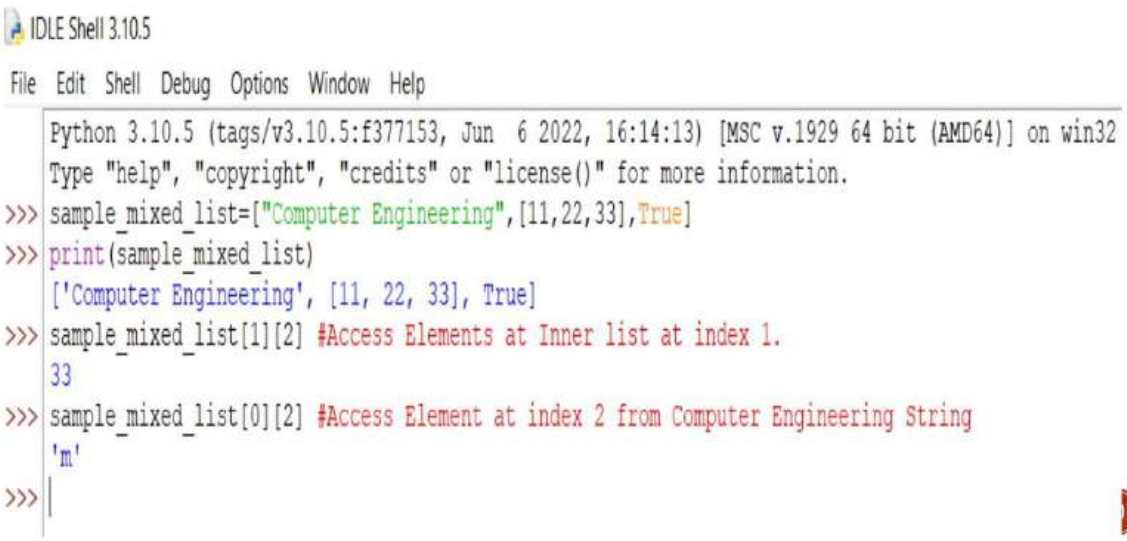
sample_mixed_list[0][2] # String at index 0 and in the Element at index 2 → 'm'

79

P P SAVANI UNIVERSITY

School of Engineering

NESTED LIST DEMO



P P SAVANI UNIVERSITY

School of Engineering

DELETE LIST AND LIST ITEMS

Syntax:

❑**Delete specified Index Item**

del listName[Index]

❑**Delete whole List**

del listName

P P SAVANI UNIVERSITY

School of Engineering

DELETE LIST AND LIST ITEMS

Example:

samplelist=[11,"Yatri","Yatharth","Yuvraj",44.5,55]

❑**Delete a Element from list**

del samplelist[3] #delete Element at Index 3

❑**Delete range of Elements from list**

del samplelist[2:4] #delete Element from Index 2 to 4

❑**Delete whole list**

del samplelist #delete all elements

P P SAVANI UNIVERSITY

School of Engineering

DELETE LIST

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

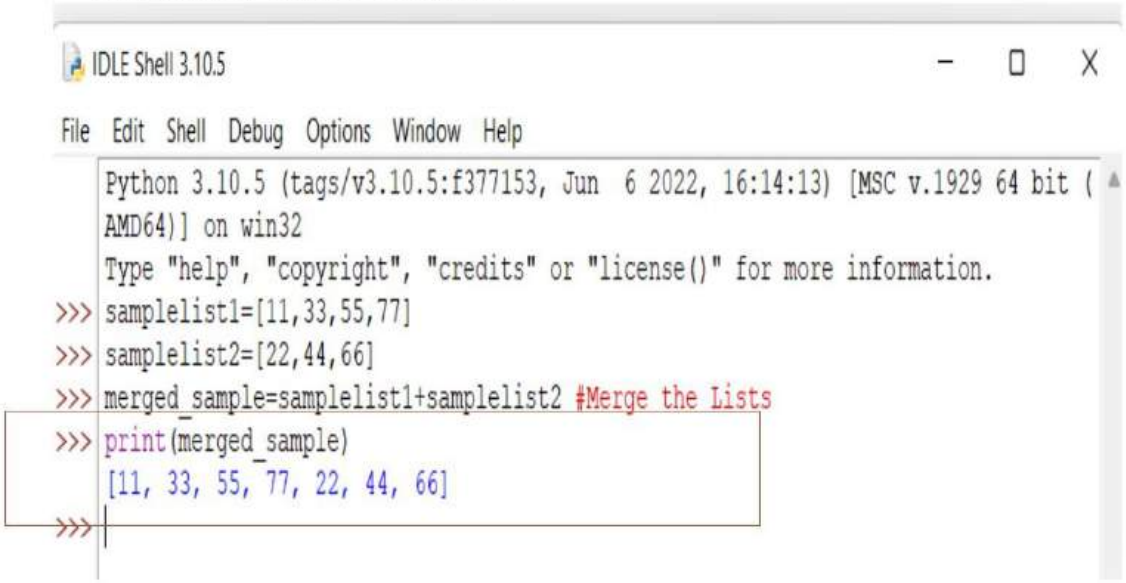
```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> samplelist=[11,"Yatri","Yatharth","Yuvraj",44.5,55]
>>> del samplelist[3] #delete Element at Index 3
>>> samplelist
[11, 'Yatri', 'Yatharth', 44.5, 55]
>>> del samplelist[2:4]
>>> samplelist
[11, 'Yatri', 55]
>>> del samplelist #delete whole list
>>> print(samplelist)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(samplelist)
NameError: name 'samplelist' is not defined
>>>
```

List deleted

P P SAVANI UNIVERSITY

School of Engineering

MERGE THE LISTS



84

P P SAVANI UNIVERSITY

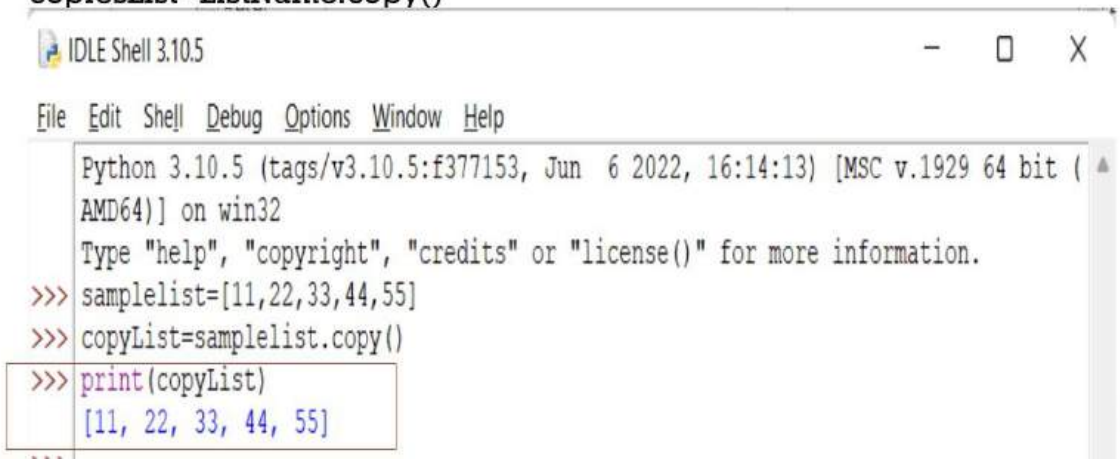
School of Engineering

LIST FUNCTIONS

copy() : It is used to make a copy of a List.

Syntax:

copiesList=ListName.copy()



85

P P SAVANI UNIVERSITY

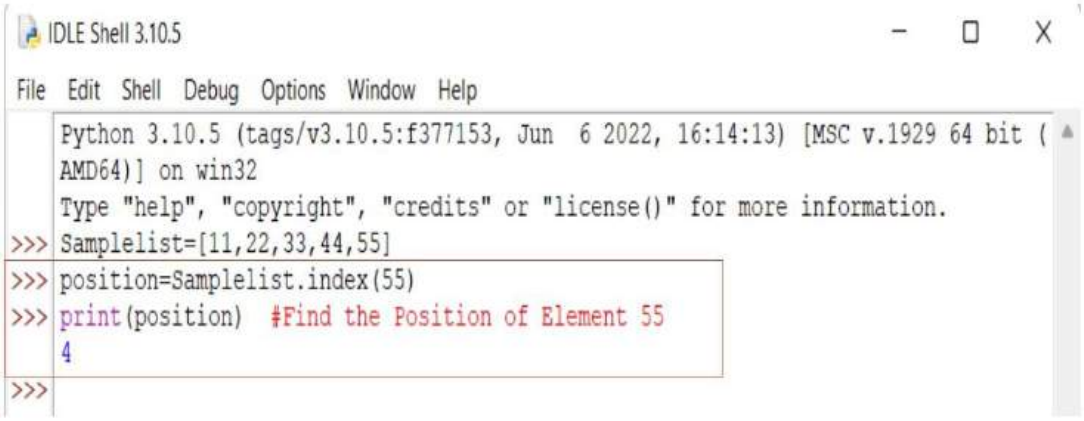
School of Engineering

LIST FUNCTIONS

index() : It is used to return the position of a specified list element.

Syntax:

position=ListName.index(ElementName)



86

P P SAVANI UNIVERSITY

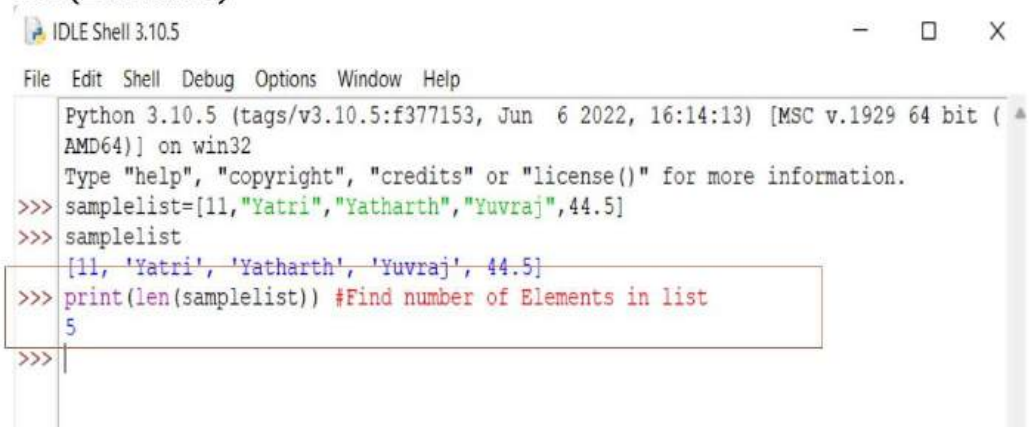
School of Engineering

LIST FUNCTIONS

Len() : It is used to count the number of elements in List.

Syntax:

Len(ListName)



87

P P SAVANI UNIVERSITY

School of Engineering

LIST FUNCTIONS

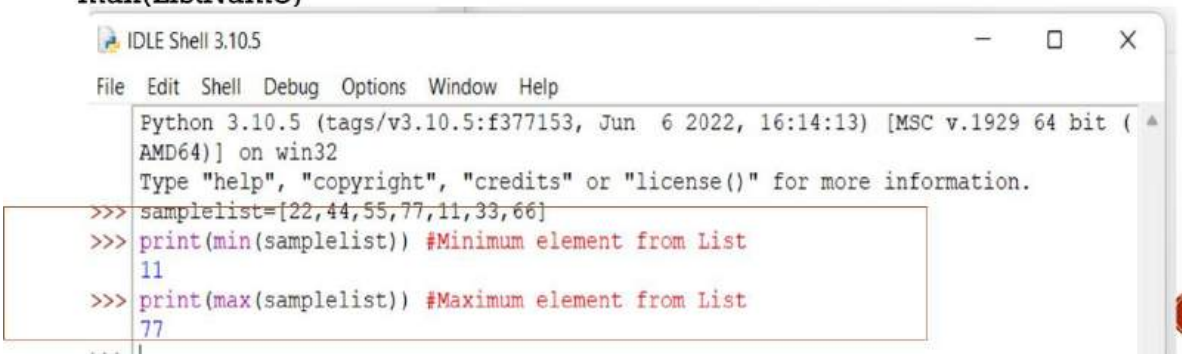
min() : It is used to find the minimum elements from List.

max() : It is used to find the maximum elements from List.

Syntax:

min(ListName)

max(ListName)



88

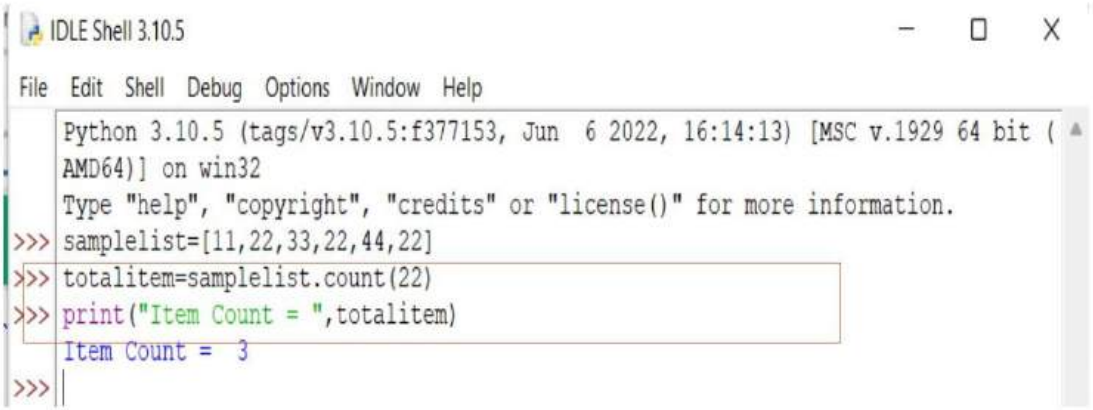
P P SAVANI UNIVERSITY

School of Engineering

LIST FUNCTIONS

count() : It is used to count the number of times the specified element available in List.

Syntax:



89

P P SAVANI UNIVERSITY

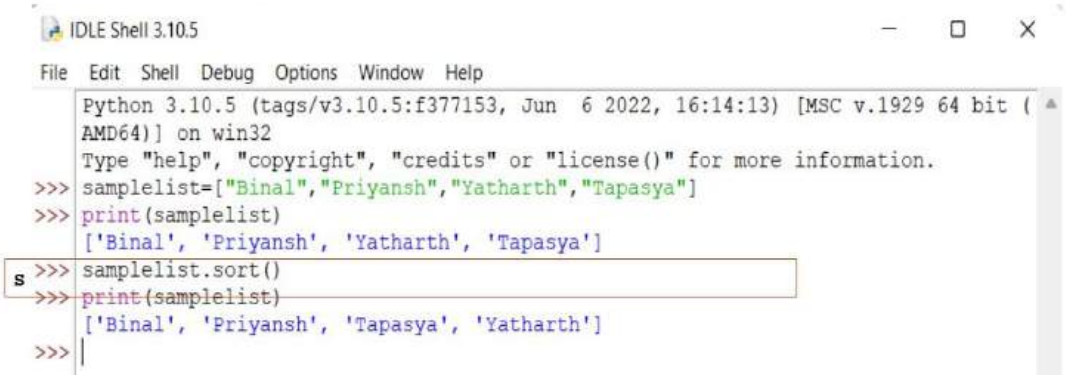
School of Engineering

LIST FUNCTIONS

sort() : It is used to arrange the elements of List. By default arranged in ascending order.

Syntax:

samplelist.sort()



90

P P SAVANI UNIVERSITY

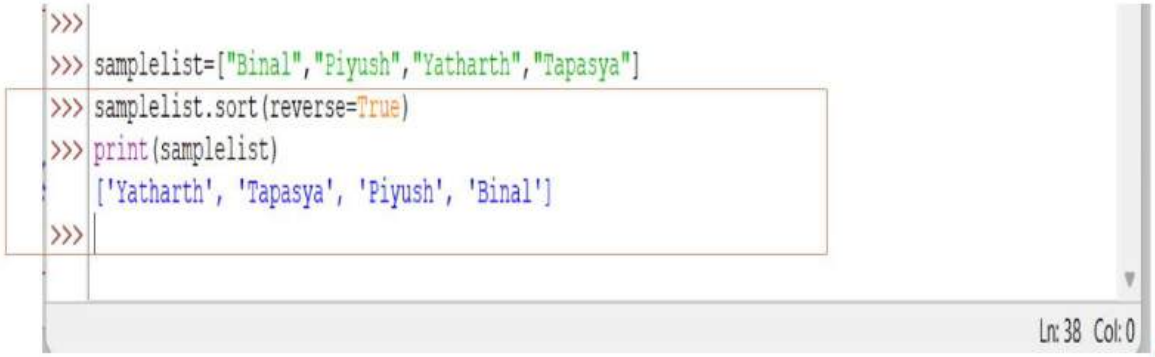
School of Engineering

LIST FUNCTIONS

sort() : It is used to arrange the elements of List.

Syntax: (for descending order)

samplelist.sort(reverse=True)



91

P P SAVANI UNIVERSITY

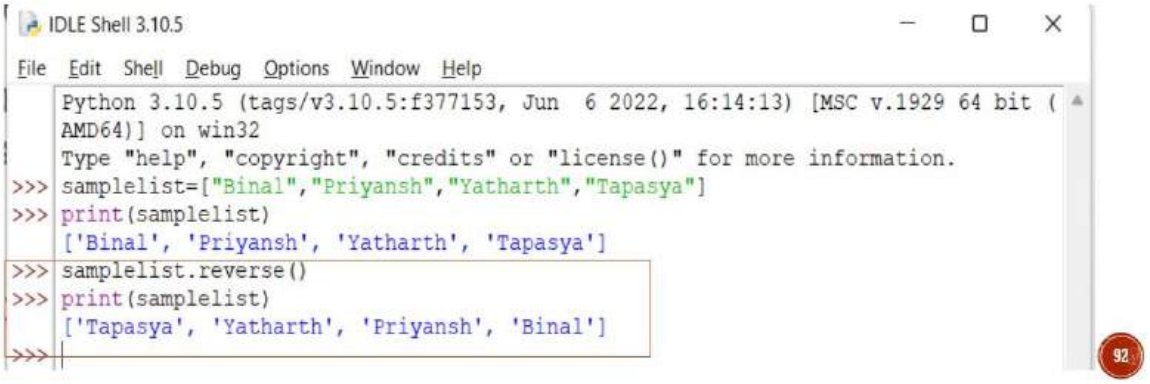
School of Engineering

LIST FUNCTIONS

reverse() : It is used to print the reverse the elements of List in backward.

Syntax:

samplelist.reverse(ListName)



P P SAVANI UNIVERSITY

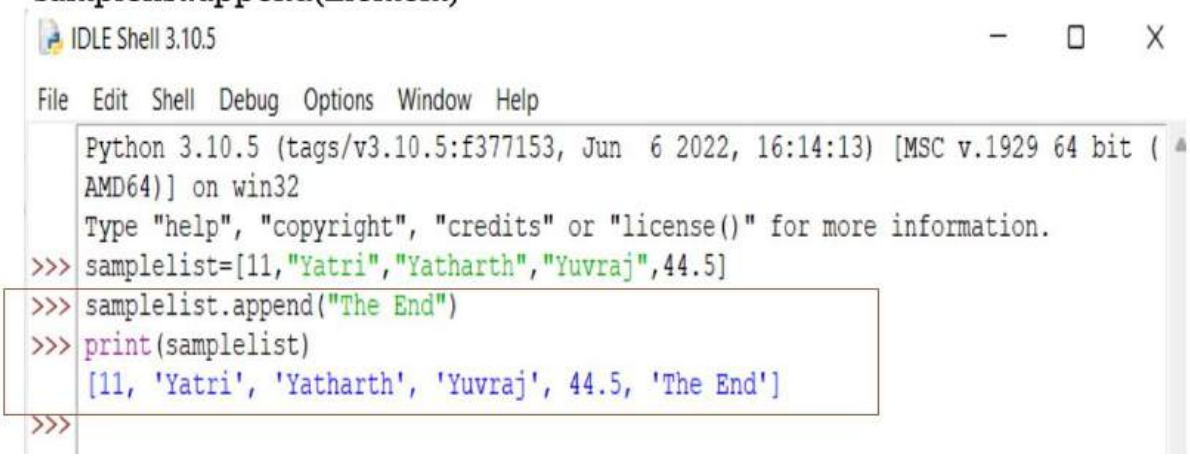
School of Engineering

LIST FUNCTIONS

append() : It is used to add the element at the end of list.

Syntax:

samplelist.append(Element)



P P SAVANI UNIVERSITY

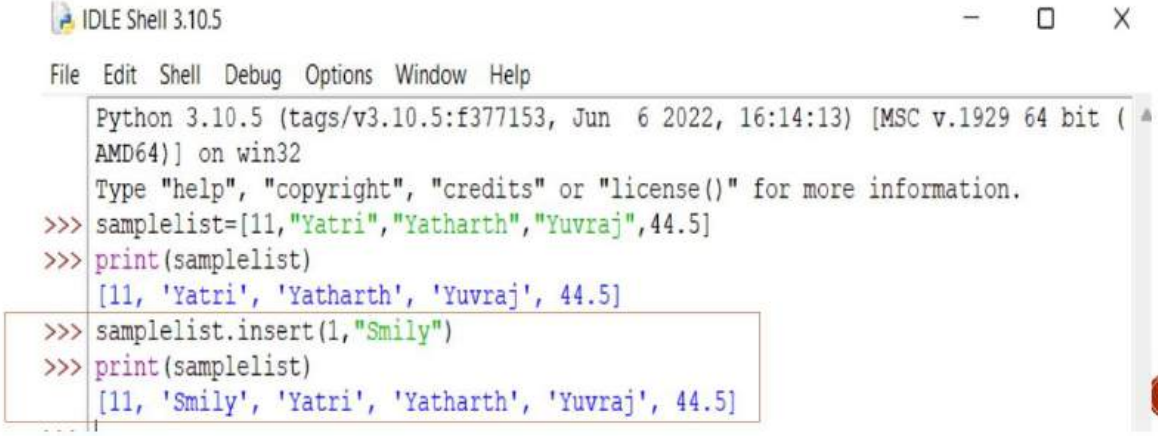
School of Engineering

LIST FUNCTIONS

insert() : It is used to add an element at a specified position.

Syntax:

samplelist.insert(position,Element)



94

P P SAVANI UNIVERSITY

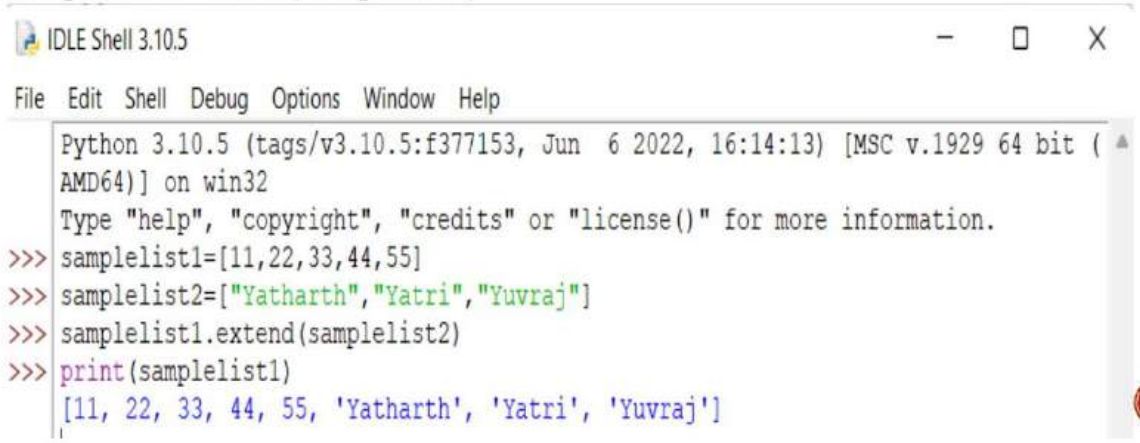
School of Engineering

LIST FUNCTIONS

extend() : It is used to add specified list elements to the end of the current list.

Syntax:

Samplelist1.extend(samplelist2)



95

P P SAVANI UNIVERSITY

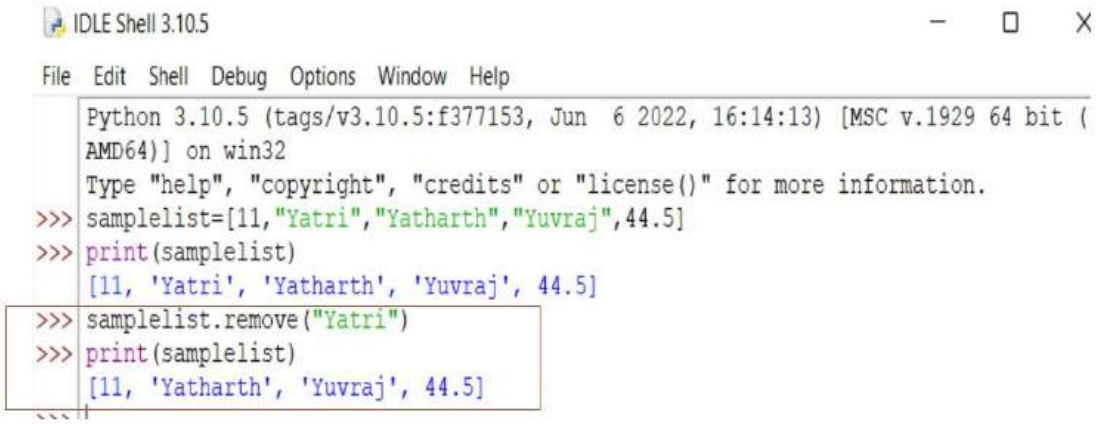
School of Engineering

LIST FUNCTIONS

remove() : It is used to delete an element from a list.

Syntax:

samplelist.remove(Element)



96

P P SAVANI UNIVERSITY

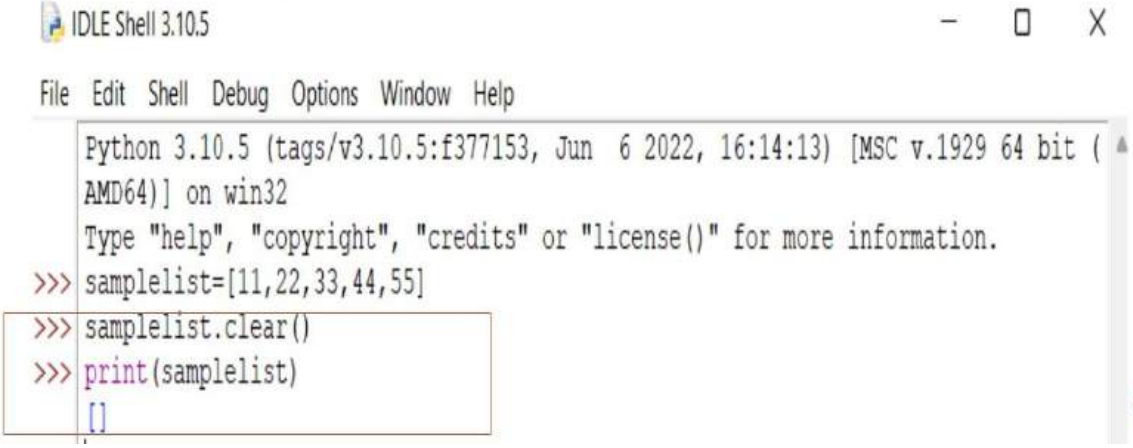
School of Engineering

LIST FUNCTIONS

clear() : It is used to remove all elements from a list.

Syntax:

samplelist.clear()



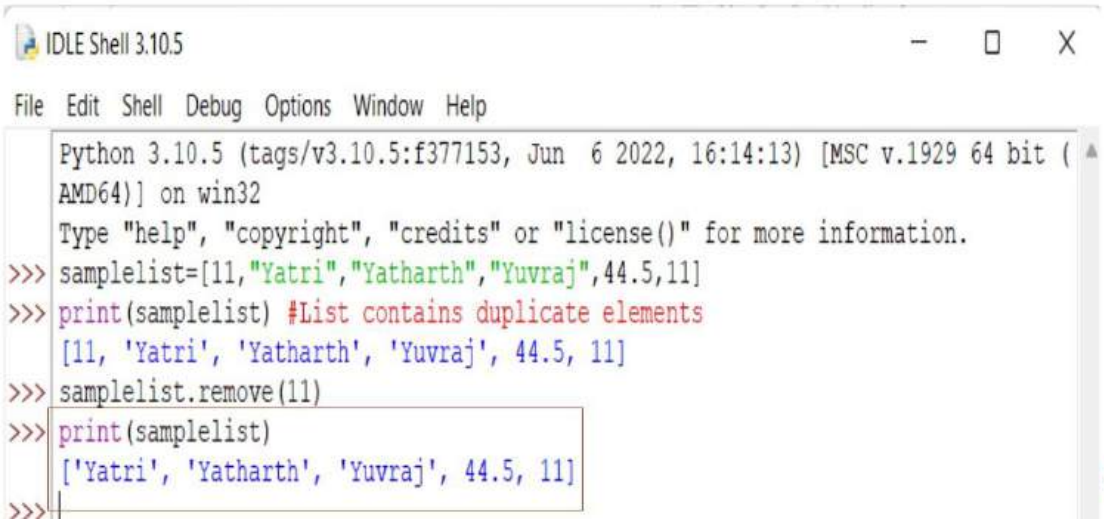
97

P P SAVANI UNIVERSITY

School of Engineering

LIST CONTAINS DUPLICATE VALUE

1st matched value removed.



98

P P SAVANI UNIVERSITY

School of Engineering

TUPLE

- Tuples are used to store multiple items (collection of data) in a single variable like List.
- Tuples are created using **round brackets**.
- Tuple items allow duplicate values.
- A tuple is a collection of objects which ordered and **immutable.**
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

99

P P SAVANI UNIVERSITY

School of Engineering

TUPLE EXAMPLE

```
names = (“Yuvraj", “Yath", “Yatri“)
Numbers = (11,22,33,44)
MixedTuples = (11, ”computer”, 11.5)
sampleTuple=tuple((11,22,”Yuvraj”,”Yatharth”,True))
```

100

P P SAVANI UNIVERSITY

School of Engineering

TUPLE

Syntax

```
variableName=(Elements)
variableName=tuple((elements))
```

101

P P SAVANI UNIVERSITY

School of Engineering

TUPLE DEMO

```
>>> sampleTuple=tuple((11,22,"Yuvraj","Yatharth",True))
...
>>> print(sampleTuple)
...
(11, 22, 'Yuvraj', 'Yatharth', True)
>>>
```

Tuple created using Tuple constructor.

102

P P SAVANI UNIVERSITY

School of Engineering

TUPLE DEMO

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> sampletuple=()
>>> type(sampletuple)
<class 'tuple'>
>>> sampletuple=(1)
>>> print(sampletuple)
1
>>> sampletuple=(1,2,3)
>>> print(sampletuple)
(1, 2, 3)
```

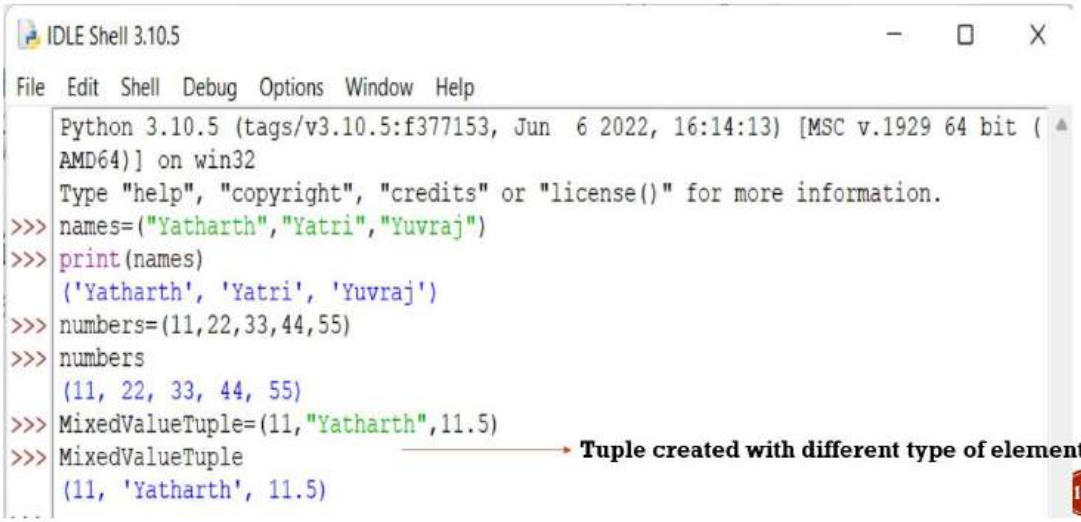
Empty Tuple created.

103

P P SAVANI UNIVERSITY

School of Engineering

TUPLE DEMO



Tuple created with different type of elements.

104

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING TUPLE ELEMENTS BY USING INDEX

Tuple Index starts from zero.

-1 represents last Element of Tuple.

Syntax:

tupleName[index]

105

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING TUPLE ELEMENTS BY USING INDEX

Example

sampletuple = [11, "Yatri","Yuvraj", "Yatharth",15.5]

- **Access first Element of a Tuple**

Print(sampletuple[0])

- **Access Last Element of a Tuple (negative sign index used to access the tuple from backwards)**

Print(sampletuple[-1])

106

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING TUPLE ELEMENTS BY SLICING

Tuple elements can be accessed by providing start and stop value.

Syntax:

TupleName[startIndex:EndIndex:Skip Index]

107

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING TUPLE ELEMENTS BY SLICING

Example

- **Access all Tuple elements (from Tuple start to end)**

Print(sampletuple(:))

- **Access specific range of Element in Tuple** (0 to 3 Index elements access)

Print(sampletuple[0:3])

- **Access specific range of Element in Tuple by jumping 2 elements instead of default value 1**

Print(sampletuple[0:3:2])

108

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING TUPLE ELEMENTS (USING INDEXING AND SLICING)

IDLE Shell 3.10.5

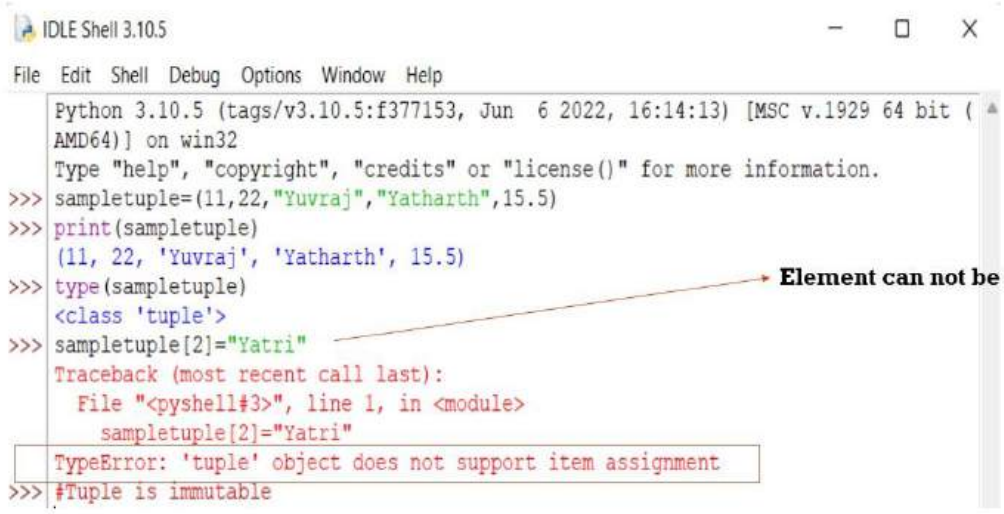
File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> sampletuple=(11,"Yatri","Yuvraj","Yatharth",15.5)
>>> print(sampletuple)
(11, 'Yatri', 'Yuvraj', 'Yatharth', 15.5)
>>> print(sampletuple[0]) #Access First Element of Tuple
11
>>> print(sampletuple[-1]) #Access Last Element of Tuple
15.5
>>> print(sampletuple[1:3]) #Access specified range of Element of Tuple
('Yatri', 'Yuvraj')
>>> print(sampletuple[::2]) #Accesss elements by jumping 2 elements in tuples
(11, 'Yuvraj', 15.5)
>>> print(sampletuple[1:3:2]) #Accesss elements by jumping 2 elements in tuples from range 1 to 3
('Yatri',)
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

IMMUTABLE TUPLE DEMO



110

P P SAVANI UNIVERSITY

School of Engineering

CHANGE TUPLE ELEMENTS

- Tuple elements can not be changed.
- Tuple can be converted to List, can be mutable and various list functions can be used.

Syntax:

Convert Tuple to List

list(tupleName)

Convert List to Tuple

tuple(listName)

111

P P SAVANI UNIVERSITY

School of Engineering

CHANGE TUPLE ELEMENTS

```
>>> sampleTuple=(11,22,"Yuvraj","Yatharth",15.5)
>>> sampleTuple=list(sampleTuple) #Convert Tuple to List
>>> sampleTuple[2]="Yatri"
>>> print(sampleTuple)
[11, 22, 'Yatri', 'Yatharth', 15.5]
>>> sampleTuple=tuple(sampleTuple) #Convert List back to Tuple
>>> print(sampleTuple)
(11, 22, 'Yatri', 'Yatharth', 15.5)
>>>
```

Element updated

112

P P SAVANI UNIVERSITY

School of Engineering

DELETE TUPLE

Tuple object doesn't support item deletion.

Syntax:

❑**Delete Tuple**

del listName

Example:

sampleTuple=(11,"Yatri","Yatharth","Yuvraj",44.5,55)

❑**Delete Tuple**

del sampleTuple

113

P P SAVANI UNIVERSITY

School of Engineering

MERGE THE TUPLES

```
>>> sampletuple1=[11,33,55,77]
>>> sampletuple2=[22,44,66]
>>> merged_sample=sampletuple1+sampletuple2
>>> print(merged_sample) #Tuples merged
[11, 33, 55, 77, 22, 44, 66]
>>>
```

114

P P SAVANI UNIVERSITY

School of Engineering

TUPLES FUNCTIONS

```
>>> sampletuple=tuple((11,22,44,77,55))
>>> print(len(sampletuple)) #count number of elements in tuple
5
>>> print(max(sampletuple)) #Maximum element in Tuple
77
>>> print(min(sampletuple)) #Minimum element in Tuple
11
>>>
```

115

P P SAVANI UNIVERSITY

School of Engineering

TUPLE FUNCTIONS

index() : It is used to return the position of a specified Tuple element.

Syntax:

position=TupleName.index(ElementName)

```
>>> sampletuple=tuple((11,22,33,44,55))
>>> position=sampletuple.index(55)
>>> print("Position : ", position)
Position :  4
>>>
```

116

P P SAVANI UNIVERSITY

School of Engineering

TUPLE FUNCTIONS

count() : It is used to count the number of times the specified element available in Tuple.

Syntax:

TupleName.count(element)

```
>>>
>>> sampletuple=tuple((11,22,33,22,44,22))
>>> totalitem=sampletuple.count(22)
>>> print("Item Count = ",totalitem)
Item Count =  3
>>>
```

117

P P SAVANI UNIVERSITY

School of Engineering

DELETE TUPLE

```
>>>
>>> sampletuple=(11,"Yatri","Yatharth","Yuvraj",44.5,55)
>>> del sampletuple[0] # Try to delete an item from Tuple
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    del sampletuple[0] # Try to delete an item from Tuple
TypeError: 'tuple' object doesn't support item deletion
>>> del sampletuple #Delete the Tuple
>>> print(sampletuple)
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print(sampletuple)
NameError: name 'sampletuple' is not defined
>>>
```

118

P P SAVANI UNIVERSITY

School of Engineering

LIST MAY CONTAINS TUPLES

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mixed=[(11,22,33),[44,55,67.8],"Yatharth",True]
>>> print(mixed)
[(11, 22, 33), [44, 55, 67.8], 'Yatharth', True]
>>> mixed[3]="Yatri" #Update the Element
>>> print(mixed)
[(11, 22, 33), [44, 55, 67.8], 'Yatharth', 'Yatri']
>>> mixed[1][2]=66 #Try to update Element at Index 1 List element 3
>>> print(mixed)
[(11, 22, 33), [44, 55, 66], 'Yatharth', 'Yatri']
>>> mixed[0][2]=30 #Try to update Element at Index 0 Tuple element 3
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    mixed[0][2]=30 #Try to update Element at Index 0 Tuple element 3
TypeError: 'tuple' object does not support item assignment
```

List contains Tuple

List element updated

Tuple element can not be updated

P P SAVANI UNIVERSITY

School of Engineering

DICTIONARY

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered, changeable and does not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values.
- Dictionaries are mutable.
- Example : Phone Book

120

P P SAVANI UNIVERSITY

School of Engineering

HOW TO CREATE DICTIONARY

Syntax

DictionaryName={}

DictionaryName=dict()

Example

students = {1001:"Yuvraj", 1002: "Yatri" , 1003 : "Yatharth"}

121

P P SAVANI UNIVERSITY

School of Engineering

DICTIONARY DEMO

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> sampleDictionary={} #create Empty dictionary
>>> sampleDictionary2=dict()
>>> print(sampleDictionary)
{}
>>> print(sampleDictionary2)
{}
>>> studentDictionary={1001:"Yatharth",1002:"Yuvraj",1003:"Yatri"} #StudentData
>>> print(studentDictionary)
{1001: 'Yatharth', 1002: 'Yuvraj', 1003: 'Yatri'}
>>>
```

Create Empty Dictionary

122

P P SAVANI UNIVERSITY

School of Engineering

DICTIONARY DEMO

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> studentDictionary={"Name":"Yatharth","Email":"yathu@gmail.com","Contact":[72876677,67767676]}
>>> print(studentDictionary)
{'Name': 'Yatharth', 'Email': 'yathu@gmail.com', 'Contact': [72876677, 67767676]}
>>> print(studentDictionary['Contact']) #Print Contact of a Student
[72876677, 67767676]
>>> print(studentDictionary['Contact'][0]) #Print 1st Contact number of student
72876677
```

123

P P SAVANI UNIVERSITY

School of Engineering

DICTIONARY DOES NOT ALLOW DUPLICATE

- Dictionary does not allow duplicates.
- Duplicate **values will overwrite existing value.**

```
>>>
>>> sampleDictionary=dict({1001:"Yatharth",1002:"Yuvraj",1003:"Yatri",1002:"Yuvi"})
>>> print(type(sampleDictionary))
<class 'dict'>
>>> print(sampleDictionary)
{1001: 'Yatharth', 1002: 'Yuvi', 1003: 'Yatri'}
>>>
```

124

P P SAVANI UNIVERSITY

School of Engineering

ACCESS DICTIONARY ELEMENT

Syntax:

- **Access the elements by using Keys**

DictionaryName[Key]

- **Access the elements by using get()**

DictionaryName.get("Key")

- **Access all Keys of Dictionary by using keys()**

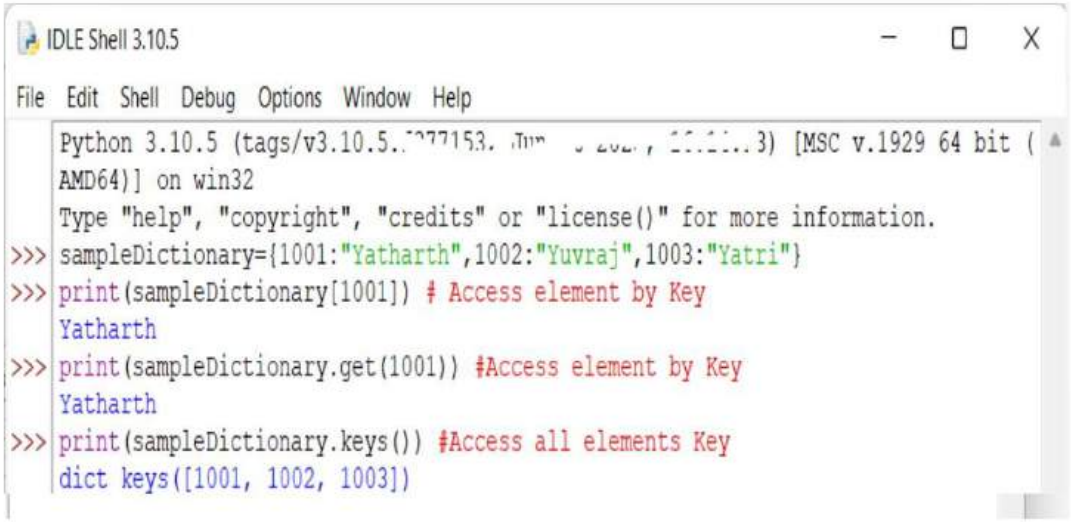
DictionaryName.keys()

125

P P SAVANI UNIVERSITY

School of Engineering

ACCESS DICTIONARY ELEMENT



126

P P SAVANI UNIVERSITY

School of Engineering

ACCESS DICTIONARY ELEMENT

Syntax:

- **Access all elements or values of Dictionary**

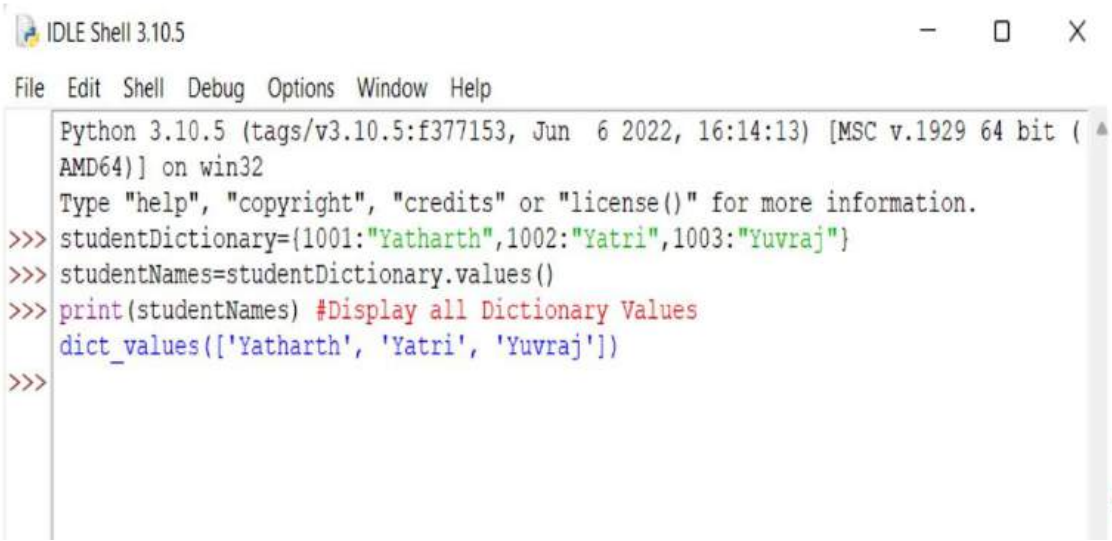
DictionaryName.values()

127

P P SAVANI UNIVERSITY

School of Engineering

ACCESS DICTIONARY ELEMENT



128

P P SAVANI UNIVERSITY

School of Engineering

UPDATE DICTIONARY

Syntax:

- **Access the elements by using Keys**

DictionaryName[Key]=UpdatedElement

- **Access the elements by using update()**

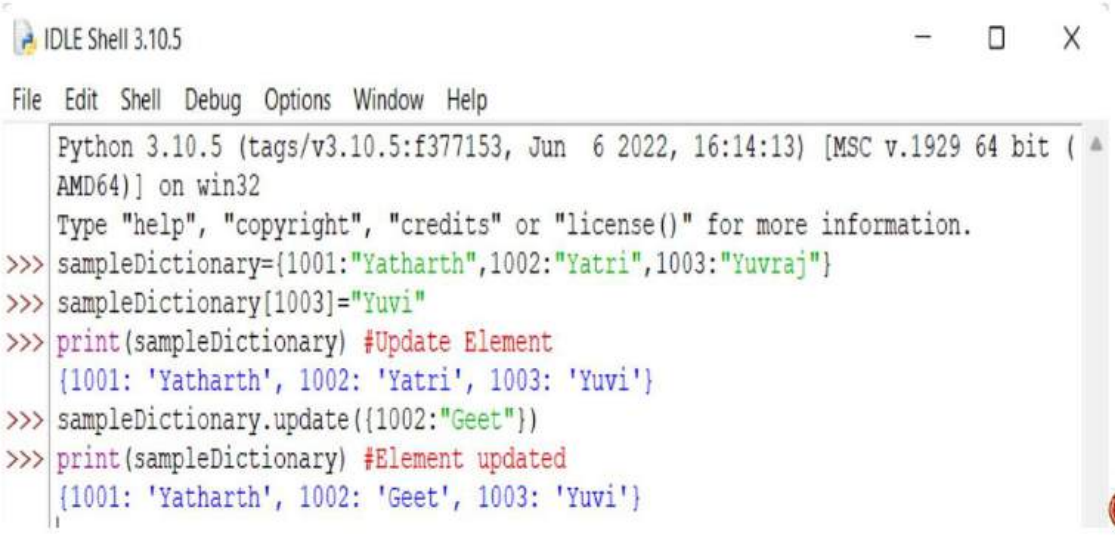
DictionaryName.update({key : UpdatedElement})

129

P P SAVANI UNIVERSITY

School of Engineering

UPDATE DICTIONARY



130

P P SAVANI UNIVERSITY

School of Engineering

DELETE DICTIONARY AND ITEMS

Syntax:

❑**Delete specified Index Item**

del dictionaryName[Index]

❑**Delete whole dictionary**

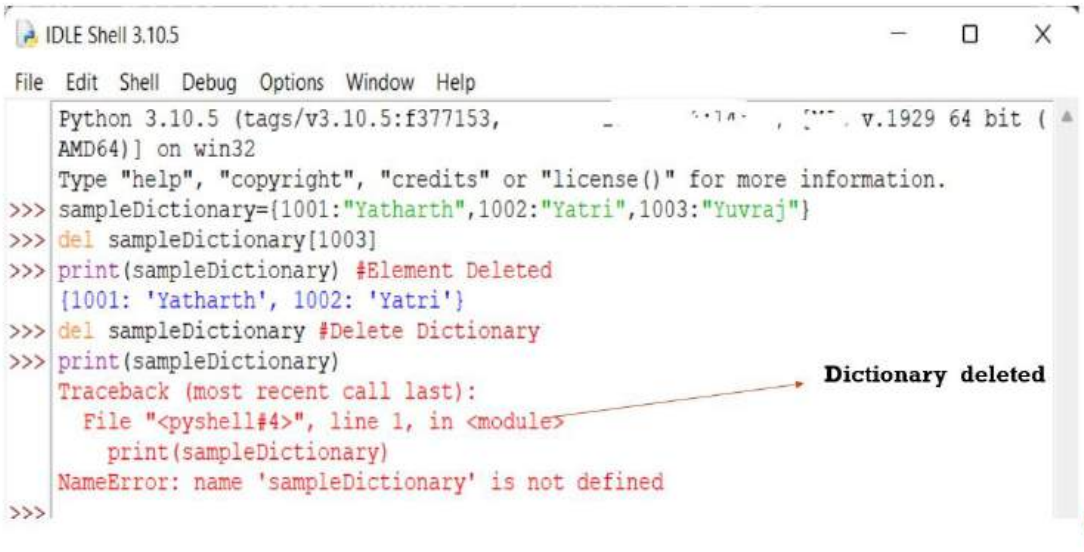
del dictionaryName

131

P P SAVANI UNIVERSITY

School of Engineering

DELETE DICTIONARY



P P SAVANI UNIVERSITY

School of Engineering

CLEAR DICTIONARY

Syntax:

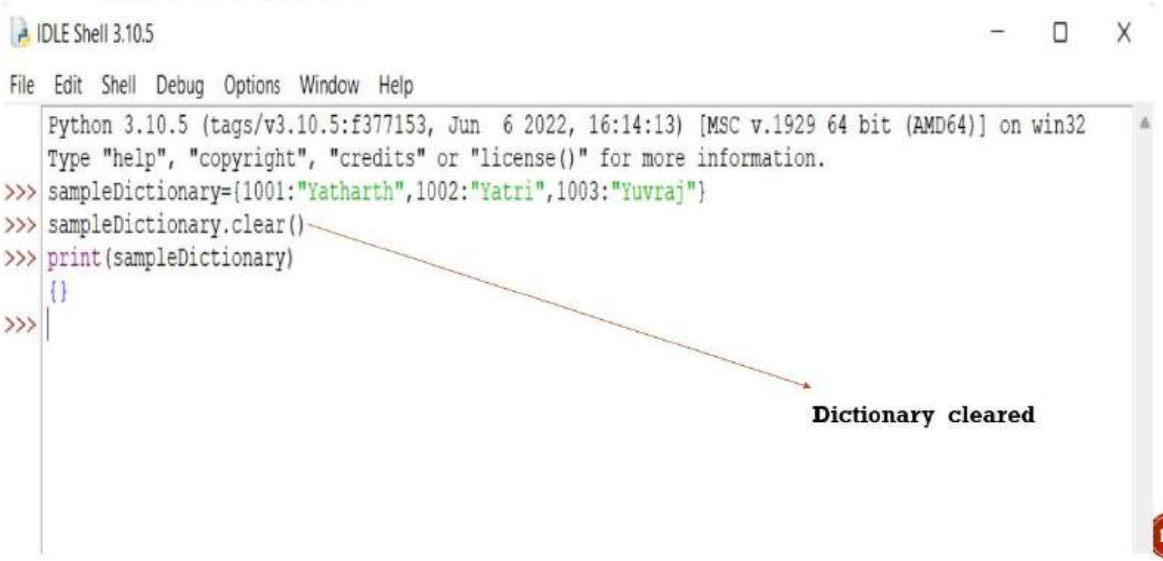
❑**Delete all Items**

dictionaryName.clear()

P P SAVANI UNIVERSITY

School of Engineering

CLEAR LIST



P P SAVANI UNIVERSITY

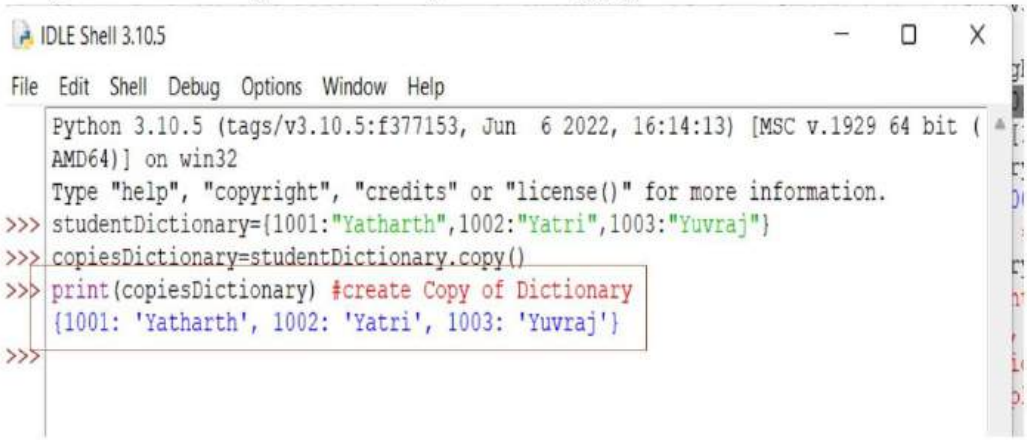
School of Engineering

COPY DICTIONARY

copy() : It is used to make a copy of a Dictionary.

Syntax:

copiesDictionary=dictionaryName.copy()



P P SAVANI UNIVERSITY

School of Engineering

SET

- Set is used to **store multiple items (collection of data) of unique elements** in a single variable.
- A set is a collection which is both **unordered and unindexed**.
- Sets are **mutable**.
- Sets are written with **curly brackets.**
- Once a set is created, you cannot change its items, but you can add new items.

136

P P SAVANI UNIVERSITY

School of Engineering

SET

Example

names = {“Yuvraj", “Yath", “Yatri“}

Numbers = {11,22,33,44}

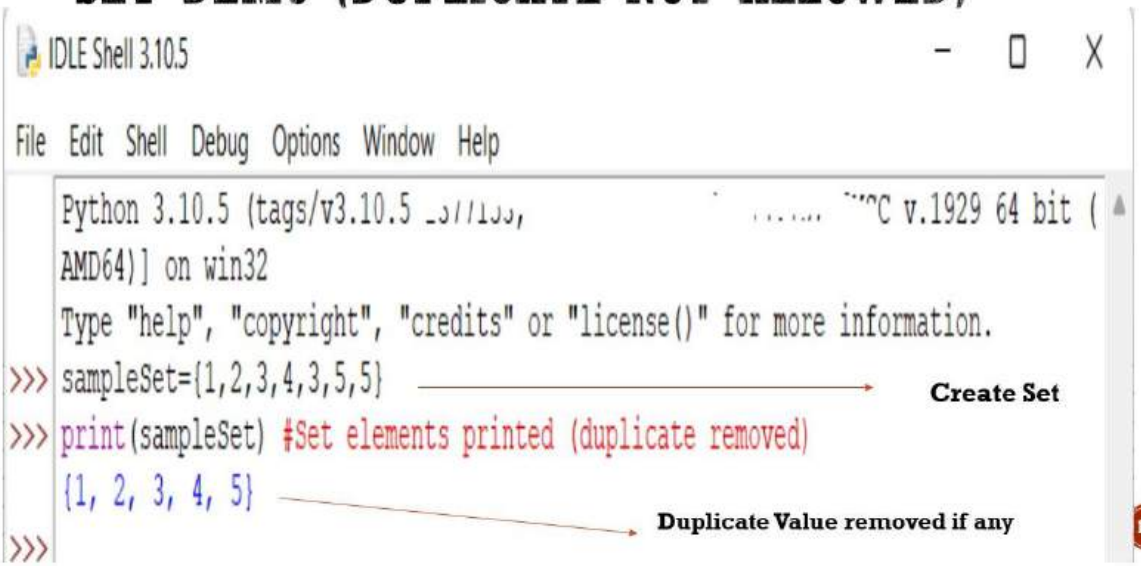
MixedSet = {11, ”computer”, 11.5}

137

P P SAVANI UNIVERSITY

School of Engineering

SET DEMO (DUPLICATE NOT ALLOWED)



P P SAVANI UNIVERSITY

School of Engineering

ACCESS SET

- Set elements **can not be accessed** through Index or Key.
- Elements can be access through loop or we can check whether value is present in set or not.

```
>>> sampleSet={1,2,3,4,5}
>>> print(5 in sampleSet) #Check whether element is present in Set or not
True
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

ADD ELEMENT IN SET

- Add(): It is used to add element in set.

Syntax

SetName.add(Element)

```
>>> sampleSet={1,2,3,4,5}
>>> sampleSet.add(6)
>>> print(sampleSet) #Element added in Set
{1, 2, 3, 4, 5, 6}
>>>
>>>
>>>
```

140

P P SAVANI UNIVERSITY

School of Engineering

ADD ELEMENTS FROM ANOTHER SET

- update(): It is used to update current set by adding elements from another set.

Syntax

SetName.update(Set2)

```
>>> sampleSet1={1,2,3,4,5}
>>> sampleSet2={6,7,8}
>>> sampleSet1.update(sampleSet2)
>>> print("Now Set be :" , sampleSet1)
Now Set be : {1, 2, 3, 4, 5, 6, 7, 8}
>>>
```

141

P P SAVANI UNIVERSITY

School of Engineering

REMOVE SET ELEMENTS

```
IDLE Shell 3.10.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> sampleSet={1,2,3,4,5,6}
>>> sampleSet.remove(7)                                   By using remove()
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    sampleSet.remove(7)
KeyError: 7
>>> sampleSet.discard(7)
>>> print("After Element deletion set is : " , sampleSet)
After Element deletion set is :  {1, 2, 3, 4, 5, 6}
>>> #if element not present in set then remove() will give error
>>> #if element not present in set then discard() will not give error
>>> sampleSet.clear() #Remove all elements in set          By using clear()
>>> print("Now Set is : ",sampleSet)
Now Set is :  set()
>>> del sampleSet #Delete the set                          By using del
>>> print("Now set is :", sampleSet)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    print("Now set is :", sampleSet)
NameError: name 'sampleSet' is not defined
```

142

P P SAVANI UNIVERSITY

School of Engineering

SET FUNCTIONS

```
IDLE Shell 3.10.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> sample_set1={1,2,3,4,5}
>>> sample_set2={3,4,5,6}
>>> print(sample_set1.union(sample_set2))
{1, 2, 3, 4, 5, 6}
>>> # Union() returns set containing union of sets
>>> print(sample_set1.intersection(sample_set2))
{3, 4, 5}
>>> #intersection() returns the set containing common elements in set
>>> print(sample_set1.difference(sample_set2))
{1, 2}
>>> #difference returns the difference between set (The elements present in set1 and
 not in set2)
>>> print(sample_set1.symmetric_difference(sample_set2))
{1, 2, 6}
>>> #symmetric_difference returns the differences of two sets
>>>
```

143

P P SAVANI UNIVERSITY

School of Engineering

STRING

- Strings are collection of characters.
- Strings can be written in Single (‘ ‘) or Double (“ “) quotation marks.
- Strings are not mutable.

144

P P SAVANI UNIVERSITY

School of Engineering

CREATE STRING

Syntax:

Assign String to a Variable

- VariableName=“String”

Assign Multiline String to a Variable

- VariableName=“””” Multiline String “””””

Example

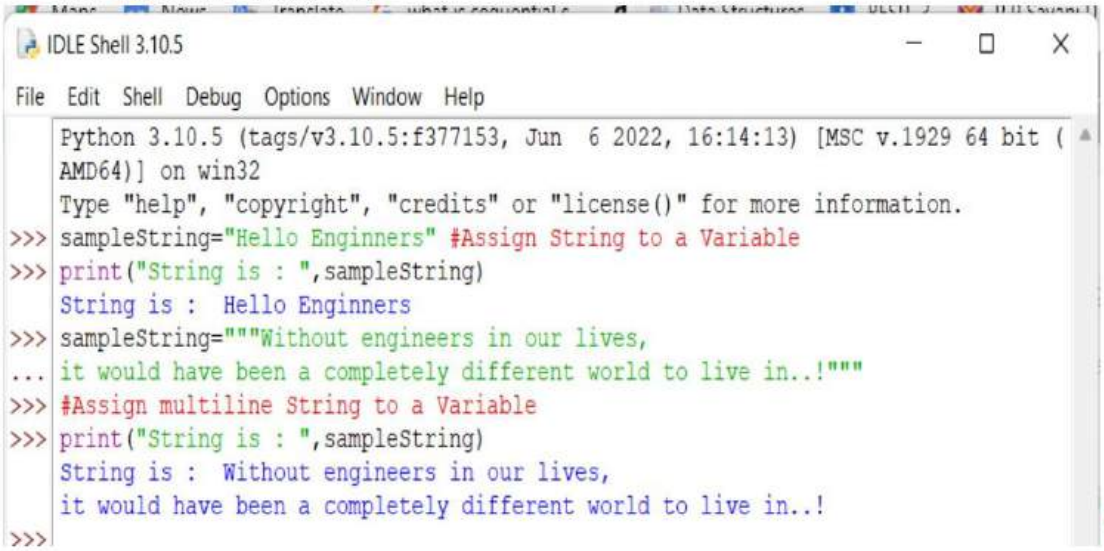
- sampleString =“Hello Engineers”
- Print(“String is :“, sampleString)

145

P P SAVANI UNIVERSITY

School of Engineering

STRING DEMO



146

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING STRING USING INDEX

- String Index starts from zero.
- -1 represents last Element of List.

Syntax:

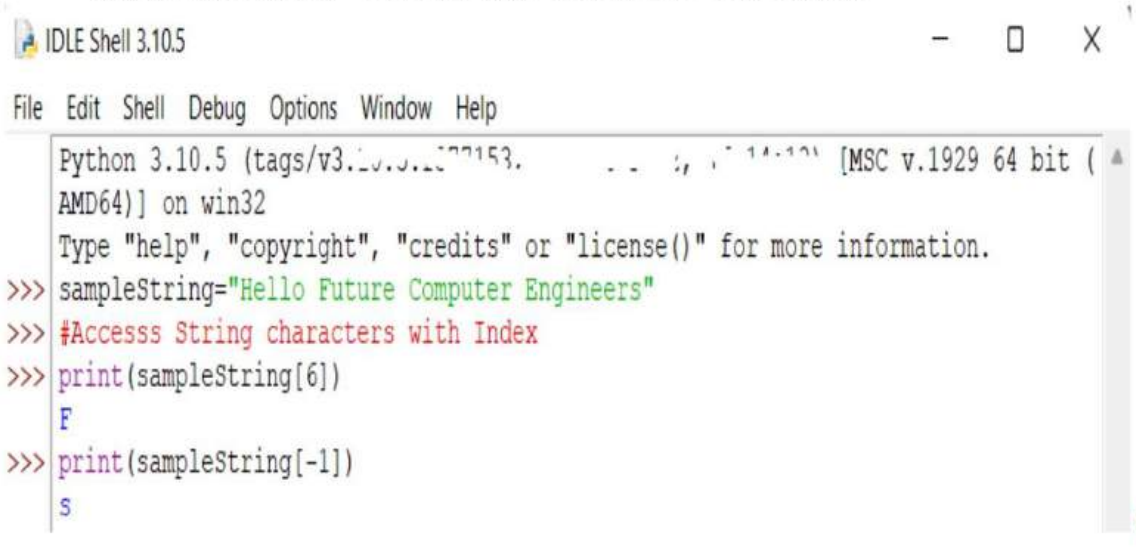
stringName[index]

147

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING STRING USING INDEX



148

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING STRING BY SLICING

String elements can be accessed by providing start and stop value.

Syntax:

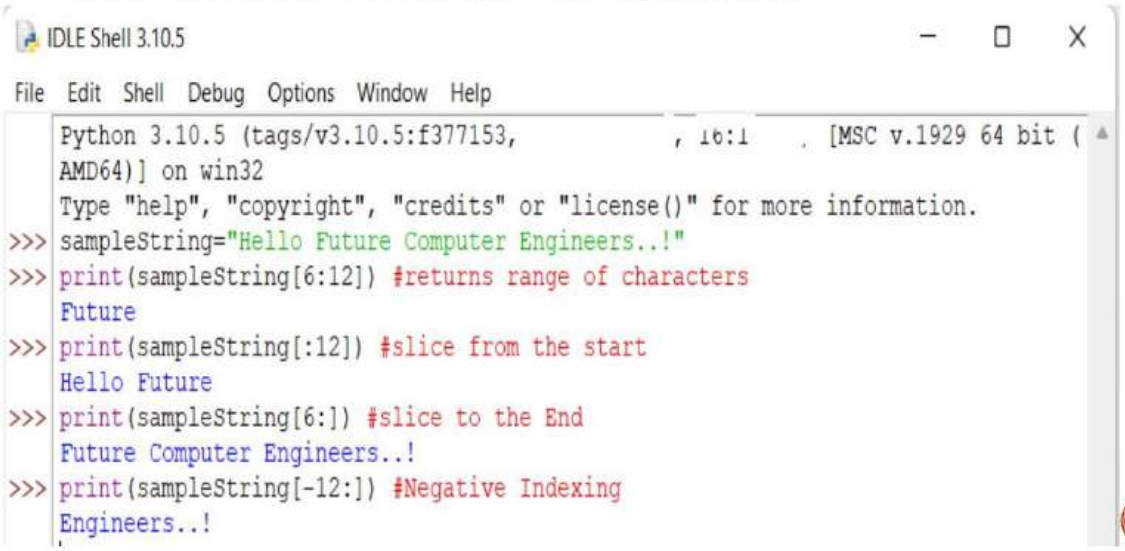
stringName[startIndex:EndIndex:skip index]

149

P P SAVANI UNIVERSITY

School of Engineering

ACCESSING STRING BY SLICING



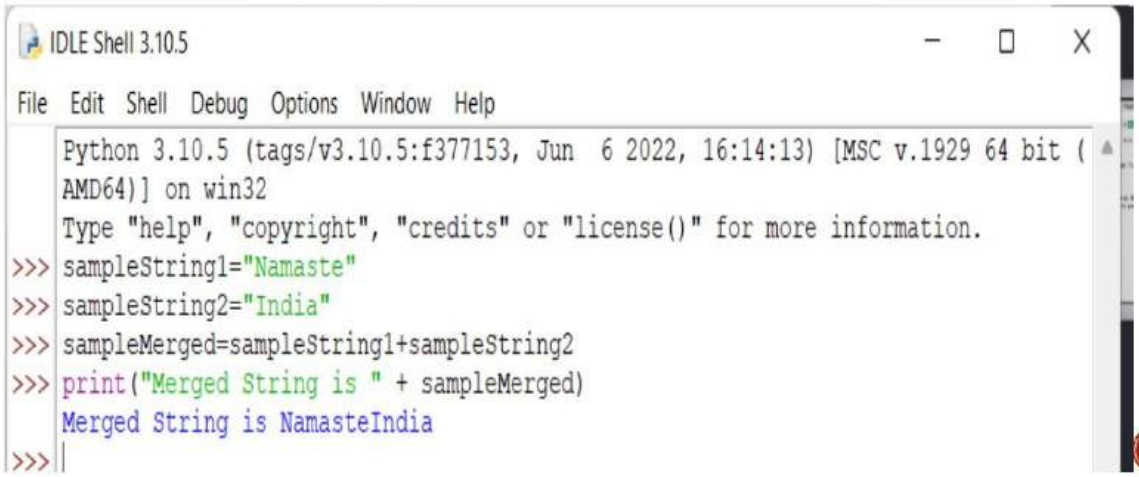
150

P P SAVANI UNIVERSITY

School of Engineering

STRING CONCATENATION

String Merge or combine.



151

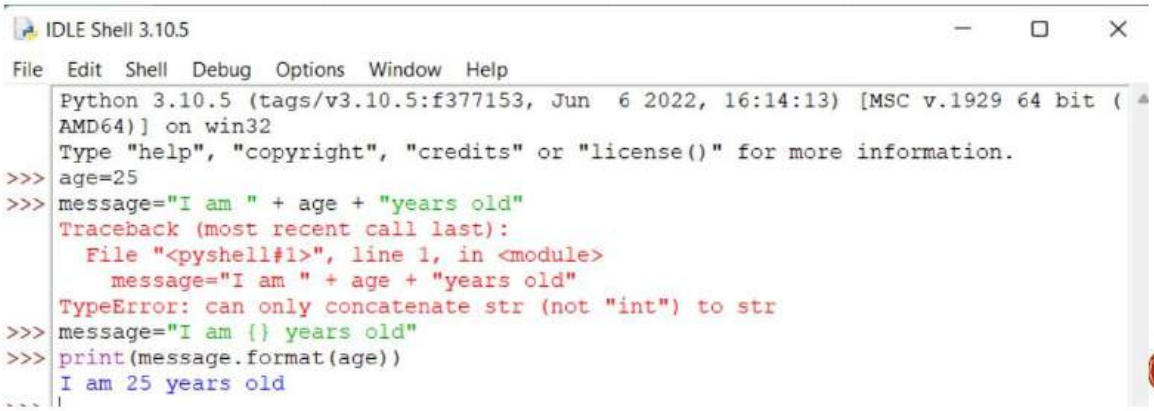
P P SAVANI UNIVERSITY

School of Engineering

STRING FORMAT

- format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are.
- **Syntax**

StringName.format(arguments to be placed in string)



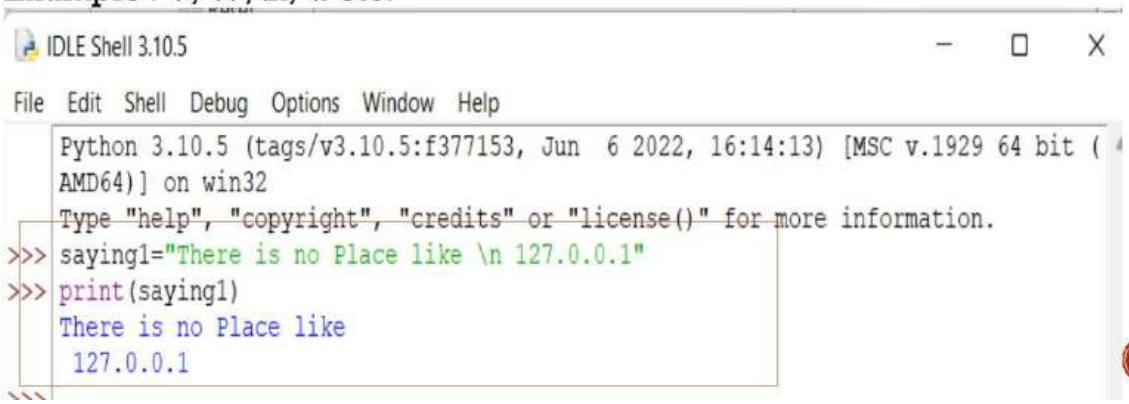
P P SAVANI UNIVERSITY

School of Engineering

ESCAPE CHARACTER

- In Strings, the backslash / is a special escape character.
- An escape character backslash that is followed by the special character to be insert.

Example : \', \\ ,\n, \t etc.



P P SAVANI UNIVERSITY

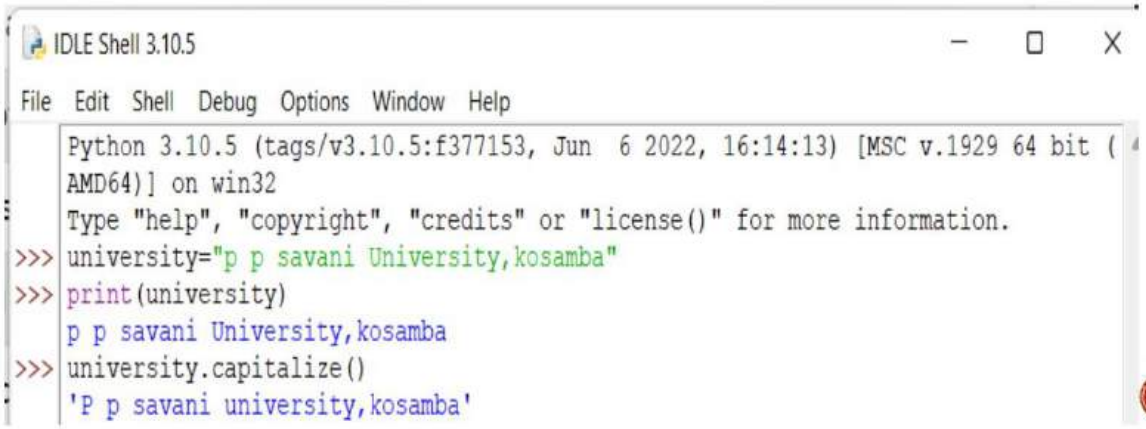
School of Engineering

STRING FUNCTIONS

capitalize() : Converts first character to upper case.

Syntax :

StringName.capitalize()



154

P P SAVANI UNIVERSITY

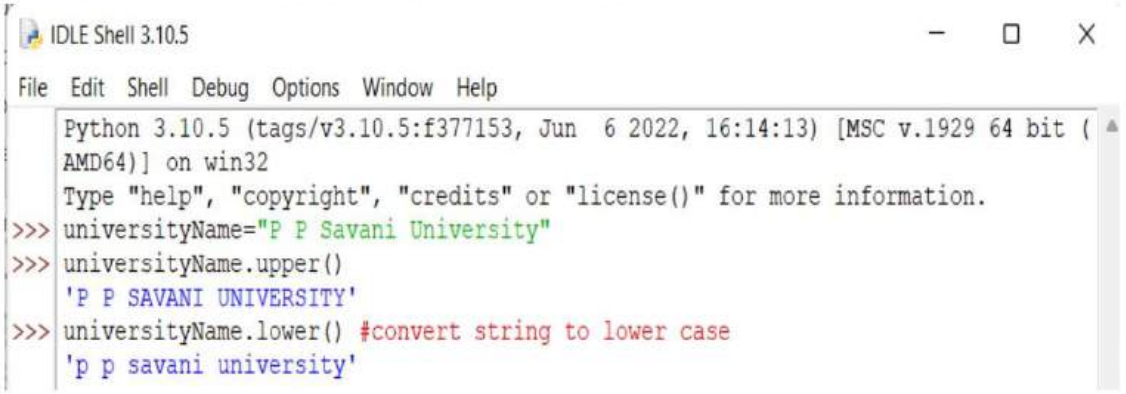
School of Engineering

STRING FUNCTIONS

- **upper() :** converts string to Upper case.
- **lower() :** converts string to Lower case.

Syntax :

StringName.upper() , StringName.lower()



155

P P SAVANI UNIVERSITY

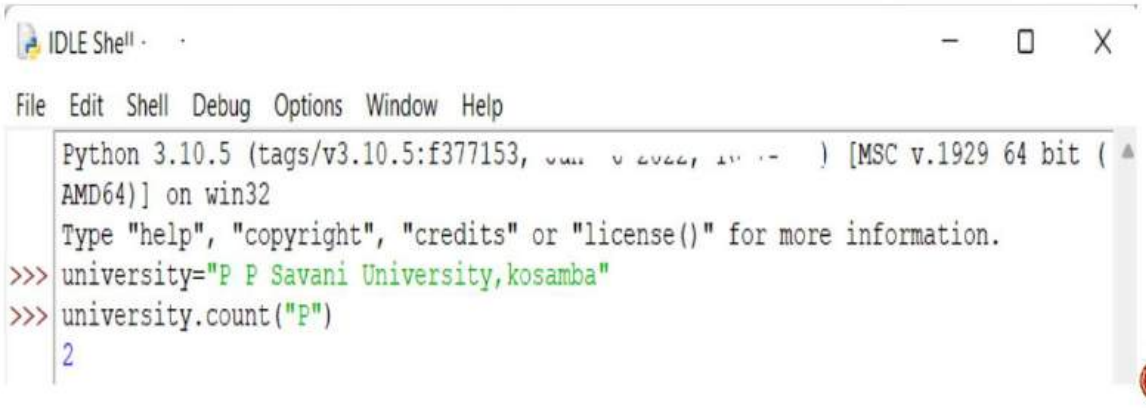
School of Engineering

STRING FUNCTIONS

Count() : Returns the number of times a specified value occurs in a string.

Syntax :

StringName.count(Element)



P P SAVANI UNIVERSITY

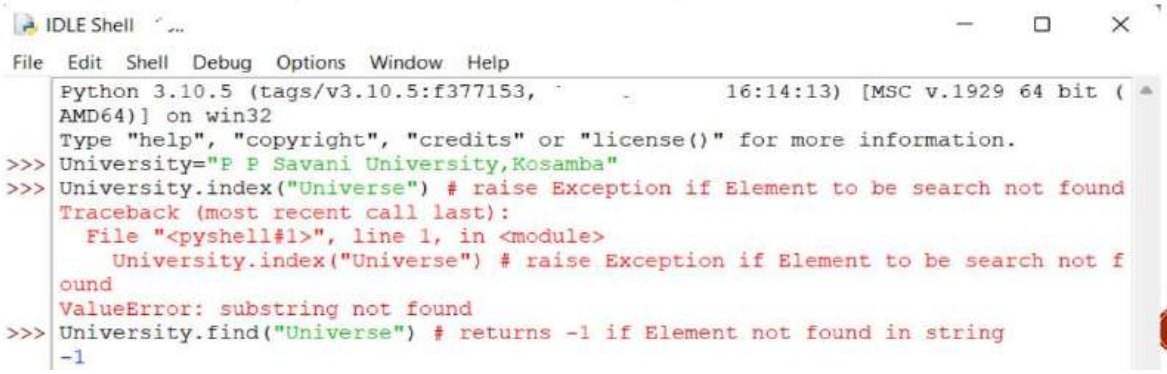
School of Engineering

STRING FUNCTIONS

- **index() , Find() :** Searches the string for a specified value and returns the position of where it was found.
- The index() method raises an exception if the value is not found.

Syntax :

StringName.index(Element, start,end) , StringName.find(Element, start,end)



P P SAVANI UNIVERSITY

School of Engineering

STRING FUNCTIONS

Isdigit() : returns True if all characters in the string are digits.

Isalnum():returns True if all characters in the string are alphanumeric.

Isalpha() : returns True if all characters in the string are in the alphabet.

Syntax :

StringName.isdigit(), StringName.**isalnum**(), StringName.**isalpha**()

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, ˜ ˜, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> number="123"
>>> number.isdigit() # returns true if all String characters are digits
True
>>> LabNumber="B-201-A"
>>> LabNumber.isalnum()  # returns true if all String characters are alpha and digits
False
>>> Department="CSE3"
>>> Department.isalnum() # returns true if all String characters are alpha and digits
True
>>> className="ComputerEngineering"
>>> className.isalpha()   # returns true if all String characters are alphabets
True
```


P P SAVANI UNIVERSITY

School of Engineering

STRING FUNCTIONS

replace() : It replaces a specified string with another specified phrase.

Syntax :

stringName.replace(oldvalue, newvalue, count)

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ˜ ˜ ˜ (tags/v3.10.5:f377153, ˜ ˜ , 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> Saying="There is no place like 127.0.0.1"
>>> print(Saying)
There is no place like 127.0.0.1
>>> Saying.replace("127.0.0.1","Home")
'There is no place like Home'
>>>
```

P P SAVANI UNIVERSITY

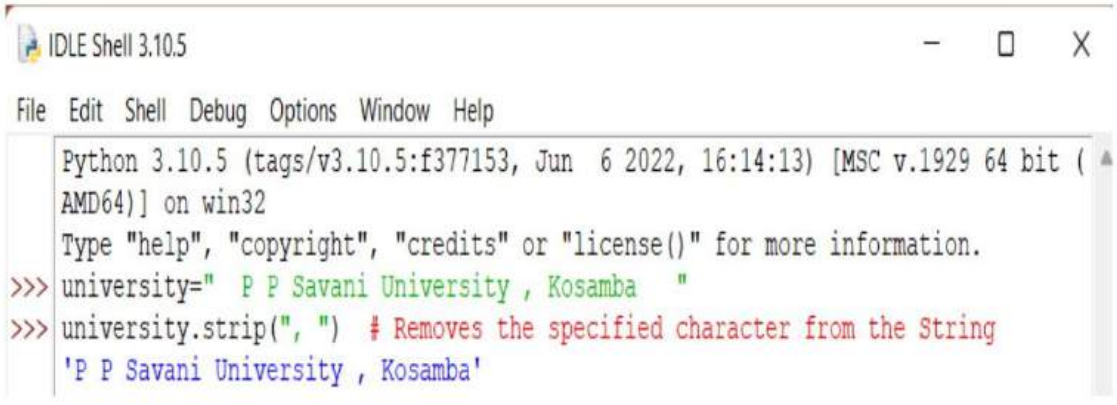
School of Engineering

STRING FUNCTIONS

strip() : It removes any leading (spaces at the beginning) and trailing (spaces at the end) characters (space is the default leading character to remove)

Syntax :

stringName.strip(characters)



160

P P SAVANI UNIVERSITY

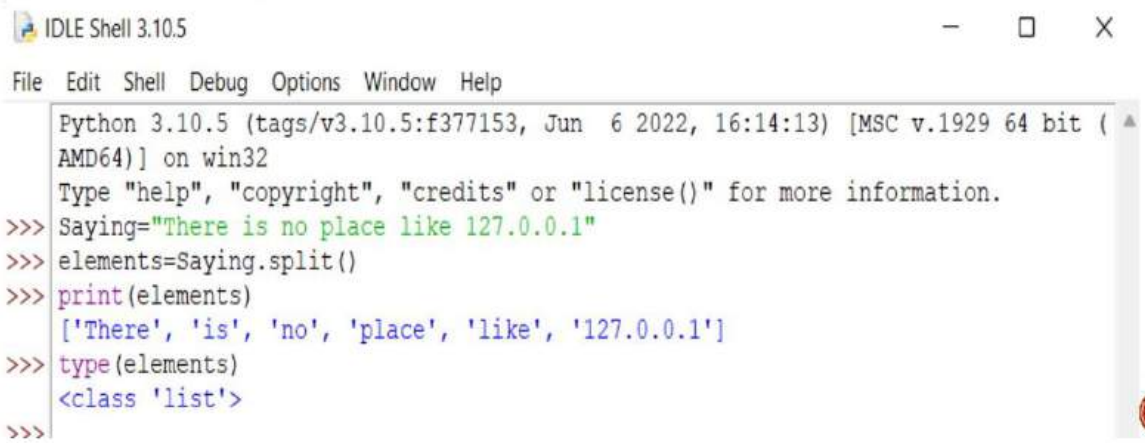
School of Engineering

STRING FUNCTIONS

split() : It splits a string into a list.

Syntax :

stringName.split()



161

P P SAVANI UNIVERSITY

School of Engineering

DIFFERENCE BETWEEN FUNCTION & METHOD

Function	Method
Functions have an **independent existence**. It can be defined outside of the class.	Methods **do not have an independent existence**. It is always defined within a class, struct, or enum.
Functions **do not have any reference variables**.	Methods are called **using reference variables**.
Functions are called **independently**.	Methods are called using **instances or objects of the class.**
Functions are **the properties of structured languages.**	Methods are the **properties of Object-oriented language.**
It is a **self-describing piece of code.**	It is used to **manipulate the instance variable of a class.**
A particular type should **not be associated with the functions.**	A method is **associated with a particular class,** struct, or enum.
Every Function is not a method.	Every method is a function.

P P SAVANI UNIVERSITY

School of Engineering

TYPE CONVERSION

Type conversion is **converting the one type of datatype to another datatype.**

- **Implicit Type Conversion**

Implicit Type Conversion is automatically performed by the Python interpreter.

- **Explicit Type Conversion**

Explicit Type conversion, we enforce the object to a specific data type. So loss of data may occur.

165

P P SAVANI UNIVERSITY

School of Engineering

IMPLICIT TYPE CONVERSION DEMO

implicitCastDemo.py - D:/Python Material/Python Programs/implicitCastDemo.py (3.10.5)

File Edit Format Run Options Window Help

```
number1=1
print("number1 type is " , type(number1))
number2=5.0
print("number2 type is " , type(number2))
ans=number1+number2
print(number1 , "+" , number2 , "=", ans)
print("ans is " ,type(ans))
```

Ln: 6 Col: 0

```
>>
======== RESTART: D:/Python Material/Python Programs/implicitCastDemo.py =======
number1 type is  <class 'int'>
number2 type is  <class 'float'>
1 + 5.0 = 6.0
ans is  <class 'float'>
>>
```

Implicit datatype assigned

166

P P SAVANI UNIVERSITY

School of Engineering

TYPE CONVERSION DEMO

calculator.py - D:/Python Material/Python Programs/calculator.py (3.10.5)

File Edit Format Run Options Window Help

```
number1=input("Please enter Number1 : ")
print(type(number1))
number2=input("Please enter Number2 : ")
print(type(number2))
ans=number1+number2
print(number1 , "+" , number2,"=", ans)
```

Ln: 4 Col: 40

```
Please enter Number1 : 1
<class 'str'>
Please enter Number2 : 2
<class 'str'>
1 + 2 = 12
>>
```

Input() always accept user input in form of String.

Merged as a String

167

P P SAVANI UNIVERSITY

School of Engineering

EXPLICIT DATA CONVERSION METHODS

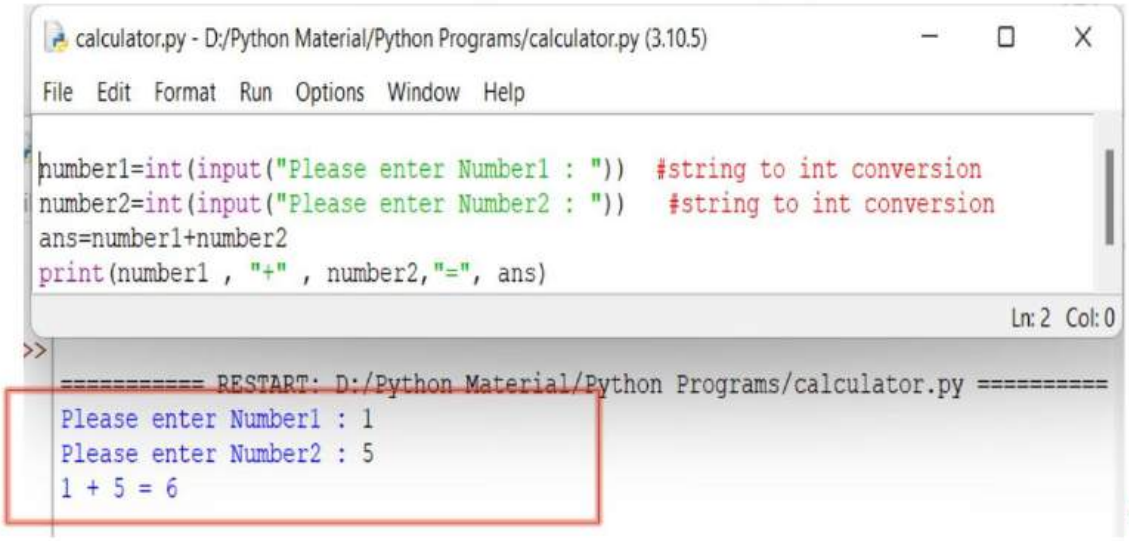
- **int(a):** This function converts **any data type to integer.**
- **float():** This function is used to convert **any data type to a** floating-point **number.**
- **ord() :** This function is used to convert a **character to integer.**
- **hex() :** This function is to convert **integer to hexadecimal string.**
- **oct() :** This function is to convert **integer to octal string.**
- **tuple() :** This function is used to **convert to a tuple.**
- **set() :** This function returns the **type after converting to set.**
- **list() :** This function is used to convert **any data type to a list type.**
- **str() :** Used to **convert integer into a string.**
- **dict() :** This function is used to **convert a tuple of order (key,value) into a dictionary.**

168

P P SAVANI UNIVERSITY

School of Engineering

EXPLICIT TYPE CONVERSION DEMO



169

P P SAVANI UNIVERSITY

School of Engineering

PYTHON OPERATORS

Operators

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

Types of Operators

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

170

P P SAVANI UNIVERSITY

School of Engineering

ARITHMETIC OPERATORS

Operators	Description	Example
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns exponential power	X**y

171

P P SAVANI UNIVERSITY

School of Engineering

MERGE THE LISTS BY USING + OPERATOR

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> samplelist1=[11,33,55,77]
>>> samplelist2=[22,44,66]
>>> merged_sample=samplelist1+samplelist2 #Merge the Lists
>>> print(merged_sample)
[11, 33, 55, 77, 22, 44, 66]
>>>
```

172

P P SAVANI UNIVERSITY

School of Engineering

COMPARISON OPERATORS

Operators	Description	Example
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to True if the left operand is greater than or equal to the right	x >= y
<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y

P P SAVANI UNIVERSITY

School of Engineering

ASSIGNMENT OPERATORS

Operators	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b used for value assignment
+=	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%=	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Performs exponential (power) calculation on operators and assign value to the left operand	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

P P SAVANI UNIVERSITY

School of Engineering

LOGICAL OPERATORS

Operators	Description	Example
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

175

P P SAVANI UNIVERSITY

School of Engineering

BITWISE OPERATORS

Operators	Description	Example
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

176

P P SAVANI UNIVERSITY

School of Engineering

MEMBERSHIP OPERATORS

- Python's membership operators **test for membership** in a sequence, such as strings, lists, or tuples.

Operators	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

177

P P SAVANI UNIVERSITY

School of Engineering

MEMBERSHIP OPERATORS

IDLE Shell .

File Edit Shell Debug Options Window Help

```
Python :    : (tags/v3.10.5:f377153, ...    ...    [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> students=["Yatharth","Yuvraj","Yatri"]
>>> print("Yatharth" in students) #Check whether Yatharth in the (part of) List
True
>>> # in Operator
>>> print("yatharth" in students)
False
>>> print("Riya" not in students) # not in [if Riya is part of students List]
True
>>>
```

178

P P SAVANI UNIVERSITY

School of Engineering

IDENTITY OPERATORS

- Identity operators are used to **check if two values are located on the same part of the memory**. Two variables that are equal do not imply that they are identical.

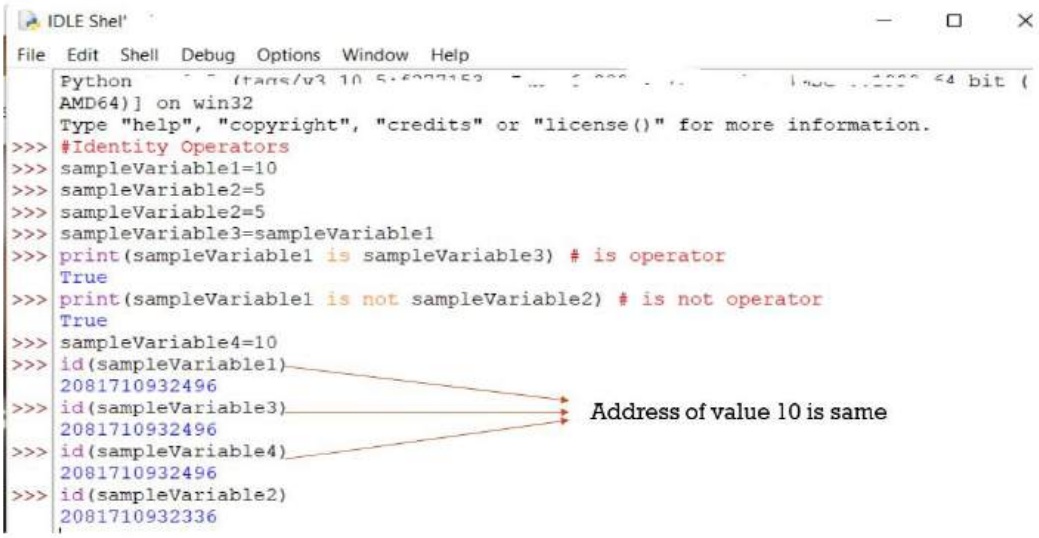
Operators	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

179

P P SAVANI UNIVERSITY

School of Engineering

IDENTITY OPERATORS



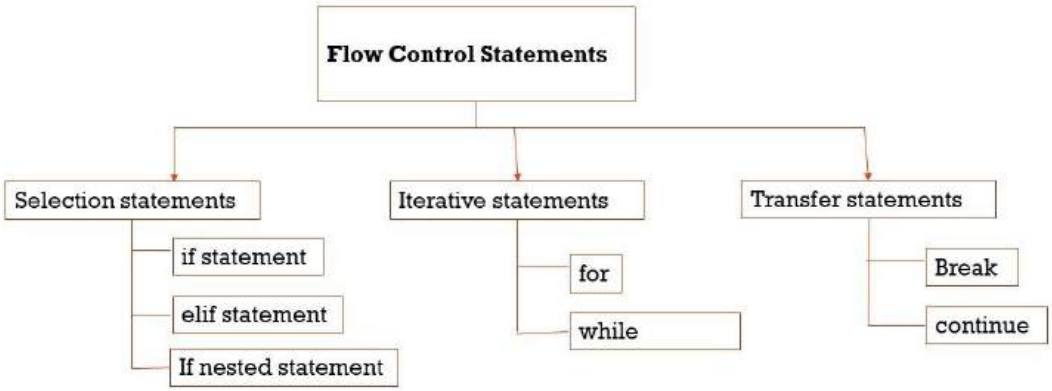
180

P P SAVANI UNIVERSITY

School of Engineering

FLOW CONTROL STATEMENTS

Flow control statements used to **interrupt the execution flow** and determines which statement to be executed next and jumping to a statement different from the successive one in sequence of program.



181

P P SAVANI UNIVERSITY

School of Engineering

FLOW CONTROL STATEMENTS

Flow Control Statements

If , elif, If Nested

While Loop

For Loop

Continue

Break

Note : There is no concept of Switch, Go to and do while statement.

182

P P SAVANI UNIVERSITY

School of Engineering

IF STATEMENT

Syntax :

```
if condition :      #with or without parentheses
        statement1  #all statement with same indentation (under if )
        statement2
        statement3
Statement4      # Not in if statement
```

Note : if condition is true then Statements-1,2,3 will get executed.

183

P P SAVANI UNIVERSITY

School of Engineering

IF STATEMENT

```
File  Edit  Format  Run  Options  Window  Help
#if Statement
university=input("Please enter your University Name  \t")
if university=="PPSU":
    print("Hello PPSU Students..")
print("Have a Knowlagable Day..!!")
```

```
IDLE
File  Edit  Shell  Debug  Options  Window  Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/ifDemo.py ==========
Please enter your University Name       PPSU
Hello PPSU Students..
Have a Knowlagable Day..!!
>>>
========== RESTART: D:/Python Material/Python Programs/ifDemo.py ==========
Please enter your University Name       GTU
Have a Knowlagable Day..!!
>>>
```

If condition is true

If condition is false

184

P P SAVANI UNIVERSITY

School of Engineering

IF-ELSE STATEMENT

Syntax :

if condition:

Action- statements-1

else:

Action – statements-2

Note : if condition is true then Action-statements-1 will get executed , if false the Action-statements-2 will get executed.

185

P P SAVANI UNIVERSITY

School of Engineering

IF-ELSE STATEMENT

ifelseDemo.py - D:/Python Material/Python Programs/ifelseDemo.py

File Edit Format Run Options Window Help

```
#if Statement
university=input("Please enter your University Name  \t")
if university=="PPSU":
    print("Hello PPSU Students..")
else:
    print("Hello Students..")
print("Have a Knowlagable Day..!!")
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/ifelseDemo.py ==========
Please enter your University Name        GTU
Hello Students..
Have a Knowlagable Day..!!
>>>
========== RESTART: D:/Python Material/Python Programs/ifelseDemo.py ==========
Please enter your University Name        PPSU
Hello PPSU Students..
Have a Knowlagable Day..!!
```

If condition is false

If condition is true

186

P P SAVANI UNIVERSITY

School of Engineering

IF-ELIF-ELSE STATEMENT

Syntax :

if condition1:

Action- statements-1

elif condition2:

Action – statements-2

elif condition3:

Action – statements-3

else:

default action statements

Note : Based on the condition Action statements will get executed ,

if no condition matched then else default action block will get executed.

187

P P SAVANI UNIVERSITY

School of Engineering

IF-ELIF-ELSE STATEMENT

nestedifDemo.py - D:/Python Material/Python Programs/nestedifDemo.py

File Edit Format Run Options Window Help

```
#if Statement
department=input("Please enter your Department Name  \t")
if department=="CE":
    print("Hello Computer Engineering Students..")
elif department=="IT":
    print("Hello IT Students..")
elif department=="CV":
    print("Hello Civil Engineering Students..")
elif department=="CH":
    print("Hello Chemical Engineering Students..")
else:
    print("Hello Engineering Students..")
print("Have a Knowlagable Day..!!")
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, ...
AMD64)
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/nestedifDemo.py ==========
Please enter your Department Name        CE
Hello Computer Engineering Students..
Have a Knowlagable Day..!!
>>>
========== RESTART: D:/Python Material/Python Programs/nestedifDemo.py ==========
Please enter your Department Name        CSE
Hello Engineering Students..
Have a Knowlagable Day..!!
```

Condition matched

Condition not matched

188

P P SAVANI UNIVERSITY

School of Engineering

EXCERCISE

- Write a Python Script to check whether the number is Positive or not.
- Write a Python Script to check whether the number is Positive or Negative.
- Write a Python Script to check whether the number is Positive, Negative or Zero.
- Write a Python script to find maximum number among two numbers.
- Write a Python script to find maximum number among three numbers.
- Write a Python script to create calculator that performs arithmetic operation on two numbers.

189

P P SAVANI UNIVERSITY

School of Engineering

NUMBER IS POSITIVE, NEGATIVE OR ZERO

```
n = int(input("Enter number: "));
if (n>0):
        print("Number is Positive")
elif (n<0):
        print("Number is Negative")
else:
        print("Number is Zero")
```

190

P P SAVANI UNIVERSITY

School of Engineering

FIND MAXIMUM AMONG TWO NUMBERS

```
n1 = int(input("Enter first number: "));
n2 = int(input("Enter second number: "));
if (n1 >= n2):
        l = n1
else:
        l = n2
print("A Largest number among the two is", l)
```

191

P P SAVANI UNIVERSITY

School of Engineering

FIND MAXIMUM AMONG THREE NUMBERS

```
n1 = int(input("Enter first number: "));
n2 = int(input("Enter second number: "));
n3 = int(input("Enter Third number: "));
if (n1 >= n2) and (n1 >= n3):
        l = n1
elif (n2 >= n3):
        l = n2
else:
        l = n3
print("A Largest number among the three is", l)
```

192

P P SAVANI UNIVERSITY

School of Engineering

BASIC CALCULATOR

CalcDemo.py - D:/Python Material/Python Programs/CalcDemo.py

File Edit Format Run Options Window Help

```
# take input from the user
choice = input("Enter choice(+,-,*,/,%,**): ")

# check if choice is one of the four options
if choice in ('+', '-', '*', '/','%','**'):
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == '+':
        print(num1, "+", num2, "=",num1+num2)

    elif choice == '-':
        print(num1, "-", num2, "=",num1-num2)

    elif choice == '*':
        print(num1, "*", num2, "=",num1*num2)

    elif choice == '/':
        print(num1, "/", num2, "=",num1/num2)

    elif choice == '%':
        print(num1, "%", num2, "=",num1%num2)

    elif choice == '**':
        print(num1, "**", num2, "=",num1**num2)
else:
    print("Invalid Choice")
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/CalcDemo.py ==========
Enter choice(+,-,*,/,%,**): *
Enter first number: 2
Enter second number: 3
2.0 * 3.0 = 6.0
>>>
========== RESTART: D:/Python Material/Python Programs/CalcDemo.py ==========
Enter choice(+,-,*,/,%,**): -
Enter first number: 1
Enter second number: 2
1.0 - 2.0 = -1.0
>>>
========== RESTART: D:/Python Material/Python Programs/CalcDemo.py ==========
Enter choice(+,-,*,/,%,**): =
Invalid Choice
>>>
```

Choice not matched

193

P P SAVANI UNIVERSITY

School of Engineering

FOR LOOP

- The for loop statement has **variable iteration in a sequence(list, tuple, set, dictionary or string)** and executes statements until the loop does not reach the false condition.

- If we want to perform any activity on each element present in the given collection then we can use for loop.

- **Syntax**

For x in Collection:
    Activity Statements

194

P P SAVANI UNIVERSITY

School of Engineering

FOR LOOP

```
forDemo1.py - D:/Python Material/Python Programs/forDemo1.py (
File  Edit  Format  Run  Options  Window  Help
department="Computer Engineering"
for character in department:
    print (character)
```

```
IDLE Shell 3.10.5
File  Edit  Shell  Debug  Options  Window  Help
Python ... win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/forDemo1.py ==========
C
o
m
p
u
t
e
r

E
n
g
i
n
e
e
r
i
n
g
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

FOR LOOP (PRINT INDEX)

```
forDemo2.py - D:/Python Material/Python Programs/forDemo2.py (
File  Edit  Format  Run  Options  Window  Help
department="Computer Engineering"
i=0
for character in department:
    print ("character {} is present at index {}".format(character,i))
    i=i+1
```

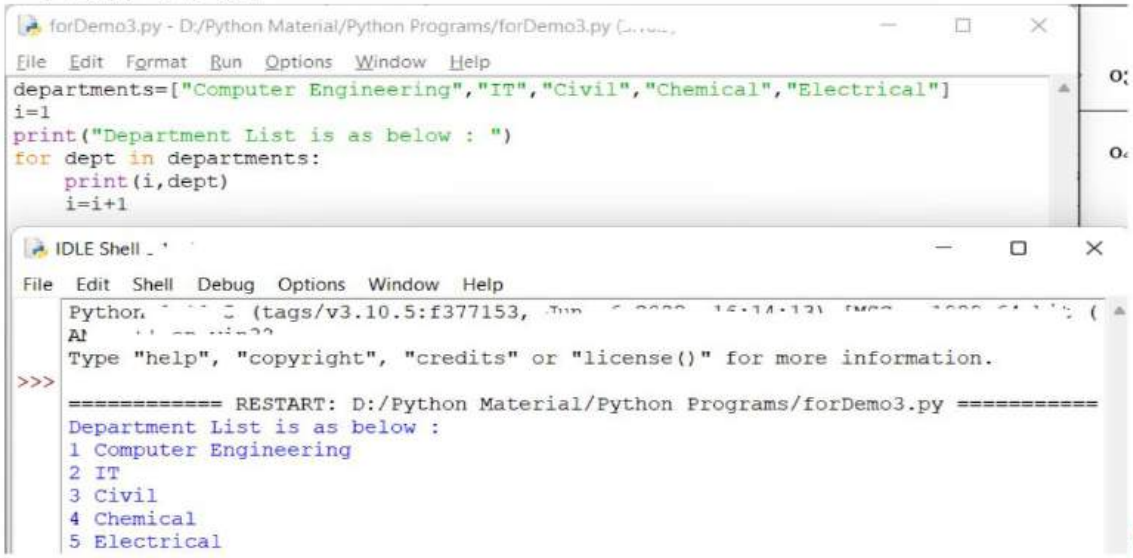
```
IDLE Shell ...
File  Edit  Shell  Debug  Options  Window  Help
Python ... 16:14:13) [MSC v.1929 64 bit (
AMD ... win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/forDemo2.py ==========
character C is present at index 0
character o is present at index 1
character m is present at index 2
character p is present at index 3
character u is present at index 4
character t is present at index 5
character e is present at index 6
character r is present at index 7
character   is present at index 8
character E is present at index 9
character n is present at index 10
character g is present at index 11
character i is present at index 12
character n is present at index 13
character e is present at index 14
character e is present at index 15
character r is present at index 16
character i is present at index 17
character n is present at index 18
character g is present at index 19
>>>
```

→ Print location along with member character

P P SAVANI UNIVERSITY

School of Engineering

FOR LOOP



P P SAVANI UNIVERSITY

School of Engineering

RANGE()

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax:

range(start, stop, step)

Where,

Start : Optional. An integer number specifying at which position to start. Default is 0

Stop : Required. An integer number specifying at which position to stop (not included).

Step : Optional. An integer number specifying the incrementation. Default is 1.

P P SAVANI UNIVERSITY

School of Engineering

RANGE()

- The range() is one of the data types in python.
- The range() only works with integers and whole numbers.
- Arguments passed in the range cannot be any other data type other than integer datatype.
- All the arguments passed can be either positive or negative.
- Step argument value cannot be zero otherwise it will throw a valueError Exception.
- Range elements can be accessed using Index, just like another datatype.

199

P P SAVANI UNIVERSITY

School of Engineering

RANGE EXAMPLE

IDLE Shell _.

File Edit Shell Debug Options Window Help

```
Python :       (tags/v3.10.5:f377153,        :  , 16:14:13) [MSC
Type "help", "copyright", "credits" or "license()" for more infor
>>> sampleRange=range(10)
>>> print(sampleRange)
range(0, 10)
>>> # range created above
>>>
>>>
>>> sampleList=list(sampleRange) #Range converted to List
>>> print(sampleList)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

200

P P SAVANI UNIVERSITY

School of Engineering

RANGE EXAMPLE

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python '     (tags/v3.10.5:f377153,              16:14:12) [MSC v.1929
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information
>>> #Print 1 to 10 numbers
>>> for i in range(1,11):
...     print(i)
...

1
2
3
4
5
6
7
8
9
10
>>>
```

Print 1 to 10 numbers.

201

P P SAVANI UNIVERSITY

School of Engineering

RANGE EXAMPLE

```
>>> for i in range(2,11,2):
...     print(i)
...

2
4
6
8
10
```

Print Even numbers from 1 to 10

202

P P SAVANI UNIVERSITY

School of Engineering

RANGE EXAMPLE

sumDemo.py - D:/Python Material/Python Programs/sumDemo.py

File Edit Format Run Options Window Help

```
sum=0
for i in range(0,11):
    sum=sum+i

print("Sum of 1 to 10= ", sum)
55
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153, [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/sumDemo.py ==========
Sum of 1 to 10=  55
```

Sum of first 10 numbers

203

P P SAVANI UNIVERSITY

School of Engineering

RANGE()

*IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153,
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>>
========== RESTART: D:/Python Material/Python Programs/sumDem
Sum of 1 to 10=  55
>>>
>>>
>>>
>>> for i in range(10,1,-1):
...     print(i)
...
10
9
8
7
6
5
4
3
2
>>> #Print reverse sequence
```

204

P P SAVANI UNIVERSITY

School of Engineering

RANGE()

displayDemo.py - D:/Python Material/Python Programs/displayDemo.py

File Edit Format Run Options Window Help

```
departments=["Computer Engineering","IT","Civil","Chemical","Electrical"]

for i in range(len(departments)):
    print(i+1,departments[i])
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python .      (tags/v3.10.5:f377153, ...  ..., 16:14:13) [MSC v.1929 64 b
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/displayDemo.py ======
1 Computer Engineering
2 IT
3 Civil
4 Chemical
5 Electrical
```

205

P P SAVANI UNIVERSITY

School of Engineering

FOR EXAMPLE

*IDLE Shell

File Edit Shell Debug Options Window Help

```
Python        (tags/v3.10.5:f377153, ...                 ...  [MS
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>>
============ RESTART: D:/Python Material/Python Programs/sumDem
Sum of 1 to 10=  55
>>>
>>>
>>>
>>> for i in range(10,1,-1):
...     print(i)
...
10
9
8
7
6
5
4
3
2
>>> #Print reverse sequence
```

204

P P SAVANI UNIVERSITY

School of Engineering

RANGE()

displayDemo.py - D:/Python Material/Python Programs/displayDemo.py

File Edit Format Run Options Window Help

```
departments=["Computer Engineering","IT","Civil","Chemical","Electrical"]

for i in range(len(departments)):
    print(i+1,departments[i])
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python .     . (tags/v3.10.5:f377153, .  .  .  , 16:14:13) [MSC v.1929 64 b
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/displayDemo.py ======
1 Computer Engineering
2 IT
3 Civil
4 Chemical
5 Electrical
```

205

P P SAVANI UNIVERSITY

School of Engineering

WHILE STATEMENT [ITERATIVE STATEMENT]

While loop we can execute a set of statements as long as a conditions are met.

Syntax:

While expression:

    statements

206

P P SAVANI UNIVERSITY

School of Engineering

WHILE STATEMENT

whileDemo.py - D:/Python Material/Python Programs/whileDemo.py

File Edit Format Run Options Window Help

```
#Perform the addition of first 10 elements
i=1
sum=0
while i<=10:
    sum=sum+i
    i=i+1
print("Addition of first 10  numbers are : ",sum)
```

IDLE Shell .

File Edit Shell Debug Options Window Help

```
Python  (tags/v3.10.5:f377153,
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/whileDemo.py ==========
Addition of first 10  numbers are :  55
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

BREAK STATEMENT (TRANSFER STATEMENT)

The break statement is used to terminate the loop or statement in which it is present.

breakDemo.py - D:\Python Material\Python Programs\breakDemo.py

File Edit Format Run Options Window Help

```
#Search a number from a list
numberList=[11,55,77,33,66]
i=0
for number in numberList:
    if number==77:
        print("Number at position : ",i)
        break
    i=i+1
```

After finding the Location of element ,loop ends

IDLE Shell .

File Edit Shell Debug Options Window Help

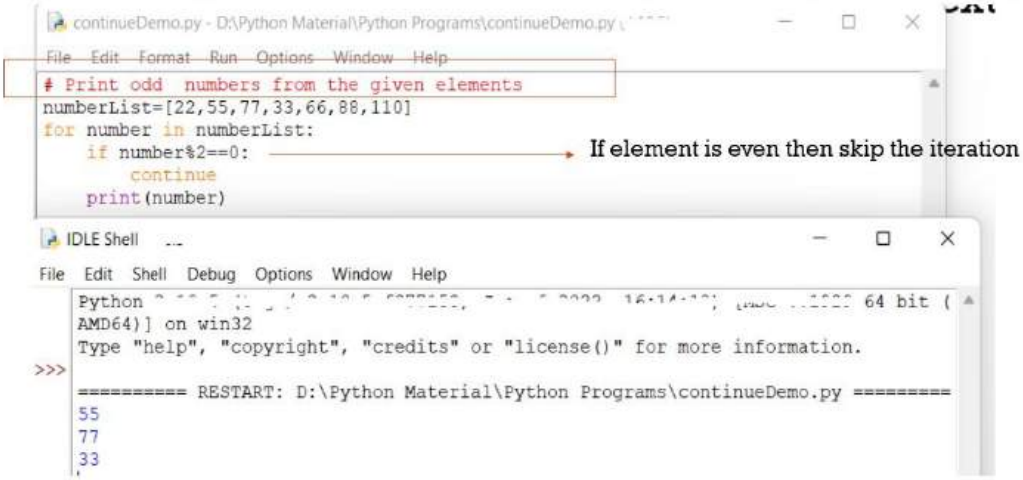
```
Python  (tags/v3.10.5:f377153,  [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\breakDemo.py ==========
Number at position :  2
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

CONTINUE STATEMENT (TRANSFER STATEMENT)

Continue statement forces to execute the next iteration of the loop.



P P SAVANI UNIVERSITY

School of Engineering

EVAL()

- It is **evaluates** the provided input value (python expression) dynamically which are provided as string and converted to corresponding type automatically.
- It is also be used as an alternative to type cast functions.
- Eval() is unsecure as it accept to querystrings.

Syntax:

- eval(expression, globals=None, locals=None)

Expression – It is the python expression passed onto the method.

globals – A dictionary of available global methods and variables.

locals – A dictionary of available local methods and variables.

P P SAVANI UNIVERSITY

School of Engineering

EVAL() DEMO

*IDLE Shell .

File Edit Shell Debug Options Window Help

```
Python ' ' ' (tags/v3.10.5:f377153, .          , 16 '     [MSC v.1929 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> eval("1+2*3/2") # Evaluates the Expression
4.0
>>>
```

211

P P SAVANI UNIVERSITY

School of Engineering

EVAL() DEMO

evalDemo.py - D:/Python Material/Python Programs/evalDemo.py (3.10.5)

File Edit Format Run Options Window Help

```
#Eval is alternative to type cast functions

value=eval(input("Enter Anything"))
print("You have entered", type(value) , "type of value")
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything 5
You have entered <class 'int'> type of value
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything 5.5
You have entered <class 'float'> type of value
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything "Computer ENgineering"
You have entered <class 'str'> type of value
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything True
You have entered <class 'bool'> type of value
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything [11,22,33]
You have entered <class 'list'> type of value
>>>
========== RESTART: D:/Python Material/Python Programs/evalDemo.py ==========
Enter Anything ("Yatharth","Yatri")
You have entered <class 'tuple'> type of value
>>>
```

212

P P SAVANI UNIVERSITY

EVAL() DEMO

School of Engineering

```
>>> x=5
>>> y=10
>>> eval("x+y",{},{}) #Globals and Locals are empty
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    eval("x+y",{},{}) #Globals and Locals are empty
  File "<string>", line 1, in <module>
NameError: name 'x' is not defined
>>> eval("x+y") # Variables value taken directly
15
>>> eval("x+y",{"x":1,"y":2})
3
>>> #Local Variables taken above
```


P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo1.py - D:/Python Material/Python Programs/inputDemo1.py

File Edit Format Run Options Window Help

```
#Perform Addition of three numbers
number1,number2,number3=int(input("Enter no1 ")),int(input("Enter no2 ")),int(in

ans=number1+number2+number3

print(number1,"+", number2,"+",number3,"=",ans)
```

Multiple Variables

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153, [illegible]) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyri[illegible]" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/inputDemo1.py ==========
Enter no1 11
Enter no2 22
Enter no3 33
11 + 22 + 33 = 66
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo2.py - D:\Python Material\Python Programs\inputDemo2.py

File Edit Format Run Options Window Help

```
#Perform Addition of three numbers
number1,number2,number3=input("Enter three numbers").split()
ans=int(number1)+int(number2)+int(number3)
print(number1,"+",number2,"+",number3,"=",ans)
```

Split the user input in multiple variables

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153,
AMD64)] on
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\inputDemo2.py ==========
Enter three numbers 11 22 33
11 + 22 + 33 = 66
```

215

P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo3.py - D:/Python Material/Python Programs/inputDemo3.py

File Edit Format Run Options Window Help

```
#Perform Addition of three numbers
number1,number2,number3=input("Enter three numbers").split(",")
ans=int(number1)+int(number2)+int(number3)
print(number1,"+",number2,"+",number3,"=",ans)
```

Split the user input in multiple variables

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153,
AMD64)] on wi
Type "help", "copyright",
>>>
========== RESTART: D:/Python Material/Python Programs/inputDemo3.py ==========
Enter three numbers 11,22,33
 11 + 22 + 33 = 66
```

216

P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo4.py - D:/Python Material/Python Programs/inputDemo4.py

File Edit Format Run Options Window Help

```
#Take list of numbers from User
numberList=input("Enter numbers separated by space\n").split()

print("List is below \n ", numberList)
```

List will be created

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python . . (tags/v3.10.5:f377153 . . . 1929, . . . 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/inputDemo4.py ==========
Enter numbers separated by space
11 22 33
List is below
  ['11', '22', '33']
```


P P SAVANI UNIVERSITY

School of Engineering

MAP()

The map () : function executes a specified function for each item in an iterable.

It returns the map object.

Syntax:

map(function, iterables)

Where,

Function : Required. The function to execute for each item.

Iterable : Required. A sequence, collection or an iterator object. You can send as many iterables as you like, just make sure the function has one parameter for each iterable.

P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo5.py - D:/Python Material/Python Programs/inputDemo5.py (

File Edit Format Run Options Window Help

```
#Perform the addition of List elements
numberList=list(map(int,input("Enter numbers separated by space\n").split()))
print("List is below \n ", numberList)
print("Sum of List elements : ",sum(numberList))
```

Map each element to int type and create a List

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/inputDemo5.py ==========
Enter numbers separated by space
11 22 33
List is below
  [11, 22, 33]
Sum of List elements :  66
```

219

P P SAVANI UNIVERSITY

School of Engineering

TAKE MULTIPLE VALUES FROM USER

inputDemo5.py - D:/Python Material/Python Programs/inputDemo5.py (

File Edit Format Run Options Window Help

```
#Perform the addition of List elements
# Consider first three number
numberList=list(map(int,input("Enter three numbers separated by space\n").split()))[:3]
print("List is below \n ", numberList)
print("Sum of List elements : ",sum(numberList))
```

First three elements to be taken from user

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/inputDemo5.py ==========
Enter three numbers separated by space
11 22 33 44 55
List is below
  [11, 22, 33]
Sum of List elements :  66
```

220

P P SAVANI UNIVERSITY

School of Engineering

ARRAY

Array is the **collection of elements of same type.**

Syntax

```
from array import *
numbers=array(typecode,[Elements])
numbers
```

Or

```
import array as arr
arr.array(typecode,[Elements])
```

221

P P SAVANI UNIVERSITY

School of Engineering

ARRAY TYPE CODE

Type Code	C Type	Python Type	Minimum size in Bytes
'b'	Signed Char	int	1
'B'	unsigned Char	int	1
'u'	Py_UNICODE	unicode character	2
'h'	Signed short	int	2
'H'	unsigned short	int	2
'i'	Signed Int	int	2

222

P P SAVANI UNIVERSITY

School of Engineering

ARRAY TYPE CODE

Type Code	C Type	Python Type	Minimum size in Bytes
'I'	unsigned int	int	2
'l'	Singed long	int	4
'L'	unsinged long	int	4
'f'	float	float	4
'd'	double	float	8

223

P P SAVANI UNIVERSITY

School of Engineering

ARRAY

```
arrayDemo.py - D:/Python Material/Python Programs/arrayDemo.py (3.10.5)
File  Edit  Format  Run  Options  Window  Help
from array import *
values=array('i',[11,55,88,44,22])  ----> Integer array created
print(values)
```

```
IDLE Shell 3.10.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/arrayDemo.py ==========
array('i', [11, 55, 88, 44, 22])
```

224

P P SAVANI UNIVERSITY

School of Engineering

ARRAY

arrayDemo.py - D:/Python Material/Python Programs/arrayDemo.py (3.10.5)

File Edit Format Run Options Window Help

```
from array import *
values=array('i',[11.5,55,88,44,22])
print(values)
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/arrayDemo.py ==========
Traceback (most recent call last):
  File "D:/Python Material/Python Programs/arrayDemo.py", line 2, in <module>
    values=array('i',[11.5,55,88,44,22])
TypeError: 'float' object cannot be interpreted as an integer
>>>
```

Error as 1 float element given

P P SAVANI UNIVERSITY

School of Engineering

ARRAY

arrayDemo.py - D:/Python Material/Python Programs/arrayDemo.py

File Edit Format Run Options Window Help

```
from array import *
values=array('i',[11,55,88,44,22])
print(values)
print(values.buffer_info()) #Print location and number of elements
print(values.typecode) #Print typecode
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/arrayDemo.py ==========
array('i', [11, 55, 88, 44, 22])
(1695631682448, 5)
i
```

P P SAVANI UNIVERSITY

School of Engineering

LOOPING ARRAY ELEMENT

arrayDemo2.py - D:/Python Material/Python Programs/arrayDemo2.py

File Edit Format Run Options Window Help

```
from array import *
values=array('i',[11,55,88,44,22])

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

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [...] 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/arrayDemo2.py ==========
11
55
88
44
22
```

227

P P SAVANI UNIVERSITY

School of Engineering

ARRAY FUNCTIONS

arrayDemo3.py - D:/Python Material/Python Programs/arrayDemo3.py

File Edit Format Run Options Window Help

```
from array import *
values=array('i',[11,55,88,44,22,55])

values.append(33) #Append element in Array
print(values)
print("55 comes ", values.count(55),"times") #Count the specified element presen
values.extend([66,99]) #Extend the array with elements
print(values)
values.insert(1,110) #Insert element at a particular position
print(values)
print(values.index(55)) #Find Position of an element
values.pop(2) #Removes element at specified position
print(values)
values.reverse() #Reverse Array ELements
print(values)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [...] 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/arrayDemo3.py ==========
array('i', [11, 55, 88, 44, 22, 55, 33])
55 comes  2 times
array('i', [11, 55, 88, 44, 22, 55, 33, 66, 99])
array('i', [11, 110, 55, 88, 44, 22, 55, 33, 66, 99])
2
array('i', [11, 110, 88, 44, 22, 55, 33, 66, 99])
array('i', [99, 66, 33, 55, 22, 44, 88, 110, 11])
>>>
```

228

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS IN PYTHON

Barkha Wadhvani
Assistant Professor
P.P. Savani University
School Of Engineering

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION OF FUNCTIONS

- Functions are useful for breaking up a large program to make it easier to read and maintain.
- They are also useful if you find yourself writing the same code at several different points in your program.
- You can put that code in a function and call the function whenever you want to execute that code.

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS CREATION

Functions are defined with the **def** statement.

The statement ends with a colon, and the code that is part of the function is indented below the def statement.

Syntax:

def functionName:

    statements

Barkha Wadhvani

3

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS DEMO WITH NO ARGUMENT

*functionDemo1.py - D:/Python Material/Python Programs/functionDemo1.py

File Edit Format Run Options Window Help

```
#Function creation
def greetings():
    print("Hello Engineers")

#Calling function
greetings()

print("Good Luck")
```

IDLE She

File Edit Shell Debug Options Window Help

```
Pyth  3.10.9 (tags/v3.10  23/7152.  v.1929 64 bit (
AMD64)] on
Type "help", "copyrig  "  "  s" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo1.py ==========
Hello Engineers
Good Luck
>>>
```

Barkha Wadhvani

4

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS DEMO WITH ONE ARGUMENT

functionDemo3.py - D:/Python Material/Python Programs/functionDemo3.py

File Edit Format Run Options Window Help

```
#Function to print function N times
def greetings(no):
    print('Hello Engineers\n ' *no)


n=int(input("How many times you want to print the message??\n"))
#Call Function
greetings(n)
```

IDLE Shell .

File Edit Shell Debug Options Window Help

```
Python (tags/ 16:14:13) bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo3.py ==========
How many times you want to print the message??
5
Hello Engineers
 Hello Engineers
 Hello Engineers
 Hello Engineers
 Hello Engineers
```

Barkha Wadhvani

5

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS WITH MULTIPLE ARGUMENTS)

functionDemo4.py - D:/Python Material/Python Programs/functionDemo4.py

File Edit Format Run Options Window Help

```
#Function to print a function N times
def greetings(message,no):
    print(message * no)

msg=input("What would you like to print?\n")
n=int(input("How many times you want to print the message??\n"))
#Call Function
greetings(msg,n)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo4.py ==========
What would you like to print?
Hello Enginees
How many times you want to print the message??
5
Hello EngineesHello EngineesHello EngineesHello EngineesHello Enginees
>>>
```

Barkha Wadhvani

6

P P SAVANI UNIVERSITY

School of Engineering

FUNCTIONS DEMO TO DRAW SQUARE

functionDemo2.py - D:\Python Material\Python Programs\functionDemo2.py

File Edit Format Run Options Window Help

```
#Function creation
def draw_square():
    print("*" * 15)
    print("*", " " * 11, "*")
    print("*", " " * 11, "*")
    print("*" * 15)

#Call Function
draw_square()
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\functionDemo2.py ==========
***************
*             *
*             *
***************
```

Barkha Wadhvani 7

P P SAVANI UNIVERSITY

School of Engineering

FUNCTION WITH RETURN

functionDemo5.py - D:/Python Material/Python Programs/functionDemo5.py

File Edit Format Run Options Window Help

```
#Function to convert Celsius to Fahrenheit
def convert(t):
    return t*9/5+32 #return send function result back to caller

#Call Function
print("Temperaure in Fahrenheit : ", convert(40))
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo5.py ==========
Temperaure in Fahrenheit :  104.0
>>>
```

Barkha Wadhvani 8

P P SAVANI UNIVERSITY

School of Engineering

DEFAULT ARGUMENTS

functionDemo6.py - D:/Python Material/Python Programs/functionDemo6.py (

File Edit Format Run Options Window Help

```
#Function for Greeting Message
def greetings(what,whom='Engineers'):
    print(what,"\t",whom)


#Call Function
greetings('Hi') #Second Argument taken as default value
greetings("Hi","Computer Engineers") # No default value considered
```

If no arguments provided for whom then 'Engineers' value will be taken by default

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo6.py ==========
Hi        Engineers
Hi        Computer Engineers
```

Barkha Wadhvani

9

P P SAVANI UNIVERSITY

School of Engineering

ARBITRARY ARGUMENTS

- Sometimes, **we do not know in advance the number of arguments** that will be passed into a function.
- Python allows us to handle this kind of situation through function calls with an arbitrary number of arguments.
- In the function definition, **we use an asterisk (*) before the parameter name** to denote this kind of argument we have called the function with multiple arguments.
- These arguments get **wrapped up into a tuple** before being passed into the function.
- Inside the function, we use a for loop to retrieve all the arguments back.

Barkha Wadhvani

10

P P SAVANI UNIVERSITY

School of Engineering

ARBITRARY ARGUMENTS

functionDemo7.py - D:\Python Material\Python Programs\functionDemo7.py

File Edit Format Run Options Window Help

```
#Function for Student Greetings
def greeting(*studentname):

    for name in studentname:
        print("Hello",name)


#Call Function
greeting('Yatharth','Yatri','Yuvraj','Riya','Khushi','Sahil')
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\functionDemo7.py ==========
Hello Yatharth
Hello Yatri
Hello Yuvraj
Hello Riya
Hello Khushi
Hello Sahil
>>>
```

Barkha Wadhvani 11

P P SAVANI UNIVERSITY

School of Engineering

PYTHON RECURSION

- Recursion is the process of defining something in terms of itself.

functionDemo8.py - D:/Python Material/Python Programs/functionDemo8.py

File Edit Format Run Options Window Help

```
#Function for Student Greetings
def factorial(x):
    if x==1:
        return 1
    else:
        return (x * factorial(x-1))

number=int(input("Please enter the number whose factorial you want to calculate\
#Call Function
print(factorial(number))
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo8.py ==========
Please enter the number whose factorial you want to calculate
5
120
```

Barkha Wadhvani 12

P P SAVANI UNIVERSITY

School of Engineering

MAP() USE IN FUNCTIONS

functionDemo9.py - D:/Python Material/Python Programs/functionDemo9.py

File Edit Format Run Options Window Help

```
#Function to calculate the square of each number in list
def squarenumbers(x):
    return x*x
numberList=[11,22,33,44,55]
#Call Function for task
s=map(squarenumbers,numberList)
print(s)
s=print(list(s))
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [MSC v.1929 64
AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo9.py ==========
<map object at 0x000002010B742020>
[121, 484, 1089, 1936, 3025]
>>>
```

Barkha Wadhvani

13

P P SAVANI UNIVERSITY

School of Engineering

LAMBDA PYTHON

- A lambda function is a **small anonymous function** that is defined without a name.
- A lambda function **can take any number of arguments, but can only have one expression.**
- The expression is executed and the result is returned.

Syntax:

- lambda arguments : expression

Barkha Wadhvani

14

P P SAVANI UNIVERSITY

School of Engineering

USE OF LAMBDA

- We use lambda functions when we require a **nameless function for a short period of time.**
- In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments).
- Lambda functions are used along with built-in functions like filter(), map() etc.

Barkha Wadhvani 15

P P SAVANI UNIVERSITY

School of Engineering

LAMBDA PYTHON

functionDemo10.py - D:/Python Material/Python Programs/functionDemo10.py

File Edit Format Run Options Window Help

```
#Function to calculate the power of 2 of a number
power=lambda number : number * 2

#Call Function for task
result=power(2)
print("Result :" ,result)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... bit (
AMD64)] o... win32
Type "help", "copyright", "credits" or ... more information.
>>>
========= RESTART: D:/Python Material/Python Programs/functionDemo10.py ========
Result : 4
```

Barkha Wadhvani 16

P P SAVANI UNIVERSITY

School of Engineering

LAMBDA PYTHON WITH MULTIPLE ARGUMENTS

functionDemo10.py - D:/Python Material/Python Programs/functionDemo10.py (...

File Edit Format Run Options Window Help

```
#Function to calculate the adddition of three numbers
addition=lambda number1,number2,number3 : number1+number2+number3

#Call Function for task
result=addition(11,22,33)
print("Result :" ,result)
```

IDLE Shell ...

File Edit Shell Debug Options Window Help

```
Python ... bit (
AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo10.py ========
Result : 66
>>>
```

Barkha Wadhvani

17

P P SAVANI UNIVERSITY

School of Engineering

LAMBDA AND MAP() USE IN FUNCTION

functionDemo12.py - D:/Python Material/Python Programs/functionDemo12.py

File Edit Format Run Options Window Help

```
#Function to double the each number in List
numberList=[11,22,33,44,55]
doubleList=list(map(lambda number:number*number,numberList))
print(doubleList)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... bit (
AMD64)] on win32
Type "help", "...", ... or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/functionDemo12.py ========
[121, 484, 1089, 1936, 3025]
>>>
```

Barkha Wadhvani

18

P P SAVANI UNIVERSITY

School of Engineering

FILTER()

- The filter() function returns an iterator were the items are filtered through a function to test if the item is accepted or not.

- **Syntax**

filter(function,iterable)

Parameter	Description
function	A Function to be run for each item in the iterable
iterable	The iterable to be filtered

Barkha Wadhvani 19

P P SAVANI UNIVERSITY

School of Engineering

FILTER()

filter.py - D:/Python Material/Python Programs/filter.py

File Edit Format Run Options Window Help

```
ages = [55, 2, 7, 18, 23, 32]
def Eligible(x):
  if x < 18:
    return False
  else:
    return True
adults = list(filter(Eligible, ages))

print("Eligible for Votes \n")
for x in adults:
  print(x)
```

IDLE Shell

File Edit Shell Debug Options Window Help

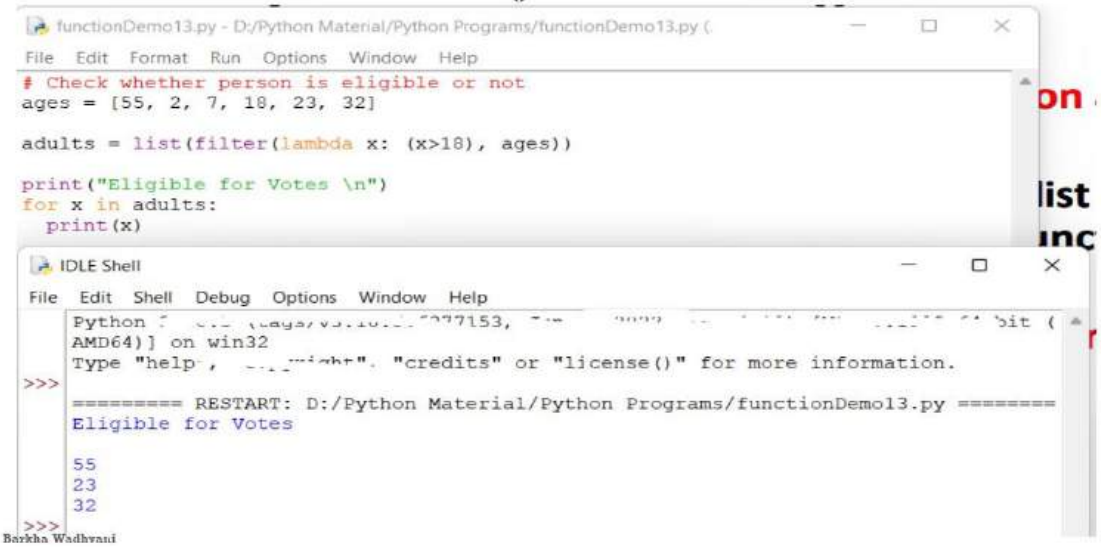
```
Python ... 64 bit (
AMD64)] on wir...
Type "help", "...right" ... "...ses" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/filter.py ==========
Eligible for Votes

55
18
23
32
>>>
```

Barkha Wadhvani 20

P P SAVANI UNIVERSITY

School of Engineering

LAMBDA AND FILTER() USE IN FUNCTION



P P SAVANI UNIVERSITY

School of Engineering

PYTHON GLOBAL VARIABLES

- In Python, a variable declared outside of the function or in global scope is known as a global variable.
- This means that a global variable can be accessed inside or outside of the function.

P P SAVANI UNIVERSITY

School of Engineering

PYTHON GLOBAL VARIABLES

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [tags/v3.10.5:f377153, ... MSC v.1929 ... bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: D:/Python Material/Python Programs/globalVariables.py ========
Inside Function (Name value is): Yatharth
Outside Function (Name value is): Yatharth
>>>
```

globalVariables.py - D:/Python Material/Python Programs/globalVariables.py

File Edit Format Run Options Window Help

```
sampleName="Yatharth"
def GetName():
    print("Inside Function (Name value is):" ,sampleName)

GetName() # Call Function

print("Outside Function (Name value is):" ,sampleName)
```

Global Variable

P P SAVANI UNIVERSITY

School of Engineering

PYTHON LOCAL VARIABLES

- A variable declared inside the function's body or in the local scope is known as a local variable.
- Accessing local variable outside the scope.

P P SAVANI UNIVERSITY

School of Engineering

PYTHON LOCAL VARIABLES

globalKeyword.py - D:/Python Material/Python Programs/globalKeyword.py

File Edit Format Run Options Window Help

```
sampleName="Yatharth"  #Global Variable
print("Address of Global Variable  ", id(sampleName))

def GetName():
    sampleName="Yatri"    #Local Variable
    print("Address of Local Variable", id(sampleName))
    print("Inside Function (Name value is):" ,sampleName)

GetName() # Call Function

print("Outside Function (Name value is):" ,sampleName)
```

Two copies will be created

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/globalKeyword.py ==========
Address of Global Variable   2769719776304
Address of Local Variable 2769719776432
Inside Function (Name value is): Yatri
Outside Function (Name value is): Yatharth
```


P P SAVANI UNIVERSITY

School of Engineering

PYTHON GLOBAL KEYWORD

- global keyword allows you to **modify the variable outside of the current scope.**
- It is used to create a global variable and make changes to the variable in a local context.
- If a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.

P P SAVANI UNIVERSITY

School of Engineering

PYTHON GLOBAL KEYWORD

globalKeyword.py - D:/Python Material/Python Programs/globalKeyword.py

File Edit Format Run Options Window Help

```
sampleName="Yatharth"  #Global Variable

def GetName():
    global sampleName   #Allow to modify out of scope
    sampleName="Yatri"
    print("Inside Function (Name value is):" ,sampleName)

GetName() # Call Function

print("Outside Function (Name value is):" ,sampleName)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/globalKeyword.py ==========
Inside Function (Name value is): Yatri
Outside Function (Name value is): Yatri
>>>
```

Barkha Wadhvani

27

P P SAVANI UNIVERSITY

School of Engineering

PYTHON MODULE

- Modules refer to a **file containing Python statements and definitions.**
- A file containing Python code, for example: example.py, is called a module, and its module name would be example.
- We use modules to **break down large programs into small manageable and organized files**. Furthermore, **modules provide reusability of code.**
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Barkha Wadhvani

28

P P SAVANI UNIVERSITY

School of Engineering

CREATE AND IMPORT PYTHON MODULE

moduleDemo.py - D:/Python Material/Python Programs/moduleDemo.py

File Edit Format Run Options Window Help

```
#Module creation
def add(number1,number2):
    result=number1+number2
    return result
```

→ **Module created**
[Add two numbers and returns the result]

moduleCallDemo.py - D:/Python Material/Python Programs/moduleCallDemo.py

File Edit Format Run Options Window Help

```
import moduleDemo
print("result :", moduleDemo.add(11,22))
```

→ **Module imported and method add() called**

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/moduleCallDemo.py ========
result : 33
>>>
```

Barkha Wadhvani

29

P P SAVANI UNIVERSITY

School of Engineering

MATH MODULE

- Python has in-built module that is used for **mathematical tasks.**

Syntax

- Import math → With package name itself you can access the different methods of a package

Or

- Import math as m → With reference name you can access the different methods of a package

Or

- From math import * → Without any reference you can access the different methods of a package

Barkha Wadhvani

31

P P SAVANI UNIVERSITY

School of Engineering

MATH MODULE

```
IDLE Shell
File  Edit  Debug  Options  Window  Help
Python ... 16:14:12) ... bit (AMD64)] ... win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> help('math')
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

        The result is between 0 and pi.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.
```

32

P P SAVANI UNIVERSITY

School of Engineering

MATH MODULE

```
mathDemo.py - D:\Python Material\Python Programs\mathDemo.py (...
File  Edit  Format  Run  Options  Window  Help
# Import math module
import math

x=int(input("Enter the number to perform operations\n"))
print("Factorial is ", math.factorial(5))
x=math.sqrt(x)
print("Sqrt of {} is ".format(x))
print("Floor value is ", math.floor(x))
print("Ceil value is", math.ceil(x))

print("Power {} ** 2 is {}".format(x,math.pow(x,2)))
```

```
IDLE Shell :
File  Edit  Shell  Debug  Options  Window  Help
Python ... tags/ ... ( ...
AMD64)] on win...
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\mathDemo.py ==========
Enter the number to perform operations
5
Factorial is  120
Sqrt of 2.23606797749979 is
Floor value is  2
Ceil value is 3
Power 2.23606797749979 ** 2 is 5.000000000000001
>>>
```

33

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

MATH MODULE

mathDemo3.py - D:/Python Material/Python Programs/mathDemo3.py

```
File  Edit  Format  Run  Options  Window  Help
# Import math module (reference variable m)
import math as m

x=int(input("Enter the number to perform operations\n"))
print("Factorial is ", m.factorial(5))
x=m.sqrt(x)
print("Sqrt of {} is ".format(x))
print("Floor value is ", m.floor(x))
print("Ceil value is", m.ceil(x))
print("Power {} ** 2 is {}".format(x,m.pow(x,2)))
```

*IDLE Shell

```
File  Edit  Shell  Debug  Options  Window  Help
Python ... [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/mathDemo3.py ==========
Enter the number to perform operations
5
Factorial is  120
Sqrt of 2.23606797749979 is
Floor value is  2
Ceil value is 3
Power 2.23606797749979 ** 2 is 5.000000000000001
>>> https://forms.gle/9t78zsppp1TTSDNbA
```

34

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

MATH MODULE

*mathDemo2.py - D:/Python Material/Python Programs/mathDemo2.py

```
File  Edit  Format  Run  Options  Window  Help
# Import whole math module so that functions can be called directlly
from math import *

x=int(input("Enter the number to perform operations\n"))
print("Factorial is ", factorial(5))
x=sqrt(x)
print("Sqrt of {} is ".format(x))
print("Floor value is ",floor(x))
print("Ceil value is",ceil(x))
print("Power {} ** 2 is {}".format(x,pow(x,2)))
```

IDLE Shell

```
File  Edit  Shell  Debug  Options  Window  Help
Python ... [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/mathDemo2.py ==========
Enter the number to perform operations
5
Factorial is  120
Sqrt of 2.23606797749979 is
Floor value is  2
Ceil value is 3
Power 2.23606797749979 ** 2 is 5.000000000000001
>>>
```

35

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

RANDOM MODULE

- Python **Random module** is an in-built module of Python which is used to generate random numbers.

Syntax:

- **Import random**

Or

- **Import random as r**

Or

- **From random import ***

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

RANDOM MODULE DEMO

randomModuleDemo1.py - D:/Python Material/Python Programs/randomModuleDemo1.py

File Edit Format Run Options Window Help

```
#import random module
import random

#Creating random Integers from given range
r1 = random.randint(5, 15)
print("Random number between 5 and 15 is ",r1)

# Get random Float number
r2= random.random()
print("Random number ",r2)
```

Random.randint(Start,End)

Generate random number from 0 to 1

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... (tags/v3.10.5:...) ...
AMD64)] on win32
Type "help", ... for more information.
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo1.py =======
Random number between 5 and 15 is  6
Random number  0.8947477501912867
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo1.py =======
Random number between 5 and 15 is  6
Random number  0.5022267202587183
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo1.py =======
Random number between 5 and 15 is  8
Random number  0.42823716478678076
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo1.py =======
Random number between 5 and 15 is  5
Random number  0.08683809800562592
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

RANDOM MODULE DEMO

randomModuleDemo2.py - D:/Python Material/Python Programs/randomModuleDemo2.py ...

File Edit Format Run Options Window Help

```
#import random module
import random

list=[11,22,33,"Yatharth","Yatri"]
print("List is :", list)

#Get random value form list, tuple or strings
print("Random number from list is : ",random.choice(list))
```

Random.choice(Group)

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo2.py =======
List is : [11, 22, 33, 'Yatharth', 'Yatri']
Random number from list is :  33
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo2.py =======
List is : [11, 22, 33, 'Yatharth', 'Yatri']
Random number from list is :  Yatri
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo2.py =======
List is : [11, 22, 33, 'Yatharth', 'Yatri']
```

Barkha Wadhvani

38

P P SAVANI UNIVERSITY

School of Engineering

RANDOM MODULE DEMO

randomModuleDemo3.py - D:/Python Material/Python Programs/randomModuleDemo3.py ...

File Edit Format Run Options Window Help

```
#import random module
import random

list=[11,22,33,"Yatharth","Yatri"]
print("List is :", list)

#Shuffle list
random.shuffle(list)
print("List after shuffle ",list) # Original list will be shuffled
```

Random.shuffle(group)

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo3.py =======
List is : [11, 22, 33, 'Yatharth', 'Yatri']
List after shuffle  ['Yatharth', 33, 11, 22, 'Yatri']
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo3.py =======
List is : [11, 22, 33, 'Yatharth', 'Yatri']
List after shuffle  ['Yatharth', 'Yatri', 33, 11, 22]
>>>
```

Barkha Wadhvani

39

P P SAVANI UNIVERSITY

School of Engineering

SAMPLE()

```
randomModuleDemo5.py - D:/Python Material/Python Programs/randomModuleDemo5.py ...
File  Edit  Format  Run  Options  Window  Help
from random import *
myGroup=[11, 22, "Yatharth", "Yatri", "Yuvraj", True]

print(sample(myGroup,3))
```

→ Returns N random elements from specified Group

```
IDLE Shell
File  Edit  Shell  Debug  Options  Window  Help
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:/Python Material/Python Programs/randomModuleDemo5.py =======
[22, 'Yatri', True]
```


P P SAVANI UNIVERSITY

School of Engineering

SEED IN RANDOM MODULE

- Seed(value) is used to save the state of a random function, so that it can generate same random numbers on multiple executions of the code.

```
seedDemo.py - D:/Python Material/Python Programs/seedDemo.py (...
File  Edit  Format  Run  Options  Window  Help
import random

#Generate Random numbers
print("Different numbers")
for i in range(5):
    # Generated random number will be between 1 to 100.
    print(random.randint(1, 100))

# Generate same numbers
print("Same numbers")
for i in range(5):
    # Any number can be used in place of '1'.
    random.seed(1)

    # Generated random number will be between 1 to 100.
    print(random.randint(1, 100))
```

```
========== RESTART: D:/Python Ma
Different numbers
77
14
86
5
80
Same numbers
18
18
18
18
18
```

Output-1

```
>>>
Different numbers
12
65
16
27
28
Same numbers
18
18
18
18
18
```

Output-2

```
Different numbers
57
46
47
52
81
Same numbers
18
18
18
18
```

Output-3

P P SAVANI UNIVERSITY

School of Engineering

PACKAGES IN PYTHON

- Packages are way to organize your code to manageable modules or files.
- A Python module may contain several classes, functions, variables, etc. whereas a Python package can contains several module.

P P SAVANI UNIVERSITY

School of Engineering

NUMPY

- **NumPy** is the fundamental package for **scientific computing in python.**
- **NumPy** stands for **Numerical Python**. It is open-source and we can use it freely.
- **NumPy** is a python library that is **used for working with arrays** [single, double, or multiple dimensional].
- **Python NumPy** also contains **random number generators**. It also has functions for working in the **domain of linear algebra, Fourier transforms, and matrices,** etc.

P P SAVANI UNIVERSITY

School of Engineering

PIP

- PIP stands for **Package installer for python.**
- It connects to an **online repository of public packages**, called the Python Package Index for downloading the package.

P P SAVANI UNIVERSITY

School of Engineering

PIP

```
C:\Users\LENOVO>pip install pandas
Collecting pandas
  Downloading pandas-1.4.3-cp310-cp310-win_amd64.whl (10.5 MB)
     ---------------------------------------- 10.5/10.5 MB 158.7 kB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2022.2.1-py2.py3-none-any.whl (500 kB)
     ---------------------------------------- 500.6/500.6 KB 85.3 kB/s eta 0:00:00
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ---------------------------------------- 247.7/247.7 KB 98.7 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.0 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pandas) (1.23.1)
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, six, python-dateutil, pandas
Successfully installed pandas-1.4.3 python-dateutil-2.8.2 pytz-2022.2.1 six-1.16.0
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.
```

P P SAVANI UNIVERSITY

School of Engineering

ARRAY CREATION USING NUMPY

numpyDemo1.py - D:/Python Material/Python Programs/numpyDemo1.py (3.10.5)

File Edit Format Run Options Window Help

```
from numpy import *

arr = array([11, 22, 33, 44.5, 55],int)
#arr = array([11, 22, 33, 44.5, 55])

print(arr)

print(type(arr))
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo1.py ==========
[11 22 33 44 55]
<class 'numpy.ndarray'>
>>>
```

N-dimensional Array object

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

2-D ARRAY CREATION USING NUMPY

numpyDemo2.py - D:/Python Material/Python Programs/numpyDemo2.py

File Edit Format Run Options Window Help

```
from numpy import *

arr = array([[11,22,33],[44,55,66]])

print(arr)
print(type(arr))

print('2nd element on 2nd row: ', arr[1, 1])
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo2.py ==========
[[11 22 33]
 [44 55 66]]
<class 'numpy.ndarray'>
2nd element on 2nd row:  55
>>>
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

3-D ARRAY CREATION USING NUMPY

numpyDemo3.py - D:/Python Material/Python Programs/numpyDemo3.py

File Edit Format Run Options Window Help

```
from numpy import *

arr = array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr)
print(type(arr))

print("1st array's 2nd element on 2nd row: ", arr[0,1, 1])
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo3.py ==========
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
<class 'numpy.ndarray'>
1st array's 2nd element on 2nd row:  5
>>>
```

Barkha Wadhvani

48

P P SAVANI UNIVERSITY

School of Engineering

ARRAY SLICING USING NUMPY

numpyDemo2.py - D:\Python Material\Python Programs\numpyDemo2.py

File Edit Format Run Options Window Help

```
from numpy import *

arr = array([[11,22,33],[44,55,66]])
print(arr)
print(type(arr))

print('First and Second element of 1st and 2nd row: ', arr[0:2,0:2])
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\numpyDemo2.py ==========
[[11 22 33]
 [44 55 66]]
<class 'numpy.ndarray'>
First and Second element of 1st and 2nd row:  [[11 22]
 [44 55]]
```

Barkha Wadhvani

49

P P SAVANI UNIVERSITY

School of Engineering

ARRAY SEARCH

numpyDemo4.py - D:/Python Material/Python Programs/numpyDemo4.py

File Edit Format Run Options Window Help

```
from numpy import *

arr = array([11,22,33,11,44,55,11])
position=where(arr==11)
print("Position of 11 is ", position)
```

Returns the position where condition matched

IDLE Shell ...

File Edit Shell Debug Options Window Help

```
Python ... (tags/v3.10.5:...) ... 16:14:13) [MSC v.1929 64 b
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo4.py ======
Position of 11 is  (array([0, 3, 6], dtype=int64),)
>>>
```

Barkha Wadhvani

50

P P SAVANI UNIVERSITY

School of Engineering

SEARCH USING NUMPY

numpyDemo5.py - D:/Python Material/Python Programs/numpyDemo5.py

File Edit Format Run Options Window Help

```
# Find Even numbers position
from numpy import *

arr = array([11,22,33,11,44,55,11])
position=where(arr%2==0)
print("Position of Even numbers are \n", position,"\n")
```

IDLE Shell ...

File Edit Shell Debug Options Window Help

```
Python ... (tags/v3.10.5:...) ... [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo5.py ==========
Position of Even numbers are
 (array([1, 4], dtype=int64),)
```

Barkha Wadhvani

51

P P SAVANI UNIVERSITY

School of Engineering

RANDOM NUMBER USING NUMPY

```
numpyDemo6.py - D:/Python Material/Python Programs/numpyDemo6.py (...)
File  Edit  Format  Run  Options  Window  Help
#import Package
from numpy import random

#Generate random numbers from range (0 to 10)
x=random.randint(10)
print("Random Number generated is ", x)

#Data Distribution
x = random.choice([11, 22, 33, 44], p=[0.1, 0.3, 0.6, 0.0], size=(200))
print("200 numbers generated from the specified group with probability", x)
```

```
IDLE Shell
File  Edit  Shell  Debug  Options  Window  Help
Python ... (MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo5.py ==========
Position of Even numbers are
 (array([1, 4], dtype=int64),)

>>>
========== RESTART: D:/Python Material/Python Programs/numpyDemo6.py ==========
Random Number generated is  9
200 numbers generated from the specified group with probability [33 33 22 33 22
33 33 22 22 22 11 33 11 33 33 11 22 22 33 33 33 33 22
 33 33 33 33 33 22 22 33 22 33 33 33 22 11 33 11 33 33 33 22 33 33 22 11
 33 22 22 22 33 22 11 22 33 11 22 33 33 11 33 33 33 33 22 33 33 33 22 33
 22 33 33 22 22 33 33 22 22 11 22 22 22 22 22 33 22 33 33 33 33 22 33 33
 33 33 33 22 22 22 33 11 33 33 33 22 33 22 22 33 33 33 33 33 33 22 22 22
 22 33 22 11 33 33 33 33 33 33 33 11 22 33 33 22 11 33 33 33 33 33 33 33
 33 33 33 33 22 22 33 33 22 22 22 22 33 33 33 22 33 11 33 33 33 33 22 33
 33 33 33 22 22 33 11 33 33 22 22 22 11 33 22 33 22 33 33 33 22 33 33 33
 22 11 22 33 11 33 33 11]
```


P P SAVANI UNIVERSITY

School of Engineering

6 DIGIT PIN USING NUMPY PACKAGE

```
randomModuleDemo6.py - D:\Python Material\Python Programs\randomModuleDemo6.py ...
File  Edit  Format  Run  Options  Window  Help
#Generate Six digit pin
## importing modules
import random
import math

## initializing a string
random_str = ""

## we can generate any lenght of string we want
## generating a random index between 0 to 10 (10 is excluded)
for i in range(6):
    index=random.randint(0,10)
    random_str += str(index)

## displaying the random string
print("6 digit Pin : ", random_str)
                                                              Ln: 1  Col: 23
>>>
======= RESTART: D:\Python Material\Python Programs\randomModuleDemo6.py =======
6 digit Pin :  911618
>>>
======= RESTART: D:\Python Material\Python Programs\randomModuleDemo6.py =======
6 digit Pin :  472462
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

6 DIGIT PIN GENERATION

randomModuleDemo6.py - D:\Python Material\Python Programs\randomModuleDemo6.py ...

File Edit Format Run Options Window Help

```
#Generate Six digit pin
## importing modules
import random
import math

## initializing a string
random_str = ""

## we can generate any lenght of string we want
## generating a random index between 0 to 10 (10 is excluded)
for i in range(6):
    index=random.randint(0,10)
    random_str += str(index)

## displaying the random string
print("6 digit Pin : ", random_str)
```

Ln: 1 Col: 23

```
>>>
======= RESTART: D:\Python Material\Python Programs\randomModuleDemo6.py =======
6 digit Pin :  911618
>>>
======= RESTART: D:\Python Material\Python Programs\randomModuleDemo6.py =======
6 digit Pin :  472462
>>>
```

Barkha Wadhvani

42

P P SAVANI UNIVERSITY

School of Engineering

PANDAS

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
- Panel Data refers to multi-dimensional data frequently involving measurements over time.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Barkha Wadhvani

54

P P SAVANI UNIVERSITY

School of Engineering

PANDAS ADVANTAGES

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets.

Barkha Wadhvani

55

P P SAVANI UNIVERSITY

School of Engineering

PANDAS DATA FRAME

- It is a widely used **data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns).**
- DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index.

It consists of the following properties:

- The columns can be heterogeneous types like int, bool, and so on.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

Barkha Wadhvani

56

P P SAVANI UNIVERSITY

School of Engineering

PANDAS SERIES

- A pandas Series is a **one-dimensional labelled data structure** which can hold data such as strings, integers and even other Python objects.
- It is built on top of numpy array and is the primary data structure to hold one-dimensional data in pandas.

Barkha Wadhvani

57

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES FROM TUPLE

PandaDemo2.py - D:/Python Material/Python Programs/PandaDemo2.py

File Edit Format Run Options Window Help

```
# Make a series from a Tuple
import pandas as pd
# Create the list
data_list = ('Yatharth', 'Yatri','Yuvraj', 'Vihani', 'Riya')

# pass the list to the data parameter of the series constructor
series = pd.Series(data=data_list)
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo2.py ==========
0     Yatharth
1        Yatri
2       Yuvraj
3       Vihani
4         Riya
dtype: object
>>>
```

Barkha Wadhvani

60

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES FROM DICTIONARY

PandaDemo3.py - D:/Python Material/Python Programs/PandaDemo3.py

File Edit Format Run Options Window Help

```
# Make a series from a Dictionary
import pandas as pd
# Create the list
data_list = {1:'Yatharth', 2:'Yatri',3:'Yuvraj',4:'Vihani',5:'Riya'}

# pass the list to the data parameter of the series constructor
series = pd.Series(data=data_list)
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo3.py ==========
1     Yatharth
2        Yatri
3       Yuvraj
4       Vihani
5         Riya
dtype: object
>>>
```

Barkha Wadhvani

61

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES FROM NUMPY ARRAY

PandaDemo4.py - D:/Python Material/Python Programs/PandaDemo4.py (

File Edit Format Run Options Window Help

```
# Make a series from a Numpy array
import pandas as pd
import numpy as np
# Create the list
data_list = np.array(['Yatharth','Yatri','Yuvraj','Vihani','Riya'])
print("Array Form",data_list)

# pass the list to the data parameter of the series constructor
series = pd.Series(data=data_list)
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153, ...) [MSC v.1929 64 b
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo4.py ======
Array Form ['Yatharth' 'Yatri' 'Yuvraj' 'Vihani' 'Riya']
0     Yatharth
1        Yatri
2       Yuvraj
3       Vihani
4         Riya
dtype: object
>>>
```

Barkha Wadhvani

62

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES FROM LIST

PandaDemo5.py - D:/Python Material/Python Programs/PandaDemo5.py (

File Edit Format Run Options Window Help

```
# Make a series from a Numpy array
import pandas as pd
import numpy as np
# Create the list
data_list = np.array([11,22,33,44,55])
print("Array Form",data_list)

# pass the list to the data parameter of the series constructor
series = pd.Series(data=data_list)
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python (tags/v3.10.5:f377153, ...) [MSC v.1929 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo5.py =========
Array Form [11 22 33 44 55]
0    11
1    22
2    33
3    44
4    55
dtype: int32
>>>
```

Barkha Wadhvani

63

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES FROM LIST

pandasDemo8.py - D:/Python Material/Python Programs/pandasDemo8.py

File Edit Format Run Options Window Help

```
# Make a series from a Numpy array
import pandas as pd
import numpy as np
# Create the list
data_list = np.array([11.5,22.0,33.45,44.2,55.1])
print("Array Form",data_list)

# pass the list to the data parameter of the series constructor
series = pd.Series(data=data_list)
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... v.1929 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo8.py ==========
Array Form [11.5  22.   33.45 44.2  55.1 ]
0    11.50
1    22.00
2    33.45
3    44.20
4    55.10
dtype: float64
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES USING SCALAR VALUE

PandaDemo6.py - D:/Python Material/Python Programs/PandaDemo6.py

File Edit Format Run Options Window Help

```
# Make a series using scalar value
import pandas as pd
import numpy as np

# Create the list
data_list = 11

# pass the list to the data parameter of the series constructor
series = pd.Series(data_list,index=[1,2,3,4,5])
print(series)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... SC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo6.py ==========
1    11
2    11
3    11
4    11
5    11
dtype: int64
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

CREATE SERIES WITH A DEFINED INDEX

PandaDemo7.py - D:/Python Material/Python Programs/PandaDemo7.py (...

File Edit Format Run Options Window Help

```
# Make a series using scalar value
import pandas as pd
import numpy as np

# Create the list
data_list = ["Yatharth","Yatri","Yuvraj","Vihanee","Khushi"]
#create the indices
indices=["Computer Engineering","IT","Civil","Electrical","CSE(AIML)"]
# pass the list to the data parameter of the series constructor
series = pd.Series(data_list,index=indices)
print(series)
```

IDLE Shell

ile Edit Shell Debug Options Window Help

```
Python [illegible] bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
========== RESTART: D:/Python Material/Python Programs/PandaDemo7.py ========
Computer Engineering     Yatharth
IT                         Yatri
Civil                     Yuvraj
Electrical               Vihanee
CSE(AIML)                 Khushi
dtype: object
>>
```

Barkha Wadhvani

66

P P SAVANI UNIVERSITY

School of Engineering

DATAFRAME

- A dataframe is a **two-dimensional table of data (structured representation of data)**, typically arranged with columns corresponding to variables and rows corresponding to observations.

- Dataframes are an efficient way of storing and analyzing data.

Barkha Wadhvani

67

P P SAVANI UNIVERSITY

School of Engineering

DATAFRAME

Syntax:

class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)

Data : array (structured or homogeneous) Iterable, dict, or DataFrame.

Index : Index to use for resulting frame.

Columns : Column labels to use for resulting frame.

dtypedtype: (default None) Data type to force. Only a single dtype is allowed.

Copy : bool or None, default None. Copy data from inputs.

P P SAVANI UNIVERSITY

School of Engineering

CREATE DATA FRAME USING LIST

pandasDemo1.py - D:/Python Material/Python Programs/pandasDemo1.py

File Edit Format Run Options Window Help

```
import pandas

# a list of strings
Names=['Yatharth','Yatri']
# Calling DataFrame constructor on list
df=pandas.DataFrame(Names)
print(df)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python                                                    4SC v.1
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo1.py
          0
0  Yatharth
1     Yatri
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

CREATE DATA FRAME USING DICTIONARY

pandasDemo9.py - D:/Python Material/Python Programs/pandasDemo9.py

File Edit Format Run Options Window Help

```
import pandas as pd

# Create the Data
data = {"Name":["Yatharth","Yatri","Vihanee"], "Percentage":[99,98,97]}

# Load data into a Dataframe object
df = pd.DataFrame(data)
print(df)
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... 3) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo9.py ==========
      Name  Percentage
0  Yatharth          99
1     Yatri          98
2   Vihanee          97
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

LOCATE ROWS USING INDEX

pandasDemo9.py - D:\Python Material\Python Programs\pandasDemo9.py

File Edit Format Run Options Window Help

```
import pandas as pd

# Create the Data
data = {"Name":["Yatharth","Yatri","Vihanee"], "Percentage":[99,98,97]}

# Load data into a Dataframe object
df = pd.DataFrame(data)

print(df)

#Locate Row by using Index
print("First Student Data \n ", df.loc[0])#returns Pandas Series

print("Second and Third Student Data\n ", df.loc[[1,2]])#returns Pandas DataFram
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Python Material\Python Programs\pandasDemo9.py ==========
      Name  Percentage
0  Yatharth          99
1     Yatri          98
2   Vihanee          97
First Student Data
  Name          Yatharth
Percentage          99
Name: 0, dtype: object
Second and Third Student Data
      Name  Percentage
1     Yatri          98
2  Vihanee          97
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

LOCATE ROWS USING NAMED INDEX

pandasDemo10.py - D:/Python Material/Python Programs/pandasDemo10.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Create the Data
data = {"Name":["Yatharth","Yatri","Vihanee"], "Percentage":[99,98,97]}

# Load data into a Dataframe object
df = pd.DataFrame(data, index=["21CS101","21CS102","21CS103"])
print("Data is as below :", df)

#Locate Row by using Named Index
print("First Student Data \n ", df.loc[["21CS102"]])#returns Pandas Dataframe by
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python [... v.1929 64 bi
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo10.py ======
Data is as below :              Name  Percentage
21CS101  Yatharth          99
21CS102     Yatri          98
21CS103    Vihanee          97
First Student Data
             Name  Percentage
21CS102  Yatri          98
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

CSV FILE(COMMA SEPARATED VALUE)

studentData.csv - Notepad

File Edit View

```
Name,Department,Percentage
Yatharth,Computer Engineering,99
Yuvraj,IT,98
Vihanee,Civil,97
Raju,IT,95
Avasti,Chemical,80
Ashima,Civil,66
Mahi,Electrical,89
Riya,CSE,90
Nisha,Computer Engineering,50
Pooja,Civil,45
Komal,CSE,40
Sagar,IT,46
Akash,Chemical,40
Yatharth,Computer Engineering,99
Yuvraj,IT,98
Vihanee,Civil,97
Raju,IT,95
Avasti,Chemical,80
Ashima,Civil,66
Mahi,Electrical,89
Riya,CSE,90
Nisha,Computer Engineering,50
Pooja,Civil,45
Komal,CSE,40
Sagar,IT,46
Akash,Chemical,40
Yatharth,Computer Engineering,99
Yuvraj,IT,98
Vihanee,Civil,97
Raju,IT,95
Avasti,Chemical,80
Ashima,Civil,66
Mahi,Electrical,89
Riya,CSE,90
```

P P SAVANI UNIVERSITY

School of Engineering

KNOW YOUR DATA

```
pandasDemo17.py - D:/Python Material/Python Programs/pandasDemo17.py
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')
# Data Information
print("Data details :", data.info())
```

```
IDLE Shell
File  Edit  Shell  Debug  Options  Window  Help
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDem
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 3 columns):
 #    Column       Non-Null Count  Dtype
---   ------       --------------  -----
 0    Name         65 non-null     object
 1    Department   65 non-null     object
 2    Percentage   65 non-null     int64
dtypes: int64(1), object(2)
memory usage: 1.6+ KB
Data details : None
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

READ CSV FILE USING PANDAS

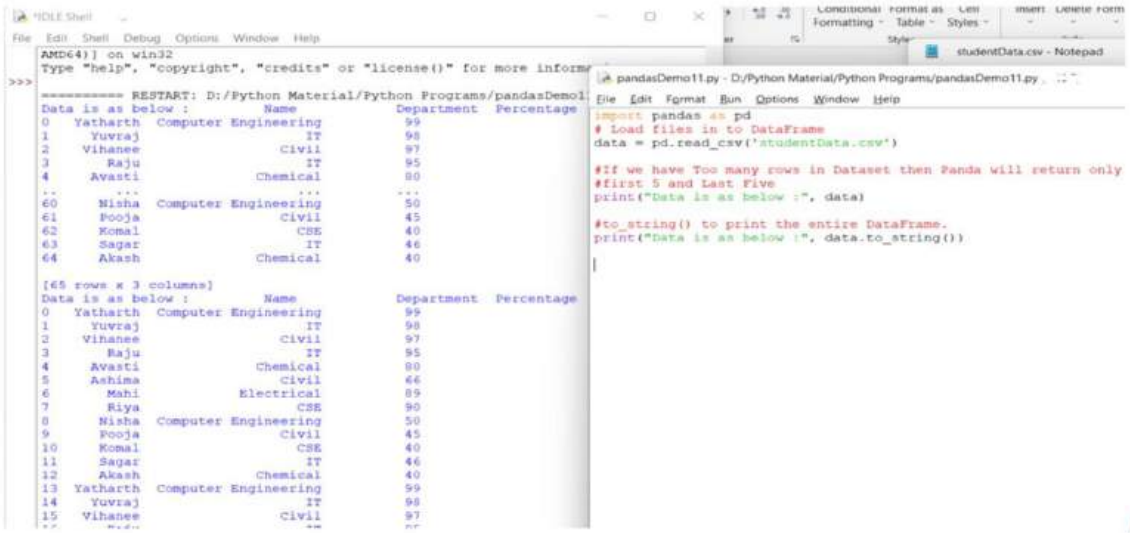
```
pandasDemo12.py - D:/Python Material/Python Programs/pandasDemo12.py
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')

#f the DataFrame contains more than max count rows, the
#print(data) statement will return only the headers and the first and last 5 rows.
print("Max no of rows support in system :", pd.options.display.max_rows)

print("Data is as below :\n", data)
```

```
IDLE Shell
File  Edit  Shell  Debug  Options  Window  Help
Python
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo12.py ==========
Max no of rows support in system : 60
Data is as below :
          Name                Department  Percentage
0     Yatharth  Computer Engineering          99
1        Yuvraj                    IT          98
2       Vihanee                 Civil          97
3          Raju                    IT          95
4        Avasti              Chemical          80
..         ...                   ...         ...
60       Nisha  Computer Engineering          50
61       Pooja                 Civil          45
62       Komal                   CSE          40
63       Sagar                    IT          46
64       Akash              Chemical          40

[65 rows x 3 columns]
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

READ CSV FILE USING PANDAS



```
Data is as below :          Name          Department  Percentage
0   Yatharth  Computer Engineering          99
1     Yuvraj                   IT          98
2    Vihanee                Civil          97
3       Raju                   IT          95
4     Avasti             Chemical          80
..      ...                   ...         ...
60     Nisha  Computer Engineering          50
61     Pooja                Civil          45
62     Komal                  CSE          40
63     Sagar                   IT          46
64     Akash             Chemical          40

[65 rows x 3 columns]
Data is as below :          Name          Department  Percentage
0   Yatharth  Computer Engineering          99
1     Yuvraj                   IT          98
2    Vihanee                Civil          97
3       Raju                   IT          95
4     Avasti             Chemical          80
5     Ashima                Civil          66
6       Mahi           Electrical          89
7       Riya                  CSE          90
8      Nisha  Computer Engineering          50
9      Pooja                Civil          45
10     Komal                  CSE          40
11     Sagar                   IT          46
12     Akash             Chemical          40
13  Yatharth  Computer Engineering          99
14    Yuvraj                   IT          98
15   Vihanee                Civil          97
```

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')

#If we have Too many rows in Dataset then Panda will return only
#first 5 and Last Five
print("Data is as below :", data)

#to_string() to print the entire DataFrame.
print("Data is as below :", data.to_string())
```


P P SAVANI UNIVERSITY

School of Engineering

CHANGE MAX NUMBER OF ROWS

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')
#update Max row number count
pd.options.display.max_rows= 100

print("Data is as below :\n", data)
```

```
53     Yuvraj                   IT          98
54    Vihanee                Civil          97
55       Raju                   IT          95
56     Avasti             Chemical          80
57     Ashima                Civil          66
58       Mahi           Electrical          89
59       Riya                  CSE          90
60      Nisha  Computer Engineering          50
61      Pooja                Civil          45
62      Komal                  CSE          40
63      Sagar                   IT          46
64      Akash             Chemical          40
```

All 65 rows will be displayed

P P SAVANI UNIVERSITY

School of Engineering

ACCESS FIRST SPECIFIED NUMBER OF ROWS

pandasDemo14.py - D:\Python Material\Python Programs\pandasDemo14.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')
# Display Top 5 rows
print("First 5 row Data is as below :\n", data.head())
# Display First 10 rows
print("First 10 Row Data is as below :\n", data.head(10))
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153, ... (MSC v.1931
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information
>>>
========== RESTART: D:\Python Material\Python Programs\pandasDemo14.py
First 5 row Data is as below :
        Name                Department  Percentage
0   Yatharth  Computer Engineering          99
1     Yuvraj                    IT          98
2    Vihanee                 Civil          97
3       Raju                    IT          95
4     Avasti              Chemical          80
First 10 Row Data is as below :
        Name                Department  Percentage
0   Yatharth  Computer Engineering          99
1     Yuvraj                    IT          98
2    Vihanee                 Civil          97
3       Raju                    IT          95
4     Avasti              Chemical          80
5     Ashima                 Civil          66
6       Mahi           Electrical          89
7       Riya                  CSE          90
8      Nisha  Computer Engineering          50
9      Pooja                 Civil          45
>>>
```

Barkha Wadhvani

78

P P SAVANI UNIVERSITY

School of Engineering

ACCESS LAST SPECIFIED NUMBER OF ROWS

pandasDemo16.py - D:/Python Material/Python Programs/pandasDemo16.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')
# Display Last 5 rows
print("Last 5 row Data is as below :\n", data.tail())
# Display Last 10 rows
print("Last 10 Data is as below :\n", data.tail(10))
```

IDLE Shell

File Edit Shell Debug Options Window Help

```
Python ... (tags/v3.10.5:f377153, ... 14:14:13) [MSC ...
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo
Last 5 row Data is as below :
        Name                Department  Percentage
60     Nisha  Computer Engineering          50
61     Pooja                 Civil          45
62     Komal                   CSE          40
63     Sagar                    IT          46
64     Akash              Chemical          40
Last 10 Data is as below :
        Name                Department  Percentage
55      Raju                    IT          95
56    Avasti              Chemical          80
57    Ashima                 Civil          66
58      Mahi           Electrical          89
59      Riya                  CSE          90
60     Nisha  Computer Engineering          50
61     Pooja                 Civil          45
62     Komal                   CSE          40
63     Sagar                    IT          46
64     Akash              Chemical          40
>>>
```

Barkha Wadhvani

79

P P SAVANI UNIVERSITY

School of Engineering

DISPLAY SPECIFIED COLUMNS

pandasDemo15.py - D:/Python Material/Python Programs/pandasDemo15.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_csv('studentData.csv')
# Select specified columns only
print("First 10 Data is as below :\n", data[["Name","Department"]])
```

IDLE She

File Edit Shell Debug Options Window Help

```
Python ... [MSC v.1929 ...
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: D:/Python Material/Python Programs/pandasDemo15.py ==
>>>
========== RESTART: D:/Python Material/Python Programs/pandasDemo15.py ==
First 10 Data is as below :
          Name              Department
0    Yatharth  Computer Engineering
1      Yuvraj                    IT
2     Vihanee                 Civil
3        Raju                    IT
4      Avasti              Chemical
..       ...                   ...
60      Nisha  Computer Engineering
61      Pooja                 Civil
62      Komal                   CSE
63      Sagar                    IT
64      Akash              Chemical

[65 rows x 2 columns]
>>>
```

Barkha Wadhvani

80

P P SAVANI UNIVERSITY

School of Engineering

CLEAN EMPTY CELL (REMOVES NULL VALUE RAWS)

pandasDemo18.py - D:\Python Material\Python Programs\pandasDemo18.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# return a new Data Frame with no empty cells
newlyData=data.dropna() # Removes all rows with NULL values
print(newlyData.head(5))
```

Barkha Wadhvani

81

P P SAVANI UNIVERSITY

School of Engineering

CLEAN EMPTY CELL

```
========== RESTART: D:\Python Material\Python Programs\pandasDemo18.py =======
Data are :
          Name            Department       Admission Date  Percentage
0   Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
3       Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti              Chemical  2021-09-02 00:00:00        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             65 non-null     object
 1   Department       63 non-null     object
 2   Admission Date   65 non-null     object
 3   Percentage       64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
          Name            Department       Admission Date  Percentage
0   Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
4     Avasti              Chemical  2021-09-02 00:00:00        80.0
5     Ashima                 Civil  2022-09-02 00:00:00        66.0
>>>
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

REMOVES ROWS FROM ORIGINAL DATAFRAME)

pandasDemo19.py - D:/Python Material/Python Programs/pandasDemo19.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# return a new Data Frame with no empty cells
data.dropna(inplace = True) # Remove all rows containing NULL Values from the original data
print(data.head(5))
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

REMOVES ROWS FROM ORIGINAL DATAFRAME)

```
========== RESTART: D:/Python Material/Python Programs/pandasDemo19.py ======
Data are :
         Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1    Yuvraj                    IT  2020-09-02 00:00:00        98.0
2   Vihanee                 Civil  2020-09-02 00:00:00        97.0
3      Raju                   NaN  2020-09-02 00:00:00        95.0
4    Avasti              Chemical  2021-09-02 00:00:00        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             65 non-null     object
 1   Department       63 non-null     object
 2   Admission Date   65 non-null     object
 3   Percentage       64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
         Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1    Yuvraj                    IT  2020-09-02 00:00:00        98.0
2   Vihanee                 Civil  2020-09-02 00:00:00        97.0
4    Avasti              Chemical  2021-09-02 00:00:00        80.0
5    Ashima                 Civil  2022-09-02 00:00:00        66.0
>>>
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

REPLACE EMPTY DATA WITH SOME DATA

pandasDemo20.py - D:/Python Material/Python Programs/pandasDemo20.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# replace Empty Cells with some data
data.fillna(0,inplace = True) # Remove all rows containing NULL Values from the
print("After Replacement Datas are :", data.head(5))
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

REPLACE EMPTY DATA WITH SOME DATA

```
Data are :
          Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1    Yuvraj                    IT  2020-09-02 00:00:00        98.0
2   Vihanee                 Civil  2020-09-02 00:00:00        97.0
3      Raju                   NaN  2020-09-02 00:00:00        95.0
4    Avasti              Chemical                  NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            65 non-null     object
 1   Department      63 non-null     object
 2   Admission Date  64 non-null     object
 3   Percentage      64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
After Replacement Datas are :        Name            Department       Admission
Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1    Yuvraj                    IT  2020-09-02 00:00:00        98.0
2   Vihanee                 Civil  2020-09-02 00:00:00        97.0
3      Raju                     0  2020-09-02 00:00:00        95.0
4    Avasti              Chemical                    0        80.0
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

REPLACE ONLY SPECIFIED COLUMNS WITH SOME DATA

pandasDemo21.py - D:/Python Material/Python Programs/pandasDemo21.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# replace specified columns  empty Cells with some data
data["Department"].fillna("Electical",inplace = True)
print("After Replacement Datas are :", data.head(5))
```

P P SAVANI UNIVERSITY

School of Engineering

REPLACE ONLY SPECIFIED COLUMNS WITH SOME DATA

```
Data are :
          Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
3       Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti              Chemical                  NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            65 non-null     object
 1   Department      63 non-null     object
 2   Admission Date  64 non-null     object
 3   Percentage      64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
After Replacement Datas are :         Name            Department       Admission
Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
3       Raju             Electical  2020-09-02 00:00:00        95.0
4     Avasti               Chemical                  NaN        80.0
>>>
```

Barkha Wadhvani

88

P P SAVANI UNIVERSITY

School of Engineering

REPLACE EMPTY DATA WITH SOME DATA

```
Data are :
          Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00         99.0
1    Yuvraj                    IT  2020-09-02 00:00:00         98.0
2   Vihanee                 Civil  2020-09-02 00:00:00         97.0
3      Raju                   NaN  2020-09-02 00:00:00         95.0
4    Avasti              Chemical                   NaN         80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            65 non-null     object
 1   Department      63 non-null     object
 2   Admission Date  64 non-null     object
 3   Percentage      64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
After Replacement Datas are :        Name            Department       Admission
Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00         99.0
1    Yuvraj                    IT  2020-09-02 00:00:00         98.0
2   Vihanee                 Civil  2020-09-02 00:00:00         97.0
3      Raju                     0  2020-09-02 00:00:00         95.0
4    Avasti              Chemical                     0         80.0
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

REPLACE ONLY SPECIFIED COLUMNS WITH SOME DATA

pandasDemo21.py - D:/Python Material/Python Programs/pandasDemo21.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# replace specified columns  empty Cells with some data
data["Department"].fillna("Electical",inplace = True)
print("After Replacement Datas are :", data.head(5))
```

P P SAVANI UNIVERSITY

School of Engineering

REPLACE ONLY SPECIFIED COLUMNS WITH SOME DATA

```
Data are :
        Name            Department       Admission Date  Percentage
0   Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
3       Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti              Chemical                   NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            65 non-null     object
 1   Department      63 non-null     object
 2   Admission Date  64 non-null     object
 3   Percentage      64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
After Replacement Datas are :         Name            Department       Admission
Date  Percentage
0   Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2    Vihanee                 Civil  2020-09-02 00:00:00        97.0
3       Raju             Electical  2020-09-02 00:00:00        95.0
4     Avasti              Chemical                   NaN        80.0
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

REPLACE VALUE OF A CELL

pandasDemo23.py - D:/Python Material/Python Programs/pandasDemo23.py (.

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# replace values of a cell
data.loc[2,'Name']= 'Vihanee'
print("Now Datas are \n :", data.head(5))
```

P P SAVANI UNIVERSITY

School of Engineering

REPLACE VALUE OF A CELL

```
========== RESTART: D:/Python Material/Python Programs/pandasDemo23.py ========
Data are :
          Name              Department       Admission Date  Percentage
0   Yatharth   Computer Engineering  2021-09-02 00:00:00        99.0
1      Yuvraj                    IT  2020-09-02 00:00:00        98.0
2         NaN                 Civil  2020-09-02 00:00:00        97.0
3        Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti                Chemical                  NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             64 non-null     object
 1   Department       63 non-null     object
 2   Admission Date   64 non-null     object
 3   Percentage       64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
Now Datas are
 :          Name              Department       Admission Date  Percentage
0   Yatharth   Computer Engineering  2021-09-02 00:00:00        99.0
1      Yuvraj                    IT  2020-09-02 00:00:00        98.0
2     Vihanee                 Civil  2020-09-02 00:00:00        97.0
3        Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti                Chemical                  NaN        80.0
>>>
```


P P SAVANI UNIVERSITY

School of Engineering

DISCOVERING DUPLICATES

pandasDemo25.py - D:/Python Material/Python Programs/pandasDemo25.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# Discover Duplicate value
print(data.duplicated())
print("Data details :", data.info())
```

Returns True if row is duplicated, False otherwise

P P SAVANI UNIVERSITY

School of Engineering

DISCOVERING DUPLICATES

```
Data are :
          Name            Department       Admission Date  Percentage
0  Yatharth  Computer Engineering  2021-09-02 00:00:00        99.0
1     Yuvraj                    IT  2020-09-02 00:00:00        98.0
2        NaN                 Civil  2020-09-02 00:00:00        97.0
3       Raju                   NaN  2020-09-02 00:00:00        95.0
4     Avasti              Chemical                  NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             64 non-null     object
 1   Department       63 non-null     object
 2   Admission Date   64 non-null     object
 3   Percentage       64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
0     False
1     False
2     False
3     False
4     False
      ...
60     True
61     True
62     True
63     True
64     True
Length: 65, dtype: bool
<class 'pandas.core.frame.DataFrame'>
```


P P SAVANI UNIVERSITY

School of Engineering

REMOVE DUPLICATES

pandasDemo24.py - D:/Python Material/Python Programs/pandasDemo24.py

File Edit Format Run Options Window Help

```
import pandas as pd
# Load files in to DataFrame
data = pd.read_excel('StudentDetails.xlsx')

print("Data are : \n ", data.head(5))
# Data Information
print("Data details :", data.info())

# removes Duuplicates value
data.drop_duplicates(inplace= True)
print("Data details :", data.info())
```

P P SAVANI UNIVERSITY

School of Engineering

REMOVE DUPLICATES

```
Data are :
          Name              Department        Admission Date  Percentage
0  Yatharth   Computer Engineering   2021-09-02 00:00:00        99.0
1     Yuvraj                     IT   2020-09-02 00:00:00        98.0
2        NaN                  Civil   2020-09-02 00:00:00        97.0
3       Raju                    NaN   2020-09-02 00:00:00        95.0
4     Avasti               Chemical                    NaN        80.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             64 non-null     object
 1   Department       63 non-null     object
 2   Admission Date   64 non-null     object
 3   Percentage       64 non-null     float64
dtypes: float64(1), object(3)
memory usage: 2.2+ KB
Data details : None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 0 to 26
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             19 non-null     object
 1   Department       18 non-null     object
 2   Admission Date   19 non-null     object
 3   Percentage       19 non-null     float64
dtypes: float64(1), object(3)
memory usage: 800.0+ bytes
Data details : None
>>
```

Barkha Wadhvani

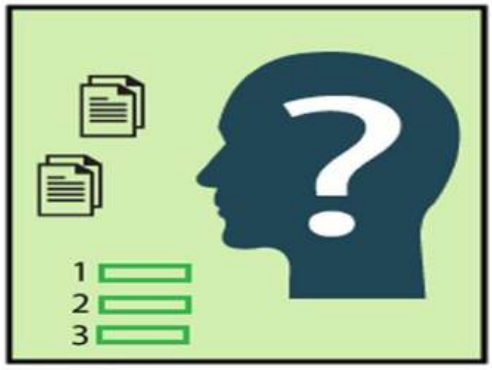
97

P P SAVANI UNIVERSITY

School of Engineering

WHY DATA VISUALIZATION?

- Data visualization allows us to quickly interpret the data and adjust different variables to see their effect.



Barkha Wadhvani

100

P P SAVANI UNIVERSITY

School of Engineering

DATA VISUALIZATION

- Human minds are more adaptive for the visual representation of data rather than textual data.
- We can **easily understand things** when they are visualized.
- It is **better to represent the data through the graph** where we can analyze the data more efficiently and make the specific decision according to data analysis.

Barkha Wadhvani

101

P P SAVANI UNIVERSITY

School of Engineering

MATPLOTLIB

- Matplotlib is **a multi-platform data visualization library.**
- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib is mostly **written in python, a few segments are written in C, Objective-C and Javascript** for Platform compatibility.
- Matplotlib is **open source** and we can use it freely.
- Matplotlib was created by **John D. Hunter.**

P P SAVANI UNIVERSITY

School of Engineering

INSTALL MATPLOTLIB

```
C:\Users\LENOVO>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.5.3-cp310-cp310-win_amd64.whl (7.2 MB)
     ---------------------------------------- 7.2/7.2 MB 1.4 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp310-cp310-win_amd64.whl (55 kB)
     ---------------------------------------- 55.3/55.3 KB 714.6 kB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-
packages (from matplotlib) (2.8.2)
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
     ---------------------------------------- 98.3/98.3 KB 256.1 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages
(from matplotlib) (1.23.1)
Collecting packaging>=20.0
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
     ---------------------------------------- 40.8/40.8 KB 492.1 kB/s eta 0:00:00
Collecting fonttools>=4.22.0
  Downloading fonttools-4.37.1-py3-none-any.whl (957 kB)
     ---------------------------------------- 957.2/957.2 KB 239.8 kB/s eta 0:00:00
Collecting pillow>=6.2.0
  Downloading Pillow-9.2.0-cp310-cp310-win_amd64.whl (3.3 MB)
     ---------------------------------------- 3.3/3.3 MB 168.3 kB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (fr
om python-dateutil>=2.7->matplotlib) (1.16.0)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, packaging, matplotlib
Successfully installed cycler-0.11.0 fonttools-4.37.1 kiwisolver-1.4.4 matplotlib-3.5.3 packaging-21.3 pillow-9.2.0 pypa
rsing-3.0.9
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe -m pip install
 --upgrade pip' command.
```

P P SAVANI UNIVERSITY

School of Engineering

PLOTING X AND Y POINTS

- The plot() function is used to draw points (markers) in a diagram.
- By default, the plot() function draws a line from point to point.
- The function takes parameters for specifying points in the diagram.
- Parameter 1 is an array containing the points on the x-axis.
- Parameter 2 is an array containing the points on the y-axis

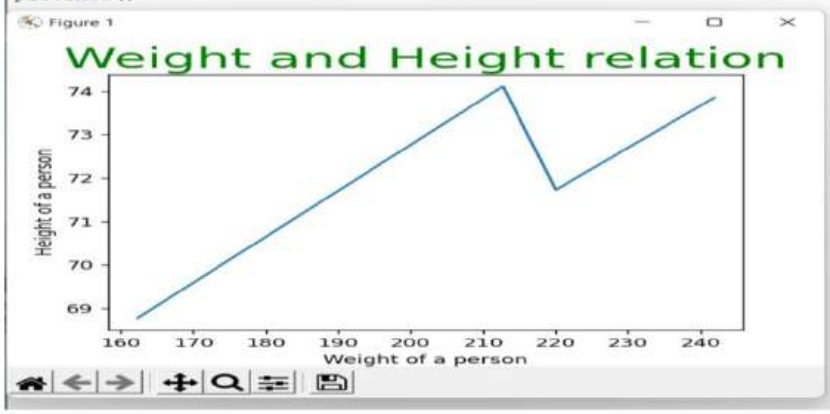
Barkha Wadhvani 104

P P SAVANI UNIVERSITY

School of Engineering

PLOTING X AND Y POINTS

```
File  Edit  Format  Run  Options  Window  Help
from matplotlib import pyplot as plt  # import pyplot
weight=[162.3104725,212.7408556,220.0424703,241.8935632]
height=[68.78190405,74.11010539,71.7309784,73.84701702]
plt.plot(weight,height) #Plot of Weight and Height
plt.title("Weight and Height relation", fontsize=25, color="green")
plt.xlabel("Weight of a person") # Label for X axis
plt.ylabel("Height of a person") # Label for Y axis
plt.show()
```



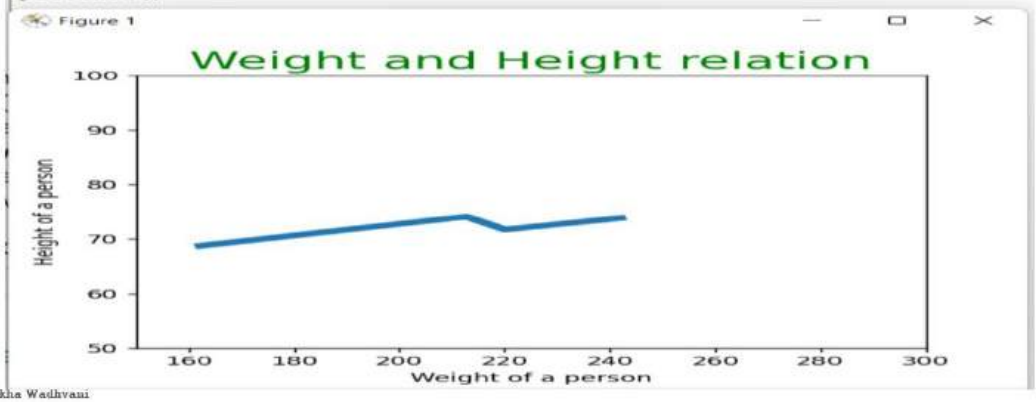
Barkha Wadhvani 105

P P SAVANI UNIVERSITY

School of Engineering

SET LIMIT FOR X AND Y AXIS

```
metplotlibDemo2.py - D:/Python Material/Python Programs/metplotlibDemo2.py
File  Edit  Format  Run  Options  Window  Help
from matplotlib import pyplot as plt  # import pyplot
weight=[162.3104725,212.7408556,220.0424703,241.8935632]
height=[68.78190405,74.11010539,71.7309784,73.84701702]
plt.plot(weight,height, linewidth=5) #Plot of Weight and Height
plt.title("Weight and Height relation", fontsize=20, color="green")
plt.xlabel("Weight of a person") # Label for X axis
plt.ylabel("Height of a person") # Label for Y axis
plt.xlim(150,300)  # Set upper and lower limit of x axis
plt.ylim(50,100) # Set upper and lower limit of Y axis
plt.show()
```



106

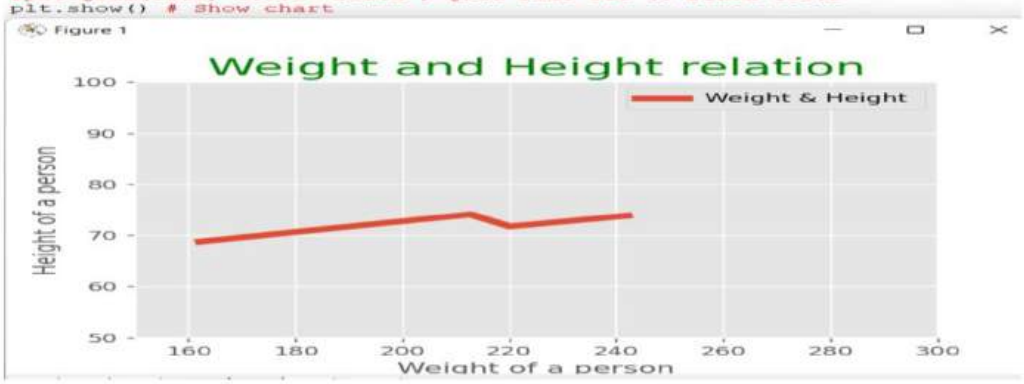
Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

SET STYLE TO GRAPH

```
from matplotlib import pyplot as plt
from matplotlib import style  # Use chart Style
style.use('ggplot')
weight=[162.3104725,212.7408556,220.0424703,241.8935632]
height=[68.78190405,74.11010539,71.7309784,73.84701702]
plt.plot(weight,height, linewidth=5, label="Weight & Height ")
plt.title("Weight and Height relation", fontsize=20, color="green")
plt.xlabel("Weight of a person")
plt.ylabel("Height of a person")
plt.xlim(150,300)
plt.ylim(50,100)
plt.legend() # Displayed the Label what line represents
#plt.grid(True, color="black") grid line set to black color
plt.show() # Show chart
```



107

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

WEIGHT-HEIGHT-GENDER.CSV

	A	B	C
	Gender	Height	Weight
	Male	68.7819	162.3105
	Male	74.11011	212.7409
	Male	71.73098	220.0425
	Male	73.84702	241.8936

108

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

GET DATA FROM FILE FOR VISUALIZATION

metplotlibDemo4.py - D:/Python Material/Python Programs/metplotlibDemo4.py (3.10.5)

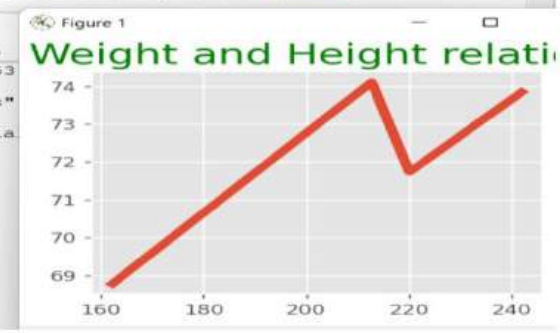
File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style  # Use chart Style
import pandas as pd
data=pd.read_csv('weight-height-Gender.csv')
print("Data are\n ", data)
style.use('ggplot')
weight=data["Weight"]  # Extract Weight column from Dataset
height=data["Height"]  # Extract Height column from Dataset
#print(weight)    #print(height)
plt.plot(weight,height, linewidth=5, label="Weight & Height ")
plt.title("Weight and Height relation", fontsize=20, color="green")
plt.show() # Show chart
```

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

```
Python 3.10.5 (tags/v3.10.5:f377153
AMD64)] on win32
Type "help", "copyright", "credits"
>>>

======== RESTART: D:/Python Materia
Data are
     Gender      Height      Weight
0    Male   68.781904  162.310473
1    Male   74.110105  212.740856
2    Male   71.730978  220.042470
3    Male   73.847017  241.893563
```



109

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

STUDENT_DATA.XSLX

A	B	C	D	E	F	G
Id	Name	Gender	Marks	Admission_Date	Department	Fees
1	Janvi	Female	90	3/5/2018	IT	10000
2	OM	Male	85	30/6/2019	IT	8000
3	Dhruv	Male	88	3/5/2018	IT	15000
4	Ajay	Male	92	15/6/2019	Chemical	7000
5	Nisarg	Male	91	30/7/2019	Chemical	7000
6	Dishang	Male	88	18/6/2019	Civil	10000
7	Parth	Male	85	17/5/2019	Civil	6000
8	Mansi	Male	83	21/6/2019	Computer	15000
9	Sameera	Female	70	21/6/2019	Electrical	1000
10	Sanjay	Male	90	21/6/2019	Computer	1000

110

Barkha Wadhvani

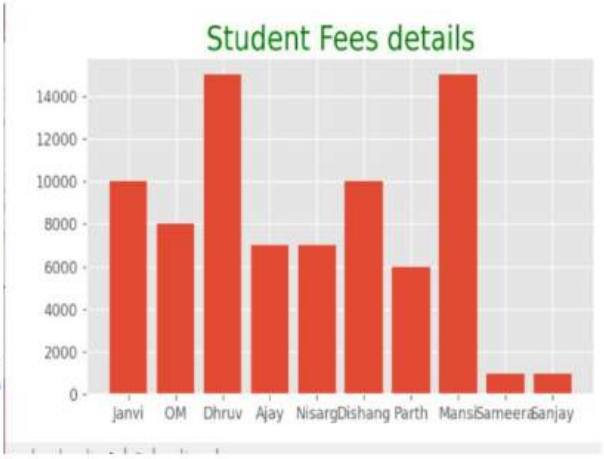
P P SAVANI UNIVERSITY

School of Engineering

BAR CHART

```
from matplotlib import pyplot as plt
from matplotlib import style  # Use chart Style
import pandas as pd
data=pd.read_excel('Student_Data.xlsx')
print("Data are\n ", data)
style.use('ggplot')

plt.bar(data['Name'], data['Fees'])
plt.title("Student Fees details", fontsize=20, color="green")
plt.show() # Show chart
```

```
Data are
    Id     Name  Gender  Marks       Admission_Date  Department   Fees
0   1    Janvi  Female     90  2018-03-05 00:00:00          IT  10000
1   2       OM    Male     85           30/6/2019          IT   8000
2   3    Dhruv    Male     88  2018-03-05 00:00:00          IT  15000
3   4     Ajay    Male     92           15/6/2019    Chemical   7000
4   5   Nisarg    Male     91           30/7/2019    Chemical   7000
5   6  Dishang    Male     88           18/6/2019       Civil  10000
6   7    Parth    Male     85           17/5/2019       Civil   6000
7   8    Mansi    Male     83           21/6/2019    Computer  15000
8   9  Sameera  Female     70           21/6/2019  Electrical   1000
9  10   Sanjay    Male     90           21/6/2019    Computer   1000
```



111

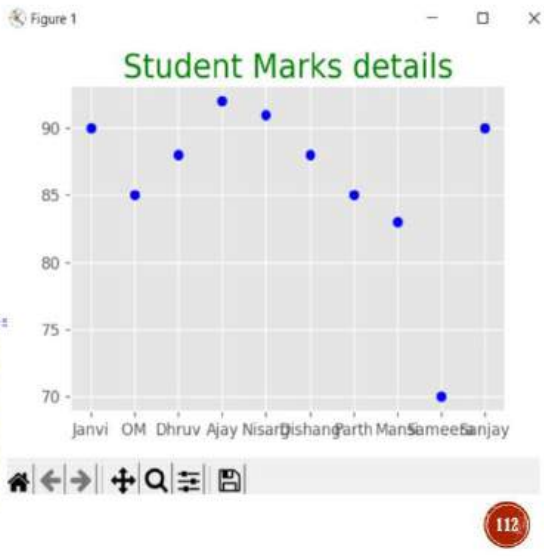
P P SAVANI UNIVERSITY

School of Engineering

SCATTTER PLOT

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_excel('Student_Data.xlsx')
print("Data are\n ", data)
style.use('ggplot')
plt.scatter(data['Name'], data['Marks'], color="blue", marker="o") #Scatter plot
plt.title("Student Marks details", fontsize=20, color="green")
plt.show() # Show chart
```

```
Data are
     Id     Name  Gender  Marks       Admission_Date  Department   Fees
0    1    Janvi  Female     90  2018-03-05 00:00:00          IT  10000
1    2       OM    Male     85            30/6/2019          IT   8000
2    3    Dhruv    Male     88  2018-03-05 00:00:00          IT  15000
3    4     Ajay    Male     92            15/6/2019    Chemical   7000
4    5   Nisarg    Male     91            30/7/2019    Chemical   7000
5    6  Dishang    Male     88            18/6/2019       Civil  10000
6    7    Parth    Male     85            17/5/2019       Civil   6000
7    8    Mansi    Male     83            21/6/2019    Computer  15000
8    9  Sameera  Female     70            21/6/2019  Electrical   1000
9   10   Sanjay    Male     90            21/6/2019    Computer   1000
```



Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

PIE CHART

metplotlibDemo6.py - D:/Python Material/Python Programs/metplotlibDemo6.py

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_excel('Student_Data.xlsx')
print("Data are\n ", data)
style.use('ggplot')
Fees=data["Fees"]
Students=data["Name"]
cols=["orange","red","blue","green"]
plt.pie(Fees, labels=Students, colors=cols, autopct="%0.0f%%") # Pie chart
#autopct represent auto percentage
#explode define how further away the wedge is from the center
plt.title("Fees collected details", fontsize=20, color="green")
plt.show() # Show chart
```

Radius can also be adjust by,
Radius=2

```
Data are
     Id     Name  Gender  Marks       Admission_Date  Department   Fees
0    1    Janvi  Female     90  2018-03-05 00:00:00          IT  10000
1    2       OM    Male     85            30/6/2019          IT   8000
2    3    Dhruv    Male     88  2018-03-05 00:00:00          IT  15000
3    4     Ajay    Male     92            15/6/2019    Chemical   7000
4    5   Nisarg    Male     91            30/7/2019    Chemical   7000
5    6  Dishang    Male     88            18/6/2019       Civil  10000
6    7    Parth    Male     85            17/5/2019       Civil   6000
7    8    Mansi    Male     83            21/6/2019    Computer  15000
8    9  Sameera  Female     70            21/6/2019  Electrical   1000
9   10   Sanjay    Male     90            21/6/2019    Computer   1000
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

PIE CHART

School of Engineering

metplotlibDemo6.py - D:/Python Material/Python Programs/metplotlibDemo6.py

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_excel('Student_Data.xlsx')
print("Data are\n ", data)
style.use('ggplot')
Fees=data["Fees"]
explodeList=[0.1] * len(Fees) # Generate list of explode for fees length
Students=data["Name"]
cols=["orange","red","blue","green"]
plt.pie(Fees, labels=Students, colors=cols, autopct="%0.0f%%", explode=explodeList)
#autopct represent auto percentage
#explode define how further away the wedge is from the center
plt.title("Fees collected details", fontsize=20, color="green")
plt.show() # Show chart
```

```
Data are
   Id     Name  Gender  Marks       Admission_Date  Department   Fees
0   1    Janvi  Female     90  2018-03-05 00:00:00          IT  10000
1   2       OM    Male     85              30/6/2019          IT   8000
2   3    Dhruv    Male     88  2018-03-05 00:00:00          IT  15000
3   4     Ajay    Male     92              15/6/2019    Chemical   7000
4   5   Nisarg    Male     91              30/7/2019    Chemical   7000
5   6  Dishang    Male     88              18/6/2019       Civil  10000
6   7    Parth    Male     85              17/5/2019       Civil   6000
7   8    Mansi    Male     83              21/6/2019    Computer  15000
8   9  Sameera  Female     70              21/6/2019  Electrical   1000
9  10   Sanjay    Male     90              21/6/2019    Computer   1000
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

HISTOGRAM

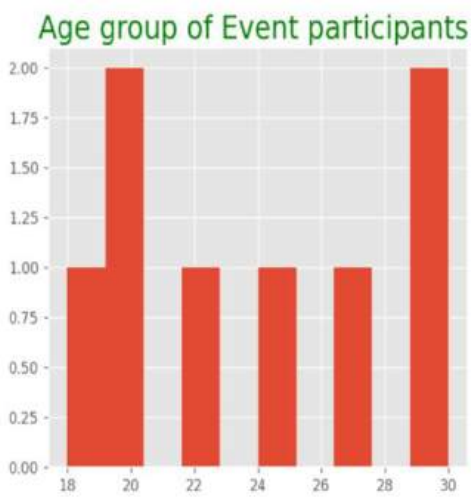
School of Engineering

metplotlibDemo7.py - D:/Python Material/Python Programs/metplotlibDemo7.py

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_csv('Eventdetails.csv')
print("Data are\n ", data)
style.use('ggplot')
plt.hist(data["Age"])
plt.title("Age group of Event participants ", fontsize=20, color="green")
plt.show() # Show chart
```

```
Data are
   ParticipatorName  Gender  Age
0          Yatharth    Male   20
1             Yatri  Female   18
2            Khushi  Female   30
3           Vihanee  Female   20
4             Akash    Male   22
5             Samar    Male   25
6             Sagar    Male   30
7             Komal  Female   27
```



Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

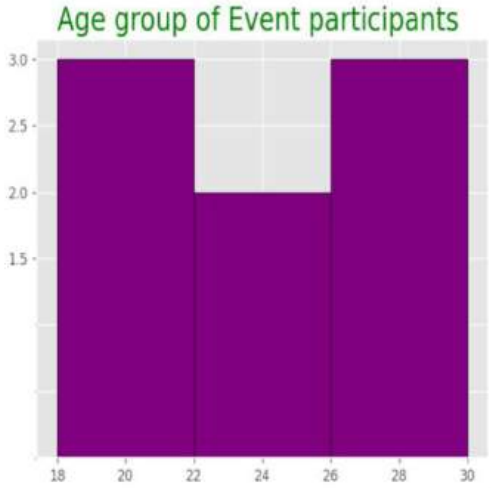
HISTOGRAM

metplotlibDemo8.py - D:/Python Material/Python Programs/metplotlibDemo8.py

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_csv('Eventdetails.csv')
print("Data are\n ", data)
style.use('ggplot')
plt.hist(data["Age"],bins=3,color="purple",edgecolor="black")
plt.title("Age group of Event participants ", fontsize=20, color="green")
plt.show() # Show chart
```

```
Data are
     ParticipatorName  Gender  Age
0           Yatharth    Male   20
1              Yatri  Female   18
2             Khushi  Female   30
3            Vihanee  Female   20
4              Akash    Male   22
5              Samar    Male   25
6              Sagar    Male   30
7              Komal  Female   27
```



Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

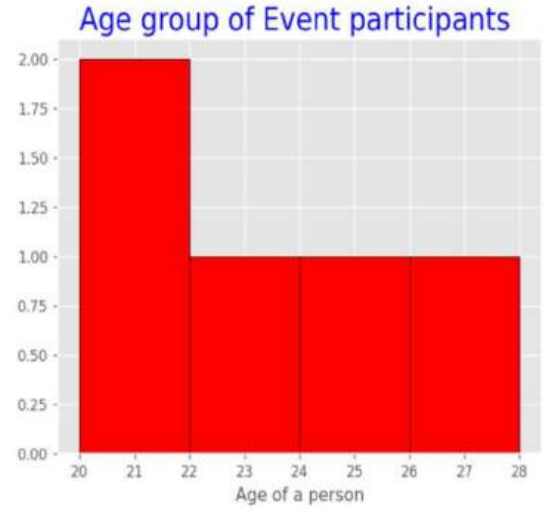
HISTOGRAM

metplotlibDemo9.py - D:/Python Material/Python Programs/metplotlibDemo9.py

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_csv('Eventdetails.csv')
print("Data are\n ", data)
style.use('ggplot')
plt.hist(data["Age"],bins=range(20,30,2) ,color="red",edgecolor="black")
plt.title("Age group of Event participants ", fontsize=20, color="blue")
plt.show() # Show chart
```

```
Data are
     ParticipatorName  Gender  Age
0           Yatharth    Male   20
1              Yatri  Female   18
2             Khushi  Female   30
3            Vihanee  Female   20
4              Akash    Male   22
5              Samar    Male   25
6              Sagar    Male   30
7              Komal  Female   27
```



Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

FEESDATA.XSLX

	A	B	C	D	E	F	G	H
1	Id	Name	Gender	Marks	Admission_Date	Department	Paid_Fees	Pending_Fees
2	1	Janvi	Female	90	3/5/2018	IT	10000	5000
3	2	OM	Male	85	30/6/2019	IT	8000	7000
4	3	Dhruv	Male	88	3/5/2018	IT	15000	0
5	4	Ajay	Male	92	15/6/2019	Chemical	7000	8000
6	5	Nisarg	Male	91	30/7/2019	Chemical	7000	8000
7	6	Dishang	Male	88	18/6/2019	Civil	10000	5000
8	7	Parth	Male	85	17/5/2019	Civil	6000	9000
9	8	Mansi	Male	83	21/6/2019	Computer	15000	0
10	9	Sameera	Female	70	21/6/2019	Electrical	1000	14000
11	10	Sanjay	Male	90	21/6/2019	Computer	1000	14000

118

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

MULTIPLE THINGS ON SAME PLOTS

```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_excel('FeesData.xlsx')
print("Data are\n ", data)
style.use('ggplot')
plt.plot(data['Name'], data['Paid_Fees'], color="blue", marker="o",label="Paid Fees ")
plt.plot(data['Name'], data['Pending_Fees'], color="red", marker="*",label="Pending Fees ")
plt.title("Student Fees details", fontsize=20, color="green")
plt.legend()
plt.show() # Show chart
```

```
Data are
   Id     Name  Gender  ...  Department Paid_Fees Pending_Fees
0   1    Janvi  Female  ...          IT     10000         5000
1   2       OM    Male  ...          IT      8000         7000
2   3    Dhruv    Male  ...          IT     15000            0
3   4     Ajay    Male  ...    Chemical      7000         8000
4   5   Nisarg    Male  ...    Chemical      7000         8000
5   6  Dishang    Male  ...       Civil     10000         5000
6   7    Parth    Male  ...       Civil      6000         9000
7   8    Mansi    Male  ...    Computer     15000            0
8   9  Sameera  Female  ...  Electrical      1000        14000
9  10   Sanjay    Male  ...    Computer      1000        14000

[10 rows x 8 columns]
```

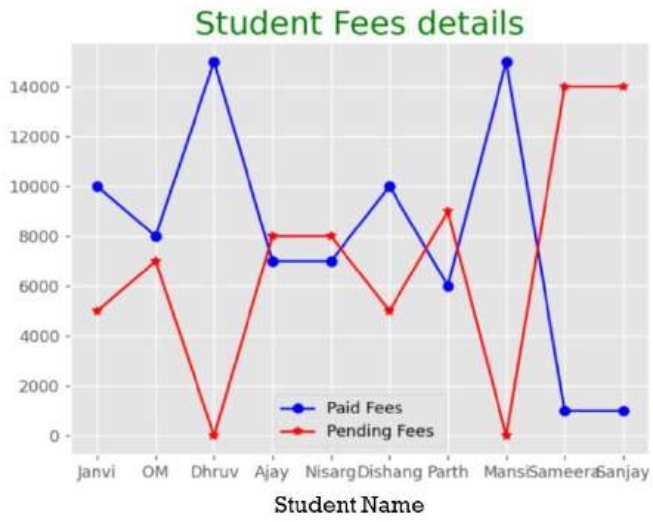
119

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

MULTIPLE THINGS ON SAME PLOTS



P P SAVANI UNIVERSITY

School of Engineering

PLOT SUB PLOTS

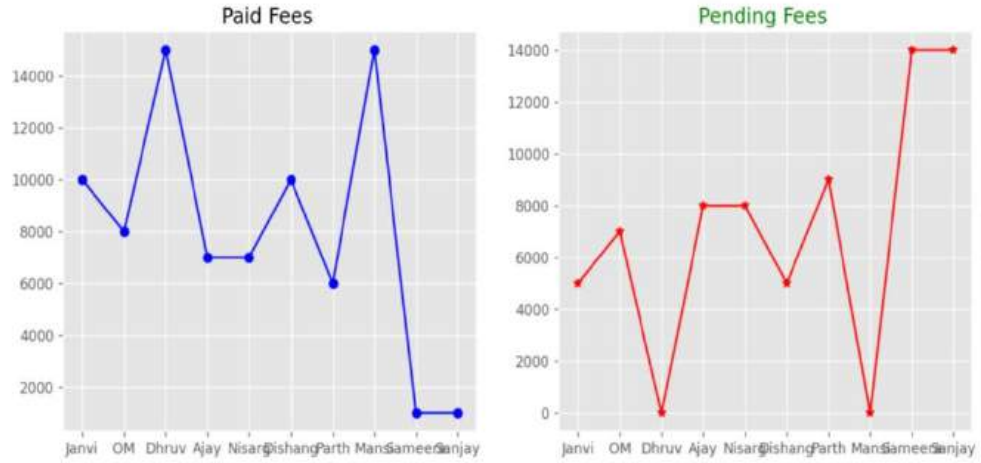
```
from matplotlib import pyplot as plt
from matplotlib import style
import pandas as pd
data=pd.read_excel('FeesData.xlsx')
print("Data are\n ", data)
style.use('ggplot')
plt.subplot(1,2,1) # 1 row and 2 column [in that 1st column we're going to fill]
plt.plot(data['Name'], data['Paid_Fees'], color="blue", marker="o")
plt.title("Paid Fees ")
plt.subplot(1,2,2) # 1 row and 2 column [in that 2nd column we're going to fill]
plt.plot(data['Name'], data['Pending_Fees'], color="red", marker="*")
plt.title("Student Fees details", fontsize=20, color="green")
plt.title("Pending Fees ")
plt.show() # Show chart
```

```
Data are
   Id     Name  Gender  ...  Department Paid_Fees Pending_Fees
0   1    Janvi  Female  ...          IT     10000         5000
1   2       OM    Male  ...          IT      8000         7000
2   3    Dhruv    Male  ...          IT     15000            0
3   4     Ajay    Male  ...    Chemical      7000         8000
4   5   Nisarg    Male  ...    Chemical      7000         8000
5   6  Dishang    Male  ...       Civil     10000         5000
6   7    Parth    Male  ...       Civil      6000         9000
7   8    Mansi    Male  ...    Computer     15000            0
8   9  Sameera  Female  ...  Electrical      1000        14000
9  10   Sanjay    Male  ...    Computer      1000        14000

[10 rows x 8 columns]
```

P P SAVANI UNIVERSITY

School of Engineering

PLOT SUB PLOTS



Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

OBJECT ORIENTED PROGRAMMING

Barkha Wadhvani
Assistant Professor
P.P. Savani University
School Of Engineering

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION OF OBJECT ORIENTED

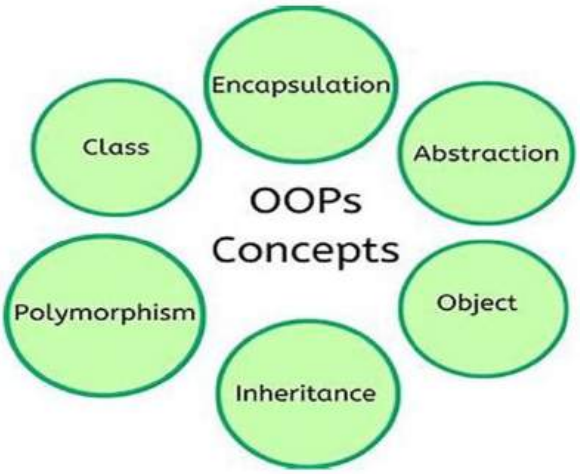
- Object-oriented programming is a programming paradigm that provides a means of structuring programs so that properties and behaviors are bundled into individual **objects.**

- Object-oriented programming is an approach for modeling concrete, real-world things, like cars, as well as relations between things, like companies and employees, students and teachers, and so on. OOP models real-world entities as software objects that have some data associated with them and can perform certain functions

P P SAVANI UNIVERSITY

School of Engineering

OBJECT ORIENTED PROGRAMMING CONCEPTS



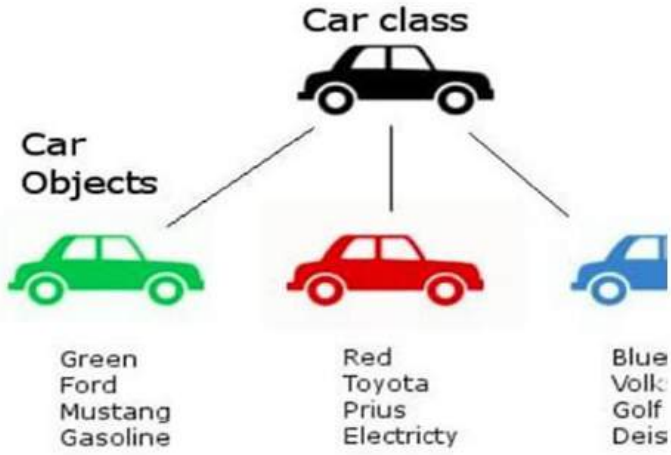
Barkha Wadhvani

3

P P SAVANI UNIVERSITY

School of Engineering

UNDERSTAND OBJECT ORIENTED



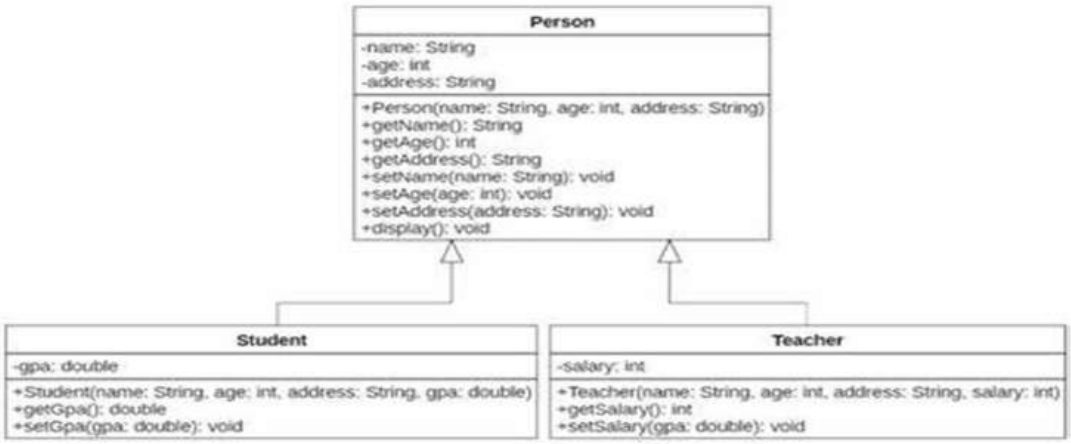
Barkha Wadhvani

4

P P SAVANI UNIVERSITY

School of Engineering

UNDERSTAND OBJECT ORIENTED



Barkha Wadhvani

5

P P SAVANI UNIVERSITY

School of Engineering

CLASS

- A class is a blueprint for the object.

Create Class in python

Syntax

```
Class className:
    Statements
```

Barkha Wadhvani

6

P P SAVANI UNIVERSITY

School of Engineering

OBJECT

- An object (instance) is an instantiation of a class.
- When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
- An Object has Attributes(Properties) and Behaviours (Methods).

Create Object in python

Syntax

Obj= classname()

Barkha Wadhvani

7

P P SAVANI UNIVERSITY

School of Engineering

CLASS AND OBJECT DEMO

```
# Create Class
class Person:
  pass    # Pass is the null statement in Python

#Create Object
p1 = Person()

print("Person class object datatype: ", type(p1)) # Datatype of class
```

Each Object will be allocated to different memory location
Size of the object will be depend on the no. of variables

```
Person class object datatype:  <class '__main__.Person'>
>>>
```

The class is User defined

Barkha Wadhvani

8

P P SAVANI UNIVERSITY

School of Engineering

CREATE METHOD

classDemo2.py - D:/Python Material/Python Programs/classDemo2.py

File Edit Format Run Options Window Help

```
# Create Class
class Person:
  def greet():
      print("Hello....Namaste...!!")

#Create Object
p1 = Person()

Person.greet()
```

```
========== RESTART: D:/Python Material/Python Pr
Hello....Namaste...!!
>>
```

Barkha Wadhvani 9

P P SAVANI UNIVERSITY

School of Engineering

CREATE METHOD

```
# Create Class
class Person:
  def greet(self):        # Self points to particular object
      print("Hello....Namaste...!!")

#Create Object
p1 = Person()
p2 = Person()


p1.greet()    # It goes like Person.greet(p1)
p2.greet()    # same as Person.greet(p2)
```

Method to greet Persons

```
Hello....Namaste...!!
Hello....Namaste...!!
>>>
```

Barkha Wadhvani 10

P P SAVANI UNIVERSITY

School of Engineering

SELF PARAMETER

- The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
- **It does not have to be named self** , you can call it whatever you like, but it has to be the first parameter of any function in the class:

P P SAVANI UNIVERSITY

School of Engineering

CLASS METHOD DEMO

*classDemo3.py - D:/Python Material/Python Programs/classDemo3.py

File Edit Format Run Options Window Help

```
# Create Class
class Person:
    # Self points to particular object . Object name can be anything
    def greet(hello):
        print("Hello....Namaste...!!")

#Create Object
p1 = Person()
p2 = Person()

p1.greet()    # It goes like Person.greet(p1)
p2.greet()    # same as Person.greet(p2)
```

Object name can be anything

```
========== RESTART: D:/Python Material/Python Programs/classDemo3.py
Hello....Namaste...!!
Hello....Namaste...!!
```

P P SAVANI UNIVERSITY

School of Engineering

SET AND GET DATA METHODS

classDemo4.py - D:/Python Material/Python Programs/classDemo4.py

File Edit Format Run Options Window Help

```
# Create Class
class Person:
  def setData(self):
    self.Name="Yatri"
    self.Contact="8787675654"

  def getData(self):
    print("Name : \n ", self.Name)
    print("Name : \n ", self.Contact)

  def greet(self):
    print("Hello....Namaste...!!")

#Create Object
p1 = Person()
p1.setData()   # Set Data
p1.getData()   # Get Data
```

Ln: 9

```
Name :
  Yatri
Name :
  8787675654
```

Barkha Wadhvani

13

P P SAVANI UNIVERSITY

School of Engineering

CLASS CONSTRUCTOR

- The job of the class constructor is to **assign the values to the data members of the class when an object of the class is created.**

- **__init() is used to declare constructor** in class.

Barkha Wadhvani

14

P P SAVANI UNIVERSITY

School of Engineering

INIT() FUNCTION

- All classes have a function called **__init__(), which is always executed when the class is being initiated.**
- Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:
- The __init__() function is **called automatically** every time the class is being used to create a new object.

P P SAVANI UNIVERSITY

School of Engineering

__INIT__()

classDemo5.py - D:/Python Material/Python Programs/classDemo5.py (3...

File Edit Format Run Options Window Help

```
# Create Class
class Person:
    def __init__(self,Name,Contact):
        self.Name=Name
        self.Contact=Contact

    def getData(self):
        print("Name : \n ", self.Name)
        print("Name : \n ", self.Contact)

    def greet(self):
        print("Hello....Namaste...!!\n ")

#Create Object
p1 = Person("Yatri","8787675654")
p2 = Person("Rahul","9987675654")
p1.greet()          # Same as Person.greet(P1)
p1.getData()    # Get Data
```

```
Hello....Namaste...!!

Name :
  Yatri
Name :
  8787675654
Hello....Namaste...!!

Name :
  Rahul
Name :
  9987675654
>>>
```

P P SAVANI UNIVERSITY

School of Engineering

DEMO

overloadingDemo.py - D:/Python Material/Python Programs/overloadingDemo.py

File Edit Format Run Options Window Help

```
# Create Class
class Marks:
    def __init__(self,m1,m2,m3):
        self.mark1=m1
        self.mark2=m2
        self.mark3=m3

    def calculate(self):
        total=self.mark1+self.mark2+self.mark3
        percentage=(total*100)/300
        return percentage

#Create Object
p1 = Marks(90,80,70)

print("Percentage of a person", p1.calculate())
```

Calculate Percentage of a Student

```
========= RESTART: D:/Python Material/Python Programs/overl
Percentage of a person 80.0
>>>
```

Barkha Wadhvani

17

P P SAVANI UNIVERSITY

School of Engineering

ENCAPSULATION

- Using OOP in Python, **we can restrict access to methods and variables.** This prevents data from direct modification which is called encapsulation.

- In Python, we denote private attributes using underscore as the prefix i.e single _ or double __.

Barkha Wadhvani

18

P P SAVANI UNIVERSITY

School of Engineering

OVERLOADING BUILT-IN FUNCTIONS

- Function overloading is the feature when **multiple functions have the same name but the number of parameters in the functions varies.**
- Since the method arguments do not have a data type, **python does not support Function overloading.**

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

FUNCTION OVERLOADING

overloadingDemo3.py - D:\Python Material\Python Programs\overloadingDemo3.py

File Edit Format Run Options Window Help

```
# Create Class
class Addition:

    def add(self,a,b):
        return (a+b)

    def add(self,a,b,c):
        return (a+b+c)

#Create Object
p1 = Addition()
print("Addition of Three numbers", p1.add(11,22,33)) # COrrect Output
print("Addition of Two numbers", p1.add(11,22))    # throws Error
```

Function Overloading does not supported in python

Ln: 1

```
======== RESTART: D:\Python Material\Python Programs\overloadingDemo3.py
Addition of Three numbers 66
Traceback (most recent call last):
  File "D:\Python Material\Python Programs\overloadingDemo3.py", line 14,
dule>
    print("Addition of Two numbers", p1.add(11,22))
TypeError: Addition.add() missing 1 required positional argument: 'c'
>>>
```


Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

WAY TO ACHIEVE FUNCTION OVERLOADING

overloadingDemo4.py - D:\Python Material\Python Programs\overloadingDemo4.py (

File Edit Format Run Options Window Help

```
# Calculate addition of numbers
class Addition:

   # Way to acheive Function Overloading
   def add(self,a=None,b=None,c=None):
      if a != None and b != None and c != None:
         return a+b+c
      elif a!= None and b != None:
         return a+b
      elif a!= None:
         return a
      else:
         print("Please enter more numbers")

p1 = Addition()
print("If No nuumbers provided", p1.add())
print("Addition of one numbers", p1.add(11))
print("Addition of Two numbers", p1.add(11,22))
print("Addition of Three numbers", p1.add(11,22,33))
```

Function Overloading achieved

```
Please enter more numbers
If No nuumbers provided None
Addition of one numbers 11
Addition of Two numbers 33
Addition of Three numbers 66
```

Barkha Wadhvani

21

P P SAVANI UNIVERSITY

School of Engineering

INHERITANCE

- Inheritance is a way of **creating a new class for using details of an existing class without modifying it.**
- The newly formed class is a derived class (or **child class**). Similarly, the existing class is a base class (or **parent class**).

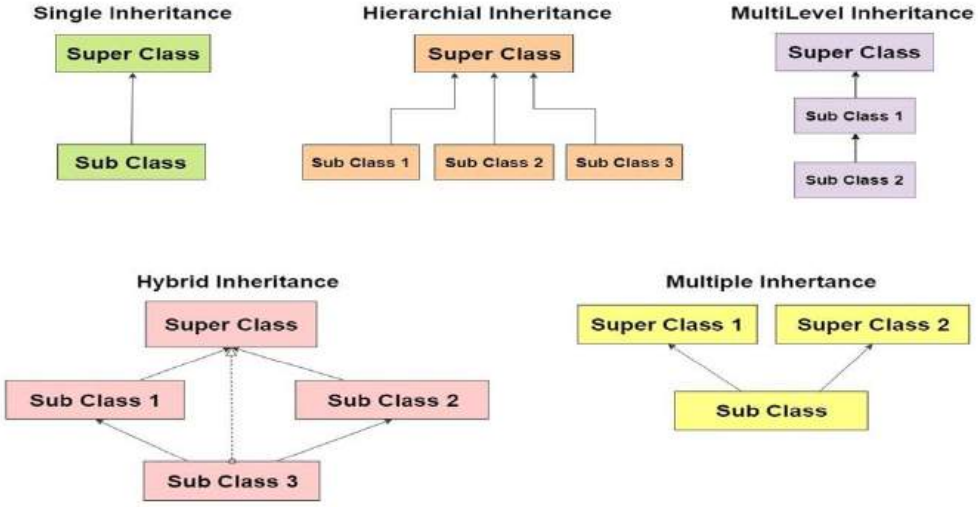
Barkha Wadhvani

22

P P SAVANI UNIVERSITY

School of Engineering

TYPES OF INHERITANCE



Barkha Wadhvani

23

P P SAVANI UNIVERSITY

School of Engineering

INHERITANCE DEMO

```
# Demonstrate Inheritance [Single Inheritance]
class Person:
  def __init__(self,Name,Contact):
    self.Name=Name
    self.Contact=Contact

  def getData(self):
    print("Name : \n ", self.Name)
    print("Contact : \n ", self.Contact)

  def greet(self):
    print("Hello....Namaste...!!\n ")

class Student(Person):    # Inherit Person Class
  def __init__(self,Name,Contact):
    Person.__init__(self,Name,Contact)   # Call Person class Init method

  def Calculate(self,m1,m2,m3):
    total=m1+m2+m3
    percentage=(total*100)/300
    return percentage

#Create Object
p1 = Student("Yatri","8787675654")
p1.greet()   # Person (Parent) class method called with Student (Child) class
p1.getData()
print("Student Percentage",p1.Calculate(90,50,99))
```

```
Hello....Namaste...!!

Name :
  Yatri
Contact :
  8787675654
Student Percentage 79.66666666666667
>>>
```

Barkha Wadhvani

24

P P SAVANI UNIVERSITY

School of Engineering

INHERITANCE DEMO

```
# Demonstrate Inheritance [Hierarchial Inheritance]
class Person:
  def __init__(self,Name,Contact):
    self.Name=Name
    self.Contact=Contact

  def getData(self):
    print("Name : \n ", self.Name)
    print("Contact : \n ", self.Contact)

  def greet(self):
    print("\nHello....Namaste...!!\n ")

class Student(Person):     # Inherit Person Class
  def __init__(self,Name,Contact):
    Person.__init__(self,Name,Contact)    # Call Person class Init method

  def Calculate(self,m1,m2,m3):
    total=m1+m2+m3
    percentage=(total*100)/300
    return percentage

class Teacher(Person):     # Inherit Person Class
  def __init__(self,Name,Contact,Salary):
    self.Salary=Salary
    Person.__init__(self,Name,Contact)    # Call Person class Init method

  def getSalary(self):
    print("Salary :" , self.Salary)
```


Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

INHERITANCE DEMO

```
#Create Object
s1 = Student("Yatri","8787675654")
s1.greet()    # Person (Parent) class method called with Student (Child) class
s1.getData()
print("Student Percentage",s1.Calculate(90,50,99))

t1=Teacher("Akash","9878678787",50000)
t1.greet()
t1.getData()
```

```
Hello....Namaste...!!

Name :
  Yatri
Contact :
  8787675654
Student Percentage 79.66666666666667

Hello....Namaste...!!

Name :
  Akash
Contact :
  9878678787
Salary : 50000
>>>
```


Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

INHERITANCE DEMO

```
#Create Object
s1 = Student("Yatri","8787675654")
s1.greet()    # Person (Parent) class method called with Student (Child) class
s1.getData()
print("Student Percentage",s1.Calculate(90,50,99))

t1=Teacher("Akash","9878678787",50000)
t1.greet()
t1.getData()
```

```
Hello....Namaste...!!

Name :
   Yatri
Contact :
   8787675654
Student Percentage 79.66666666666667

Hello....Namaste...!!

Name :
   Akash
Contact :
   9878678787
Salary : 50000
>>>
```

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

SUPER()

- The super() function is used to give access to methods and properties of a parent or sibling class.
- The super () function returns an object that represents the parent class.

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

SUPER()

```
class Person:
  def __init__(self,Name,Contact):
    self.Name=Name
    self.Contact=Contact

  def getData(self):
    print("Name : \n ", self.Name)
    print("Contact : \n ", self.Contact)

  def greet(self):
    print("\nHello....Namaste...!!\n ")

class Student(Person):     # Inherit Person Class
  def __init__(self,Name,Contact):
    #Person.__init__(self,Name,Contact)    # Call Person class Init method
    super().__init__(Name,Contact)

  def Calculate(self,m1,m2,m3):
    total=m1+m2+m3
    percentage=(total*100)/300
    return percentage
```


P P SAVANI UNIVERSITY

School of Engineering

ACCESS MODIFIERS

- Specifiers determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class.
- Access Specifiers can be used to restrict access.

P P SAVANI UNIVERSITY

School of Engineering

PUBLIC ACCESS MODIFIER

`public` Access Modifier

By default, all the variables and member functions of a class are `public` in a python program.

```
# defining a class Employee
class Employee:
    # constructor
    def __init__(self, name, sal):
        self.name = name
        self.sal = sal
```

All the member variables of the class in the above code will be by default `public`, hence we can access them as follows:

```
>>> emp = Employee("Ironman", 999000)
>>> emp.sal
999000
```

Barkha Wadhvani

31

P P SAVANI UNIVERSITY

School of Engineering

PROTECTED ACCESS MODIFIER

`protected` Access Modifier

According to Python convention adding a prefix `_` (single underscore) to a variable name makes it `protected`. Yes, no additional keyword required.

```
# defining a class Employee
class Employee:
    # constructor
    def __init__(self, name, sal):
        self._name = name    # protected attribute
        self._sal = sal      # protected attribute
```

In the code above we have made the class variables **name** and **sal** `protected` by adding an `_` (underscore) as a prefix, so now we can access them as follows:

```
>>> emp = Employee("Captain", 10000)
>>> emp._sal
10000
```

Barkha Wadhvani

32

P P SAVANI UNIVERSITY

School of Engineering

PROTECTED ACCESS MODIFIER

Similarly if there is a child class extending the class `Employee` then it can also access the protected member variables of the class `Employee`. Let's have an example:

```
# defining a child class
class HR(Employee):

    # member function task
    def task(self):
        print "We manage Employees"
```

Now let's try to access protected member variable of class `Employee` from the class `HR`:

```
>>> hrEmp = HR("Captain", 10000)
>>> hrEmp._sal
10000
>>> hrEmp.task()
We manage Employees
```

Barkha Wadhvani

33

P P SAVANI UNIVERSITY

School of Engineering

PRIVATE ACCESS MODIFIER

`private` Access Modifier

While the addition of prefix `__` (double underscore) results in a member variable or function becoming `private`.

```
# defining class Employee
class Employee:
    def __init__(self, name, sal):
        self.__name = name     # private attribute
        self.__sal = sal       # private attribute
```

If we want to access the **private** member variable, we will get an error.

```
>>> emp = Employee("Bill", 10000)
>>> emp.__sal
```

OUTPUT:

```
AttributeError: 'employee' object has no attribute '__sal'
```

Barkha Wadhvani

34

P P SAVANI UNIVERSITY

School of Engineering

```
# define parent class Company
class Company:
    # constructor
    def __init__(self, name, proj):
        self.name = name      # name(name of company) is public
        self._proj = proj     # proj(current project) is protected

    # public function to show the details
    def show(self):
        print("The code of the company is = ",self.ccode)

# define child class Emp
class Emp(Company):
    # constructor
    def __init__(self, eName, sal, cName, proj):
        # calling parent class constructor
        Company.__init__(self, cName, proj)
        self.name = eName   # public member variable
        self.__sal = sal    # private member variable

    # public function to show salary details
    def show_sal(self):
        print("The salary of ",self.name," is ",self.__sal,)
```

```
# creating instance of Company class
c = Company("Stark Industries", "Mark 4")
# creating instance of Employee class
e = Emp("Steve", 9999999, c.name, c._proj)

print("Welcome to ", c.name)
print("Here ", e.name," is working on ",e._proj)

# only the instance itself can change the __sal variable
# and to show the value we have created a public function show_sal()
e.show_sal()
```

Barkha Wadhvani

35

P P SAVANI UNIVERSITY

School of Engineering

CLASS ATTRIBUTES

- It is shared between all the objects of this class.
- It is defined outside the constructor function, __init__(self,...) , of the class

Barkha Wadhvani

36

P P SAVANI UNIVERSITY

School of Engineering

CLASS ATTRIBUTES

```
class countObject:

    # this is used to print the number
    # of instances of a class
    counter = 0

    # constructor of geeks class
    def __init__(self):

        # increment
        countObject.counter += 1

# object or instance of geeks class
ob1 = countObject()
ob2 = countObject()
ob3 = countObject()
print("Number of Object created : ", countObject.counter)
```

```
======== RESTART: D:/Python Material/Python Programs/ObjectCountDemo.p
Number of Object created :  3
>>>
```

Barkha Wadhvani

37

Regular Expression in Python

What is regular expression?

- A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.
- RegEx can be used to check if a string contains the specified search pattern.
- Python has a built-in package called re, which can be used to work with Regular Expressions.

- Import the re module

  Import re

RegEx functions

The `re` module offers a set of functions that allows us to search a string for a match:

Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

Findall(pattern, String)

- It returns a list of all matching patterns.

```
import re

string = "I felt happy because I saw the others were happy and
 because I knew I should feel happy, but I wasn’t really happy."
x = re.findall("happy", string)

print(x)

# OUTPUT
# ['happy', 'happy', 'happy', 'happy']
```

re.search(pattern,string)

It returns a match object of the first location of the pattern in the string.

```
import re

string = "I felt happy because I saw the others were happy and because I knew I should feel happy, but I wasn’t really happy."

x = re.search("happy", string)

print(x)

if x:
  print("First location of pattern is",x.start())
else:
  print("No match")


  # OUTPUT
  # <_sre.SRE_Match object; span=(7, 12), match='happy'>
  # First location of pattern is 7
```

re.split(pattern,string)

It returns a list where the string has been split at each match.

```
import re
string = "I felt happy because I saw the others were happy."
x = re.split(" ", string)
print(x)

# OUTPUT
# ['I', 'felt', 'happy', 'because', 'I', 'saw', 'the', 'others', 'were', 'happy.']
```

Re.sub(pattern, replace, string, count=0)

- It replaces one or many matches with a string. Here, the default value of count is equal to zero means it will replace all the matching patterns in the string. If we pass the value of count other than zero, then that would be the number of the matching patterns in the string.

```
import re

string = "I felt happy because I saw the others were happy"
x = re.sub("happy", "sad", string)
print(x)

# OUTPUT
# I felt sad because I saw the others were sad
```

Special characters

- ^ : Matches the start of the string
- $: Matches the end of the string
- . : Matches any character except a newline
- ? : Repeats a character zero or one time
- + : Matches one or more repetitions of the preceding RE
- +? : Matches one or more repetitions of the preceding RE(non-greedy)
- * : Matches zero or more repetitions of the preceding RE
- *? : Matches zero or more repetitions of the preceding RE(non-greedy)
- \ : Signals a special sequence
- | : Used for alternation
- [] : Used to indicate a set of characters
- () : Matches whatever regular expression is inside the parentheses, and indicates the start and end of a group
- {m} : Specifies that exactly m copies of the previous RE should be matched
- {m,n} : Matches from m to n repetitions of the preceding RE, attempting to match as many repetitions as possible

Metacharacters: characters with special meaning

1. **‘[]’ A set of characters**

```
import re
str = "The rain in Spain"
#Find all lower case characters alphabetically between "a" and "m":
x = re.findall("[a-m]", str)
print(x)
```

Output: ['h', 'e', 'a', 'i', 'i', 'a', 'i']

2. **'\' Signals a special sequence**

```
str = "That will be 59 dollars"
#Find all digit characters:
x = re.findall("\d", str)
print(x)
```

Output: ['5','9']

3. '.' Any character (except newline character)

```
str = "hello world"
#Search for a sequence that starts with "he", followed by two (any) characters, and an "o":
x = re.findall("he..o", str)
print(x)
```

Output: ['hello']

- ‘^’ starts with

```
str = "hello world"
#Check if the string starts with 'hello':
x = re.findall("^hello", str)
if (x):
 print("Yes, the string starts with 'hello'")
else:
  print("No match")
Output: “Yes, the string starts with 'hello'"
```

- ‘$’ ends with

```
str = "hello world"
#Check if the string ends with 'world':
x = re.findall("world$", str)
if (x):
  print("Yes, the string ends with 'world'")
else:
  print("No match")
```

Output:

- ‘*’ zero or more occurences

```
str = "The rain in Spain falls mainly in the plain!"
#Check if the string contains "ai" followed by 0 or more "x" characters:
x = re.findall("aix*", str)
print(x)
if (x):
  print("Yes, there is at least one match!")
else:
  print("No match")
```

Output: [‘ai’,’ai’,’ai’,’ai’]

- ‘+’ one or more occurences

```
str = “hello heley”
#Check if the string contains “e" followed by 1 or more “l" characters:
x = re.findall(“el+", str)
print(x)
Output=[‘ell’, ‘el’]
x = re.findall(“e+", str)
x = re.findall(“el*", str)
x=re.findall('[a-z]',str)
```

```
str="hello 8 9 89«
x=re.findall('[0-9][0-9]',str)
x=re.findall('[012]',str)
```

Special Sequences

1. ‘\A’

- Returns a match if the specified characters are at the beginning of the string

```
x = re.findall("\AThe", str)
```

2. ‘\d’ match digits

3. ‘\D’ Returns a match where the string DOES NOT contain digits

4. ‘\s’ Returns a match where the string contains a white space character

```
x = re.findall("\s",” The rain in the spain”)
```

- ‘\S’ Returns a match where the string DOES NOT contain a white space character

```
x = re.findall("\S", str)
```

- ‘\w’ Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)
- ‘\z’ Returns a match if the specified characters are at the end of the string

```
x = re.findall("Spain\Z", str)
```

Sets

Set	Description
[arn]	Returns a match where one of the specified characters (`a` , `r` , or `n`) are present
[a-n]	Returns a match for any lower case character, alphabetically between `a` and `n`
[^arn]	Returns a match for any character EXCEPT `a` , `r` , and `n`
[0123]	Returns a match where any of the specified digits (`0` , `1` , `2` , or `3`) are present

RegEx functions

- Findall: returns all the matches else empty list as an output

```
import re
str = "The rain in Spain"
x = re.findall("ai", str)
print(x)
```

Output [‘ai’,’ai’]

- Search():

The search() function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned

```
str = "The rain in Spain"
x = re.search("\s", str)
print("The first white-space character is located in position:",
x.start())
```

Output 3

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.

.string returns the string passed into the function

.group() returns the part of the string where there was a match

Split(): The split() function returns a list where the string has been split at each match

```
x = re.split("\s", str)
```

using maxsplit parameter

```
x = re.split("\s", str, 1)
```

Sub() function:

```
x = re.sub("\s", "9", str)
```

With count parameter

```
x = re.sub("\s", "9", str, 2)
```

Module 9 : Exceptions Handling

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them:

- **Exception Handling:** This would be covered in this tutorial.

What is Exception?

- An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
- In general, when a Python script encounters a situation that it can't cope with, it raises an exception. An exception is a Python object that represents an error.
- When a Python script raises an exception, it must either handle the exception immediately otherwise it would terminate and come out.

Handling an exception:

- If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

Syntax:

```
try:
  You do your operations here;
  ......................
except Exception I:
  If there is ExceptionI, then execute this block.
except Exception II:
  If there is ExceptionII, then execute this block.
  ......................
else:
  If there is no exception then execute this block.
```

Here are few important points above the above mentioned syntax:

- A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.
- You can also provide a generic except clause, which handles any exception.
- After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.
- The else-block is a good place for code that does not need the try: block's protection.

Example:

```
try:
  fh = open("testfile", "w")
  fh.write("This is my test file for exception
  handling!!")
except IOError:
  print "Error: can\'t find file or read data"
else:
  print "Written content in the file
  successfully"
fh.close()
```

- This will produce following result:

```
Written content in the file successfully
```

```
try:
        a = float(input('first number --> '))
        b = float(input('second number --> '))
        z = a / b
        print(z)

except ArithmeticError:
        print(“Divide by Zero”)
```

The *except* clause with no exceptions:

You can also use the except statement with no exceptions defined as follows:

```
try:
  You do your operations here;
  ......................
except:
  If there is any exception, then execute this
  block. ......................
else:
  If there is no exception then execute this
  block.
```

This kind of a **try-except** statement catches all the exceptions that occur. Using this kind of try-except statement is not considered a good programming practice, though, because it catches all exceptions but does not make the programmer identify the root cause of the problem that may occur.

The *except* clause with multiple exceptions:

You can also use the same *except* statement to handle multiple exceptions as follows:

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]):
   If there is any exception from the given exception
   list, then execute this block
   ......................
else:
   If there is no exception then execute this block.
```

Standard Exceptions:

Here is a list standard Exceptions available in Python: Standard Exceptions

The try-finally clause:

You can use a **finally:** block along with a **try:** block. The finally block is a place to put any code that must execute, whether the try-block raised an exception or not. The syntax of the try-finally statement is this:

```
try:
   You do your operations here;
   ......................
   Due to any exception, this may be skipped.
finally:
   This would always be executed.
   ......................
```

Note that you can provide except clause(s), or a finally clause, but not both. You can not use *else* clause as well along with a finally clause.

Example:

```
try:
  fh = open("testfile", "w")
  fh.write("This is my test file for exception
  handling!!")
finally:
  print "Error: can\'t find file or read data"
```

If you do not have permission to open the file in writing mode then this will produce following result:

```
Error: can't find file or read data
```

Reading and Writing Files:

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

The *write()* Method:

- The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.
- The write() method does not add a newline character ('\n') to the end of the string:

Syntax:

```
fileObject.write(string);
```

Argument of an Exception:

An exception can have an *argument*, which is a value that gives additional information about the problem. The contents of the argument vary by exception. You capture an exception's argument by supplying a variable in the except clause as follows:

```
try:
   You do your operations here;
   ......................
except ExceptionType, Argument:
   You can print value of Argument here...
```

- If you are writing the code to handle a single exception, you can have a variable follow the name of the exception in the except statement. If you are trapping multiple exceptions, you can have a variable follow the tuple of the exception.
- This variable will receive the value of the exception mostly containing the cause of the exception. The variable can receive a single value or multiple values in the form of a tuple. This tuple usually contains the error string, the error number, and an error location.

Example:

Following is an example for a single exception:

```
def temp_convert(var):
  try:
      return int(var)
  except ValueError, Argument:
      print "The argument does not contain
  numbers\n", Argument
temp_convert("x");
```

- This would produce following result:

```
The argument does not contain numbers
invalid literal for int() with base 10: 'x'
```

Raising an exceptions:

You can raise exceptions in several ways by using the raise statement. The general syntax for the **raise** statement.

Syntax:

```
raise [Exception [, args [, traceback]]]
```

- Here *Exception* is the type of exception (for example, NameError) and *argument* is a value for the exception argument. The argument is optional; if not supplied, the exception argument is None.
- The final argument, traceback, is also optional (and rarely used in practice), and, if present, is the traceback object used for the exception

Example:

```
def functionName( level ):
  if level < 1:
   raise "Invalid level!", level
  # The code below to this would not be executed
  # if we raise the exception
```

Note: In order to catch an exception, an "except" clause must refer to the same exception thrown either class object or simple string. For example to capture above exception we must write our except clause as follows:

```
try:
  Business Logic here...
except "Invalid level!":
  Exception handling here...
else:
  Rest of the code here...
```

User-Defined Exceptions:

- Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.
- Here is an example related to *RuntimeError*. Here a class is created that is subclassed from *RuntimeError*. This is useful when you need to display more specific information when an exception is caught.
- In the try block, the user-defined exception is raised and caught in the except block. The variable e is used to create an instance of the class Networkerror.

```
class Networkerror(RuntimeError):
  def __init__(self, arg):
    self.args = arg
```

- So once you defined above class, you can raise your exception as follows:

```
try:
  raise Networkerror("Bad hostname")
except Networkerror,e:
  print e.args
```

P P SAVANI UNIVERSITY

School of Engineering

FILES IN PYTHON

Barkha Wadhvani
Assistant Professor
P.P. Savani University
School Of Engineering

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION

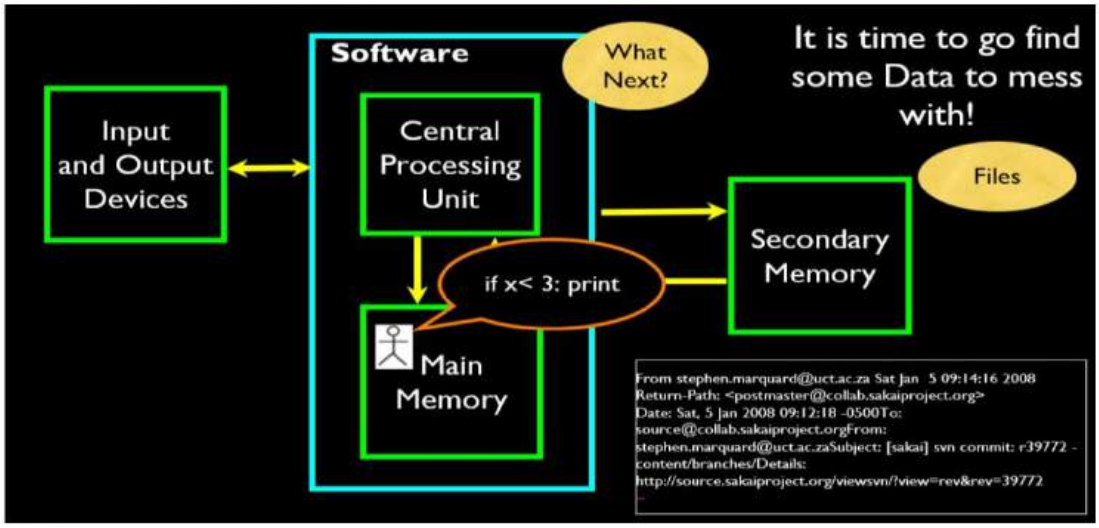
- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).

- A file operation takes place in the following order:
  - ✓Open a file
  - ✓Read or write (perform operation)
  - ✓Close the file

P P SAVANI UNIVERSITY

School of Engineering

INTRODUCTION



Barkha Wadhvani

3

P P SAVANI UNIVERSITY

School of Engineering

OPENING FILES

- Before we can read the contents of the file we must tell Python which file we are going to work with and what we will be doing with the file.
- This is done with the open() function.
- Open() returns a "file handle" - a variable used to perform operations on the file.
- Kind of like "File -> Open" in a Word Processor.

Barkha Wadhvani

4

P P SAVANI UNIVERSITY

School of Engineering

FILE MODES

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Barkha Wadhvani

5

P P SAVANI UNIVERSITY

School of Engineering

OPENING FILES

Syntax

handle = open(filename, mode, <buffering>)

Example

fhand = open('FileDemo.txt','r') # Read File

f = open("test.txt",'w') # write in text mode

f = open("img.bmp",'r+b') # read and write in binary mode

- Open() returns a handle use to manipulate the file
- filename is a string
- mode is optional

Barkha Wadhvani

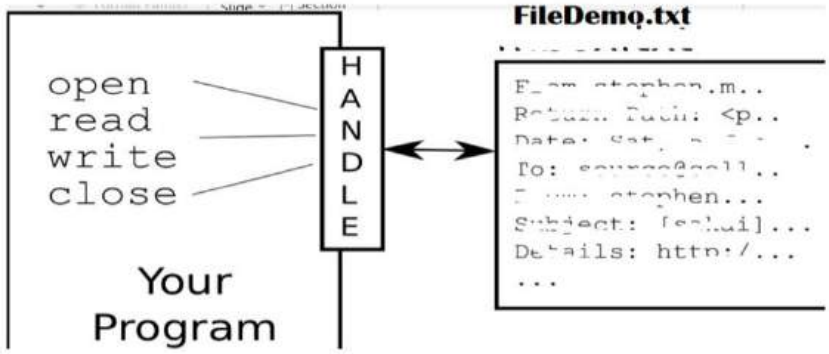
6

P P SAVANI UNIVERSITY

School of Engineering

WHAT IS HANDLE?

- handle = open(FIileDemo.txt')
- print handle
- <open file 'FileDemo.txt', mode 'r' at 0x1005088b0>



Barkha Wadhvani

7

P P SAVANI UNIVERSITY

School of Engineering

FILE OBJECT ATTRIBUTES

- Once a file is opened and you have one file object, you can get various information related to that file.

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.

Barkha Wadhvani

8

P P SAVANI UNIVERSITY

School of Engineering

CLOSE THE FILE

- The close() method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done.

- **Python automatically closes a file when the reference object of a file is reassigned to another file.** It is a good practice to use the close() method to close a file.

Syntax :

fileObject.close()

Barkha Wadhvani

9

P P SAVANI UNIVERSITY

School of Engineering

OPEN FILE DEMO

```
#Open File in Read Mode
handle = open('D://FileDemo.txt',"r")
#handle holds the file object if successfully opened
print(handle)

if handle:
    print("File opened successfully")

#Close the file
handle.close()
```


```
<_io.TextIOWrapper name='D://FileDemo.txt' mode='r' encoding='cp1252'>
File opened successfully
```

Barkha Wadhvani

10

P P SAVANI UNIVERSITY

School of Engineering

READ FILE

- The read() method returns the specified number of bytes from the file into a single string.
- Default is -1 which means the whole file.

Syntax:

handle.read()

P P SAVANI UNIVERSITY

School of Engineering

READ FILE DEMO

```
#Open File in Read Mode
handle = open('D://FileDemo.txt',"r")

if handle:
    #Read file contents
    content=handle.read()
    print("File contents are as below \n", content)

#Close the file
handle.close()
```

```
File contents are as below
 Computer engineering (CoE or CpE) is a branch of electrical engineering that in
tegrates several fields of computer science and electronic engineering required
to develop computer hardware and software.[1] Computer engineers usually have tr
aining in electronic engineering, software design, and hardware-software integra
tion instead of only software engineering or electronic engineering. It uses the
 techniques and principles of electrical engineering and computer science, but a
lso covers areas such as artificial intelligence (AI), robotics, computer networ
ks, computer architecture and operating systems. Computer engineers are involved
 in many hardware and software aspects of computing, from the design of individu
al microcontrollers, microprocessors, personal computers, and supercomputers, to
 circuit design. This field of engineering not only focuses on how computer syst
ems themselves work but also how they integrate into the larger picture.[2] Robo
ts are one of the applications of computer engineering.
```

P P SAVANI UNIVERSITY

School of Engineering

READ LINES OF A FILE

```
#Open File
handle = open('D://FileDemo.txt','r')

if handle:
    #Read File
    content=handle.readline()
    print("Below is a Line\n", content)

    #content=handle.readlines() #returns List of all lines as an elements
    #print("Below are all Lines\n", content)

#Close the file
handle.close()
```

```
Below is a Line
 This is CE Era.
```

FileDemo.txt - Notepad

File Edit View

```
This is CE Era.

This is an IT Era too.
```

Barkha Wadhvani

13

P P SAVANI UNIVERSITY

School of Engineering

COUNT NUMBER OF LINES IN A FILE

```
# Count Lines in a File
handle = open('D://FileDemo.txt','r')

if handle:
    content=handle.readlines() # Returns List of Lines
    print("Below are all Lines", content)
    count=0
    # Count number of Lines
    for line in content:
        count=count+1

print("Number of Lines in a file",count)
#Close the file
handle.close()
```

FileDemo.txt - Notepad

File Edit View

```
This is CE Era.

This is an IT Era too.
```

```
========== RESTART: D:/Python Material/Python Programs/FileDemo11.py ==========
Below are all Lines ['This is CE Era.\n', '\n', 'This is an IT Era too.']
Number of Lines in a file 3
>>
```

Barkha Wadhvani

14

P P SAVANI UNIVERSITY

School of Engineering

COUNT NUMBER OF LINES AND WORDS IN A FILE

```
# Count number of Lines and words in a File
handle = open('D://FileDemo.txt','r')

if handle:
    content=handle.readlines() # Returns List of Lines
    print("Below are all Lines", content)
    count=0
    countword=1
    # Count number of Lines
    for line in content:
        count=count+1
        for word in line:
            if word==' ':
                countword+=1

print("Number of Lines in a file",count)
print("Number of words in a file",countword)
#Close the file
handle.close()
```


```
Below are all Lines ['This is CE Era.\n', '\n', 'This is an IT Era too.']
Number of Lines in a file 3
Number of words in a file 1
```

Barkha Wadhvani

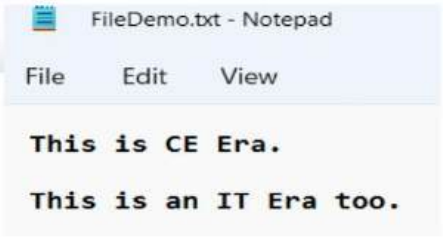
15

P P SAVANI UNIVERSITY

School of Engineering

READ FILE THROUGH LOOP

```
#Open File
handle = open('D://FileDemo.txt','r')

if handle:
    #Read File through Loop
    for line in handle:
        print(line) # Line variable contains each line of the file

#Close the file
handle.close()
```




```
This is CE Era.


This is an IT Era too.
>>>
```

Barkha Wadhvani

16

P P SAVANI UNIVERSITY

School of Engineering

FILE POINTER POSITION

- The tell() method which is used to print the byte number at which the file pointer currently exists.
- Initially the file pointer is at location 0.

Syntax

- **Handle.tell()**

P P SAVANI UNIVERSITY

School of Engineering

FILE POINTER POSITION DEMO

```
#Open File
handle = open('D://FileDemo.txt','r')

if handle:
    # Initially the filepointer is at 0
    print("Below is a Line\n", handle.tell())

    # Read the content from File
    content=handle.read(10)
    print("Content that we read are : \n", content)

    print("After reading , the handle (File pointer) is at", handle.tell())

#Close the file
handle.close()
```

FileDemo.txt - Notepad

File Edit View

This is CE Era.

This is an IT Era too.

```
Below is a Line
 0
Content that we read are :
 This is CE
After reading , the handle (File pointer) is at 10
>>
```

P P SAVANI UNIVERSITY

School of Engineering

MODIFYING FILE POINTER POSITION

- seek() method which enables us to modify the file pointer position externally.

Syntax

Handle.seek(offset,from)

- **offset:** It refers to the new position of the file pointer within the file.
- **from:** It indicates the reference position from where the bytes are to be moved. If it is set to 0, the beginning of the file is used as the reference position. If it is set to 1, the current position of the file pointer is used as the reference position. If it is set to 2, the end of the file pointer is used as the reference position.

19

Barkha Wadhvani

P P SAVANI UNIVERSITY

School of Engineering

MODIFYING FILE POINTER POSITION DEMO

```
#Open File
handle = open('D://FileDemo.txt','r')

if handle:
    # Initially the filepointer is at 0
    print("File pointer isat location \n", handle.tell())

    # Changing the File pointer location to 10
    handle.seek(10)
    print("After updating pointer,the handle is at", handle.tell())

    # Read the content from File
    content=handle.read()
    print("Content that we read are : \n", content)


#Close the file
handle.close()
```

```
File pointer isat location
 0
After updating pointer,the handle is at 10
Content that we read are :
   Era.

This is an IT Era too.
>>>
```

20

P P SAVANI UNIVERSITY

School of Engineering

WRITE IN A FILE

- The write() method writes a specified text to the file.
- Where the specified text will be inserted depends on the file mode and stream position.
- "a": The text will be inserted at the current file stream position, default at the end of the file.
- "w": The file will be emptied before the text will be inserted at the current file stream position, default 0.

Syntax

Handle.write(byte)

Barkha Wadhvani

21

P P SAVANI UNIVERSITY

School of Engineering

WRITE IN A FILE

```
#Open File
handle = open('D://FileDemo.txt','a')

if handle:
    handle.write("This is CE Era.")
    #Read file contents
    handle = open('D://FileDemo.txt','r')
    content=handle.read()
    print("File contents are as below \n", content)

#Close the file
handle.close()
```

Append mode

```
File contents are as below
 Computer engineering (CoE or CpE) is a branch of electrical engineering that in
tegrates several fields of computer science and electronic engineering required
to develop computer hardware and software.[1] Computer engineers usually have tr
aining in electronic engineering, software design, and hardware-software integra
tion instead of only software engineering or electronic engineering. It uses the
 techniques and principles of electrical engineering and computer science, but a
lso covers areas such as artificial intelligence (AI), robotics, computer networ
ks, computer architecture and operating systems. Computer engineers are involved
 in many hardware and software aspects of computing, from the design of individu
al microcontrollers, microprocessors, personal computers, and supercomputers, to
 circuit design.
This field of engineering not only focuses on how computer systems themselves wo
rk but also how they integrate into the larger picture.[2]
Robots are one of the applications of computer engineering.This is CE Era.
```

Barkha Wadhvani

22

P P SAVANI UNIVERSITY

School of Engineering

WRITE IN A FILE

```
#Open File in write Mode
handle = open('D://FileDemo.txt','w')

if handle:
     handle.write("This is CE Era.")
     #Read file contents
     handle = open('D://FileDemo.txt','r')
     content=handle.read()
     print("File contents are as below \n", content)

#Close the file
handle.close()
```

```
File contents are as below
This is CE Era.
```

Barkha Wadhvani

23

P P SAVANI UNIVERSITY

School of Engineering

FILES IN PYTHON

Barkha Wadhvani
Assistant Professor
P.P. Savani University
School Of Engineering

P P SAVANI UNIVERSITY

School of Engineering

WHAT IS FLASK

- Flask is a web framework written in python, it's a Python module that lets you develop web applications easily.
- It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco.
- It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
- Flask supports extensions that can add application features as if they were implemented in Flask itself.

P P SAVANI UNIVERSITY

School of Engineering

INSTALL FLASK

```
C:\Users\LENOVO>pip install flask
Collecting flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
     ---------------------------------------- 101.5/101.5 KB 343.0 kB/s eta 0:00:00
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     ---------------------------------------- 96.6/96.6 KB 1.4 MB/s eta 0:00:00
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
     ---------------------------------------- 133.1/133.1 KB 2.6 MB/s eta 0:00:00
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
     ---------------------------------------- 232.7/232.7 KB 1.4 MB/s eta 0:00:00
Collecting colorama
  Downloading colorama-0.4.5-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.1-cp310-cp310-win_amd64.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, Werkzeug, Jinja2, click, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.5 flask-2.2.2 itsdangerous-
2.1.2
WARNING: You are using pip version 22.0.4; however, version 22.3 is available.
You should consider upgrading via the 'C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe -m pip install
 --upgrade pip' command.
```

Barkha Wadhvani

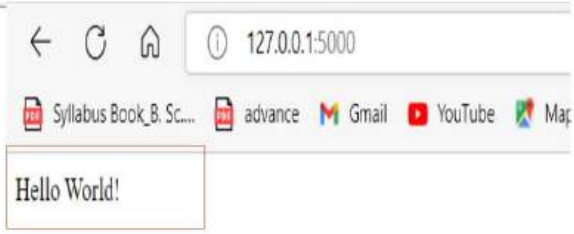
3

P P SAVANI UNIVERSITY

School of Engineering

FLASK DEMO

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```



Print Hello World on your Browser

Barkha Wadhvani

4

P P SAVANI UNIVERSITY

School of Engineering

FLASK

- The current module's name (__name__) is passed as an input to the Flask function Object().
- The Flask class's route() function is a decorator that instructs the application which URL should be used to call the related function. The output of this function will be released when the homepage of the webserver is opened in the browser.
- To start the flask application, we use the run() function.

Barkha Wadhvani

5

P P SAVANI UNIVERSITY

School of Engineering

___NAME___ , ___MAIN___

- **__name__ is a built-in variable which evaluates to the name of the current module.**
- Thus it can be used to check whether the current script is being run on its own or being imported somewhere else by combining it with if statement.
- Example : when File1.py is run directly, the interpreter sets the __name__ variable as __main__ and when it is run through any other File2.py by importing, the __name__ variable is set as the name of the python script.

Barkha Wadhvani

6

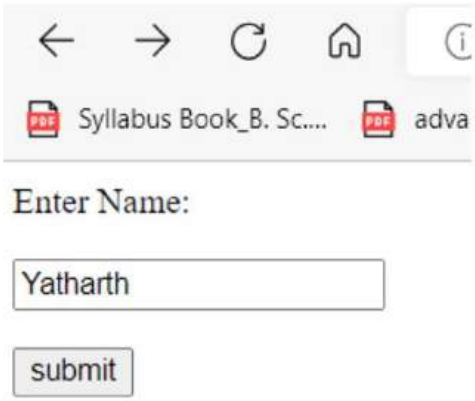
P P SAVANI UNIVERSITY

School of Engineering

FLASK DEMO

HTML page for Accepting the User Name

```
<html>
   <body>
      <form action = "http://localhost:5000/login" method = "post"
         <p>Enter Name:</p>
         <p><input type = "text" name = "nm" /></p>
         <p><input type = "submit" value = "submit" /></p>
      </form>
   </body>
</html>
```



Barkha Wadhvani

7

P P SAVANI UNIVERSITY

School of Engineering

FLASK DEMO

```
from flask import Flask, redirect, url_for, request
app = Flask(__name__)

@app.route('/success/<name>')
def success(name):
    return 'welcome %s' % name

@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        user = request.form['nm']
        return redirect(url_for('success', name=user))
    else:
        user = request.args.get('nm')
        return redirect(url_for('success', name=user))

if __name__ == '__main__':
    app.run(debug=True)
```

Barkha Wadhvani

8