INTRODUCTION TO PYTHON

INTRODUCTION TO PYTHON

1 Python Installation

2 Variable Assignment

```
>>> a=4
>>>a
4
```

3 Type and Type Conversion

str( )	'Jigsaw'	Variables to strings
int( )	5, 7, 89	Variables to integers
float ( )	5.8, 9.7	Variables to floats
bool( )	True, False	Variables to booleans

4 Integer Operations

a+6	Sum of two variables
a-3	Subtraction of two variables
a*2	Multiplication of two variables
a**2	Exponentiation of a variable
a%2	Remainder of a variable
a/2	Division of a variable

5 Strings

```
Course = "Data Science with Python"
print(type(Course))
```

```
<class 'str'>
```

6 String Operations

```
>>>'Jigsaw'  * 2
    'JigsawJigsaw'
>>> 'Jigsaw '  + ' Academy '
    'JigsawAcademy
```

Python has Zero-Based Indexing i.e Index starts at 0
Syntax for indexing - [inclusive:exclusive] for items

Course[3]	will return 'a'
Course[0:4]	will return 'Data'
Course[5:]	returns 'Science with Python'
Course [: -1]	returns all elements except last element n

7 String Methods

Course.upper( )	String to uppercase
Course.lower( )	String to lowercase
Course.count(' i ')	Count String elements
Course.replace(' i ', ' e')	Replace elements
Course.strip( )	Strip Whitespaces

INTRODUCTION TO PYTHON

1 Lists

list1 = [1,2,3,4,5]
list2 = [[1, ' a ',2, ' b '],[3, ' c ',4, ' d ']]

- Single Value for *collections of elements*.
- Contain any type - values can be floats, integers, Booleans, strings even lists
- Create List by using a *squared bracket*.
- *Can Contain different types* in a single list

2 Selecting List Elements

- **Subset**

list1 [0] select the first item from list1
list1[-2] select the second last item

- **Slicing**

list1 [1:3] select the item at index 1 & 2
list1 [1:] select items after index 0
list1 [: 3] select items before index 3
list3 = list1[:] } copy list1 to a new list3
list3 = list1.copy() }

- **Subsetting Lists of Lists**

list2 [1][0:3] selects first three items from second list in list2

3 List Operations

```
>>>list1  + list1
      [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
>>> list1 * 2
      [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
>>> list1[1] = 10       #Change items in a list
      [1, 10, 3, 4, 5]
>>> list1[0:2] = ['a',10] #Replace items in a list
      ['a', 10, 3, 4, 5]
>>> list3 = sorted(list1, reverse = True)
      [5, 4, 3, 2, 1]
```

4 List Methods

list1.index('a')	Get the index of an item
list1.count('a')	Count an item
list1.append('i')	Append an item at a time
list1.remove(' i ')	Remove an item
del(list1[0:3])	Remove items from a list
list1.reverse( )	Reverse the list
list1.extend('i')	Append an item
list1.pop(-1)	Remove an item stating index position
list1.insert(5,'i')	Insert item 'i' in the position of index
list1.sort( )	Sort the List

INTRODUCTION TO PYTHON

1 Copying a List

```
Sample_new = Sample.copy()
Sample_new = Sample[:]
```

2 Iterators and Iterables

Iteratable is an object that can be looped over
Eg- String, Integer, Dictionary, Tuple, File
Iterator is the object that can be iterated upon

3 For Loop in Python

```
Location = ['Delhi','Mumbai','Chennai']
for places in location:
        print(places)
```

Location – is a list also an iterables
Places is an iterator used to loop over an iterables

#Write a loop to square elements in a list:-

```
sq = [1,2,3,4]
sq2 = [ ]
for i in sq:
    sq2.append(i**2)
print(sq2)
```

4 List Comprehension

- Use single lines instead of for loops
- Can be used over iterables/iterator
- Can consist of conditions

5 List Comprehension- Example

Syntax for List Comprehension –

[output_expression iterator iterable condition]

```
age = [23,12,67,78,84]
new_age = [ ]
For y in age:
        new_age.append(age+1)
print (new_age)
[24,13,68,79,85]
```

A list comprehension –

new_age = [y +1 for y in age]

6 More Examples

```
[x**2 for x in sq ]  #Square of all elements in a List
[x**2 for x in sq if x > 2] #List Comprehension with
                            condition
a=[1,2,3,4]
b=['a','b','c','d']
[[x,y] for x in a for y in b if a.index(x)==b.index(y)]
 Out: [[1, 'a'], [2, 'b'], [3, 'c'], [4, 'd']]
```

INTRODUCTION TO PYTHON

1 Lambdas and Map

Lambdas and map can replicate the functionality of list comprehension.
Lambda operator or lambda function is a way to create small anonymous functions.

2 Example

```
mass=[45,55,65,76]
ht=[1.65,1.70,1.55,1.80]
bmi=map(lambda x,y:x/y**2,mass,ht)
print (list(bmi))
```

3 Basic Syntax for Lambda

lambda arguments : expression

lambda operator can have any number of arguments, but it can have only one expression.
It returns a function object which can be assigned to any variable.

```
def add(x, y):
    return x + y
# Call the function
add(2, 3)  # Output: 5
#convert the above function into a lambda function
add = lambda x, y : x + y
print add(2, 3) # Output: 5
```

4 Map Object

- map functions expects a function object and any number of iterables like list, dictionary, etc.

5 Example

```
def multiply2(x):
  return x * 2
```

map(multiply2, [1, 2, 3, 4]) # Output [2, 4, 6, 8]
#map executes multiply2 function for each element in the list i.e. 1, 2, 3, 4 and returns [2, 4, 6, 8]

#write the above code using map and lambda.
map(lambda x : x*2, [1, 2, 3, 4]) #Output [2, 4, 6, 8]

6 Generator Expression

List Comprehensions returns a list whereas Generators return a generator object.
Does not store list in memory/ It does not construct a list
Looping over a generator expression produces elements of the list.
Use ( ) instead of [ ] to create a generator. Rest is same as list comprehension.
Example: comp = (x for x in range(5))

INTRODUCTION TO PYTHON

1 Python Functions

Function is a reusuable code aimed at solving a particular task.
Function is used with a round bracket "()"

2 Popular Functions

abs( ) Returns the absolute value of a no.
dir( ) Returns funcs, mthds, attribts of an obj
enumerate( ) Adds index to an iterable
float( ) Returns a floating point no. from a no. or string.
help( ) Invoke the Built-in help system
len( ) Returns the length of an object
map( ) Returns an iterator that applies function to every item of iterable.
max ( ) Returns the largest item in an iterable
min ( ) Returns the smallest item in an iterable
next( ) Retrieve the next item from the iterator
print( ) Prints the content of an object
round(no,ndigits) Returns number rounded to ndigits precision after the decimal point.
sorted( ) Returns a new sorted list from the items in an iterable.
zip(*iterables) Make an iterator that aggregates elements from each of the iterables.

3 Python Methods

Python objects have associated methods depending on type. These are functions belonging to Python objects.
Method is used as a suffix to an object.

4 Methods Examples

```
a = " data"
a.capitalize() #converts the first character of a string to capital (uppercase) letter
Output: 'Data'
```

capitalize(), replace() are methods for str
index(),count() for list
bit_length(), conjugate() for float
Index() can be applied on both strings and list

5 Opening a File in Python

Reading a text file

```
f=open('file.txt','r')
fo=f.read()
f.close()
```

OR

```
with open ("file.text") as txt
Data = file.read( )
```

Writing to a text file

```
f=open('write.txt','w')
for dt in fo:
    f.write(dt+'\n')
f.close( )
```

INTRODUCTION TO PYTHON

1 Tuple

Tuples are immutable objects. Elements of a tuple cannot be modified at place.
t = (32,34,87,56)

2

- They are iterables hence they can be looped over.
- Elements can be accessed by indexing
- Can contain heterogeneous elements
- t(0) = 19 #will give an error

3 Dictionaries

Dictionaries are key:value pairs
Created using curly {} brackets
Inc = {"Rose":30000,"Joy":40000}

4

- Keys have to be immutable objects
- The values can be updated,selected or deleted using keys
- Dictionary is indexed using unique keys unlike lists that are indexed using range of numbers

5 Dictionary Methods

enrol = {"EPBA" : 20, " PGPDM" : 50}
enrol.keys() returns the keys EPBA and PGPDM in the dictionary Enrol

6

enrol.values() | returns *values* in enrol – [20,50]
EPBA in enrol | returns *True* if Key is found
enrol["EPBA"] = 50 changes values in the key EPBA from 20 to 50. values to be accessed using keys of a dictionary.
del(enrol["EPBA"]) deletes a key in a dictionary
enrol.items() prints all items in dictionary *enrol* including keys and values

enrol = {"EPBA" : [20,30], " PGPDM" : [40,50]} #dictionary
enrol["EPBA"][0] – returns 20. Index values using keys

7 Dictionary of Dictionaries

```
Income = { 'Ramesh': { 'Gender':'male', 'Income':40000},
           'Ana': { 'Gender':'female', 'Income':66000 } }

# Print out the Income of Ramesh
print(Income['Ramesh']['Income'])
```