**Python Training:**

**Day1 : 20 Feb: 10am to 12.30pm**

Python Introduction:

**Compiled Programming lang:**

Source code -----------------🡪m/c code ---------🡪O/P

compiler

Execution – Faster

**Interpreted scripting lang:**

Source code ---------------------------------🡪O/P

Python interpreter. Line by line execution

First.py -------------------------🡪O/P

**Python**

**C:\python First.py**

Execution/Performace – Slower

Development cycle – Faster

Clear syntax

Huge Library support

Data structures provided:

Int float

String

Tuple, List

Set, Dictionary

Remarkable power: i) Procedural oriented way

ii)OOP

Open Source

No pre-defined Keywords to define data types

Case sensitive

Indentation :

# comment line

Num1 = 100 # <class ‘int’>. Num1 is varaiable name – identifier

print(Num1)

Num2 = 200 # <class ‘int’>

Pi = 3.14 #<class ‘float’>

S1 = ‘Persistent’ #<class ‘str’>

S2 =”Welcome to Python session!!!” #<class ‘str’>

S3 = “”” Pune “”” #<class ‘str’>

Ans = Num1 + Num2 #’int’ + ‘int’ 300

#Result = Num1 + S1# RuntimeError **TypeError. ‘int’ + ‘str’ terminated the program execution**

print (x1) #**NameError – 1 of the pre-defined Exception**

print(“END!!!”)

**Powerful Typing:**

**Portable**

**Relative Faster to some of the other Interpreted lang**

Python Program execution:

Pre-requisite : i) Text Editor

ii) Python Soft

python.org **Python 3**.x version

if u are comfortable with any IDE like Eclipse, PyCharm, VisualStudio... u can use them... we will see all these in details eventually

Execution:

1. Open DOS prompt

And type **python**

>>>. Python interactive prompt

>>>num1 = 100

>>>num1

100

>>>s1 = ‘ABC’

>>>s1

ABC

>>>num1 + s1

TypeError

2)Create a file **First.py**

Notepad/Notepadd++

**C:\demos\day1\python First.py**

**Path env variable**

**If python not the part of ‘path’ system env var, then set it-**

**1)on dos prompt –**

**C:\set path=** **C:\Program Files\Python37;%path%**

**OR**

**Go to System PC🡪rt click-🡪properties**

**SyntaxError**

**RunTime pre defined errors**

**Indentation Errors:**

Indented code block - if elif else

loops - while , loop

functions

OOP class

exception handling - try except finally

**+, \_ \*, \*\* / %**

**Relational operators: == != < > <= >=**

**Pre-defined functions / generic function :**input(), int() float() hex() oct() …

**Day 2: 21st Feb 2023**

Documentation

IDE – IDLE editor

Operators, Keywords, control loops

Int float

String: ‘’ “” “”” “”” definations

Raw string : r”” R””

String data access : slicing syntax notations

String object methods

Mutable/ Immutable data

for loop

range()

**Q**

Whats differance between Dynamic and Static Typing

Puython: Dynamic typing

Num1 + s1 #TypeError

Int(“abc”) #ValueError

Num1= 100

Num1 = “abc”

Other lang : C, C++ Static

Int x1 = 100

X1=99

X1=”abc” #not possible

**Keywords:**

Logical operators **: and or not**

Error Handling: **assert try except finally raise exec**

Functions: **def lambda return global**

Control Structures**: if elif else for while break continue pass**

**OOP: class**

**Modules/Packages : import , from**

**print, del x1,**

membership operators: **is in**

**Try the code:**

s1 = “100” #str

num1 = int(s1) #int

s2 = “abc” #str

num2 = int(s2) #ValueError

Operators set:

**Data Structures:**

**Sequence Type:** String , Tuple, List

Immutable : String , Tuple

Mutable: List

Numerical Indexing concept

**Random/Unordered :** Set, Dictionary – Mutable

**NO N**umerical Indexing concept

**-------**

**BOOLEAN VALUE: True/False**

**if num1 == 100: #True**

**-------**

**IDLE : Comes with Python installation**

Color, indentations, self-intelligence help

**Start menu->python3.7->idle->**

**Open python prompt**

**>>>**

**for i in s1: #s1 iterable objects – str, tuple, list, set , dict**

**print (i) #i is element value**

**list(range(5)) # range(0,5) #[0,1,2,3,4]**

**range(3,8) # range(3,8). 3,4,5,6,7**

**for i in range(1,6)):**

**print(i)**

**Assignemnets: pyrmid pattern: nested for loop, range() usage : logic**

**pr**

**Queries:**

if s1== s1[::-1]: #str == str True/Fasle

print("It is palindrome." if n==n[::-1] else "It is not palindrome.")

**Day 3: 22nd Feb 2023**

Program Flow Control in Python …continue

String …cont .. “”” “””– 3 usage – i) Def string ii) Comment block iii) for multiline  
 if-elif-else  
 for loops  
 range()  
 for and continue statement  
 break  
 while

pass

**list, tuple** definitions, slicing access

s1[0]

s1[2:5]

**Methods:**

**String :**

String: s1.upper() lower() isupper() islower() count() append()

**List:**

**Append() extend() reverse() insert() pop() remove() sort() count() index()**

**Tuple**

**Count(), index()**

**(None, )**

**Day 4: 1 Mar 2023 (Wed)**

words = ['Aaaa', 'bb', 'cccccccc', 'zzzzzzzzzzzz']

words.sort()#Default sorting ascending

words.sort(key = len)#Functional Programming len(words[0])->4 len(words[1])->2

**sortedlist = sorted()**

**reverseObj = reversed(). #for loop**

**reverselist = list(reverseObj)**

**newList = str.split(“delimiter”)**

**pass**

**list1 = [10,20,999,888,777]**

List Comprehension

Newlist = [n1 for n1 in list1 if n1<100]

**Random/Unordered :** Set, {} set() unique elements

Dictionary – Mutable {key:value, }. {} is a default dict

Dict1[key]

Keys can’t be modified – immutable – int, float, string, tuple

Values : can be anything - int, float, string, tuple, set, dict…

**NO N**umerical Indexing concept.

**for loop**

**Methods : set , dictionary**

**Day 5: 2 Mar 2023 (Thu)**

Set, Dict, methods…contd…

Dictionary Comprehension = {i:i\*i for i in list1 }

Complex Data structures: Dict with inner list, Dict with inner Dict

**Dict with value as int**

**Dict with value as inner list**

**Dict of inner Dict**

**Day6: 7 Mar 2023 (Tue)**

**Functions:**

default args

keyword arg passing

variable number of arguments list

lambda – anonymous function

scope of the var : global

**Day7: 8 Mar 2023 (Wed)**

**Functions ….contd….**

Return statement

Functions Arguments: Positional, Default, Keyword, \*args and \*\*kwargs

Sequence of parameters to be followed

Default after non- default in the definition of function

Keyword arg after non-keywoard/positional parameter

If not followed – it gives us SyntaxError

TypeError : number of args are not matched

variable number of arguments list - \*args and \*\*kwargs

lambda – anonymous function

scope of the var : global

**pre – defined functions:** print() len() int() float() chr() ord(), hex(), oct() sum() sorted() reverseitr = reversed()

avg() m in() max()

type()

id()

input()

range()

exit()

list() tuple() set() dict()

for I in range(len(list1)):

I – index

List1[i]

Open()

**File Handling**

Opening and Reading Files

IN = open()

IN.readline()

#List1 = IN.readlines()

Str1 = IN.read()

Reading Files: Tell, Seek and Cursors

Reading Files into a List

Writing to Text Files

Appending to Text Files

File Processing

with Keyword

**Day8: 9 Mar 2023 (Thu)**

Modules/Packages

1.

import mymath

myamath.area(5)

2.

from mymath import \*

from mymath import pi, fib

pi

fib()

import mymath as m1

m1.area(4)

import searches for all required python libraries from the folder paths known by sys.path variable (list) – this contains by default current foldr, lib folder of Python

sys.path.append(“folder path of module file”)

ModuleNotFoundError

Package folder – collection of module multiple files

\_\_init\_\_.py file

from MathsCal import \*

area()

**Day9,10:14 Mar 2023 (Tue), 15 Mar 2023 (Wed)**

Exception Handling

Exception – User defined

**OOP :**

class , Object, Methods – instance method, class method/static method, class level variable

Inheritance

Special methods - \_\_str\_\_, \_\_repr\_\_ , \_\_add\_\_

Constructor method : \_\_init\_\_(self):

Ob1 = SayClass() #ref

def display(self):

ob1.display() #ob1 itseelf passed as parameter.

**Day11: Mar 20 2023 (Mon)**

**OOP : *continued…..***

class method/static method, class level variable

Inheritance

Special methods - \_\_str\_\_, \_\_repr\_\_ , \_\_add\_\_

**Regex**

**import re**

Regular expression - searching for pattern in a string

Pattern - ?? string “ PSL” sequential set of characaters – alphabet, digit, spaces , special chars

S1 = “Welcome to PSL”

S1.

re.compile(“pattern”)

re.match(pattern, str)

re.search(pattern, str)

special char: . $ ^ | [] {} \* + ?

re.sub(pattern , substitution, str)

re.split(pattern, split delimiter , str)

**Python Error:**

**1)SyntaxError**

**Python interpreter executes ur code in 2 phases-**

**1st phase – it checks for the Syntax**

**2nd Phase – it executes line by line, and during that if there are any Runtime errors, they will be reported**

**Example1:**

print("Welcome to Python Session!!)#SyntaxError: EOL while scanning string literal

**Example2:**

s2 = "That"s"#SyntaxError

**Exampl3:**

dislay()#NameError: name 'dislay' is not defined. Did you mean: 'display'?

**IndentationError:** expected an indented block

Example:

if (num1 == 100):

print("Inside if block....")

**Example4:**

add(num2 = 9,8)#SyntaxError

def test(n1 = 100, n2):#

pass

**Example5:**

#def add(num1, num2,\*\*y, \*x):#SyntaxError

**Example6:**

def func1():

global a = 100#SyntaxError

**2)RunTimeError – Exception : are reported by Python interpreter at the run time i.e. 2nd phase execution**

**1)NameError**

Example1:

#print(x1)#NameError: name 'x1' is not defined

**2)TypeError**

**Example1:**

num1 = 100

s1 =”ABC”

num1 + s1 #TypeError: unsupported operand type(s) for +: 'int' and 'str'

**Example2:**

s1 ="012345"

print("s1 = ", s1)

s1[0] = "A"#TypeError: 'str' object does not support item assignment

**Example 3:**

t1 = 10,20,30

print("t1 = ", t1, " Type = ", type(t1), " ID = ", id(t1))

#t1 = (10, 20, 30) Type = <class 'tuple'> ID = 4458935680

t1[0]= 100#TypeError: 'tuple' object does not support item assignment

**Example4:**

list2 = [99,88,10,33,555,"abc",345,199,56789]

list2.sort()#TypeError: '<' not supported between instances of 'str' and 'int'

**Example5:**

s1 = {1,2,3,3,3,3}

#print(s1[0])#TypeError: 'set' object is not subscriptable

**Example6:**

s3 ={[10,20,[1,2,3,4,5]]} #TypeError: unhashable type: 'list'

**Example 7:**

def test1():pass

test1(100)#TypeError

**Example 8:**

def add(num1, num2, \*x):pass

add(1,2, x1=100, y1=200)#TypeError: add() got an unexpected keyword argument 'x1'

**3)ValueError**

**Example1:**

s1 = "abc"

num1 = int(s1)#ValueError: invalid literal for int() with base 10: 'abc'

**Example2:**

t1 = 10,20,30, 30,30

ind = t1.index(202020)

print("Index of 202020 in t1 = ", ind)#ValueError: tuple.index(x): x not in tuple

**4)IndexError**

**Example1:**

list1 = [123, "abc", 3.145,999]

print(list1[5])#IndexError: list index out of range

**5) AttributeError**

list1 = [10,20,"abc", 30]

squareElements = []

#l = [ int(x)\*\*2 for x in list1 if x.isdigit()]

#AttributeError: 'int' object has no attribute 'isdigit'

**6)KeyError**

**Example1:**

empData = {'1a':25000, '2a':35000}

print(empData[0])#KeyError: 0

import mymath

ModuleNotFoundError: No module named 'mymath'