Course Title: Advance of Python Programming

Trainer: M A K Ansari & Rajesh Sola

L&T Technology Services

LTTS GLOBAL ENGINEERING ACADEMY

Agenda

- Certification Coverage
- Learning Resources
- Learning Path
- Assessment Pattern
- Workbook

Version

Version	Reviewed by	Approved by	Remarks
1.0			

Certification Coverage

Advance Python

Test Automation Frameworks

- Robot Automation Framework
- Selenium using Python

Other use cases of Python

- Sensor Interfacing, IOT Protocols
- Speech Recognition & Audio Processing Libraries
- PyQt – GUI Development for Rich Dashboards
- Machine Learning

Significance of Python in Embedded V&V BU Focus Areas

Additional Reference:-

- https://www.python.org/about/apps/

Topics – Coverage (Level-2)

- **Object Oriented Programming (OOP)**
  - Defining classes
  - Instance methods and data
  - Initializers
  - Class methods
  - Static methods
  - Inheritance
  - Multiple inheritance
  - Exception Handling – try, except, finally
  - Abstract Classes
  - Overloading

- **Regular Expressions**
  - Regular expressions for pattern matching
  - Syntax/Grammar
  - Objects - search & match

- **Modules& Packages**
  - Overview of Python sys module, command line arguments
  - Overview of Python OS module, creating sub process & execute applications, handling input, output & errors
  - Packages overview, how to create

Topics – Coverage (Level-2)

- **Exception Handling & Types of Errors**
- What is Exception?
- Why exception handling?
- Handling exception – try except block
- Try with multi except
- Handling multiple exceptions with single except block
- Finally block
- Try-except-finally
- Try with finally
- Raise keyword
- Custom exceptions / User defined exceptions
- Need to Custom exceptions

- **Multi-threading & Multi Processing**
- Multi-tasking v/s Multi-threading
- Threading module
- Creating thread – inheriting Thread class, Using callable object
- Life cycle of thread
- Sleep (), Join ()
- Synchronization – Lock class – acquire (), release () functions

- **PyTest**
- What Is PyTest?
- Writing Test Functions
- Testing a Package
- Using assert Statements
- Expecting Exceptions
- Marking Test Functions
- Skipping Tests
- Marking Tests as Expecting to Fail
- Running a Subset of Tests
- Parametrized Testing
- PyTest Fixtures
- Sharing Fixtures Through conftest.py
- Using Fixtures for Setup and Teardown
- Built-in Fixtures
- Plugins

Course Pedagogy

Blended Mode of Learning

- Self Paced Resources as mentioned in this slide deck
- Few SME connects – For query resolution or discussion on non-trivial topics (calendar will be published by Tech Praavinya - Program Manager)
- Yammer group – for queries and collaborative discussions

Assessment

- Take up assessment post above learning phase
- Directly Take up assessment, on going through syllabus and a quick walk through of workbook attached in later part of this slide deck.

Learning Resources

Learning Resources

sololearn.com/learning/1073

learnpython.org

py4e.com/lessons

Python for Everybody (PY4E)

Other Tutorials and References

Object Oriented Programming (OOP) Object-Oriented Programming (OOP) in Python 3 – Real Python

Regular Expressions Regular Expressions and Building Regexes in Python – Real Python

Modules& Packages Python Modules and Packages – An Introduction – Real Python

Exception Handling & Types of Errors (Tutorial) Exception and Error Handling in Python - DataCamp

Multi-threading & Multi-Processing

Iterators, Generators, Decorators

PyTest

Hands-on! Software Environment

Interpreter

- Downloading & Installing
- Setting up Environment

IDEs

- PyCharm IDE
- IDLE
- Spyder IDE

Interactive Notebooks

- Jupyter Notebook
- Jupyter Lab

ANACONDA

Online compilers (interprters actually)

Code Visualization

- Python Tutor – Best way of visualizing code flow, stack frames, objects on heap

Python 3.6
(known limitations)

```
1  mylist = [10, 20, 30, 40 ]
2  otherlist = mylist
3  mylist.append(50)
4
5  t1 = (10,20,30)
6  t2 = t1
7  t1 = t1 + (40,)
8
9  x = 10
10 x = x + 5
```

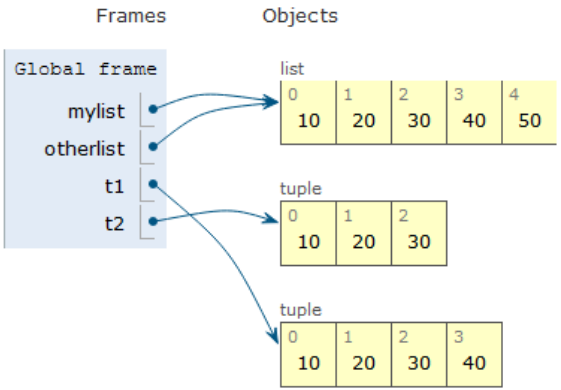
Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 7 of 8

Customize visualization (NEW!)



Helps a lot to realize the facts like everything is an object in python , object id, differentiate mutable vs immutable types, parameter passing methods, self keyword in classes etc.

Learning Path

Learnpython.org

- From learnpython.org, complete the following
  - All the basic topics (Learn The Basics)
  - From Advanced Tutorials
    - Exception Handling
    - Sets
    - List Comprehensions

If you find well familiar with the topics, You may quickly try the examples & exercises embedded in this tutorial

Assessment Pattern

Assessment Pattern

Segment/Part	Mode/Type	Weightage	Objective
Part-1	Multiple Choice Questions	40%	Knowledge Check
Part-2	Coding – Complete the logic for given function(s)	60%	Basic Coding Skill Check

Sample MCQs

```
d1 = { 1 : "A", 2: "B", 3:"C" };
if "B" in  d1:
  print("Yes")
else
  print("No")
```

What’s the output of above code.

A. Yes
B. No
C. TypeError
D. ValueError

In Python _____ keyword, holds reference of object by which an instance method is invoked.

A. this
B. super
C. Self
D. None of the above

```
l1 = [10, 20, 30 ]
l2 = [10, 20, 30 ]
print(l1 == l2)
print(l1 is l2)
```

What’s the output of above code.

A. True, True
B. False, True
C. True, False
D. False, True

Sample Coding Problems

Write a function, which returns true if given string is an isosceles or returns sum of ASCII values of each char

```
def is_isosceles_or_sum_ascii(mystr):
    pass  # TODO your code goes here.

if __name__=="__main__":
    # some test calls to check the
    # function will be placed here
    # don't change main function
```

Write a function, which returns no.of days elapsed from 1st January till given date. Date is provided in the string form of dd/mm/yyyy

```
def count_days(dstr):
    pass  # TODO your code goes here.

if __name__=="__main__":
    # some test calls to check the
    # function will be placed here
    # don't change main function
```

Your code will be validated against some default test cases and some hidden test cases at the backend.

Workbook

Test Your Python Knowledge

- You could have completed self Assessment from sololearn / learnpython / py4e.com by now.
- Online Resources – Quizzes / MCQs
  - https://www.tutorialspoint.com/python/python_online_quiz.htm
  - https://www.javatpoint.com/python-mcq
  - https://www.geeksforgeeks.org/python-multiple-choice-questions/
  - https://www.w3schools.com/quiztest/quiztest.asp

Test Your Python Knowledge

- From realpython.com [on relevant topics]
  - https://realpython.com/quizzes/python-data-types/
  - https://realpython.com/quizzes/python-variables/
  - https://realpython.com/quizzes/python-operators-expressions/
  - https://realpython.com/quizzes/python-conditional-statements/
  - https://realpython.com/quizzes/python-while-loop/
  - https://realpython.com/quizzes/python-lists-tuples/
  - https://realpython.com/quizzes/python-dicts/
  - https://realpython.com/quizzes/python-sets/
  - https://realpython.com/quizzes/python-strings/
  - https://realpython.com/quizzes/python-split-strings/
  - https://realpython.com/quizzes/read-write-files-python/
  - https://realpython.com/quizzes/python-program-structure/
  - https://realpython.com/quizzes/run-python-scripts/

Grooming Coding Skills

- Practice few problems from (Around 10 are recommended, choose python as choice)
  - https://stepik.org/course/3032
- https://py.checkio.org/
- From codewars
  - https://www.codewars.com/collections/basic-python
  - https://www.codewars.com/collections/python-challenges-1
  - https://www.codewars.com/collections/beginners-python-training
- https://www.programiz.com/python-programming/examples

Some more code snippets

```
x = 10
print(type(x)
print(id(x))
x = x + 5          # immutable type
print(id(x))
# everything is an object in python
# what happens with x = x + 5
# int being immutable type
```

```
l1 = [ 10, 20, 30 ]
print(id(t1))
l1.append(40)     # mutable type
print(id(l1))
l2 = [10, 20, 30, 40]
print( id(l1) == id(l2) )
```

```
def test(l1):
    l1.append(50)
    print(len(l1))
    l1 = [60,70,80]
    print(sum(l1))

mylist = [10,20,30,40]
test(mylist)
print(mylist)
```

Visualize above code using Python Tutor
What if a tuple is passed to function, instead of list (pass by value vs pass by reference, in case of immutable, mutable type)

Some more code snippets

```
def test(x, y, compute):
    res = compute(x, y)
    print(res)

def sum(x, y):
  return x + y

test(10, 20, sum)
test(10, 20,
      lambda x, y : x * x – y * y )
```

```
flist = [ sum, diff, multiply, divide ]
flist[index](a,b)
test (a, b, flist[index] )

fdict = { 'S': sum, 'M': multiply };
fdict['S'](a,b)
test (a, b, fdict['M’] )
```

Callback Mechanism, passing functions as arguments to other functions

- Positional Arguments, Keyword Arguments, Default Arguments
- *args, **kwargs, packing & unpacking function arguments
- global keyword, nonlocal keyword

Some more code snippets

Write a function to count no. of armstrong numbers in given range

```
def count_armstrong(start, end):
    # TODO : your logic here
```

Write a function to print frequency of each character in given string

```
def print_frequency(mystr):
    # TODO : your logic here
```

Write a function to count no. of prime numbers in given range

```
def count_primes(start, end):
    # TODO : your logic here
```

In a given list all elements are identical, except one element. Identify the odd one out

```
def diff_element(mylist):
    # TODO : your code here
```

Some more code snippets

```
f = open("sample.txt", "r") as f:
    lines=f.readlines()
    print(lines)
    for line in lines:
        val = int(line)
f.close()

# What if any line has any non
# numerics , while reading file in
# above code,i.e. behavior of code
# due to exception
```

```
with open("sample.txt", "r") as f:
    lines=f.readlines()
    print(lines)
    for line in lines:
        val = int(line)

# What if any line has non numerics
# in above code, while reading file
# in above code
```

```
f = open("simple.txt", "w")
print(dir(f))
```

```
def sum(x,y):
  pass
print(dir(sum))
```

Some more code Snippets

```
Fill the code, with suitable definitions

class Account:
  def __init__(self,id,name,bal):
      pass # TODO: your code here
  def credit ( """ TODO : your code here """):
       self.balance += amount
  def debit (""" TODO : your code here """ ):
       self.balance -= amount
  def getBalance(self):
       pass # TODO : your code here

if __name__=="__main__":
   # Write test code to create objects and
   # invoke instance methods of class
```

```
try :
  c = a / b
except ValueError:
  print("Value Error")
except KeyError:
  print("Key Error")
else:
  print("else block")
finally:
  print("finally block")

# What’s the output of
# above code
```

Thank You !

L&T Technology Services