A Short Course on Python Programming(Day-3)

Ajit Kumar

ajit.pythonclass@gmail.com

March 2, 2017

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Till Now...

1. Basic of Python Language

Till Now...

1. Basic of Python Language
2. If, while, for,function

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators
4. String

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators
4. String
5. List

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators
4. String
5. List
6. Tuple

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators
4. String
5. List
6. Tuple
7. Dictionary

Till Now...

1. Basic of Python Language
2. If, while, for,function
3. Operators
4. String
5. List
6. Tuple
7. Dictionary
8. I/O (from Terminal and File)

Table of Contents

Practice Problem -1

```
# Create a list (numbers) having integer value from 0
   to 1000.

# Create a list (squares) having square of
   numbers(0-1000) list.

# Create a list (exponents) having power of 5 (x ** 5
   ) numbers(0-1000) list.

# Create a list by filtering even numbers from
   exponents list.
```

Practice Solution -1

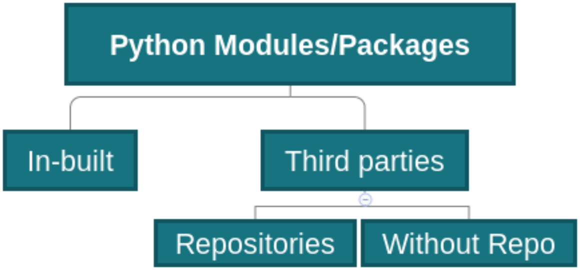
```
numbers = [ number for number in range(1001)]

squares = [ number * number for number in range(1001)]

exponents = [ number ** 5 for number in range(1001)]

evens = [ number ** 5 for number in range(1001) \
         if number % 2 == 0]
```

Table of Contents

What is Modules?

What is Modules?

Python Modules



Importing modules

```
import modulename

from modulename import *

from modulename import methodname
```

OS

```

# http://www.guru99.com/python-tutorials.html

import os

import os.path

from os import path
```

OS

```

print path.exists("foo.txt")
#True
print path.isfile("foo.txt")
#True
print path.isdir("foo.txt")
#False
print path.realpath("foo.txt")
#/home/ajit/Python-Workshop/foo.txt
```

OS

```
print path.split(path.realpath("foo.txt"))

#('/home/ajit/Python-Workshop', 'foo.txt')

print path.getmtime("foo.txt")
#1487878444.06

import time
print time.ctime(path.getmtime("foo.txt"))
#Fri Feb 24 01:04:04 2017
```

3rd Party module Installation

```
# from PyPi i.e. Python repo
# Windows you can have excutable installer also
# pip install modulename

# easy_install modulename

# From the source code

# python setup.py install

# sudo python setup.py install
```

Installation:Example

```
# pip install simplejson

# simplejson-3.10.0.win32-py2.7.exe (md5, pgp)

# https://pypi.python.org/pypi/simplejson#downloads
```

Source:
https://pypi.python.org/pypi/simplejson

Example:A user module

```
# day3modules.py
def main():
    print "Hello Python Modules-from main()!!"

def main2():
    print "Hello Python Modules-from main2()!!"

if __name__ == "__main__":
    main()
    main2()
```

Example:Calling User Module-1

```python
import day3modules

day3modules.main()

day3modules.main2()
```

Example:Calling User Module-2

```python
from day3modules import main

main()

#error
day3modules.main2()

#  day3modules.main2()
# NameError: name 'day3modules' is not defined
```

Table of Contents

Exception

- What is Exception?

Exception

- 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.

Exception

- 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.
- Why it is import to handle exception?

Exception

- 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.
- Why it is import to handle exception?
- When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Exceptions-1

```
# Exception , StopIteration , SystemExit ,

# StandardError , ArithmeticError , OverflowError ,

# FloatingPointError , ZeroDivisonError ,

# AssertionError , AttributeError , EOFError ,

# ImportError , KeyboardInterrupt , LookupError ,
```

Exceptions- 2

```

#IndexError , KeyError , NameError ,
    UnboundLocalError ,

#EnvironmentError , IOError , OSError , SyntaxError,

# IndentationError , SystemError , SystemExit ,

# TypeError , ValueError , RuntimeError ,
    NotImplementedError ,
```

Exception Handling

```
#Exception Handling
#An exception is an event, which occurs during
#the execution of a program that disrupts the
# normal flow of the program’s instructions.
try:

except:

else:

finally:
```

Exception Handling:Template

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Source:http://www.tutorialspoint.com/python/python_exceptions.htm

Exception Handling:Example

```
#!/usr/bin/python

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()
```

Importing modules-Exceptions

```

# import with exception

try:
    import mymodule
except ImportError:
    import mymodule2
```

Source: `https://www.tutorialspoint.com/python/standard_exceptions.htm`

Python Style-1

```
import this

#Beautiful is better than ugly.
#Explicit is better than implicit.
#Simple is better than complex.
#Complex is better than complicated.
#Flat is better than nested.
#Sparse is better than dense.
#Readability counts.
```

Source:
https://www.python.org/dev/peps/pep-0008/

Python Style-2

```
#Special cases aren't special enough to break the
    rules.
#Although practicality beats purity.
#Errors should never pass silently.
#Unless explicitly silenced.
#In the face of ambiguity, refuse the temptation to
    guess.
```

Python Style-3

```
#There should be one-- and preferably only one
   --obvious way to do it.
#Although that way may not be obvious at first unless
   you’re Dutch.
#Now is better than never.
```

Python Style-4

```
#Although never is often better than *right* now.
#If the implementation is hard to explain, it's a bad
    idea.
#If the implementation is easy to explain, it may be
    a good idea.
#Namespaces are one honking great idea -- let's do
    more of those!
```

PEP-8

1. Exceptions are allowed but must be used carefully.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.
5. Use parentheses sparingly.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.
5. Use parentheses sparingly.
6. Indent your code blocks with 4 spaces.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.
5. Use parentheses sparingly.
6. Indent your code blocks with 4 spaces.
7. Two blank lines between top-level definitions, one blank line between method definitions.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.
5. Use parentheses sparingly.
6. Indent your code blocks with 4 spaces.
7. Two blank lines between top-level definitions, one blank line between method definitions.
8. Follow standard typographic rules for the use of spaces around punctuation.

PEP-8

1. Exceptions are allowed but must be used carefully.
2. Avoid global variables.
3. Default Argument Values -Okay in most cases.
4. Maximum line length is 80 characters.
5. Use parentheses sparingly.
6. Indent your code blocks with 4 spaces.
7. Two blank lines between top-level definitions, one blank line between method definitions.
8. Follow standard typographic rules for the use of spaces around punctuation.
9. Generally only one statement per line.

Variable naming

```
#module_name, package_name, ClassName,

#method_name, ExceptionName, function_name,

#GLOBAL_CONSTANT_NAME, global_var_name,

#instance_var_name, function_parameter_name,

#local_var_name.
```

Source:https://google.github.io/styleguide/pyguide.html

A example:List of largest

```python
a = [1, 2, 3, 4, 5]
b = [2, 2, 9, 0, 9]

def pick_the_largest(a, b):
```

Source: https://bradmontgomery.net/blog/2013/04/01/pythons-zip-map-and-lambda/

A example:List of largest

```python
a = [1, 2, 3, 4, 5]
b = [2, 2, 9, 0, 9]

def pick_the_largest(a, b):
    result = [] # A list of the largest values

    # Assume both lists are the same length
    list_length = len(a)
    for i in range(list_length):
        result.append(max(a[i], b[i]))
    return result

print pick_the_largest(a, b)
```

zip()

```
# ZiP()
# This function takes two equal-length collections,
# and merges them together in pairs.

print zip(a, b)

# [(1, 2), (2, 2), (3, 9), (4, 0), (5, 9)]
```

zip()

```
# https://docs.python.org/2/library/functions.html#zip

x = [1, 2, 3]
y = [4, 5, 6]
zipped = zip(x, y)
print zipped

#[(1, 4), (2, 5), (3, 6)]
```

unzip

```python
x2, y2 = zip(*zipped)

print x2,y2
# (1, 2, 3) (4, 5, 6)
print x == list(x2) and y == list(y2)

#True
```

Lambda

```
#lambda

# lambda is just a shorthand to create an anonymous
    function.
# It’s often used to create a one-off function
    (usually for
# scenarios when you need to pass a function
# as a parameter into another function).
# It can take a parameter,
# and it returns the value of an expression.
```

Lambda:How to

```python
# lambda <input>: <expression>

# lambda pair: max(pair)

g = lambda x: x**2
print g(8)
#64
```

map()

```
# map()
# It takes a function, and applies it to each item
# in an iterable (such as a list).

#https://docs.python.org/2/library/functions.html#map

# map(some_function, some_iterable)
```

map():Example

```python
items = [1, 2, 3, 4, 5]
def sqr(x):
    return x ** 2

list(map(sqr, items))

#[1, 4, 9, 16, 25]
```

Largest from two list

```
# # apply the lambda to each item in the zipped list

print map(lambda pair: max(pair), zip(a, b) )

# [2, 2, 9, 4, 9]
```

Largest from two list:variants

```
print list(map(max, zip(a,b)))

print [max(pair) for pair in zip(a,b)]

print [max(ai,bi) for ai, bi in zip(a,b)]

print list(map(max,a,b))
# map return list so no need of typecast
print map(max,a,b)
```

Table of Contents

Assignments:Please refer attached file for details.

- Email the script to `ajit.pythonclass@gmail.com` with subject as **Roll Name Day-3 Mini Project**.

To be Continue...

import time
time.sleep$(7 * 24 * 60 * 60)$

print "Thank you"