Python

Session 1

Sachin

January 22, 2015

Python

- High level programming language
- Used in Scientific computing, Application development, Scripting, etc.

Why Python?

- Easy to learn, read, and modify the code
- Easy to implement
- Object Oriented

Lists

List manipulation

```
empty_list = []
list = [1, 4, 9, 2]                # List
print list                         # [1, 4, 9, 2]
list.append(8)                     # [1, 4, 9, 2, 8]
list.pop()                         # 8
print list                         # [1, 4, 9, 2]
```

Lists

List manipulation- access elements

```
org = ["gnu.org", "emacs.org", "hive.org"] # List
print org[0]                               # gnu.org
print org[2]                               # hive.org

# Last elements
print org[-1]                              # hive.org
print org[-2]                              # emacs.org
```

Lists

List manipulation- slicing

```
org = ["gnu.org", "emacs.org", "hive.org"] # List
print org[:1]  # ['gnu.org']
print org[:2]  # ['gnu.org', 'emacs.org']
print org[:3]  # ['gnu.org', 'emacs.org', 'hive.org']
```

Lists

List manipulation- slicing

```
list = [9, 6, 3, 1, 7, 5, 0, 4, 8]
list[0:3] # [9, 6, 3]
list[1:3] # [6, 3]
list[1:1] # []
list[5:7] # [5, 0]
list[4:5] # ?
```

Lists

List manipulation

```
list = [9, 6, 3]
# Add list
new_list = ["Gandolf", "Gollum", "Aragron"]
my_new_list = list + new_list
# [9, 6, 3, 'Gandolf', 'Gollum', 'Aragron']
len(my_new_list)                # 6
list_in_list = list.append(new_list)
# [9, 6, 3, ['Gandolf', 'Gollum', 'Aragron']]
list_in_list[3][2]              # 'Aragron'
# What if I want to print 'Gandolf'?
```

List

Run for-loop over list

```
new_list = ["Gandolf", "Gollum", "Aragron"]
for item in new_list:
    print item

# Gandolf
# Gollum
# Aragron
```

Dictionary

```
empty_dict = {} # Empty dictionary
status = {
    'stdout': 'Hello',
    'stderr': None,
    'exit': 0,
}

print status['exit'] # 0
print status['stdout'] # 'Hello'

print status.keys() # ['stdout', 'stderr', 'exit']
print status.values() # ['Hello', None, 0]
```

Dictionary

```
status = {
    'stdout': 'Hello',
    'stderr': None,
    'exit': 0,
}

# Change value
print status['exit'] = 1 # 0 -> 1
```

Dictionary

run for-loop over a dictionary

```
numbers = {
    'one': 1,
    'two': 2,
    'three': 3,
    'four': 4
}

for k, v in numbers.iteritems():
    print k,v
```

Functions

Define a function

```
# Function definition
def greet():
    """Greet user."""
    print "Hello "

# Call a function
greet() # Hello
```

Functions

function return a value

```
# Function definition
def greet():
    """Greet user."""
    # return a string
    return "Hello "

# Call a function
print greet() # Hello
```

Functions

Function with argument

```
# Function definition
def greet(username):
    """Greet user."""
    print "Hello ", username

# Call a function
name="Sachin"
greet(name) # Hello Sachin
```

Functions

Function with argument

```
# Function definition
def greet(username):
    """Greet user."""
    print "Hello %s" % username

# Call a function
name="Sachin"
greet(name) # Hello Sachin
```

Functions

lambda function

```
(lambda x: x > 2)(3)   # True
(lambda x: x > 2)(1)   # False
(lambda x: x+10)(45)   # 55
```

string method

format

```
"1st arg: {0}, 2nd arg: {1}".format(47, 11)
# 1st arg: 47, 2nd arg: 11

"1st arg: {0:.2f}, 2nd arg: {1:.1f}".format(47.874,
                                           11.345)
# 1st arg: 47.87, 2nd arg: 11.3
```

Simple class

```
class Animal(object):
    """Animal class"""
    def walk(self):
        print "Walking.."

    def eat(self, food):
        print "Eating %s" % food

    def fight(self):
        print "Fighting.."

if __name__=='__main__':
    animal_obj = Animal() # instance
    animal_obj.fight() # Fighting..
    animal_obj.eat("flesh") # Eating flesh
```

Simple class

```
class Animal(object):
    """Animal class"""
    def walk(self):
        print "Walking.."

    def eat(self, food="flesh"):
        print "Eating %s" % food

    def fight(self):
        print "Fighting.."

if __name__=='__main__':
    animal_obj = Animal() # instance
    animal_obj.fight() # Fighting..
    animal_obj.eat() # Eating flesh
```

Inherit a class

Inherit Animal class

```
class Cat(Animal):
    """Animal category: Cat"""
    def drink(self):
        print "Drink Milk"

if __name__=='__main__':
    cat_obj = Cat()  # instance
    cat_obj.drink() # Drink Milk
    cat_obj.walk() # Walking
    cat_obj.eat("Biscuit") # Eating Biscuit
```

Class constructor

init

```
class Calculator():
    """

    A calculator with offset.
    """
    def __init__(self, offset=0):
        self.offset = offset

    def add(self, x, y):
        return  x + y + self.offset

if __name__=='__main__':
    calc = Calculator()
    print calc.add(2, 3) # 5
```

A word about self

- `self` is similar to `.this` in Java
- Scope will be within a *Class*

```
def add(self, x, y):
    self.total = x + y # self :)
    return x + y + self.offset

def adder(self):
    """
    Simple function which make use of
    self.total defined in add()
    """
    return self.total + self.offset
```

Module

```
import Calculator

calc = Calculator()
calc.add(6, 7)  # 13

# Define 'offset'
calc = Calculator(9) # offset=9
calc.add(6, 7)  # 22

calc = Calculator(-5) # offset=-5
calc.add(6, 7)  # ?
```

Module

```python
from Calculator import add

add(6, 7)  # 13
```

Virtualenv

- Written in python

Install - Ubuntu

```
sudo apt-get install python-virtualenv
```

Create a virtual environment

Create

```
virtualenv ~/enigma
```

–no-site-packages

Don't give access to global package directory to virtual environment

```
virtualenv --no-site-packages ~/enigma
```

Activate/Deactivate

Activate

```
source ~/enigma/bin/activate
```

Deactivate

```
deactivate
```

pip

Install packages

```
pip install pep8
pip install pylint
pip install django==1.5
```

List packages

```
pip list
pip freeze
```

References

- Books
  - Byte of Python
  - Dive into Python
  - Learn Python the Hard Way
- Links
  - https://docs.python.org/2.7/tutorial/
  - https://docs.python.org/2/
  - http://learnxinyminutes.com/docs/python/

Contact

Proudly made with Emacs org-mode and LaTeX

Contact

- `isachin@iitb.ac.in`
- https://github.com/psachin/slides/python

Todo

- user input
- class method
- static methods
- list comprehension
- decorators
- super
- *args, **kwargs