Python Programming

Open Source
ITI - Alexandria

Information Technology Institute

Course Evaluation

➔ LAB 40%
➔ Project 60%

Get Course Material From

https://goo.gl/y4nY1n

Agenda

- A little Python History
- Why Python
- Who is using Python
- install and running python scripts
- Numbers
- Strings
- Sequences (List - Tuple - Dictionary - Array - set)
- if Statement
- loop (for - while)
- functions
- input / output
- File reading / writing

A Little Python History

- Created by Guido van Rossum in late 80s
- Now works at Google
- Its an interpreted language
- Python 2.7 not Python 3.4
- Python 3.x is not backward compatible with 2.x
- Why 2.7 and not 3.4?

Why Python

- Simple to get started, easy for beginners, powerful enough for professionals
- Code is clean, modular, and elegant, making it easy to read and edit
- Whitespace enforcement
- Extensive standard library, many modules to import from
- Cross platform - Windows, Mac, Linux
- Supportive, large, and helpful community
- *Extensible (could add* new built-in function or module written in c

Who is using Python

- Google
- Reddit
- Twitter
- Pinterest/Instagram
- DreamHost
- YouTube
- BitTorrent
- DropBox
- ...and countless more!

Install Python

It it already installed by default on ubuntu 14.04 LTS ;)

So let's Get Start ..

Run Python

1. using the shell by typing python on the terminal.
2. Running a python file:
    - create a file.py
    - in terminal: python file.py

OR (The Shebang way)

- make the first line in the file.py be “#!/usr/bin/env python”
- give the file execute permission “chmod +x file.py”
- in terminal “./file.py”
- boilerplate

Everything Is an Object

- In Python everything is an object. Integers, floats, strings … even functions are objects

- Indentation, Indentation, **Indentation**

Numbers

- There are int, float, Decimal, Fraction and Complex
- Simple Arithmetic Operations
- Interpreter acts as a simple calculator
- The operators +, -, * and /
- The power is simply a **
- Parentheses ( ) can be used for grouping
- Python does not have a ++ operator, but += works.

Strings

- ➔ Double and single quotes
- ➔ Printing raw string
- ➔ concatenation using + operator
- ➔ my_string[:-3]
- ➔ len(my_string)
- ➔ lower()
- ➔ upper()
- ➔ isalpha()
- ➔ isdigit()
- ➔ String Multiplication
- ➔ String Comparison (== , in , is)
- ➔ \ can be used to escape quotes

Nice thing about python is that It tries to be standardized

Strings - Cont

- To convert int to string use

```
>>> x = str(100)
```

- Strings are immutable
- Once a string is created it never changes (ready only)
- The string objects themselves are immutable.
- The variable, a, which points to the string, is mutable.

FYI: When you re-assign a string python doesn't modify the string but create a new one and make the variable rege.

Booleans

- True & False (First Character is capital)
- In integers 0 Is False and the rest of integers are true
- In strings “” empty string is False and other string is true

Command Line Input / Output

- my_input = raw_input(‘Enter your input ’)
- my_input2 = input(‘Enter your input ’)
- my_input3 = eval(raw_input(‘Enter your input ’))
- print my_input, my_input2, my_input3

When Printing different types we use , instead of +

Control Flow / if statement

```
if condition:
    statement

elif condition:
    statement

else:
    statement
```

For loop

```
for key in list:
        print key

for key, value in dictionary:
        print key, value

for value in range(2,11, 2):
        print value
```

Control Flow: While

```
while condition:
    statement
    condition change
```

Continue , Break & pass

Functions

- defining
- returning
- default arguments
- arguments un packaging
- flexible number of arguments

```
def my_function( ):
    statements

def my_func(parameter='default_value'):
    return statement

def my_func(*args):
    for a in args:
    print a
```

List

- my_list = [ ] #empty list
- my_list = [10, ‘Howdy’, [‘Strawberry’, ‘Peach’]] #different type list
- Lists are mutable
- list1 = list2 this doesn't create a copy but just a reference

List Object Methods:

- list.append(obj) #Appends object obj to list
- list.count(obj) #Returns count of how many times obj occurs in list

List – Cont-

List Object Methods:

- list.extend(seq) **#Appends the contents of seq to list**
- list.insert(index, obj) **#Inserts object obj into list at offset index**
- list.pop(obj=list[-1]) **#Removes and returns last obj from list**
- list.remove(obj) **#Removes object obj from list**
- list.reverse( ) **#Reverses objects of list in place**
- list.sort( ) **#Sorts objects of list**

List - Cont-

- Better way to sort
- Sorted(iterable, cmp=None, Key=None, reverse=False)
- Shadow list (Custom Sort)
- a == b can compare Lists just like Strings

List - Cont-

- Looping Over lists

```
for var in list:
    print var
```

- check if value in a list

```
value in list
```

- *Just like Java, When you are looping over a list you can't modify its structure or length (error)*

List - Cont-

- A function that generates a list of numbers
- range([start], stop[, step])

```
for index in range(0, 5):
    print index
```

- *You can use lists as **Stacks** with append( ) and pop( )*
- *You can use lists as **Queues** append( ) and popleft( )*
- List Comprehensions

```
squares = [x**2 for x in range(10)  if x**2<50]
```

Tuples

- ➔ Tuple is a list of immutable objects
- ➔ The tuples cannot be changed unlike lists
- ➔ Example: A point in space (x-coordinate, y-coordinate, z-coordinate)
- ➔ my_tuple = ( ) **#empty tuple**
- ➔ my_tuple = (10, ‘Howdy’, (‘Strawberry’, ‘Peach’)) **#different type tuple**

Tuples - Cont-

You can access an item using [ ] for example ***tuple[1]***

Built-in Tuple Functions:

- cmp(tuple1, tuple2) **#Compares elements of both tuples 0 true, -1 false**
- len(tuple) **#Gives the total length of the tuple.**
- max(tuple) **#Returns item from the tuple with max value.**
- min(tuple) **#Returns item from the tuple with min value.**
- tuple(seq) **#Converts a list into tuple**.

Tuples - Cont-

- To sort a list of tuples: it compares the first element in the tuple then the second element etc…
- Example for sorting a list using 2 criteria.

Dictionary

- ➔ Key-value bindings Or Hashing
- ➔ my_dic = {‘Key’ : ‘values’} **#values can be anything list, tuple, string …**
- ➔ we can use tuple as a dictionary key while we cannot use lists ! (Immutable)

- ➔ Biggest advantage is performance

Dictionaries Methods:

- ➔ dict.clear() **#Removes all elements of dictionary dict**
- ➔ dict.copy() **#Returns a shallow copy of dictionary dict**
- ➔ dict.has_key(key) **#Returns true if key in dictionary dict, false otherwise**

Dictionary

Dictionaries Methods:

- dict.items() **#Returns a list of dict's (key, value) tuple pairs**
- dict.keys() **#Returns list of dictionary dict's keys**
- dict.update(dict2) **#Adds dictionary dict2's key-values pairs to dict**
- dict.values() **#Returns list of dictionary dict2's values**

Arrays (One Type List)

from array import array
my_array = array(‘data_type_code’, [initial data]) OR
my_array = array.array(‘data_type_code’) and then my_array.append(value)
data_type_code:

- B , i for integers
- c for characters

To sum up

- Lists: use **brackets** and can shrink or expand (Mutable)
- Tuples: use **parentheses**, is fixed , immutable(can't change it)
- Strings: use **quotes** and are also immutable

File Read / Write

- open( ) returns a file object, and is most commonly used with two arguments: open(filename, mode).

  # modes: r for read, w for write
- read / write / append

- close

File Object attributes:

- my_file.name
- myfile.mode
- my_file.closed

Python Memory Management

- In python, memory allocation and deallocation method is automatic
- Python uses two strategies for memory allocation reference counting and garbage collection.

  Reference counting: counting the number of references to an object and if that count is zero the object is deleted from memory
- You can delete a reference by hand using the del command:

```
a = 4
    del a
```

Notes to be an Awesome Python Coder

- Don't write the entire program at one time

  the print utility is very efficient. USE IT

- Write a small part each time and then see what it does and build on it

Questions