A Quick Python Tour

Brought to you by pySchools
What is Python?

- Programming Language created by Guido van Rossum
- It has been around for over 20 years
- Dynamically typed, object-oriented language
- Runs on Win, Linux/Unix, Mac, OS/2 etc
- Versions: 2.x and 3.x
What can Python do?

- Scripting
- Rapid Prototyping
- Text Processing
- Web applications
- GUI programs
- Game Development
- Database Applications
- System Administrations
- And many more.
def greetings(name=''):  
    '''Function that returns a message'''
    if name=='':
        msg = "Hello Guest. Welcome!"
    else:
        msg = "Hello %s. Welcome!" % name
    return msg

>>> greetings("John") # name is 'John'
'Hello John. Welcome!'

>>> greetings()
'Hello Guest. Welcome!'
Python Data Types

• Built-in types
  • int, float, complex, long

• Sequences/iterables
  • string
  • dictionary
  • list
  • tuple
Built-in Types

• Integer
  >>> a = 5
• Floating-point number
  >>> b = 5.0
• Complex number
  >>> c = 1+2j
• Long integer
  >>> d = 12345678L
String

• Immutable sequence of characters enclosed in quotes

```python
>>> a = "Hello"
>>> a.upper()   # change to uppercase
'HELLO'
>>> a[0:2]     # slicing
'He'
```
List

- Container type that stores a sequence of items
- Data is enclosed within square brackets []

```python
>>> a = ['a', 'b', 'c', 'd']
>>> a.remove('d')  # remove item "d"
>>> a[0] = 1        # change 1st item to 1
>>> a
[1, "b", "c"]
```
Tuple

- Container type similar to list but is **immutable**
- More efficient in storage than list.
- Data is enclosed within braces ()

```python
>>> a = ('a', 'b', 'c')
>>> a[1]
'b'
>>> a[0] = 1          # invalid
>>> a += (1, 2, 3)   # invalid
>>> b = a+(1,2,3)    # valid, create new tuple
```
Dictionary

- Container type to store data in key/value pairs
- Data is enclosed within curly braces `{}`

```python
>>> a = {'a':1, 'b':2}
>>> a.keys()
['a', 'b']
>>> a.update({'c':3})  # add pair {'c':3}
>>> a.items()
[('a', 1), ('c', 3), ('b', 2)]
```
Control Structures

• Conditional
  ▪ if, elif, else - branch into different paths

• Looping
  ▪ while - iterate until condition is false
  ▪ for - iterate over a defined range

• Additional control
  ▪ break - terminate loop early
  ▪ continue - skip current iteration
  ▪ pass - empty statement that does nothing
if, else, elif

• Syntax:
  
  if condition1:
      statements
  [elif condition2:
      statements]
  [else:
      statements]

• Example:
  
  x = 1
  y = 2
  if x>y:
      print “x is greater.”
  elif x<y:
      print “y is greater.”
  else:
      print “x is equal to y.”

• Output:
  
y is greater.
while

- Syntax:
  ```
  while condition:
  statements
  ```

- Example:
  ```
  x = 1
  while x<4:
      print x
      x+=1
  ```

- Output:
  ```
  1
  2
  3
  ```
for

- Syntax:
  ```python
  for item in sequence:
      statements
  ```

- Example:
  ```python
  for x in "abc":
      print x
  ```

- Output:
  
a
  b
  c
Function

• A function or method is a group of statements performing a specific task.

• Syntax:
  ```python
def fname(parameters):
    ''' doc string '''
    statements
    [return expression]
  ```

• Example:
  ```python
def triangleArea(b, h):
    '''Return triangle area'''
    area = 0.5 * b * h
    return area
  ```

• Output:
  ```python
>>> triangleArea(5, 8)
20.0
>>> triangleArea.__doc__
‘Return triangle area’
```
class and object

• A class is a construct that represent a kind using methods and variables. An object is an instance of a class.

• Syntax:

```python
class ClassName:
    [class documentation]
    class statements
```

• Example:

```python
class Person:
    def __init__(self, name):
        self.name = name
    def introduce(self):
        return “I am %s.” % self.name
```

• Output:

```python
>>> a = Person(“John”).    # object
>>> a.introduce()          # introduce()
‘I am John.’
```
End of Tour

This is just a brief introduction.

What is next?

• Read PySchools Quick Reference
• Practice the online tutorial on PySchools

Have Fun!