

# Introduction to Python

Parham Alvani

Amirkabir University of Technology

`parham.alvani@gmail.com`

May 14, 2015

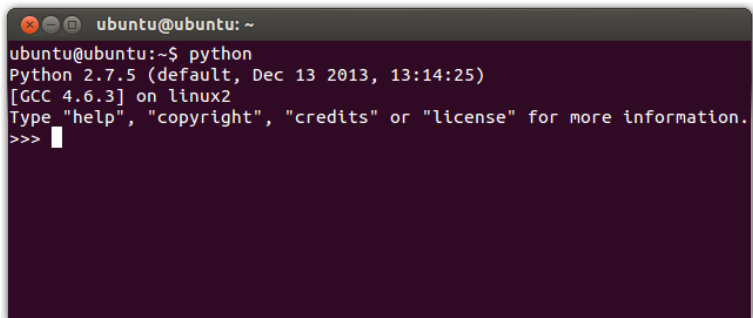


## what is python...

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable..



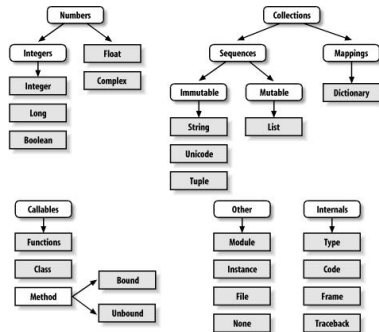
# Environment

A terminal window with a dark purple background and a grey title bar. The title bar contains three window control icons (close, minimize, maximize) and the text 'ubuntu@ubuntu: ~'. The terminal text is as follows:

```
ubuntu@ubuntu:~$ python
Python 2.7.5 (default, Dec 13 2013, 13:14:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

# Variable Types

- ▶ Numbers
- ▶ String
- ▶ List
- ▶ Tuple
- ▶ Dictionary



# Numbers

- ▶ Number data types store numeric values.
- ▶ They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

- ▶ Strings are amongst the most popular types in Python.
- ▶ We can create them simply by enclosing characters in quotes.
- ▶ Python treats single quotes the same as double quotes.

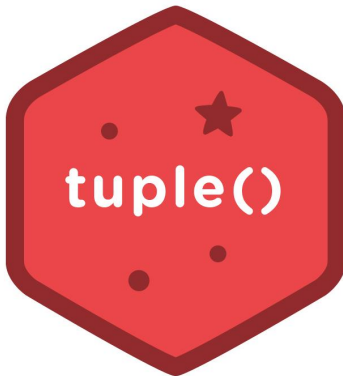
# Lists

- ▶ The most basic data structure in Python is the sequence.
- ▶ Each element of a sequence is assigned a number - its position or index.
- ▶ The first index is zero, the second index is one, and so forth.



# Tuples

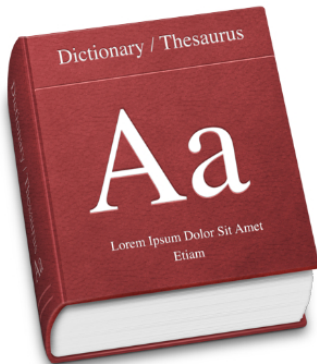
- ▶ A tuple is a sequence of immutable Python objects.
- ▶ Tuples are sequences, just like lists.
- ▶ The differences between tuples and lists are :
  - the tuples cannot be changed unlike lists
  - tuples use parentheses, whereas lists use square brackets.





# Dictionary

- ▶ Each key is separated from its value by a colon ( : )
- ▶ the items are separated by commas
- ▶ the whole thing is enclosed in curly braces.

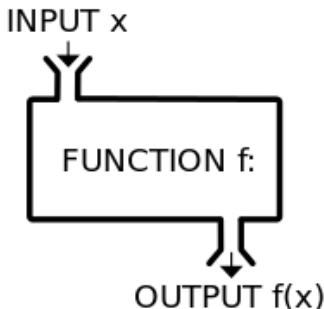


- ▶ If-Then-Else
- ▶ For
- ▶ While
- ▶ Exceptions

**FLOW**  **CONTROL**

# Functions

- ▶ Function blocks begin with the keyword `def`, followed by the function name and parentheses.
- ▶ Any input parameters or arguments should be placed within these parentheses.
- ▶ The first statement of a function can be an optional statement - the documentation string of the function or docstring.



# Functions

```
def square(x):  
    return x * x
```

```
def hello():  
    return "Hello"
```

```
def printme( str ):  
    "This prints a passed string into this function"  
    print str  
    return
```

# Classes

- ▶ The class statement creates a new class definition.
- ▶ The name of the class immediately follows the keyword `class` followed by a colon as follows



# Classes

```
class Employee:
    """
    Common base class for all employees
    """
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name, ", Salary: ", self.salary
```

```
emp1 = Employee("Zara", 2000)
```

# Garbage Collection

- ▶ Python deletes unneeded objects automatically to free the memory space.



# Inheritance

- Instead of starting from scratch, you can create a class by deriving it from a preexisting class by listing the parent class in parentheses after the new class name.





# Inheritance

```
class SubClassName (ParentClass1[, ParentClass2, ...]):  
    """  
    Optional class documentation string  
    """  
    # class_suite
```

# Base Overloading Methods

- ▶ `__init__( self [,args...] )` : Constructor (with any optional arguments)
- ▶ `__del__( self )` : Destructor, deletes an object
- ▶ `__repr__( self )` : Evaluatable string representation
- ▶ `__str__( self )` : Printable string representation
- ▶ `__lt__( self, other )`:
- ▶ `__le__( self, other )`:
- ▶ `__eq__( self, other )`:
- ▶ `__ne__( self, other )`:
- ▶ `__gt__( self, other )`:
- ▶ `__ge__( self, other )`:

These are the so-called rich comparison methods,  
and are called for comparison operators in preference to `__cmp__()` below.

# Base Overloading Methods

- ▶ `__cmp__( self, x )` : Called by comparison operations if rich comparison is not defined.
- ▶ `__add__( self, other )`:
- ▶ `__sub__( self, other )`:
- ▶ `__mul__( self, other )`:
- ▶ `__floordiv__( self, other )`:
- ▶ `__mod__( self, other )`:
- ▶ `__divmod__( self, other )`:
- ▶ `__pow__( self, other[, modulo] )`:

# Base Overloading Methods

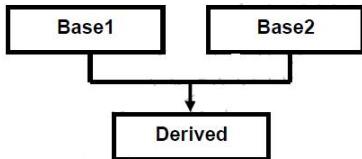
- ▶ `__lshift__( self, other ):`
- ▶ `__rshift__( self, other ):`
- ▶ `__and__( self, other ):`
- ▶ `__xor__( self, other ):`
- ▶ `__or__( self, other ):`

These methods are called to implement the binary arithmetic operations

`+`, `-`, `*`, `//`, `%`, `divmod()`, `pow()`, `**`, `<<`, `>>`, `&`, `^`, `|`

# Multiple Inheritance

- ▶ Method Resolution Order (MRO)
- ▶ C3 Algorithm



# Questions?