

Python Basics

Topics covered

What is Python

System Architecture

Python Building blocks

Why Python

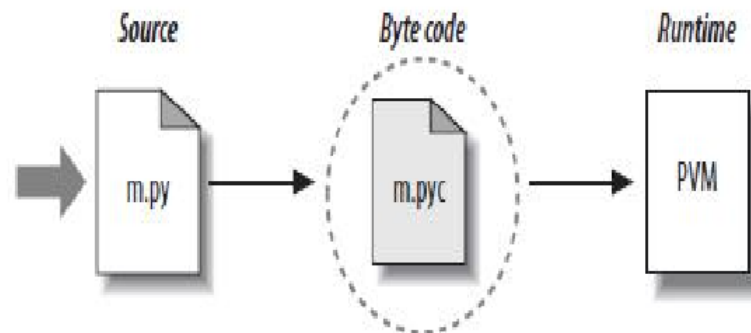
What is Python ?

- ⦿ General Purpose Programming Language.
- ⦿ Fully Object Oriented.
- ⦿ Multiple Paradigms
 - Procedural
 - Functional
 - Object Oriented
- ⦿ First Class Consideration to everything.
- ⦿ Wide range of application area
 - System Programming

- GUI Programming.
- Internet Scripting.
- Multimedia and Animation.
- Component Integration.
- Database Programming.
- Rapid Prototyping.
- Numeric and Scientific Programming.
- Gaming, Robots, AI, XML and More...

System Architecture.

Python Virtual Machine (PVM)



- ⊙ Portable Intermediate Code
- ⊙ Speed comes in between interpreted and compiled languages.
- ⊙ Python Syntax
 - Semicolon Optional, use it to break logical lines.
 - Indentation to group compound statements.
- ⊙ Interactive Python Shell
- ⊙ Multiple Implementation
- ⊙ Cpython, Jython, IronPython, pypy.

Python Building Blocks.

Python Datatypes.

- ⊙ Data Types are Object Oriented.
- ⊙ Type Information is with the Objects not with the Variables.
- ⊙ Object types
 - int, float, bool, complex, str, unicode, list, tuple, dict, and file .
 - Immutable :- Numbers, String, & Tuple
 - Mutable :- List and Dict.

Python Variables

- ⊙ Variables are references.
 - No need to declare it.
 - Dynamic Typed.
 - Initialize before Use (No default value.)
 - Mutable objects will change other references.

	Variable	Value
a = 3	a	3 (Integer)
b = a	b	
b=5	b	5(Immutable)

Numbers

- ⦿ Integer, Long integer and Float
- ⦿ Implicitly Expand to long int.
- ⦿ Print 127 - 127 :- Int
- ⦿ Print 0177 - 127 :- Octal Representation.
- ⦿ Print 0x7F - 127 :- Hexa Decimal.
- ⦿ int(x) and float(x).
- ⦿ Inbuilt support for complex numbers.

```
a = 3 + 3j ; b = 5.5 + 4j
```

```
print a + b (8.5 + 7j)
```

String

- ⦿ Set of characters.
- ⦿ Single, double and triple quote.
- ⦿ Negative indexing.
- ⦿ Support List Slicing
- ⦿ Unicode support.

```
a = 'str1' , b = "str2"
```

```
c = """str3
```

```
New line ...."""
```

```
Print a + b - str1str2
```

```
Print a[-1] - 1
```

```
Print a[0:1] - s
```

```
***[limit1:limit2)
```

List

- Ordered list of Objects indexed by 0,1,etc..
- Can Store any Objects
- Problem with shared references and mutability.
- list(a) to take copy of the object.
- Slicing and other list methods.

```
a = ['v1',32]
Print a[1] -32
b = a , b[0] = 3
print a - [3,32]
a.append(3) to
add new element.
```

Tuple

- Tuple is an immutable list.
- A = (3,) or a = (3,'34')
- Support all operations of List.

Dictionary

- Key Value Pair.
- Assign new key:value to dict. to add new element.
- Don't use mutable keys.

```
a = { 'k1' : 3 }
a[k2] = 'test'
print a
{k1:3, k2:'test'}
a['k3'] = 'dict'
Print a - {'k1':3, 'k2':'test', 'k3':'test'}
b=a.copy() and b=a
b.keys(), b.values(),b.items()
a.has_key('k3')
```

Boolean

- ⦿ True and False are Aliases of integer 0 and 1.
- ⦿ Every object hold this Boolean values.

Numbers - True if not Zero.

Other Objects - False if Empty.

None - False

Operators

- ⦿ Support most of the C Arithmetic and bit-wise operators and its precedences.
- ⦿ '**' - Exponential operator
- ⦿ '/' - Integer division.
- ⦿ Logical Operators :- and , or & not.

Files

Reading

- `f = open('test.txt','r')`, Unix style file path.
- `f.read()` :- To read all the lines into a string.
- `f.read(N)` :- To read N bits, $N > 0$
- `f.readline()` :- Read single line.

Writing

- `f = open('test.txt','w')`
- `f.write(S)` :- Write string to this file.
- `f.writeline(L)` :- write each of the string from list L into file
- `f.close()`

Input methods

- ⦿ `val = raw_input(">")` :- Read one input line.
>
- ⦿ The parameter is for prompt.
- ⦿ `val = input()` :- Evaluate the input string after reading. Dangerous one.
- ⦿ Use `int(var)` or `float(var)` or `str(var)` etc to convert the input string.

Other Statements

- ⦿ Control Statements if, elif, else
- ⦿ Now switch statement.

```
x = 22

if x < 15 :
    print 'first clause'
elif x < 25 :
    print 'second clause'
else :
    print 'the else'
```

Loop Statements

- ⦿ Loop statements :-
for, while
- ⦿ Other controls :- break , continue and pass
- ⦿ Special else clause for all loop statements (optional).
It only run if the loop completed normally.

```
x = 1
while x < 3 :
    print x
    x = x + 1
else:
    print 'hello'
```

```
for x in [1,7,13,2]:
    print x
```

Functions

- Its an Object
- We can assign it to a variable.
- `max = findmax`
`max(x,y)`

```
def findmax(x,y):
    if x < y:
        return y
    else:
        return x
```

Python Modules

- Highest level structure of python.
- Each file is a module.
- Each has its own namespace.

```
def findmax(x,y):
    if x < y:
        return y
    else:
        return x
```

```
testmax.py
import findmax
```

Module Import

```
import testmax
```

Brings all elements of testmax in, but must refer to as `mymodule.<elem>`

```
from testmax import findmax
```

Imports `findmax` from `testmax` right into this namespace

```
from testmax import *
```

Imports all elements of `testmax` into this namespace

Packages

- ⦿ Group of modules under a directory.
- ⦿ Directory name is the package name.
- ⦿ Every package hold a file called `__init__.py` , to represent it as a python package.

Class

- ⦿ Collection of data and method, or creating user defined objects.
- ⦿ `__init__(self)` is the constructor.
- ⦿ "self" is used as the first argument to all the methods. Its an equivalent of "this" under C++, PHP and others.
- ⦿ "self.variable" creates one instance variable.
- ⦿ Multiple inheritance.

```
class myclass :  
    def __init__(self, val) :  
        self.x = val  
    def printit(self) :  
        print self.x
```

Shell commands

- ⦿ `import os`
- ⦿ `os.system('ls')` :- To list files from current dir.
- ⦿ `os.system` :- output the result to screen.
- ⦿ Another module "command"
- ⦿ `import commad`
- ⦿ `output = command.getoutput(shell commands)`
- ⦿ Returns output in String format. Easy to operate.

Why Python ?

- ⦿ Software Quality.
- ⦿ Developer Productivity.
- ⦿ Program Portability.
- ⦿ Support Libraries.
- ⦿ Component Integration.
- ⦿ Enjoyment. :)

Python downsides

- ⦿ Execution speed may not be fast as C compiled code.
 - Use Compiled Libraries and Extensions.
- ⦿ GIL Degrades the Threading under Multi Processor Environment.
 - Processes Instead of Threads.

References

- ⦿ Learning Python, 3'rd Edition.
- ⦿ Google.