python™

Balamurugan S
balamurugan.sekar@gmail.com

Agenda

- History
- Basics
- Control Flow
- Functions

History

- What is Python?
- Why to learn Python?
- Application Domains
- Versions of Python

python™

What is Python?

- Python is a general purpose, object oriented, high level, interpreted language
- Developed by Guido Van Rossum in early 90's
- Developed at National Research Institute for Mathematics and Computer Science ***(Dutch: Centrum voor Wiskunde en Informatica or CWI)***
- Free Software
- Simple, Portable and Powerful
- Python is developed as successor of ABC Language

python™

Why learn Python?

- Easier to learn if you know where to start !
- Faster development of PoC code
- Elegant and Powerful, Reliable and fast
- Cross Platform
- Great Documentation
- Strong community support

python™

Application Domains

- Stand alone GUI Applications
- Web and Internet Programming
- Scientific and Numeric
- Education
- Network Programming
- Games and 3D Graphics

python™

Versions

- Python 2.5
- Python 2.6
- Python 3.0
- Why Python 2.5/2.6 and !3.0?

Basics

- Installing Python
- Editing Python
- Python Interpreter
- Hello World Program
- Structure of Python program
- Data structures

python™

Installing Python

- Debian based systems: *apt-get install python*
- Yum based installers: *yum install python*
- Or Download from: *(http://www.python.org/)*
- On Windows: ActiveState python (http://www.activestate.com/activepython/)

Editing Python

- Emacs
- Vi/Vim
- IDLE
- PythonWin (For Windows)
- Whatever editor you want

python™

Python Interpreter

- Can be invoked via script, Interactive Session
- Uses of Interactive Session
- Exit interactive session
    - quit()
    - ctrl + d on *nix
    - ctrl + z on windows

python™

Hello World

Interactive Session:

```
$ python
Python 2.6.4 (r264:75706, Nov  2 2009, 14:38:03)
[GCC 4.4.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> print "Hello World"
Hello World
>>>
```

Invoking as a script:

```
$ emacs hello_world.py
$ cat hello_world.py
#!/usr/bin/python
print "Hello World"
$ chmod +x hello_world.py
$ ./hello_world.py
Hello World
```

Structure of Python

- Indentation is very important
- Blocks are defined with same indentation
- No begin/end delimiters
- Comments starts with #

Data Types

- Integer Numbers
  - Decimal - 1, 2, 3, ...
  - Octal - 01, 02, .. 07
  - Hexa - 0x1, 0x2
  - Long - 1L, 123456789L
- Floating Point

  0.0, 1.1, 100.10, 2e3
- Complex Numbers

  -1+3j , 3-4j

Strings

- Strings can be defined in many ways

```
str = “Hello World”     #Correct
str = 'Hello World'     #Correct
str = “””
First Line              #Correct
Second line
”””
```

  - Cannot use single and double quote in same string to enclose

```
str = “Hello World'
str = 'Hello World”
```

python™

Dictionaries

- Dictionaries defines one-to-one relationships between key and value pair
- To create a dictionary

```
dict = {'k1':'v1', 'k2':3}

dict['k1'] = 'v1'
dict['k2'] = 3
dict[42] = 'test'
```

- Data can be accessed by using keys

```
>>> print dict['k1']
v1
```

- KeyError will be thrown when tried to access invalid Key
- Dictionary keys are case sensitive

Dictionaries

- Modifying a value for a key in Dictionary

```
dict['k1'] = 'newvalue'
```

- Deleting a key in Dictionary

```
del dict['k1']
```

- Deleting whole Dictionary

```
dict.clear()
dict = {}
```

python™

Lists

- List is a mutable ordered sequence of items, it is similar to arrays
- Creating List

```
li = ['a', 'b', 42, 'bala']
```

- Data can be accessed similar to an array

```
li[0]
```

- Lists can be Indexed, Sliced
- Operations allowed: append, insert, remove
  - Appends – appends to the end of the list
  - Insert – inserts at specified index
  - Remove – removes the specified element

Tuple

- Immutable ordered sequence of items / Lists
- Creating a tuple

  a = (1, 2, 3, 4)

- Data can be accessed via index
- Used to protect data

Variables

- There is no prior declaration needed
- Variables are the references to the allocated memory
- Variables can refer to any data type (like Tuple, List,Dictionary, Int, String, Complex)

Indexes and Slices

- String, List, Tuple, etc can be sliced to get a part of them
- Index -> similar to array index, it refers to 1 position of data
- Slices-> gives the data in the range
- Example ->

  a=”Free Software”

  a[:3] a[4:11] a[4:] a[:]

python™

Control Flow

- input
- if
- while
- for
- range
- break
- continue

input

- Use raw_input() to take a string input from the user
- Used as

```
<var> = raw_input("Enter a String: ")
```

- Input() is used to take a input without specifying the type

if

- If is a conditional statement, for simple “If then else“ clause in English

- Header lines are always concluded with a “ : “ followed by intended block of statements

- Optionally it can be followed by an “else if” clause known as “elif” in python

```
if <condition>:
        Statement 1
        Statement 2
elif <condition>:
        Statements
else:
        statements
```

while

- While statement is used for repeatedly executing a block of code till the condition is true, also has an optional else clause

- Use wildly for infinite loop

```
while <condition>:
      statements
else:
       statements
```

for

- It is a sequence iterator
- It works on Strings, lists, tuples, etc

```
for <target> in <iterable>:
   statements
```

range

- They are used to generate and return integer sequence as lists
- Range(5) -> [0,1,2,3,4]
- Range(1,5) -> [1,2,3,4]
- Range(0,8,2) -> [0,2,4,6]

break

- Used to terminate a loop
- If nested it terminates the inner most loop
- Practically used for conditional loop termination with an if statement

Continue

- Terminates the current iteration and executes next
- Practically used for conditional statements termination with an if statement

Functions

- What are functions?
- Defining a function
- Passing parameters

python™

What are functions?

- A Function is a group of statements that execute on request
- Function is a object
- Function block takes parameters and can return any value
- In Python, functions return None in case of return not mentioned in the definition

Defining a function

- Defining a function ->

```
def name(parameters):
    statement(s)
```

Passing Parameters

- Types of parameters
  - Mandatory Parameters
  - Optional Parameters
- Calling function using named parameters
- Default values
- Be careful when default value is a mutable object

```
def a(x,y=[]):
    y.append(x)
    print y
print a(12)
print a(34)
```

Modules

- What are modules?
- How to test modules?
- How to load modules?

Class

- How to define a class?
- How to create objects?
- How to call functions inside classes?

Thanks

?

References

- Dive into Python (http://www.diveintopython.org/)
- Senthil Kumaran Python presentation (http://www.stylesen.org/)