SecurityTube Python Scripting Expert (SPSE)

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 1:
Introduction to Python and Setting up an Environment

Vivek Ramachandran
Course Instructor

Python – a short history

- Created in 1989 by Guido Van Rossum (works for Google)
- Python 2.x in 2000
- Python 3.x in 2008
  - Not backward compatible
- 2.x is the status quo
- 3.x future

Why Python?

- Open Source
- Multi-Platform
- Rich set of libraries
- Large number of open source tools
- HLL used for Rapid Prototyping

Multiple OS Support

- Unix / Linux
- Mac OS X
- Windows
- Mobile Platforms – Android, iOS
- Embedded Systems

Implementations

- Cpython – reference implementation “Python”
- Jython – Python in Java
- IronPython - Python in C#

More:
http://wiki.python.org/moin/PythonImplementations?action=show&redirect=implementation

Why Python in Infosec?

- Rapid prototyping - POC
- Extensive library support
- Tons of tools already written

Python on different OSs

- Linux
  - Pre-Loaded
- Windows
  - ActiveState Python
- MAC
  - Pre-loaded ☺

Python 2.7 or 3.x?

- Emphasis on 2.7
- Most tools / libraries still do not support 3.x
- Eventually everything will support 3.x
  - Will take a couple of years

Platform of Choice

- Ubuntu Server 11.10 64-Bit

http://www.ubuntu.com/download/server/download

- Will be running inside Virtualbox

- Connect to it over SSH using Putty or any other client of your choice

Customary “Hello World”

- Interactive mode

- Script

Module 1: Exercise 1

- Install Python 3.x in Ubuntu 11.10
- How can you switch between different versions of Python?
  - Console
  - Script
- Explore Virtualenv in Python

SecurityTube Python Scripting Expert (SPSE)

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Exercise 1

- Install Python 3.x in Ubuntu 11.10
- How can you switch between different versions of Python?
  - Console
  - Script
- Explore Virtualenv in Python

Install Python 3

- Apt-get install python3

Install Python Virtualenv

- Install PIP: apt-get install python-pip

- Pip install virtualenv

What is Virtualenv?

- Allows creation of isolated python environments
- Takes away the pain of library version issues
- Each environment is isolated
  - Can be configured to not use globally configured libs as well
- http://www.virtualenv.org/en/latest/index.html

Creating a Virtual-Env for Python 3.x

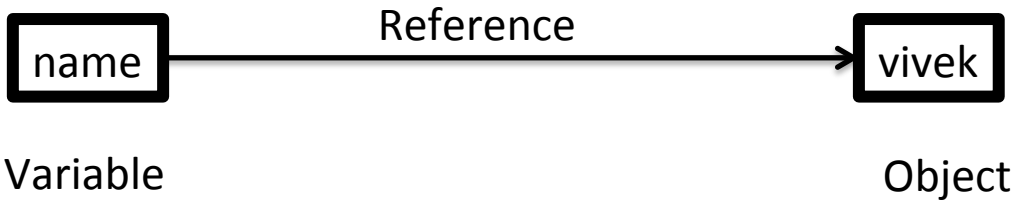
```
root@ubuntu-server:~/Python/1# virtualenv
You must provide a DEST_DIR
Usage: virtualenv [OPTIONS] DEST_DIR

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -v, --verbose         Increase verbosity
  -q, --quiet           Decrease verbosity
  -p PYTHON_EXE, --python=PYTHON_EXE
                        The Python interpreter to use, e.g.,
                        --python=python2.5 will use the python2.5 interpreter
                        to create the new environment.  The default is the
                        interpreter that virtualenv was installed with
                        (/usr/bin/python)
  --clear               Clear out the non-root install and start from scratch
  --no-site-packages    Don't give access to the global site-packages dir to
                        the virtual environment
  --unzip-setuptools    Unzip Setuptools or Distribute when installing it
  --relocatable         Make an EXISTING virtualenv environment relocatable.
                        This fixes up scripts and makes all .pth files
                        relative
  --distribute          Ignored.  Distribute is used by default. See
                        --setuptools to use Setuptools instead of Distribute.
  --setuptools          Use Setuptools instead of Distribute. Set environ
                        variable VIRTUALENV_USE_SETUPTOOLS to make it the
                        default.
  --extra-search-dir=SEARCH_DIRS
                        Directory to look for setuptools/distribute/pip
                        distributions in. You can add any number of additional
                        --extra-search-dir paths.
  --never-download      Never download anything from the network.  Instead,
                        virtualenv will fail if local distributions of
                        setuptools/distribute/pip are not present.
  --prompt==PROMPT      Provides an alternative prompt prefix for this
                        environment
root@ubuntu-server:~/Python/1# ls
root@ubuntu-server:~/Python/1#
```

Module 1: Python Language Essentials

Part 2: Variables and Data Types

Vivek Ramachandran
Course Instructor

Variables, Objects and References




```
>>> name = "vivek"
>>>
>>>
>>> name
'vivek'
>>>
```

```
>>> id(name)
23177904
>>>
>>>
>>> name.__repr__
<method-wrapper '__repr__' of str object at 0x161aab0>
>>>
>>>
>>>
>>> hex(id(name))
'0x161aab0'
>>>
>>>
```

Data Types

- Strings
- Numbers
- Lists
- Dictionaries
- Tuples
- Boolean
- …

Strings

- Definition
  - name = “vivek”
  - name = ‘vivek’
  - name = “vivek’s”
  - name = “vivek\nramachandran”
  - name = r’vivek\nramachandran’
    (raw string turns off escaping)
  - name = “””
    Vivek
    Ramachandran
    “””

Unicode String

- Used for Internationalization
- “wide characters” are they are called
- name = u’vivek’
- unicode to regular string conversion
  - str(name)
- regular string to unicode conversion
  - unicode(name)

String Operations

- strings are immutable objects in Python
- Concatenating strings
  - s1 + s2
- Repeated sequence in string
  - buffer = “A”*20
- Slicing – breaking up the string
  - string[start:end:steps]
- Int to String
  - str(42)

String Methods

- string.find(…)
- string.replace(….)
- string.split(….)
- ….

String Formatting

- “Hack this IP: %s” % ip
- “Hack %s with IP %s” (domain, ip)
- “Hack %(domain)s with IP %(ip)s” % { “domain” : ”securitytube.net”, “ip” : “192.168.1.10” }

Numbers

- Integers, Floats etc. can be represented
- Operators
  - +,-,*,/
  - x**y (x to the power y)
  - (>,=,<,>=,<=, ==)
  - x|y, x^y, x&y (bitwise operators)
  - x and y, x or y, not x (logical operators)

Lists

- Collection of objects which can be heterogeneous
- myList = [1,2,3,4]
- myList = [1, ‘vivek’, ‘SPSE’, 2.5]
- myList = [1, [3,4, ‘hello’], [3,4], 2, 3]
- len(myList)
- len(myList[1])

List Operations

- Concatenate [1,2] + [3,4] = [1,2,3,4]
- Append -- list.append()
- Extend --- list.extend([])
- Reverse -- list.reverse()
- Pop -- list.pop()
- Insert -- list.insert(index, item)
- Delete -- del list[index]

Module 1: Python Language Essentials

End of Part 2: Variables and Data Types

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 3: Data Types: Tuple, Sets, Dictionaries

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Tuple

- Tuples are similar to lists but immutable
- Can covert from list to tuple and vice versa
  - tuple(list)
  - list(tuple)
- video = (“Hello World”, 5, 10, 0)
- sequence unpacking
  - videoName, time, upvotes, downvotes = video

Sets

- Unordered collection of unique objects
- List to set : b = set(a)
- Set to list: a = list(b)
- Set Operations
  - Union: a|b
  - Intersection: a&b
  - Difference: a-b
  - …

Dictionaries

- Unordered key-value pairs
- Keys are unique and immutable objects
- Value can change
- dict = {} , dict['name'] = 'vivek'
- dict(name='vivek', age='31')
- dict = { 'name' : 'vivek', 'age' : 31 }
- Check if a given key is present
  - dict.has_key(key)
  - key in dict

Dictionary Operations

- Get tuple of items: dict.items()
- Get list of keys: dict.keys()
- Get list of values: dict.values()
- Get a particular item: dict.get(key)
- Item deletion
  - All items: dict.clear()
  - One item: del dict[key]

Getting Help on Methods etc.

- dir() – lists all attributes
- help(string.replace) - list method help

Module 1: Python Language Essentials

End of Part 3: Data Types: Tuple, Sets, Dictionaries

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 4: Conditional Statements

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

If Statement

```
if    test_condition1:
                do stuff
                do stuff
elif  test_condition2:
                do stuff
                do stuff
elif  test_condition3:
                do stuff
                do stuff
else :
                do stuff
```

While Loops

```
while    statement_is_true:
             do stuff
              do stuff
```

- break: get out of innermost loop
- continue: start the next pass of the innermost loop
- pass: do nothing, placeholder

Exercise

- While loops can also have a “else” in Python
- explore this functionality and write a simple program to illustrate

For loops

```
for item in object:
        do stuff
        do stuff
```

```
for item in [1,2,3]
for item in [‘a’, 2, ‘3’]
for (x,y) in [(“vivek”, 31), (“john”, 25)]
```

Exercise

- For loops can have a “else” statement as well
- write a simple program to illustrate this functionality

Emulating C style FOR loops

C style loops: for (i=1; i<10; i++)

Use range in python:

range(lower, upper, step) creates a list for use

range(n) – [0, ……, n-1]

Module 1: Python Language Essentials

End of Part 4: Conditional Statements

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 5: Functions

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Functions

- Functions allow sections of code to be grouped better as per functionality

- def function(arg1, arg2=default, ..) :
  - …
  - …
  - return value

Module 1: Python Language Essentials

End of Part 5: Functions

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 6: Classes and Objects

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Class

```
Class Calculator:

        def __init__(self, inp1, inp2):
                self.a = inp1
                self.b = inp2

        def sum(self):
                return self.a+self.b

        def product(self):
                return self.a*self.b
```

Inheritence

```
Class ScientificCalculator (Calculator):

        def power(self):
                return pow(self.a, self.b)
```

Exercise

- What are Global, Class and Instance variables?
- How can we override a method in parent class?

Module 1: Python Language Essentials

End of Part 6: Classes and Objects

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 7: Creating Modules

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Modules

- Better way of organizing code
- can define classes, functions and variable
- import MODULE_NAME
- from MODULE_NAME import ….

Module 1: Python Language Essentials

Part 7: Creating Modules

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 8: Creating Packages

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Packages

- hierarchical file directory structure to organize code
- consists of modules and sub-packages

Module 1: Python Language Essentials

Part 9: Exception Handling

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Exceptions

- Simply put exceptions are error conditions which disrupt the normal flow of the program

- Python allows for a simple and elegant way to handle exceptions

Exercise

- Python allows for user defined exceptions
- Code up a demo which has a user defined exception and an example use case

Module 1: Python Language Essentials

End of Part 9: Exception Handling

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 1: Python Language Essentials

Part 10: Python on other Devices

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Python on the iPhone (iOS)

- On a Jailbroken iPhone using Cydia
  - install Python scripting support
- Original Blog post by Saurik: http://www.saurik.com/id/5
- Can do a ton of stuff! Lets read the SMS DB
- With restrictions – Python for iOS

http://itunes.apple.com/us/app/python-for-ios/id485729872?mt=8&ign-mpt=uo%3D4

Python on Android

- Scripting Layer for Android

http://code.google.com/p/android-scripting/

Exercise: Install the scripting layer on your Android Phone and try the previous demo

Python in your Wi-Fi Router

- Open Source firmwares such as DD-WRT support running Python on them

http://www.dd-wrt.com/site/index

Exercise: Purchase a DD-WRT compatable router (DLINK DIR-615 E4) and run python on it

Module 1: Python Language Essentials

End of Part 10: Python on other Devices

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor