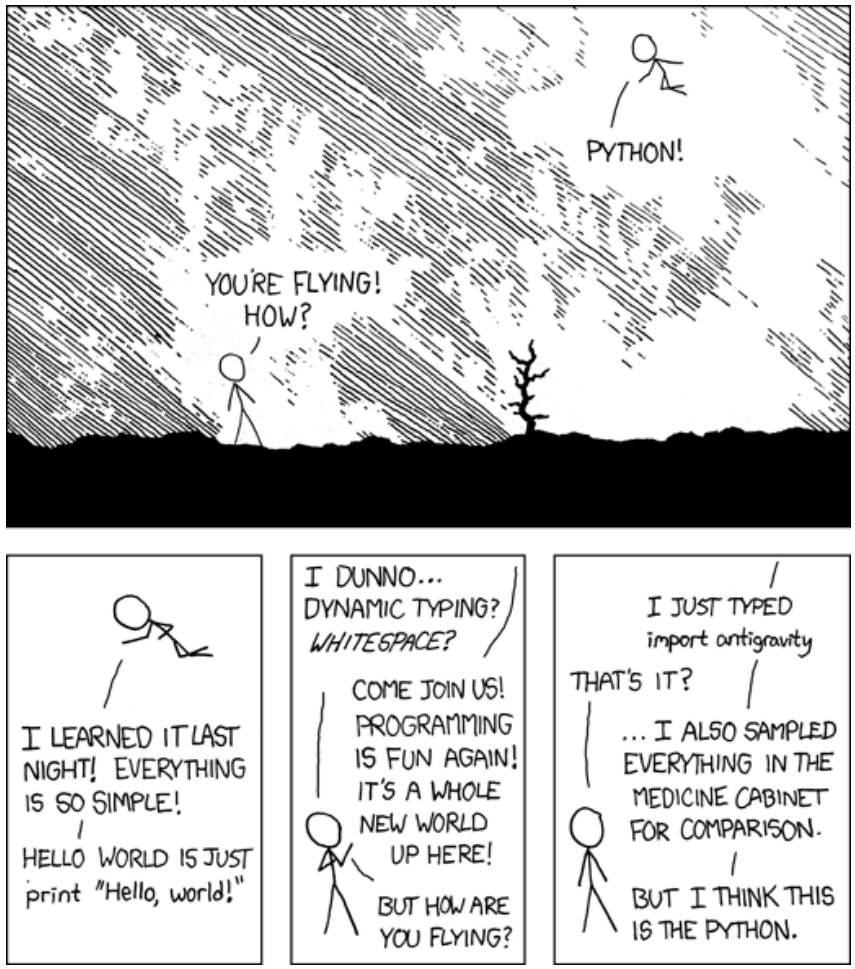
# Learning Python

Erik Edrosa

Panther Linux User Group




source: `http://xkcd.com/353/`

# What is Python?

*Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs*

- a general purpose high-level programming language
- a scripting language
- multiple programming paradigms

# How to get Python

- Linux
  - Most distros come with python
  - type `python --version` to see which version
  - if you don't have version 3, use your package manager
    - `sudo apt-get install python3` for Ubuntu
    - `sudo pacman -S python` for Arch Linux
- Windows
  - Download the lastest version from python.org
- Mac OS X
  - Mac OS X comes with python 2.7 pre-installed by apple
  - Downloaded latest following Using Python on a Mac

# Python 3 vs. Python 2

- There is two main versions to download
- Important changes were made in Python 3 which breaks compatability with some programs
- Support for Python 2 will stop soon
- Only develop in Python 2 for maintaining older code which does not support 3
- Be sure to check which version you have.
  - Some Linux distros have both versions
  - use `python --version` to check
  - Versions might sometimes be named python2 or python3

# Let's get coding

- The `python` command is the python interpreter
  - Typing `python` alone runs an interactive interpreter
  - Typing `python fileName.py` runs the code in the file
- Using the interactive interpreter
  - type `python` or `python3`
  - Version info and the symbols >>> will print
  - >>> is the python prompt
  - type in `print("Hello World")`
  - press enter
  - `Hello World` should be printed
  - type `exit()` to exit interactive mode

# Basics of Python programming

- Identation of code is important
  - use a tab or two spaces to change the semantics
  - statments must be on new lines, no semicolons
  - used to enforce code to be more readable
- Python has dynamic typing
  - No need to declare the types of variables and arguments
- No main function or method
  - Python code can run without being in the scope of a function
  - Main function can be simulated using

```
def main():
    # code goes here

if __name__ == "__main__":
    main()
```

# Basic data types

- Has all your basics like integers, floats, strings, and boolean
- A None value which represents the absence of a value

```
12345                   # Integer
3.14                    # Float
True                    # Boolean
"Hello World"           # Strings
'Hello World'
'''This can
   be on
   multiple
   lines'''
```

# Data Structures

- Lists
  - similar to arrays
  - not a fixed length
  - items in the list do not need to be the same type
- Tuples
  - consists of a number of values
  - is immutable - can not be changed once created
- Dictionaries
  - Stores keys and values
  - known as maps or associative arrays in other languages

# Data Structures

## examples

```
# list
lst = [1, 2, 3, 4, 5]
print(lst[0])           # 1
print(lst[-1])          # 5
lst.append("foo")
print(lst)              # [1,2,3,4,5,'foo']
# tuple
t = (1, 2, "foo")
u = 1, 2, "foo"
# dictionary
d = {"foo" : "bar", "n": 5}
print(d["foo"])         # bar
d["x"] = True
```

# List Comprehensions

- a concise way to create lists
- same operation can be done using a for loop but in less lines
- can even be nested

```
# Filter for odd numbers
[n for n in range(10) if n % 2 == 1]
# [1, 3, 5, 7, 9]
# Map: double each element
[x*2 for x in range(10)]
# [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

# Variables

- Remember, python has dynamic typing
- Can assign different data types to the same variable and python won't complain
- multivariable assignment from expressions which return multiple values

```
n = 12345
n = "foo"
x, y = 1, 2
x, y = (1, 2) # same thing as above
x, y, z = [1,2,3]
```

# Control flow

- Python has the standard control flow statements like if, if else, while, and for
- The condition part does not require parenthesis
- : follows the condition
- the body of the control flow statement is indented and unindented to resume the regular flow

```
x = 10
if x > 5:
    print("x is greater then 5")

lst = [1,2,3,4,5]
#range returns a list of from 0 - 4
for i in range(5):
    print(lst[i])
```

# Functions

- Functions are similar to other languages
- First class objects
  - means it can be passed around like a normal value
- use the `def` keyword to define a function
- there is also lambda (or anonymous functions)

```
def addSomeNum(y=5):
    return (lambda x: x + y)

add5 = addSomeNum()
print(add5(1))          # 6
add1= addSomeNum(1)
print(add1(2))          # 3
```

# Classes

- Python has object oriented programming
- Classes are also first class objects
- The body of a class are a series of statements, usually function definitions
- `def __init__` is the constructor function

```
class MyClass:
    x = 123
    def __init__(self, n):
        self.n = n
    def addNums(self):
        return self.x + self.n

c = MyClass(7)
print(c.addNums())  # 130
```

# Resources

- `http://python.org`
- Python 3 Tutorialj
- Dive into Python 3
- Udacity CS101 (teaches Computer Science in Python)

# Thank you

## Questions or comments?

- Contact me at eedro001@fiu.edu
- Visit the club's website `http://plug.cs.fiu.edu`
- Join our irc channel
  - plug.cs.fiu.edu
  - room #chat