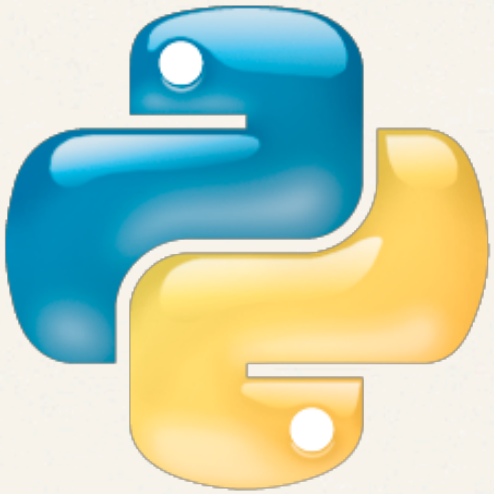


Python

Barak Stout

CS101 February 5, 2013

Python Background

- Started in 1989, 3 major releases since
- Interpreted scripting language
- Code that emphasizes readability
- Large Community
- Live Interpreter and Compiled Code
- Multi-purpose Programming language
- Used by: Google, Yahoo!, NASA, etc.

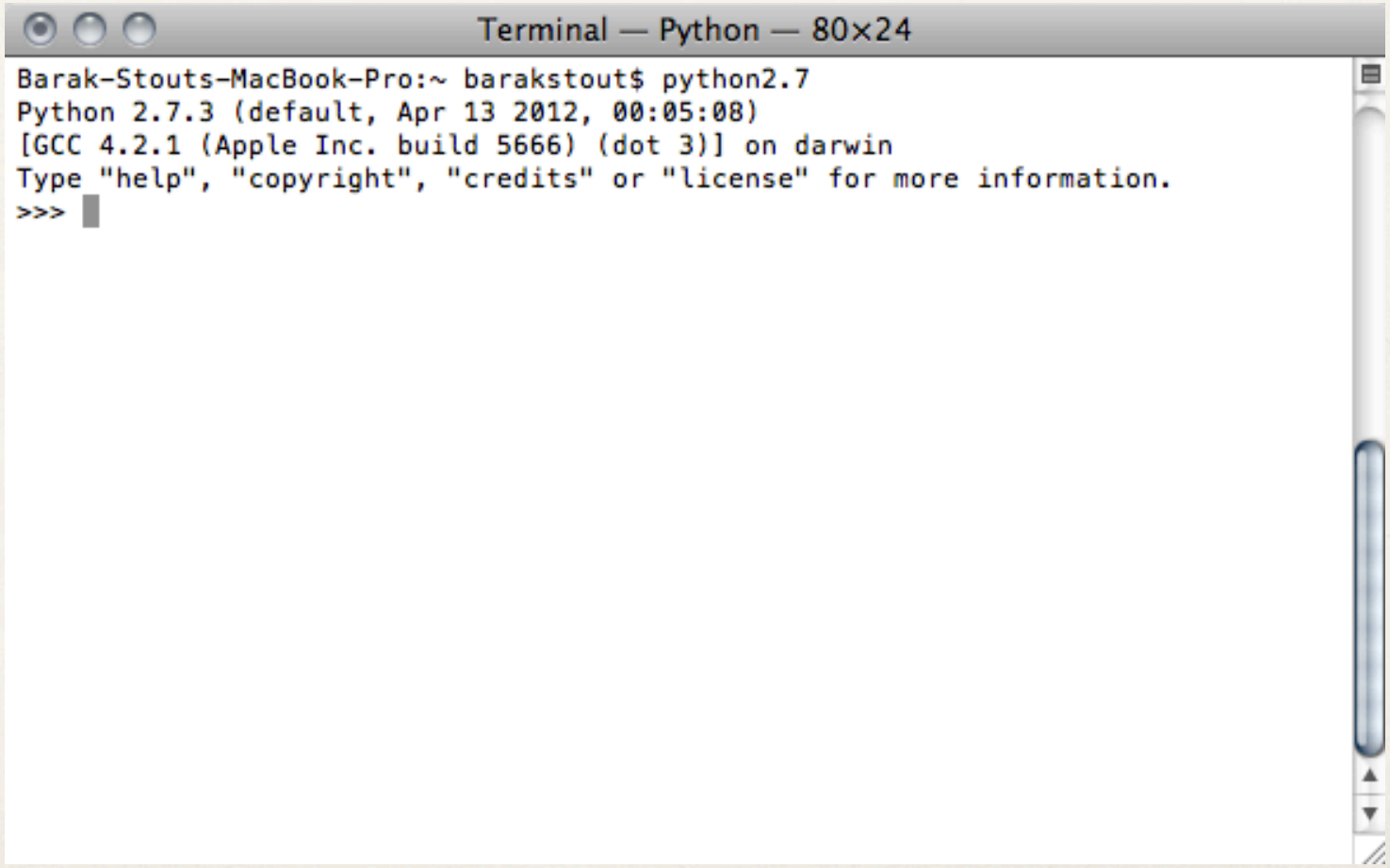
Python

- The power of python
  - http://www.youtube.com/watch?v=v1AJ_OBJUpY

Python Resources

- Website: http://www.python.org/
- Versions: 2.7.3, 3.3.0
- Documentation: http://docs.python.org/2/index.html
- Text Editor: Notepad++, TextWrangler, Aquamacs Emacs, gEdit…

Python Interpreter



Python Basic Data Types

- Basic Data Types:
  - int, long int, float, complex, strings, boolean

```
>>> x = 3
>>> type(x)
<type 'int'>
>>>
>>> x = 3L
>>> type(x)
<type 'long'>
```

```
>>> x = 3.14
>>> type(x)
<type 'float'>
>>>
>>> x = 3j
>>> type(x)
<type 'complex'>
```

```
>>> x = 'python'
>>> type(x)
<type 'str'>
>>>
>>> x = True
>>> type(x)
<type 'bool'>
```

Python Numeric Operations

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	(floored) quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`long(x)`	*x* converted to long integer
`float(x)`	*x* converted to floating point
`complex(re,im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*. (Identity on real numbers)
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

Python Identifiers and keywords

- Identifiers:

```
identifier ::=  (letter|"_") (letter | digit | "_")*
letter     ::=  lowercase | uppercase
lowercase  ::=  "a"..."z"
uppercase  ::=  "A"..."Z"
digit      ::=  "0"..."9"
```

- Reserved Keywords:

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Python Data Structures

- Data Structures :
  - list, tuple

```
>>>
>>>
>>> a = []
>>> a.append(3.14)
>>> a.append(42)
>>> a.append('hello python')
>>> a
[3.14, 42, 'hello python']
>>> a[2]
'hello python'
>>> a.pop(1)
42
>>> a
[3.14, 'hello python']
>>> a = [1,2,3,[4,5,6]]
>>> a
[1, 2, 3, [4, 5, 6]]
>>> a[3]
[4, 5, 6]
>>> a[3][1]
5
```

```
>>> t = ('a','b','c')
>>> t[0]
'a'
>>> t[0] = a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> t.append('c')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> t
('a', 'b', 'c')
>>> 'a' in t
True
>>> 'd' in t
False
>>>
```

Python Dictionary

```
>>> dic = {'a':'choice 1', 'b':'choice 2', 'c':'choice 3'}
>>> dic
{'a': 'choice 1', 'c': 'choice 3', 'b': 'choice 2'}
>>> dic['a']
'choice 1'
>>> del dic['a']
>>> dic
{'c': 'choice 3', 'b': 'choice 2'}
>>> dic['d'] = 3
>>> dic
{'c': 'choice 3', 'b': 'choice 2', 'd': 3}
>>>
>>> 3 in dic
False
>>> 'd' in dic
True
>>> x = dic.keys()
>>> x
['c', 'b', 'd']
>>>
```

Python Data Structures Set

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket)               # create a set without duplicates
>>> fruit
set(['orange', 'pear', 'apple', 'banana'])
>>> 'orange' in fruit                 # fast membership testing
True
>>> 'crabgrass' in fruit
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                 # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                             # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                             # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                             # letters in both a and b
set(['a', 'c'])
>>> a ^ b                             # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Python Control Flow

- if - elif - else
- for loop
- while loop
- try - except - finally
- pass/break statement
- Association by indentation, white space matters!

Python If, elif, else

- if <boolean> :
  - <body>
- elif <boolean> :
  - <body>
- ...
- else :
  - <body>

```
>>> a = 7
>>> b = 9
>>> c = -1
```

```
>>> if a > 1 :
...   print a
...
7
```

```
>>> if a < 1 :
...   print a
... else:
...   print b
...
9
```

```
>>> if a < 1 :
...   print a
... elif b < 1 :
...   print b
... else:
...   print c
...
-1
```

```
>>> True and False
False
>>> False or True
True
```

```
comparison    ::=  or_expr ( comp_operator or_expr )*
comp_operator ::=  "<" | ">" | "==" | ">=" | "<=" | "<>" | "!="
                   | "is" ["not"] | ["not"] "in"
```

Python For Loop

- for <var> in <sequence> :
  - <body>

```
>>> range(5)
[0, 1, 2, 3, 4]
>>>
>>> for i in range(5,30,5):
...   print i
...
5
10
15
20
25
```

```
>>> for i in range(10):
...   print i
...
0
1
2
3
4
5
6
7
8
9
>>>
>>>
>>> a = ['dog','cat','mouse']
>>> for pet in a:
...   print pet
...
dog
cat
mouse
```

Python Fizz Buzz

- Print all the numbers from 1-100
  - For multiples of 3 print “Fizz”
  - For multiples of 5 print “Buzz”
  - For multiples of both print FizzBuzz
  - Any other number print the number

Python While Loop

- while <boolean> :
  - <body>

```
>>> count = 0
>>> while (count < 9) :
...   print 'The count is: ', count
...   count += 1
...
The count is:  0
The count is:  1
The count is:  2
The count is:  3
The count is:  4
The count is:  5
The count is:  6
The count is:  7
The count is:  8
```

```
>>> while(1):
...   print 'no!'
...
no!
no!
no!
no!
no!
no!
no!
no!
no!
no!
no!
no!
^C!
Traceback (most recent call lst):
  File "<stdin>", line 2, in module>
KeyboardInterrupt
```

Python Exception Handling

- Exception: action taken outside of the normal flow of control because of errors
  - System raises the exception
  - Programmer handles the error
- Types of errors:
  - Syntax Errors, Logic Errors, Semantic Errors

Python Try, except

```
def main():
    try:
        x = input('Enter a number: ')
        print x
    except:
        print 'error'

main()
```

```
barak-stouts-macbook-pro:Desktop barak$ python try.py
Enter a number: 5
5
barak-stouts-macbook-pro:Desktop barak$ python try.py
Enter a number: a
error
```

Python Try, except

```
def main():
    while True:
        try:
            x = input('Enter a number: ')
            print x
            break
        except Exception, e:
            print e

main()
```

```
barak-stouts-macbook-pro:Desktop barak$ python try.py
Enter a number: s
name 's' is not defined
Enter a number: 1/0
integer division or modulo by zero
Enter a number: 4
4
```

Python Try, except, finally

```
def main():
    while True:
        try:
            x = input('Enter a number: ')
            print x
            break
        except Exception, e:
            print e
        finally:
            print 'Always here'

main()
```

```
barak-stouts-macbook-pro:Desktop barak$ python try.py
Enter a number: f
name 'f' is not defined
Always here
Enter a number: 4
4
Always here
```

Python Functions

- Definition:
  - def <function_name>(formal_parameter_list) :
    - <body>
- Call:
  - [var =] <function_name>(argument_list)

Python Functions

```
def fibonacci(n):
    a = 0
    b = 1
    print a,
    while b < n:
        b = a + b
        a = b - a
        print a,

def main():
    fibonacci(5)
    print "\n"
    fibonacci(10)
    print "\n"
    fibonacci(2000)
    print "\n"

main()
```

$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0$$

$$F_1 = 1$$

```
Barak-Stouts-MacBook-Pro:Desktop barakstout$ python2.7 fib.py
0 1 1 2 3

0 1 1 2 3 5 8

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Python Functions

```
def printinfo( name, age ):
   print "Name: ", name
   print "Age ", age
   return

printinfo("Bugs Bunny", 72 )

printinfo( age=77, name="Porky Pig" )
```

```
Barak-Stouts-MacBook-Pro:Desktop barakstout$ python2.7 function_example.py
Name:  Bugs Bunny
Age  72

Name:  Porky Pig
Age  77
```

Python Modules

- “A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, the module’s name (as a string) is available as the value of the global variable __name__.” (Python 2.7.1 Documentation)
- The import statement allows the use of other modules, global or local
  - import <name>
    - use as <name>.<method>()
  - from <name> import * (or the name of one or more methods)
    - use directly as <method_name>()

Python Modules Example

```
#temp_util.py

#absolute zero constants:
a0celsius = -273.15
a0fahrenheit = -459.67

#takes input fahrenheit and returns celsius
def fahrenheit2celsius(f):
    return ((f-32)*5)/9

#takes input celsius and returns fahrenheit
def celsius2fahrenheit(c):
    return (((c*9)/5)+32)
```

Python Modules Example

```
import temp_util

def main():
    x = input('Enter a temperature to convert (f2c) : ')
    print temp_util.fahrenheit2celsius(x)

    x = input('Enter a temperature to convert (c3f) : ')
    print temp_util.celsius2fahrenheit(x)

    print "\ncelsius absolute zero ", temp_util.a0celsius
    print "\nfahrenheit absolute zero ", temp_util.a0fahrenheit

main()
```

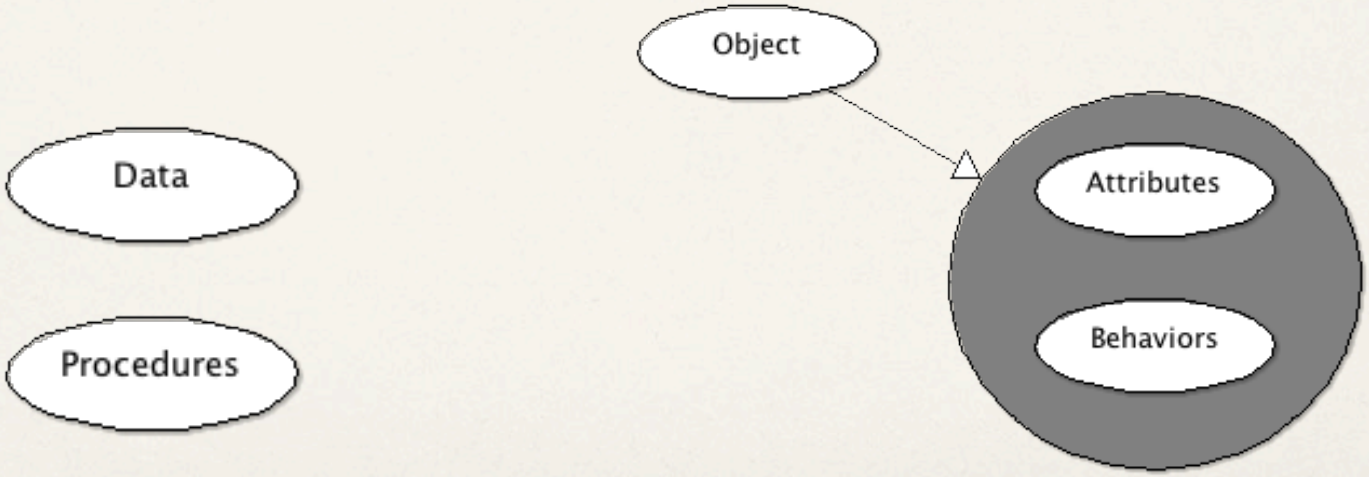
```
from temp_util import *

def main():
    x = input('Enter a temperature to convert (f2c) : ')
    print fahrenheit2celsius(x)

    x = input('Enter a temperature to convert (c3f) : ')
    print celsius2fahrenheit(x)

    print "\ncelsius absolute zero ", a0celsius
    print "\nfahrenheit absolute zero ", a0fahrenheit

main()
```

```
barak-stouts-macbook-pro:Desktop barak$ python temp.py
Enter a temperature to convert (f2c) : 42
5
Enter a temperature to convert (c3f) : 42
107

celsius absolute zero  -273.15

fahrenheit absolute zero  -459.67
```

Intro to Object Oriented (OO)

Object Oriented Programming

- “Writing software that supports a model wherein the data and their associated processing (called "methods") are defined as self-contained entities called "objects".” (PCMag)



Object Oriented Concepts

- Encapsulation - Attributes and Behaviors are enclosed
- Abstraction - Implementation is abstract
- Polymorphism - Many forms
- Inheritance - “Sub-Object”
- Aggregation - “Object of Object”

Python Classes

- A class is a blueprint containing all the instructions needed for the system to instantiate an Object
  - Imports
  - Variables (Attributes)
  - Constructor(s)
  - Methods (Behaviors)
    - Getters & Setters

Python Class Example

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def withdraw(self, amount):
        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance

>>> a = BankAccount()
>>> b = BankAccount()
>>> a.deposit(100)
100
>>> b.deposit(50)
50
>>> b.withdraw(10)
40
>>> a.withdraw(10)
90
```

Python Class Example

```
class Person():
    __firstName = ""
    __lastName = ""

    def __init__(self, fname, lname):
        self.__firstName = fname
        self.__lastName = lname

    def getFirstName(self):
        return self.__firstName

    def getLastName(self):
        return self.__lastName

    def display(self):
        print "First Name: ", self.__firstName, "\n"
        print "Last Name: ", self.__lastName, "\n"
```

```
x = Person("Bugs","Bunny")
x.display()
print x
```

```
First Name:  Bugs

Last Name:  Bunny

<__main__.Person instance at 0x1004a5b00>
```

Python Class Example

```
class Address():
    __streetNumber = 0
    __streetName = ""
    __city = ""
    __state = ""
    __zip = ""

    def __init__(self):
        pass

    def setAddress(self, stNum, stName, city, st, zip):
        self.__streetNumber = stNum
        self.__streetName = stName
        self.__city = city
        self.__state = st
        self.__zip = zip

    def getAddress(self):
        rtn = (self.__streetNumber, self.__streetName, self.__city,
               self.__state, self.__zip)
        return rtn

    def display(self):
            print "Street Number : ", self.__streetNumber, "\n"
            print "Street Name : ", self.__streetName, "\n"
            print "City : ", self.__city, "\n"
            print "State : ", self.__state, "\n"
            print "Zip : ", self.__zip, "\n"
```

```
a = Address()
a.setAddress("4400", "University Drive", "Fairfax", "VA", "22030")
string = a.getAddress()
print string, "\n"
a.display()
print a
```

```
('4400', 'University Drive', 'Fairfax', 'VA', '22030')

Street Number :  4400

Street Name :  University Drive

City :  Fairfax

State :  VA

Zip :  22030

<__main__.Address instance at 0x1004a5ab8>
```

Python Class Example

```
class Student():
    __bio = ""
    __address = ""

    def __init__(self, bio, a):
        self.__bio = Person(bio[0],bio[1])
        self.__address = Address()
        self.__address.setAddress(a[0],a[1],a[2],a[3],a[4])

    def setName(self, fName, lName):
        self.__bio = Person(fName, lName)

    def getAddress(self):
        return self.__address.getAddress()

    def display(self):
        self.__bio.display()
        self.__address.display()
```

```
First Name:  Tasmanian

Last Name:  Devil

Street Number :  4400

Street Name :  University Drive

City :  Fairfax

State :  VA

Zip :  22030

<__main__.Student instance at 0x1004a59e0>
```

```
student = Student(("Tasmanian","Devil"),
                  ("4400", "University Drive", "Fairfax", "VA", "22030"))
student.display()
print student, "\n"
```

Events

- In event driven programming, the flow of the program is determined by events - i.e., user clicks on a GUI button, user press a key on the keyboard, sensor state goes from 0 to 1...
- Key event:
  - Key is pressed - KEYDOWN
  - Key is released - KEYUP

- “Pygame is a set of Python modules designed for writing video games.” (Pygame) - www.pygame.org
- A module for:
  - Camera, CD-rom, Color, Cursors, Display, Draw, Event, Font, Gfxdraw, Image, Joystick, Key, Locals, Mask, Mixer, Mouse, Movie, Music, Overlay, Pixelarray, Rect, Scrap, Sandarray, Spirte, Surface, Surarray, Time, Transform

Pygame - Key Example

```
import pygame

def key_function():
    pygame.init()
    for event in pygame.event.get():
        if (event.type == pygame.KEYDOWN):
            keyinput = pygame.key.get_pressed()
            if keyinput[pygame.K_DOWN]:
                print "down"
            elif keyinput[pygame.K_UP]:
                print "up"
            elif keyinput[pygame.K_RIGHT]:
                print "right"
            elif keyinput[pygame.K_LEFT]:
                print "left"

            #exit
            elif keyinput[pygame.K_x]:
                print "exit"
                exit()

def main():
    while(1):
        key_function()

main()
```

Other Python Modules

- NumPy, SciPy
- OpenCV
- Tkinter
- Pyro