Global Startup

Meet-up 0: Intro to Python

Meet-up Outline

- Why Python?
- Basic Syntax
- Variables
- Control Statements
- Functions

Why Python?

Python because…

- Convenient built-in functions and data structures
- Great for rapid prototyping
  - No seperate compile step
  - No need to explicitely specify method argument types beforehand
- Syntax is readable and fast to write

```
if (x)
{
    if (y)
    {
        a();
    }
    b();
}
```

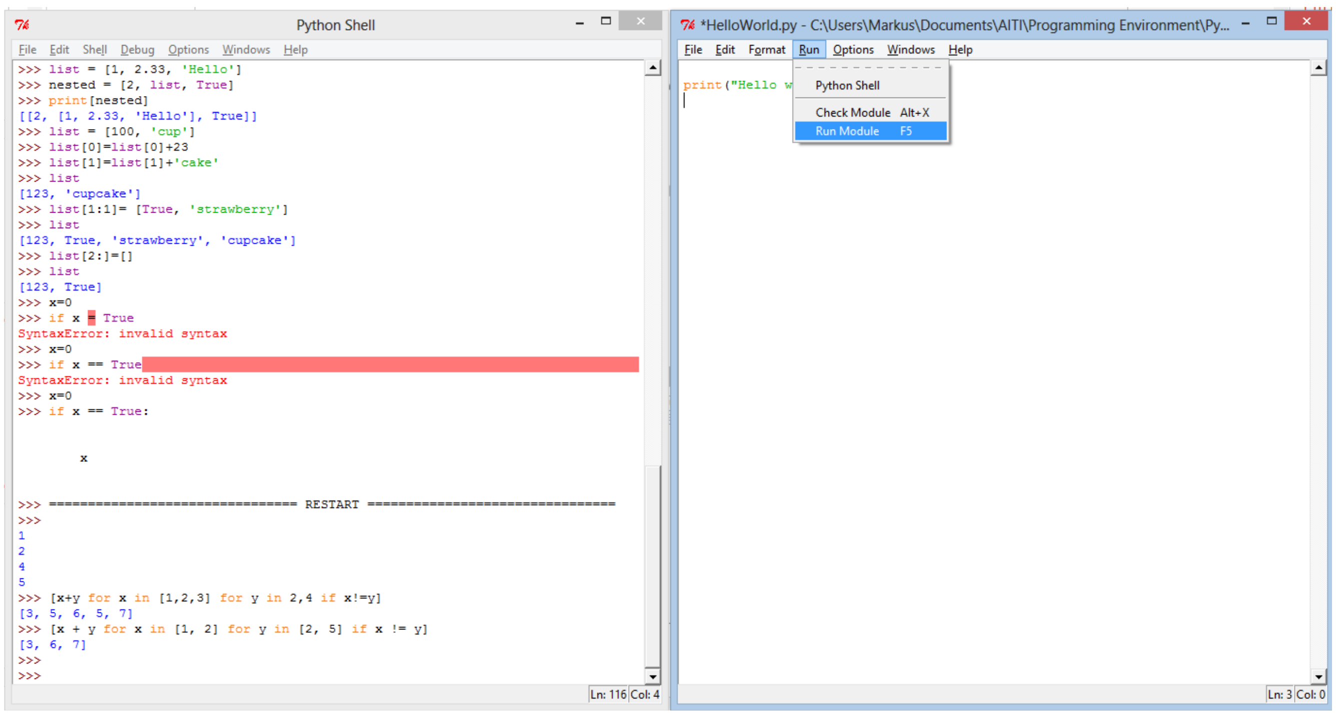
```
if x:
    if y:
        a()
    b()
```

Python because…

- We want each of you to reach millions of users, and don't want to waste time building the pipes and plumbing
- Python is supported by a number of good frameworks, including
  - Django
  - Heroku
  - Google AppEngine

Python Shell and IDLE

- IDLE is an interactive shell and text editor that comes with Python
- Great way to experiment with and write Python



Basic Syntax

Basic Syntax

- Semicolons are only used to seperate multiple statements on the same line (which is discouraged)
- Whitespace is important!

```
if x:
    if y:
        a()
    b()
```

- Use hash # to write comments

>>> # this is a comment

The range() function

- Use range() to create sequences:
  - range(4) -> [0, 1, 2, 3]
  - range(5, 10) -> [5, 6, 7, 8, 9]
  - range (1, 30, 5) -> [1, 6, 11, 16, 21, 26]

```
for i in [0,1,2,3]:
    print i


for i in range(4):
    print i
```

For loop example

```
for i in range(2, 10):
    if i%2 == 0:
        print i, "is even"
        continue
    print i, "is odd"
```

Functions

Functions

- A function is a sequence of python statements that operates on predefined input parameters and can be called by a prespecified name

Function definition

```
def functionname(parameter1, parameter2):
    body
```

Calling the function

```
functionname(100, 'Hello')
```

Functions

- Function arguments are local to function
- Program starts executing at first point after function definitions –> make sure to define or import functions BEFORE you use them
- Return statement ends function immediately

```
def convert_to_fahrenheit(c):
    tempC = 21  # this will NOT change
                # tempC outside the function!
    f = ((9.0 / 5.0) * c) + 32.0
    return f
    print("Hey") #after return -> will not print

tempC = 35
tempF = convert_to_fahrenheit(tempC)
print(tempF)
```

Functions with variable arguments

Example: range() function can be called with one two or three arguments:

- Range(4) -> [0, 1, 2, 3]
- Range(5, 8) -> [5, 6, 7]
- Range(2, 14, 3) ->[2, 5, 8, 11]

How do we achieve this?

Default arguments

- Use default argument values
  - They are used if the user does not specify a different value
- Call the function using positional or keyword arguments

```
>>>
def speak(being="a robot", action="love"):
    print "I am", being, "and I am programmed to", action


>>> speak()
I am a robot and I am programmed to love
>>> speak('Wesley', 'teach you python')
I am Wesley and I am programmed to teach you python
>>>
```

More on datastructures

Creating a new class

```
class ExampleClass:

    def __init__(self, arg1, arg2):
        self.arg1=arg1
        self.arg2=arg2

    def __str__(self):
        return ('hello world')

    # other basic customizations

    def addex(self):
        z=self.arg1+self.arg2
        print z

    def hw(self):
        print 'Hello World!'

    # other methods that you want to define
```

Define the name of the class

Basic customizations of class:

- __init__ -> called when class is first created
- __str__ -> called when print command on class
- Many more

Any functions you want to define: can be called as method objects later on

Creating a new object

```
class ExampleClass:

    def __init__(self, arg1, arg2):
        self.arg1=arg1
        self.arg2=arg2

    def __str__(self):
        return ('hello world')

    # other basic customizations

    def addex(self):
        z=self.arg1+self.arg2
        print z

    def hw(self):
        print 'Hello World!'

    # other methods that you war
```

```
>>> var = ExampleClass(12, 21)
>>> var.arg1
12
>>> var.arg2
21
>>> print var
hello world
>>> var.addex()
33
>>> var.hw()
Hello World!
>>> |
```

Create a new object of type ExampleClass

__str__ is executed

Method object var.addex() is created

Method object var.hw() is created

Class Example

```python
class example:

    def __init__(self, x):
        self.x = x

    def __str__(self):
        s='This is a wonderful number'
        return(s)

    def doublex(self):
        s=self.x*2
        print(s)

    def addx(self, y):
        return self.x + y


x = example(4)

x.doublex()

y = example(6)

print y

y.x

x.addx(y.x)
```

Program organization

```
import MODULENAME

def func1():
    BODY1
...
def funcn(a):
    BODYN

class Class1(object):
    CLASSBODY1
...
class ClassN(object):
    CLASSBODYN

# start of the program
MAINBODY
```

Import Statements

Function Definitions

Class Definitions

Your main program starts here