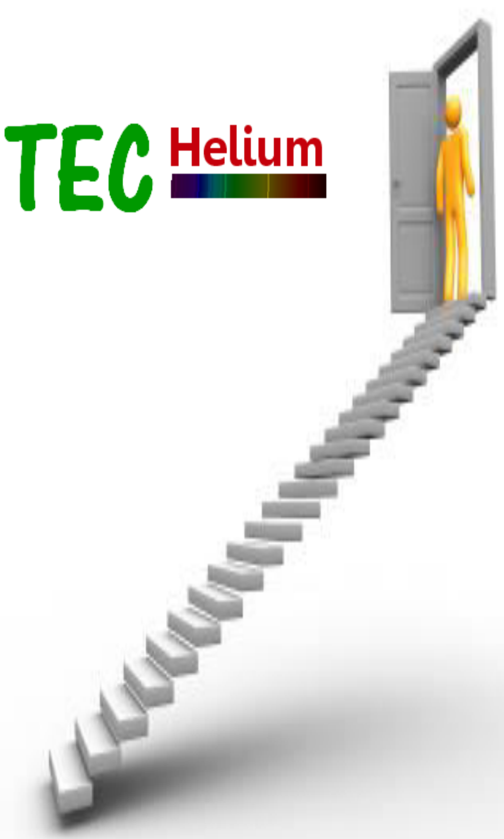
T
E
C
H
e
l
i
u
m

Lets Face it, now

Project Oriented Python

Python Introduction

TEC Helium



What is Python?

- Python is a general-purpose programming language
- Can be used for practically any programming task Instead of any higher language.
- Python is very flexible and is currently available on over two dozen operating system platforms
- Python is a free open source language.
- It is simple to learn as its syntax is easier than C
- It supports OOP – Object oriented programming like C++

TEC Helium

Python version

- Python has two major version
  - Python 2.x
  - Python 3.x
- What should I use?
  - New development better be done in python 3.x
  - However , if you join a company already having python framework of 2.0 , you may need to continue using 2.x
  - There is not much difference, you can learn any one and work with other with little changes..

Python usage

- Python can be used in a variety of situations, both online and off.
- Here are just a few interesting places where Python is used:
  - *Google* uses python in its spiders.
  - *NASA* uses Python in its Integrated Planning System as the standard scripting language at Johnson Space Center.
  - *Red Hat* uses Python for Red Hat Linux's installer (anaconda) and configuration utilities.
  - *IBM* uses Python to create the business practice logic for factory tool control applications at IBM East Fishkill.
  - *The CIA* built its website in Python with Zope.
  - *Walt Disney Feature Animation* uses Python to add scriptability to their animation production system.

Get Python (and IDLE)

- We will be using Python 2 (version 2.7.9), **not** Python 3
  - Get it from www.python.org
- The download includes an Integrated Development Environment (IDE), named "IDLE"

Running IDLE

- IDLE opens a window in which you can enter and run Python commands
  - This window is called a REPL (<u>R</u>ead-<u>E</u>val-<u>P</u>rint-<u>L</u>oop)
- Choose `File -> New Window` to open a window in which you can type entire programs
  - To execute the program, hit the F5 key

First simple script

In Linux

- Open editor
-
```
#!/usr/bin/python
# This will print "Hello, World"
  print "Hello, world"
```
- Save file as hello.py
- Chmod to execution:
  - $chmod 0755 hello.py
- Execute python script
  - ./hello.py

In IDLE

- Open editor (File->New File)
- Create following script

```
#!c:\python27\python
# This will print "Hello, World"
  print "Hello, world"
```
- Save file as hello.py
- Execute python script
  - F5

Comments in Python

- All comments in Python are written starting with a # sign.
- Anything after the # sign through to the end of the line is ignored by the interpreter.
- Comments can be placed anywhere on the line, but commands cannot follow a comment on the same line
- Multiline comments should have a # symbol as the first character on every line

```
#!/usr/bin/python
# This will print "Hello, World"
   print "Hello, world"

 # print "Not printed"

   print "Hello"  # another hello

# Multiline comments are always
#started with # and there can be
#any number of comments line
```

The #! directive

- The sole exception to # indicating a comment is on the first line of a Python program (or "script").

- All Python programs can begin with the line:
  `#!/usr/bin/python or #!c:\python`

- The #! is a hold-over from UNIX/Shell that instructs the operating system to use the /usr/bin/python program to run whatever is in this file

Block

- There are no braces to indicate blocks of code in python.
- Blocks are denoted by line indent.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount
- Indent are key to denote blocks in
  - loops,
  - logic layout, and
  - continuation of statements

```
#!/usr/bin/python

print "Here the program block starts"

i=3

while (i < 20) :
    if (i%2) :
        print i, " is a odd number"
    else :
        print i, " is a even number "
    i = i + 1

print "program end "
```

Single, Double & triple Quotes

- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

```
#!/usr/bin/python
print "Hello, \nworld"
 print ‘Hello, \nworld ‘
```

```
Output

Hello,
world
 Hello,
world
```

- The triple quotes can be used to span the string across multiple lines

```
word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph.
It is made up of multiple lines and
sentences."""
```

Variable

- Variables are nothing but reserved memory locations to store values
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory
- Python variables do not have to be explicitly declared to reserve memory space.
- The declaration happens automatically when you assign a value to a variable.

<u>Examples</u>

```
Sum = 0

Name= "Hck"

Temperature = 43.5

Inc= low = 1

Print Sum, Name

Temperature += Inc
```

Variables

- **variable**: A named piece of memory that can store a value.
  - Usage:
    - Compute an expression's result,
    - store that result into a variable,
    - and use that variable later in the program.
- **assignment statement**: Stores a value into a variable.
  - Syntax:

    name = ***value***

  - Examples:
    ```
    x = 5
    gpa = 3.14
    ```
    x $\boxed{5}$ gpa $\boxed{3.14}$

  - A variable that has been given a value can be used in expressions.

    `x + 4` is `9`

Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

```
bob   Bob   _bob   _2_bob_   bob_2   BoB
```

- There are some reserved words:

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

Naming conventions

The Python community has these recommend-ed naming conventions

- joined_lower for functions, methods and, attributes
- joined_lower or ALL_CAPS for constants
- StudlyCaps for classes
- camelCase only to conform to pre-existing conventions
- Attributes: interface, _internal, __private

Assignment

- You can assign to multiple names at the same time

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

This makes it easy to swap values

```
>>> x, y = y, x
```

- Assignments can be chained

```
>>> a = b = x = 2
```

Accessing Non-Existent Name

Accessing a name before it's been properly created (by placing it on the left side of an assignment), raises an error

```
>>> y

Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-
    y
NameError: name 'y' is not defined
>>> y = 3
>>> y
3
```

print

- `print` : Produces text output on the console.
- Syntax:

  `print` "***Message***"

  `print` ***Expression***

  - Prints the given text message or expression value on the console, and moves the cursor down to the next line.

  `print` ***Item1*** , ***Item2*** , ... , ***ItemN***

  - Prints several messages and/or expressions on the same line.
- Examples:

```
print "Hello, world!"
age = 45
print "You have", 65 - age, "years until retirement"
```

Thanks