Python Programming

Interpreted Language

Preferred Python Version 2.7.x

Environment Variables For Windows Users

Variable Name: `path`

Variable Path: `C:/Python27`

Interactive Shell Mode
Scripting Mode

Preferred Editor for scripting:

notepad++

Running Python Programs

1.Run the python script

```
ex: python hello.py
```

2. Open up the Interpreter

```
 ex: python

>>print("hello")

hello
```

Interpreter

>>> > Says That You Are Inside The Python Interpreter

The Special Quality of a Interpreter

READ - EVALUATE - PRINT - LOOP

Variable Type Declaration

There is no need to declare a variable in python programming. As the language is Dynamically Type Language. The variable is automatically set to a type

The verify the type of data it is set to. Use **`type function`**

example:

```
>>> a = "hello"

>>> type(a)
```

Numbers

```
>> 2 + 3
5
>> 45 + 30
75
```

```
>> a = 80
>> b = 90
>> a + b
170
>> c = a + b
>> print(c)
170
```

Math

Operators

`+, -  ,*, **, /, %, << , >>, &, |, ^,`
`< >, <=, >=, ==, !=`

**** The Beautiful Math Library is also your Treasure**

Strings

```
>> a = "hello"

>> b = "world"

>> c = a+b

>> print(c)

helloworld
```

Built Methods For Strings

For Strings

.strip()
.spilt()
.upper()
.capitalize()
.startswith()
.swapcase()
.islower()

Boolean

```
print (2>3)

print (24 > 9) or (89 < 0) and (19 > 8)
```

Guess the Output

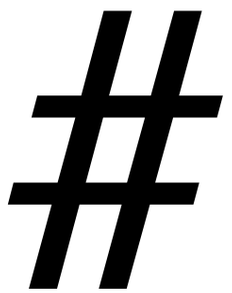
```
>>> True = False

>>> False = True

>>> print (True,False)
```

```
>>> print (0>-1)
```

Comments



Collection/Sequences

Lists

Tuples

Dictionaries

Lists

[ ] indicates that it's a list

list is mutable collection

Lists

```
>>> a = ['apple','banana','mango']
>>> b = [1,2,3,46,43,11,6]
>>> a
['apple','banana','mango']
>>> b
[1,2,3,46,43,11,6]
>>> c = ['apple',79,98,True]
>>> c
['apple',79,98,True]
>>> # List can also contain different data types
```

List Operations

```
>>> a = ['mango','cheery','pineapple','orange','apple']

>>> a[1:]

>>> a = ['mango','cheery','pineapple','orange','apple']

>>> a[2] = 'pumpkin'

>>> print(a)

['mango','cheery','pumpkin','orange','apple']
```

Methods

```
a = [98,76,87,45,90,23,65,2,9,20]

a.append(4)

a.extend(b)
```

Searching/Sorting

```
a = [98,76,87,45,90,23,65,2,9,20]

a.index(87)

a.sort()
```

Built Functions

`len()`

`min()`

`max()`

`sum()`

Helping Functions

`dir()`

`help()`

`type()`

Tuples

( ) indicates that it's a Tuple

Tuple is immutable collection

Tuples

```
>>> a = ('apple','banana','mango')
>>> b = (1,2,3,46,43,11,6)
>>> a
('apple','banana','mango')
>>> b
(1,2,3,46,43,11,6)
>>> c = ('apple',79,98,True)
>>> c[0]='mango'
TypeError: 'tuple' object does not support item
    assignment
```

Note:

Tuple are just locked sequences.

They cannot be changed

Dictionaries
{key:value}

```
>>> fruits_count = {'apples':50,'mangoes':25,'pineapple' : 6}

>>> fruits_count['apples']

50

>>> vegetable_count = {'potato':20,'tomato':35,'brinjal':20}

>>> vegetable_count['brinjal']

20
```

Indentation

```
if (x==0)
{
printf("Say Hello");
printf("World");
}
else
{
printf("Say Bye")
}
```

```
if x==0:
    print("Say Hello")
    print("World")
else:
    print("Say Bye")
```

This green bar indicates white space

Note:

```
if x==0:
    print("Say Hello")
    print("World")
    if y == 2:
        print("Ok I Need to do something")
else:
    print("Say Bye")
```

Scope of Braces

```
def hello(x,y):
    if x==0:
        print("Say Hello")
        print("World")
        if y == 2:
            print("Ok I Need to do something")
    else:
        print("Say Bye")
```

Conditionals

Keywords
if, elif ,else

```
if condition:
   ……
else:
   ……
elif condition:
   ……
```

Conditionals

PIT HOLE:

Don't forget to add ':' at the end of your conditional statement.

Indentation is must.

Iterators

for, while

for VAR in LIST

For Loop

for VAR in COLLECTION:

```
l = [23,43,12,32,2]
for i in l:
  print i
```

```
d = {'EK4':'ECE','EK5':'CSE'}
for i in d:
  print d[i]

t = (22,32)
for i in t:
  print i


file = open('names.txt','r')
for i in file:
  print i
```

The Range Function

range(start,end)

```
range(0,50)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]
```

While

while condition:

```
while a<b:
   print a
   …….
   ……. •
```

Iterators

PIT HOLE:

Don't forget to add ':' at the end of your iterative statement.

Indentation is must after your iterative statement.

Making it Clear !

The for statement iterates through a collection

The while statement simply loops until a condition is False

The End

Happy Programming

—

Vihar Kurama