Basic Python Day 2

Topics Covered:

>> Condition - if ... else ...

>> Loops - while and for

>> Function

>> Class & Object

>> Iterators

If else

In [1]:
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

Single line if else

In [2]:
```
print("A") if a > b else print("B")
```

```
A
```

In [3]:
```
print("A") if a > b else print("=") if a == b else print("B")
```

```
A
```

AND Condition

In [5]:
```
c = 500

if a > b and c > a:
  print("Both conditions are True")
```

```
Both conditions are True
```

OR Condition

In [6]:
```
if a > b or a > c:
  print("At least one of the conditions are True")
```

```
At least one of the conditions are True
```

Nested If .. elif .. else ..

In [10]:
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

While Loop

In [8]:
```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

In [15]:
```
i = 1
p = ''
while i < 10:
  p = p + '*'
  print(p)
  i += 1
```

```
*
**
***
****
*****
******
*******
********
*********
```

In [19]:
```
i = 1
p = '*'
while i < 10:
  print(i*p)
  i += 1
```

```
*
**
***
****
*****
******
*******
********
*********
```

In [24]:
```
i = 1
p = '*'
while i < 10:
  print(i*p +2*(10-i-1)*' '+i*p)
  i += 1
```

```
*                  *
**                **
***              ***
****            ****
*****          *****
******        ******
*******      *******
********    ********
*********  *********
******************
```

In [25]:
```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

In [26]:
```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

For loop

In [27]:
```
fruits = ["apple", "banana", "cherry"]

for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

In [38]:
```
i = 0
l = len(fruits)

for i in range(l):
    print(fruits[i])

print("--------------------------")

i = 0
while i < l:
    print(fruits[i])
    i += 1
```

```
apple
banana
cherry
--------------------------
apple
banana
cherry
```

In [35]:
```
from datetime import datetime
str(datetime.now())
```

Out[35]:
```
'2018-12-31 16:06:21.140520'
```

In [39]:
```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

In [3]:
```
def tri_recursion(k):
    if(k>0):
        result = k * tri_recursion(k-1)
        print(result)
    else:
        result = 1
    return result

print("\n\nRecursion Example Results")
tri_recursion(5)
```

```
Recursion Example Results
1
2
6
24
120
```

Out[3]:
```
120
```

In [2]:
```
for x in "banana":
    print(x)
```

```
b
a
n
a
n
a
```

In [4]:
```
def my_function():
    print("Hello from a function")
```

In [5]:
```
my_function()
```

```
Hello from a function
```

In [6]:
```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

In [7]:
```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

In [8]:
```
my_function()
```

```
I am from Norway
```

In [9]:
```
def my_function(x):
    return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
15
25
45
```

In [10]:
```
x = lambda a : a + 10

print(x(5))
```

```
15
```

In [11]:
```
print(x(15))
```

```
25
```

In [12]:
```
an_apple = 27
an_example = 55
any1 = 123
```

In [14]:
```
x = lambda a, b : a * b

print(x(5, 6))
```

```
30
```

In [15]:
```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
13
```

In [18]:
```
def myfunc(n):
    return lambda a : a * n

mytripler = myfunc(3)

print(mytripler(11))
```

```
33
```

In [17]:
```
def myfunc(n):
    return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

```
22
```

In [19]:
```
def myfunc(n):
    return lambda a : a * n

mydoubler = myfunc(2)
mytripler = myfunc(3)

print(mydoubler(11))
print(mytripler(11))
```

```
22
33
```

In [26]:
```
class MyClass:
    x = 5
```

In [27]:
```
MyClass.x
```

Out[27]:
```
5
```

```
All classes have a function called __init__(), which is always executed when the
class is being initiated.

#Use the __init__() function to assign values to object properties, or other
operations that are necessary to do when the object is being created:
```

In [31]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

```
John
36
```

In [32]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

```
Hello my name is John
```

In [36]:
```
class Person:
    def __init__(mysillyobject, name, age):
        mysillyobject.name = name
        mysillyobject.age = age

    def myfunc(abc):
        print("Hello my name is " + abc.name)
        print("Age is " + str(abc.age))

p1 = Person("John", 36)
p2 = Person("Rahul", 32)
p1.myfunc()
```

```
Hello my name is John
Age is 36
```

In [37]:
```
p1.age = 40
```

In [38]:
```
p1.myfunc()
```

```
Hello my name is John
Age is 40
```

In [39]:
```
del p1.age
```

In [40]:
```
p1.myfunc()
```

```
Hello my name is John

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-40-48f76da500d0> in <module>()
----> 1 p1.myfunc()

<ipython-input-36-c7e775c6e2f6> in myfunc(abc)
      6     def myfunc(abc):
      7         print("Hello my name is " + abc.name)
----> 8         print("Age is " + str(abc.age))
      9
     10 p1 = Person("John", 36)

AttributeError: 'Person' object has no attribute 'age'
```

In [41]:
```
p2 = Person("Rahul", 32)
```

In [42]:
```
del p1
```

In [43]:
```
p1.myfunc()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-43-48f76da500d0> in <module>()
----> 1 p1.myfunc()

NameError: name 'p1' is not defined
```

In [44]:
```
p2.myfunc()
```

```
Hello my name is Rahul
Age is 32
```

Python Iterators

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

In [46]:
```
# Technically, in Python, an iterator is an object which implements the iterator p
# which consist of the methods __iter__() and __next__()
```

In [47]:
```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

```
apple
banana
cherry
```

In [48]:
```
mytuple = ("apple", "banana", "cherry")

for x in mytuple:
      print(x)
```

```
apple
banana
cherry
```

In [49]:
```
mystr = "banana"

for x in mystr:
    print(x)
```

```
b
a
n
a
n
a
```

In [52]:
```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        x = self.a
        self.a += 1
        return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

```
1
2
3
4
5
```

To prevent the iteration to go on forever, we can use the *StopIteration* statement

In [54]:
```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        if self.a <= 20:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration

myclass = MyNumbers()
myiter = iter(myclass)

for x in myiter:
    print(x)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

In [66]:
```
import mymodule

mymodule.greeting("Rohit")
```

```
Hello, Rohit
```

In [65]:
```
a = mymodule.person1["age"]
print(a)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-65-ce9d8a2d7333> in <module>()
----> 1 a = mymodule.person1["age"]
      2 print(a)

AttributeError: module 'mymodule' has no attribute 'person1'
```

In [64]:
```
import mymodule
```

In [ ]:
```

```