Object References (Python)

1. What are objects in Python.

All the datatypes you have studied in Python till now are all Objects.

Eg. Creating list -

```
mylist = ['a','b','c','d']
```

Object Oriented way,

```
mylist = list('abcd')     #split every character to be a part of the list
```

Similarly, dict(), str(), etc

Every object that comes into existence, is allocated a id() that remains same for the lifetime of a program

2. Mutable & Immutable Objects.

Mutable :

mutable sequences	: list()
set type	: set()
mapping type	: dict()
classes, class instances	

Immutable :

numbers	: int(), float(), complex()
immutable sequences	: str(), tuple()

Example of Mutable :

```
> mylist = [ 1, 2, 3, 4]
> mylist[3] = 0
> print mylist
[ 1, 2, 0, 4]          #mutated object
```

Example of Immutable :

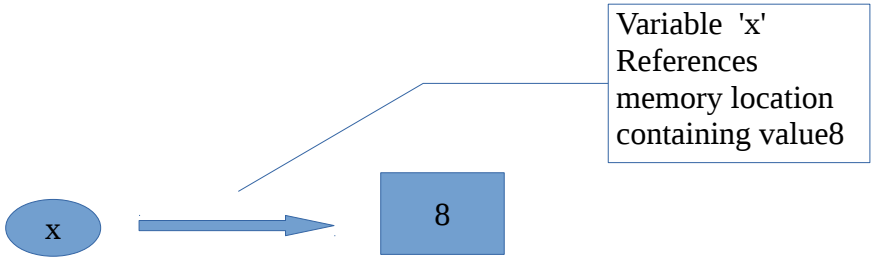
```
> mystr = “AliceInWonderland”

> mystr[2]='x'              #Error!!!! Type-Error!
```

(Find : What are other types of Errors in Python? When are they encountered?)

3. What are variables in Python?

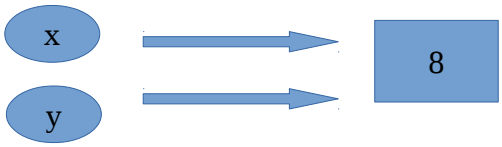
Variables link to objects



- The links aka references.
- They create/manipulate object references/bindings

5. Since objects are references, what happens if assignment of references??

like, y=x # Assignment statements do NOT copy objects.
Creates more references to the same memory location,



Try :
> x = [1,2,3,4]
> y = x

> x[2] = 0
> x
[1, 2, 0, 4]
> y
[1, 2, 0, 4]

> y[0]=9
> x
[9, 2, 0, 4]
> y
[9, 2, 0, 4]

Changing x, changed y too. Because they are referring to the same memory location.

Assignment Operator in Python works as a shallow copy

6. Shallow Copy vs Deep Copy.

As we saw, assignment operator y = x, just makes y as another reference to the same memory location.
Variable 'y' is a shallow copy of variable 'x'

[Can be verified by checking id(x) == id(y)]

What's the way to make total copy of a variable. So that changing one doesn't change the other? Deep Copy
Concept of Deep Copy is to create a new Object, with same state as the original Object.

Eg. y = x[:]

[Can be verified by checking id(x) != id(y)]

Try : Explained Deep Copy,
> x = [1,2,3,4]
> y = x[:]

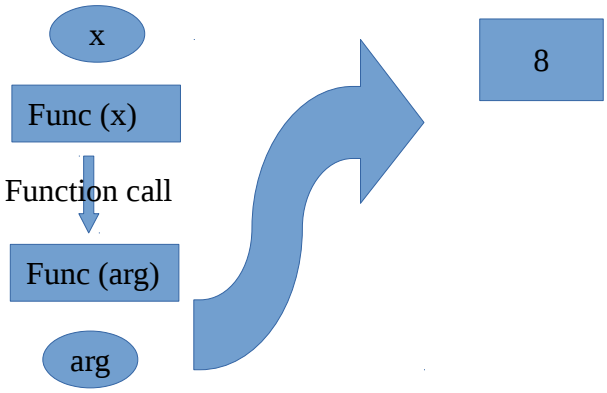
> x[2] = 0
> x
[1, 2, 0, 4]
> y
[1, 2, 3, 4]

> y[0]=9
> x
[1, 2, 0, 4]
> y
[9, 2, 3, 4]

7. How are values passed in functions?

- They are 'pass by assignment' or 'call-by-object-reference'
- References are passed as arguments
- Like a shallow copy of a variable

Eg. If I want to pass variable x to a function. What will happen to a function



Example from StackOverFlow : http://stackoverflow.com/questions/986006/how-do-i-pass-a-variable-by-reference