Python, Call by Value

```
def change(x):          # x is formal parameter
    x = x + 10          # a is Actual parameter
    print(x)
a = 10
print(a)
change(a)
print(a)
```

output
10
20
10

∴ x = x + 10 thats why x is 20 but a remain same 10.

→ In call by value actual parameters copies to formal parameters and changes in formal can't reflect in actual parameters.

```
s = "welcome"
l = [10, 20, 5, 30]
print(s)
print(l)
l[1] = l[1] + 20   # List is Mutable in Python
print(l)
string[1] = "B"    # string is immutable in Python
```

↳ This line we get error

o/p
welcome
[10, 20, 5, 30]
[10, 40, 5, 30]

Python, Call by Reference :-

```
def change(x):      # Here x is list.
    x[1] = x[1] + 20
    print(x)
a = [10, 20, 30, 40]
print(a)
change(a)
print(a)
```

output
[10, 20, 30, 40]
[10, 40, 30, 40]
[10, 40, 30, 40]

∴ value of "a" has changed, same as value of x

x = [10, 40, 30, 40]
a = [10, 40, 30, 40]

→ In call by reference actual parameters copies to formal and if formal get changed, change reflects to the actual parameter

* Instead of call by value and call by reference "call-by-object" or "call-by-object-reference" is a more accurate way of desribing Python.