FP - Some useful notes

Week 04

Aula prática nº 4 – Funções

Functions

- Natural from maths
- Represent a sequence of actions
  - Using a name
  - With explicit parameters i.e. what you need
  - With explicit result i.e. values after return

function

This function has no name. It returns a function object which is assigned to the identifier `double`. We can now call it as a normal function. The statement

```
double = lambda x: x * 2
```

is nearly the same as

```
def double(x):
    return x * 2
```

https://www.tutorialspoint.com/python/python_functions.htm
https://www.datacamp.com/community/tutorials/functions-python-tutorial

The math inspiration


```
def f(x):
    return math.sin(x)
```

```
def f(x,y):
    return x**2 + y**2
```

```
def f(x):
    return x**2
```

```
def f(x):
    return math.sqrt(x)
```

```
def f(x):
    return 1/x
```

The math inspiration

2. Escreva uma função para calcular o polinómio $p(x)=x^2+2x+3$ e use-a num programa para calcular e mostrar os valores de $p(0)$, $p(1)$, $p(2)$ e $p(p(1))$. Confira os resultados.

```
def p(x):
    return x**2 + 2*x + 3
```

```
def pol(a,b,c,x):
    return a*(x**2) + b*x + c
```

Do you agree?

```
P(1) < = >  pol(1,2,3,1)
```

The math inspiration

2. Escreva uma função para calcular o polinómio $p(x)=x^2+2x+3$ e use-a num programa para calcular e mostrar os valores de $p(0)$, $p(1)$, $p(2)$ e $p(p(1))$. Confira os resultados.

```
def p(x):
    return x**2 + 2*x + 3
```

```
def pol(a,b,c,x):
    return a*(x**2) + b*x + c
```

Do you agree?

```
p(p(1)) < = >  pol(1,2,3,pol(1,2,3,1))
```

Factorial (n!)

$U_0 = 1$ se n=0

$U_{n+1} = n * U_n$ se n>0

```
def factorial( n ):
    if n=0 :
       return 1
     else:
       return  n * factorial( n-1)
```

3! < = > factorial(3)

A way to “encapsulate” fixed content…

```
def dc1():
  message="""
  As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;
"""
  return message
```

```
def c1():
  message="""
  As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;
"""
  print( message )
  return
```

Do you agree?

< = >

```
for i in range(5) :
    print( rc1() )
```

```
for i in range(5):
    c1()
```

… and call it using a simple name

```
def dc1():
  message="""
  As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;
"""
  return message
```

```
def c1():
  message="""
  As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;
"""
  print( message )
  return
```

Do you agree?

```
for i in range(5) :
   print( dc1())
```

< = >

```
for i in range(5) :
   c1()
```

Functions as chuncks of related code…

```
def divRem(a, b):
    return a//b, a%b
```

```
def printNtimes(msg, n):
    if n > 0:
        print(msg)
        printNtimes(msg, n-1)
```

```
def fullname(first, last):
    first = first.title()    # title makes
    last = last.title()
    return first + " " + last
```

Referred using a smaller name with parameters

```
def divRem(a, b):
    return a//b, a%b

q, r = divRem(13, 4)
print(q, r)
```

```
def printNtimes(msg, n):
    if n > 0:
        print(msg)
        printNtimes(msg, n-1)
```

```
def fullname(first, last):
    first = first.title()    # title makes
    last = last.title()
    return first + " " + last

print( fullname("maria", "costa") )
print( fullname("fernando", "pessoa") )
```

Lambda

lambda

```
lambda arguments: expression
```

- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming besides other types of expressions in functions.

```
# Python code to illustrate cube of a number
# showing difference between def() and lambda().
def cube(y):
    return y*y*y;

g = lambda x: x*x*x
print(g(7))

print(cube(5))
```

```
(lambda x: x * x)(3)
```

https://www.w3schools.com/python/python_lambda.asp
https://realpython.com/python-lambda/
https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/

lambda

Syntax

As you saw in the previous sections, a lambda form presents syntactic distinctions from a normal function. In particular, a lambda function has the following characteristics:

- It can only contain expressions and can't include statements in its body.
- It is written as a single line of execution.
- It does not support type annotations.
- It can be immediately invoked (IIFE).

```
intervalo1 = lambda a, b : (b,a) if a>b else  (a,b)
```

```
intervalo1 = lambda a, b : if a>b : (b,a) else: (a,b)
```

https://realpython.com/python-lambda/

lambda

Syntax

As you saw in the previous sections, a lambda form presents syntactic distinctions from a normal function. In particular, a lambda function has the following characteristics:

- It can only contain expressions and can't include statements in its body.
- It is written as a single line of execution.
- It does not support type annotations.
- It can be immediately invoked (IIFE).

```
intervalo1 = lambda a, b : (b,a) if a>b else  (a,b)
```

```
intervalo1 = lambda a, b : if a>b : (b,a) else: (a,b)
```

https://realpython.com/python-lambda/

Just test

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

# Output: [4, 6, 8, 12]
print(new_list)
```

```
def myfunc(n):
  return lambda a : a * n

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

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

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

# Output: [2, 10, 8, 12, 16, 22, 6, 24]
print(new_list)
```

https://www.programiz.com/python-programming/anonymous-function
https://www.w3schools.com/python/python_lambda.asp

Curiosities: default values

```
#!/usr/bin/python3

# Function definition is here
def printinfo( name, age = 35 ):
   "This prints a passed info into this function"
   print ("Name: ", name)
   print ("Age ", age)
   return

# Now you can call printinfo function
printinfo( age = 50, name = "miki" )
printinfo( name = "miki" )
```

When the above code is executed, it produces the following result –

```
Name:  miki
Age  50
Name:  miki
Age  35
```

https://www.tutorialspoint.com/python3/python_functions.htm

Curiosities: using parameters names

```
#!/usr/bin/python3

# Function definition is here
def printinfo( name, age ):
   "This prints a passed info into this function"
   print ("Name: ", name)
   print ("Age ", age)
   return

# Now you can call printinfo function
printinfo( age = 50, name = "miki" )
```

When the above code is executed, it produces the following result –

```
Name:  miki
Age  50
```

https://www.tutorialspoint.com/python3/python_functions.htm

Functions and parameters

Function parameters

Pass by value

unmutable

- *safe from change*, i.e. the contents of objects cannot be changed within a method
- Just copies them… - pass by value

- Basic values
- Int, float

Pass by reference

mutable

- objects of which contents can be changed within a method
- Uses reference – send a reference to the data

- "complex" structures
- List, dictionaries…

https://medium.com/@tyastropheus/tricky-python-ii-parameter-passing-for-mutable-immutable-objects-10e968cbda35
https://www.robasworld.com/pythonpassby/

Immutable objects can't be changed.

Mutable objects can be changed.

Type	Immutable?
`int`	Yes
`float`	Yes
`bool`	Yes
`complex`	Yes
`tuple`	Yes
`frozenset`	Yes
`str`	Yes
`list`	No
`set`	No
`dict`	No

https://realpython.com/pointers-in-python/

Function parameters

Pass by value

unmutable

```
>>> def increment(n):
...     n += 1
...     return n
>>> a = 3
>>> a = increment(a)
# the return value of increment() is captured!
>>> print(a)
a = 4
# a now refers to the new object created by the function
```

Pass by reference

mutable

```
>>> def increment(n):
...     n.append([4])
>>> L = [1, 2, 3]
>>> increment(L)
>>> print(L)
L = [1, 2, 3, 4]    # a changed!
```

https://medium.com/@tyastropheus/tricky-python-ii-parameter-passing-for-mutable-immutable-objects-10e968cbda35

Pass by value

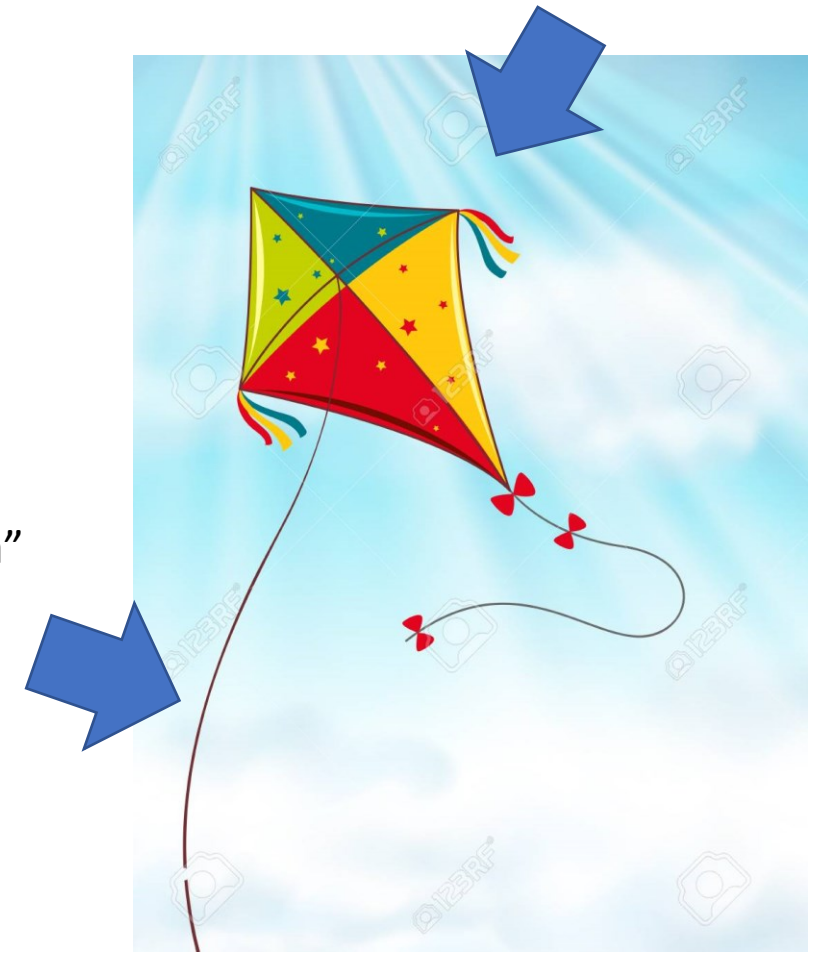
the copier
i.e. Creates the copy

The original
i.e. “data”

The copy to use
- No change on original

Pass by reference

The kite
i.e. “data”



The reference
i.e. allows to reach the “data”

Try this code in pythontutor

```
def changeme( mylist ):
    mylist.append([1,2,3,4])
    print("values inside the function", mylist)
    return

mylist=[10,20,30]
changeme( mylist)
print("values outside the function", mylist)
```

http://www.pythontutor.com/visualize.html

Get live help for free in the Python tutoring Discord chat room

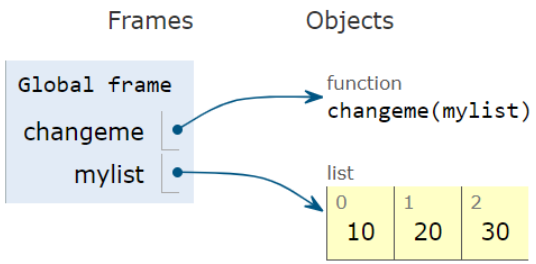
Python 3.6
(known limitations)

```
1  def changeme( mylist ):
2      mylist.append([1,2,3,4])
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)



Get live help for free in the Python tutoring Discord chat room

Python 3.6
(known limitations)

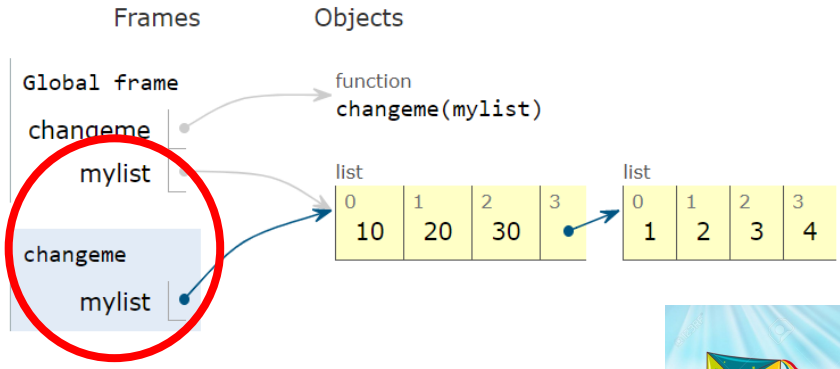
```
1  def changeme( mylist ):
2      mylist.append([1,2,3,4])
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
values inside the function [10, 20, 30, [1, 2, 3
```



Same reference (line to the same “kite”)
They are the same…

Get live help for free in the Python tutoring Discord chat room

Python 3.6
(known limitations)

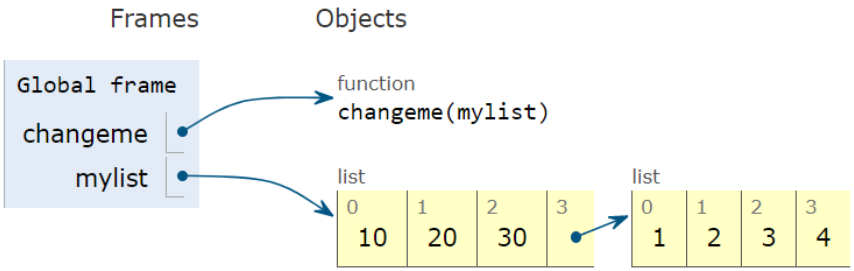
```
1  def changeme( mylist ):
2      mylist.append([1,2,3,4])
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
values inside the function [10, 20, 30, [1, 2, 3
values outside the function [10, 20, 30, [1, 2,
```



Try this code in pythontutor

```
def changeme( mylist ):
    mylist=[1,2,3,4]
    print("values inside the function", mylist)
    return

mylist=[10,20,30]
changeme( mylist)
print("values outside the function", mylist)
```

http://www.pythontutor.com/visualize.html

Get live help for free in the Python tutoring Discord chat room

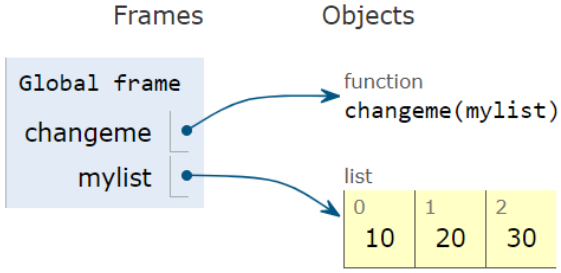
Python 3.6
(known limitations)

```
1  def changeme( mylist ):
2      mylist=[1,2,3,4]
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)



Get live help for free in the Python tutoring Discord chat room

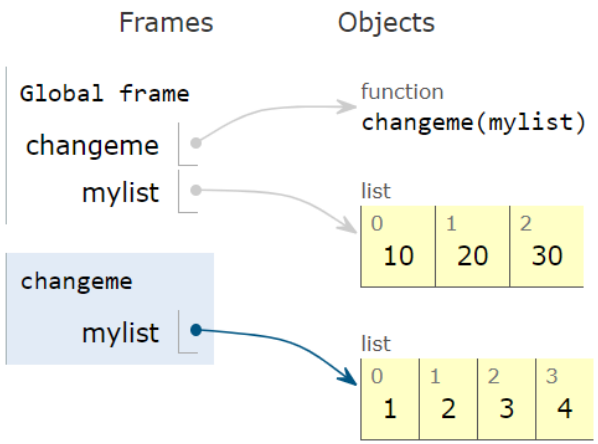
Python 3.6
(known limitations)

```
1  def changeme( mylist ):
2      mylist=[1,2,3,4]
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

→ line that just executed
→ next line to execute

Print output (drag lower right corner to resize)



Get live help for free in the Python tutoring Discord chat room

Python 3.6
(known limitations)

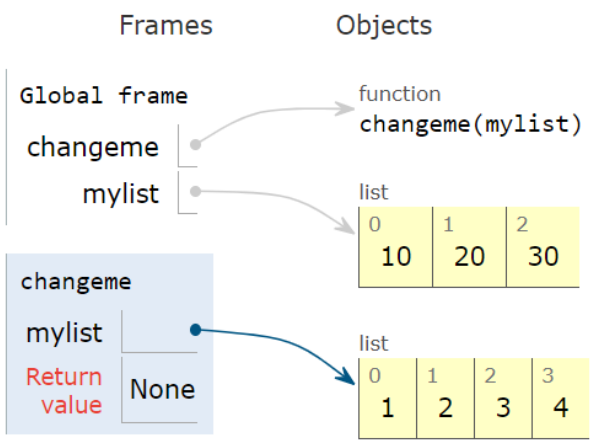
```
1  def changeme( mylist ):
2      mylist=[1,2,3,4]
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
values inside the function [1, 2, 3, 4]
```



Get live help for free in the Python tutoring Discord chat room

Python 3.6
(known limitations)

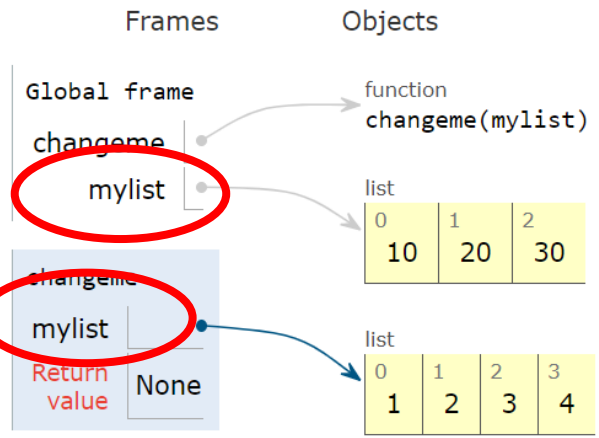
```
1  def changeme( mylist ):
2      mylist=[1,2,3,4]
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
values inside the function [1, 2, 3, 4]
```



Same name but not the same…
Not linked to the same “kite”

Get live help for free in the Python tutoring Discord chat room

Python 3.6
(known limitations)

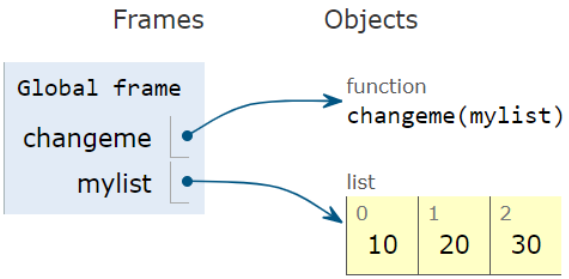
```
1  def changeme( mylist ):
2      mylist=[1,2,3,4]
3      print("values inside the function", mylist)
4      return
5
6  mylist=[10,20,30]
7  changeme( mylist)
8  print("values outside the function", mylist)
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
values inside the function [1, 2, 3, 4]
```



Pass by reference

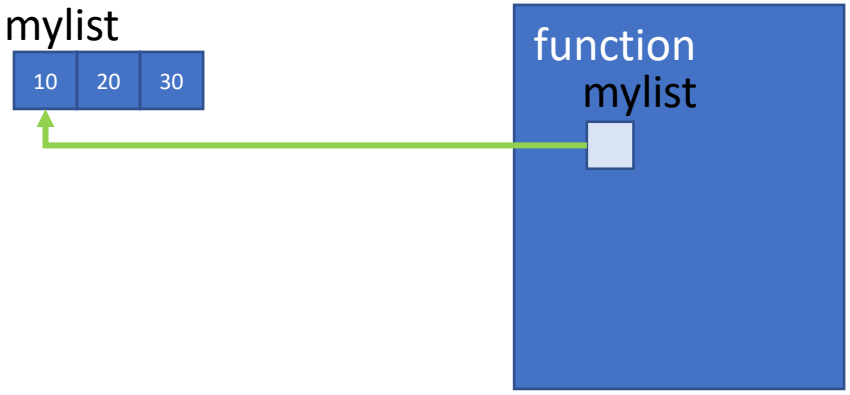
All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example −


```
#!/usr/bin/python

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

Pass by reference

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example −




```
#!/usr/bin/python

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

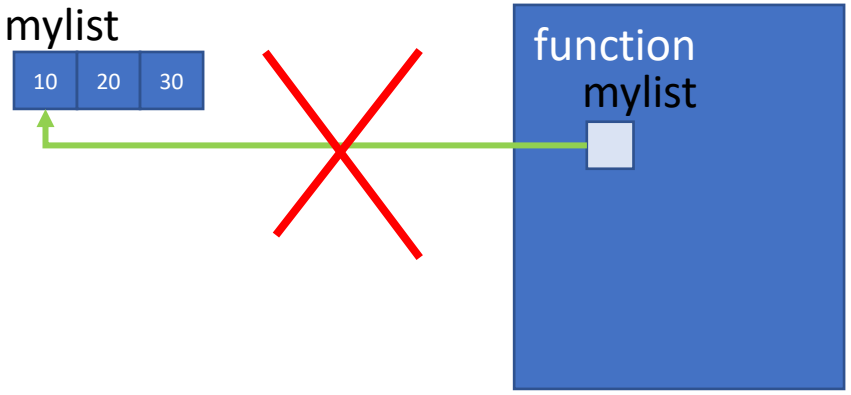
```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

Pass by reference

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example −

```
#!/usr/bin/python

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```



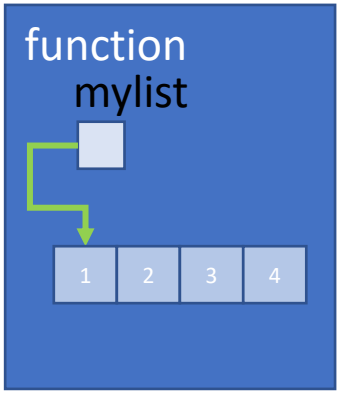
Loose reference…
i.e. override the reference
Loose the original data

Pass by reference

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example −

```
#!/usr/bin/python

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```



Loose reference…
i.e. override the reference
Loose the original data

The End