Introduction to Python

Modified Grading scheme

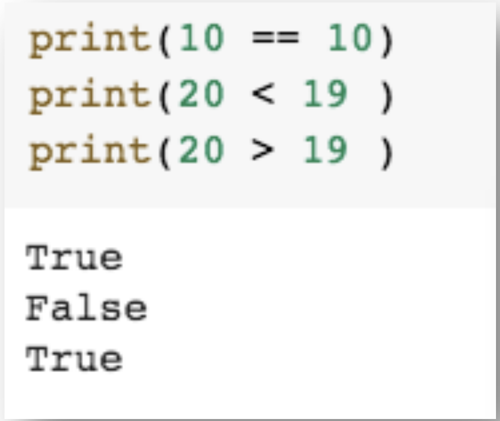
~~Quiz~~	~~- 10%~~
Mid term	– 20 marks
Continuous evaluation, viva (10) (including attendance)	– 50 marks ~~20%~~
End Semester (Theory)	– 20 marks ~~25%~~
End Semester (Practical)	– 10 marks ~~25%~~

Introduction to Python

More on functions/Operators

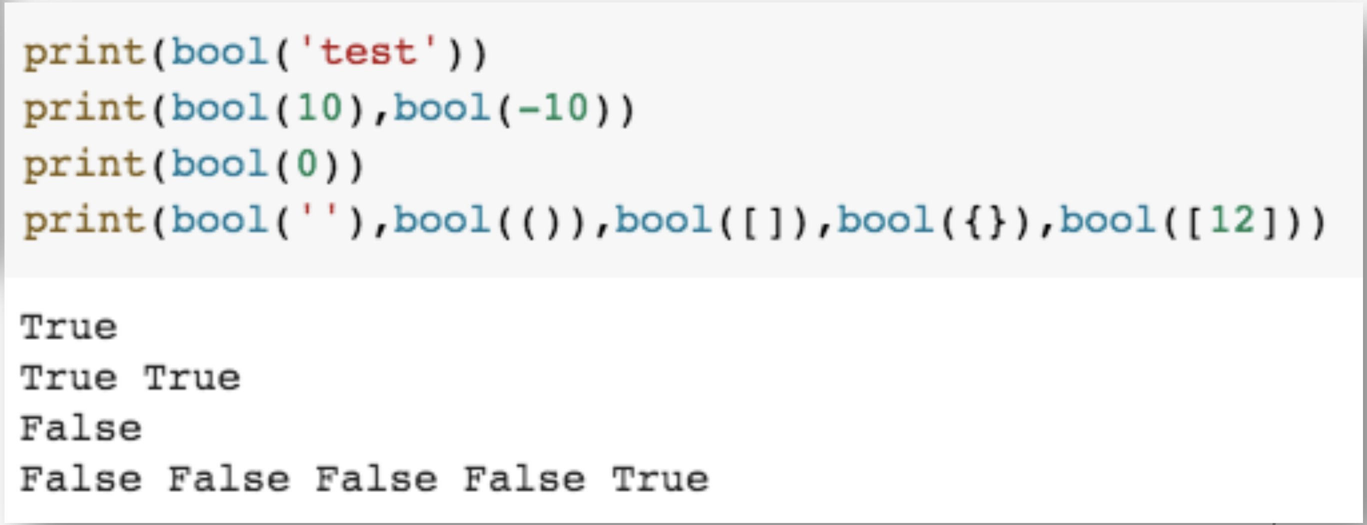
Boolean data type

Any expression in python returns True or False.
The function **bool()** evaluates value it to True or False. Every value to a bool() will return True except 0 and empty string, tuple, set, list, dict



```
print(10 == 10)
print(20 < 19 )
print(20 > 19 )
```

```
True
False
True
```



```
print(bool('test'))
print(bool(10),bool(-10))
print(bool(0))
print(bool(''),bool(()),bool([]),bool({}),bool([12]))
```

```
True
True True
False
False False False False True
```

Introduction to Python

More on functions/Operators

Logical operators:

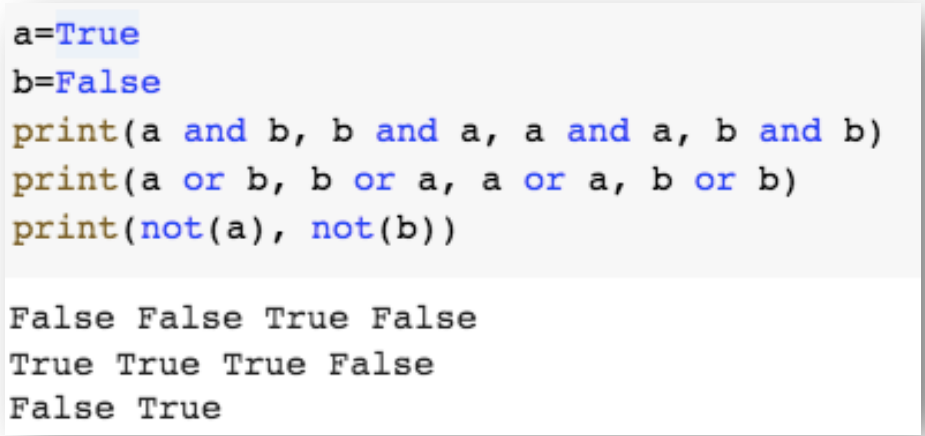
and Output True if two statements evaluates to True

or Output True if either of two statements evaluates to True

Not Output inverse of the state (T to F / F to T) Takes one input

First	Second	Output
True	True	True
True	False	False
False	True	False
False	False	False

First	Second	Output
True	False	True
False	True	True
True	False	True
False	False	False



```
a=True
b=False
print(a and b, b and a, a and a, b and b)
print(a or b, b or a, a or a, b or b)
print(not(a), not(b))
```

```
False False True False
True True True False
False True
```

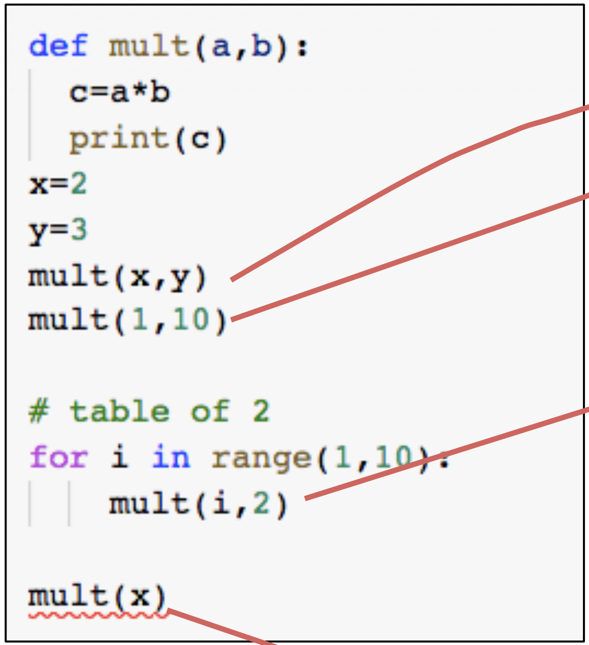
Introduction to Python

More on functions/Operators

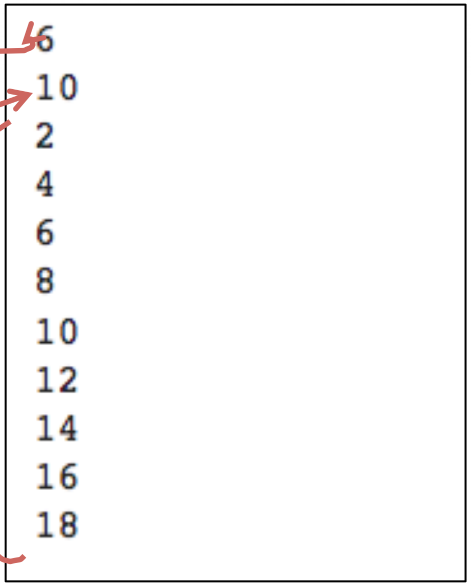
Any number of arguments could be passed to a **function**.

- The arguments could be passed as **positional argument** or **keyword assignment**.
- The arguments could be also set with default parameter.

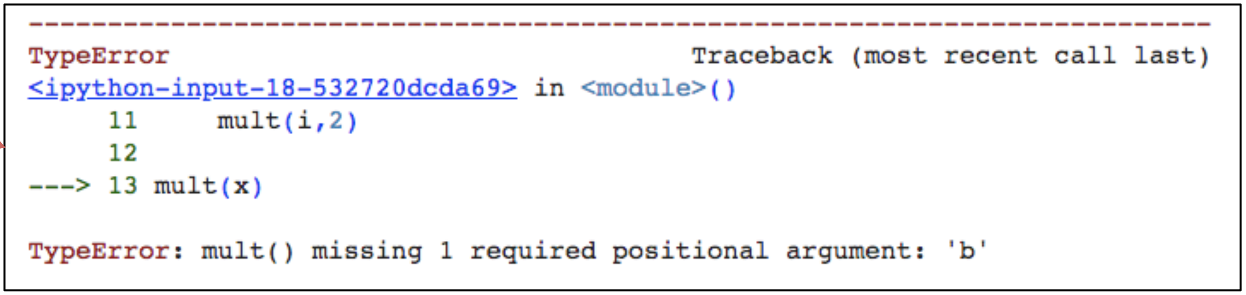
Introduction to Python



```python
def mult(a,b):
    c=a*b
    print(c)
x=2
y=3
mult(x,y)
mult(1,10)

# table of 2
for i in range(1,10):
    mult(i,2)

mult(x)
```

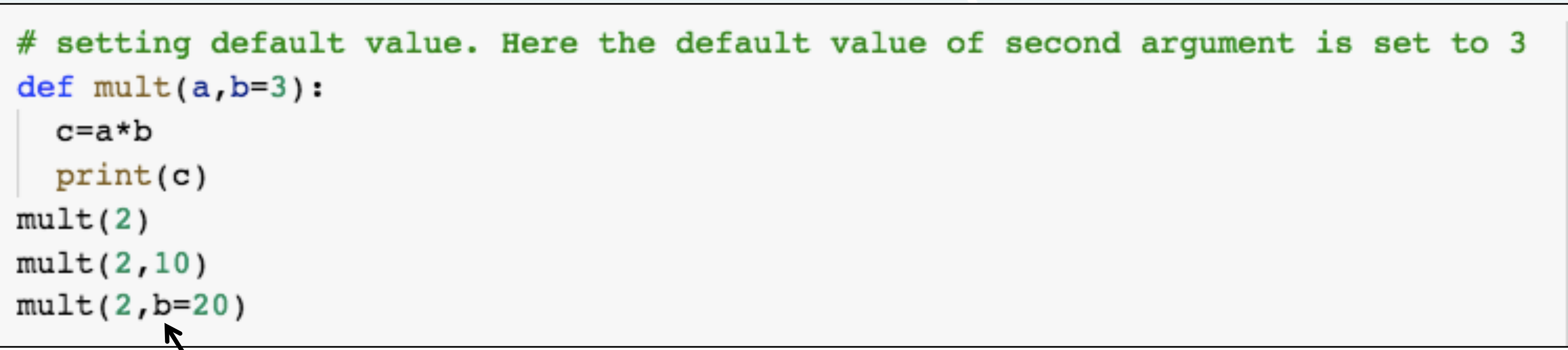


```
6
10
2
4
6
8
10
12
14
16
18
```



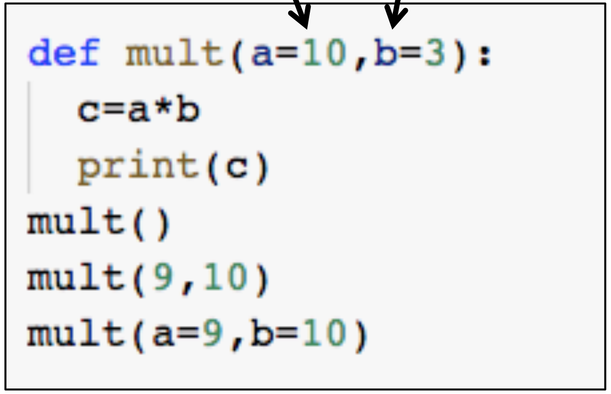
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-532720dcda69> in <module>()
     11     mult(i,2)
     12 
---> 13 mult(x)

TypeError: mult() missing 1 required positional argument: 'b'
```

Introduction to Python



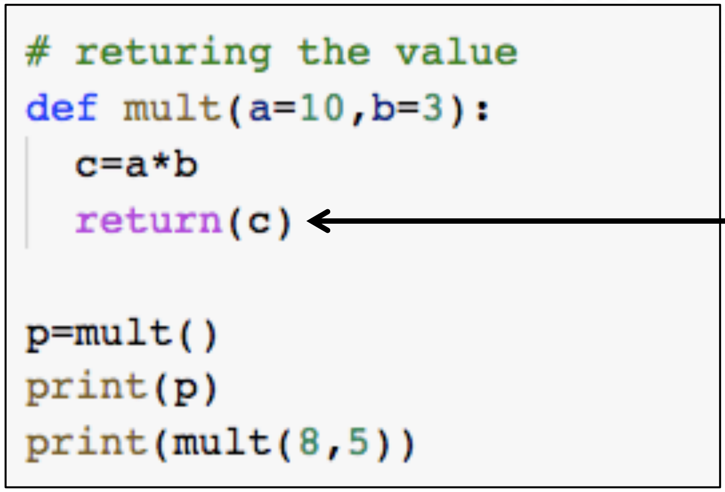
```python
# setting default value. Here the default value of second argument is set to 3
def mult(a,b=3):
    c=a*b
    print(c)
mult(2)
mult(2,10)
mult(2,b=20)
```

keyword argument



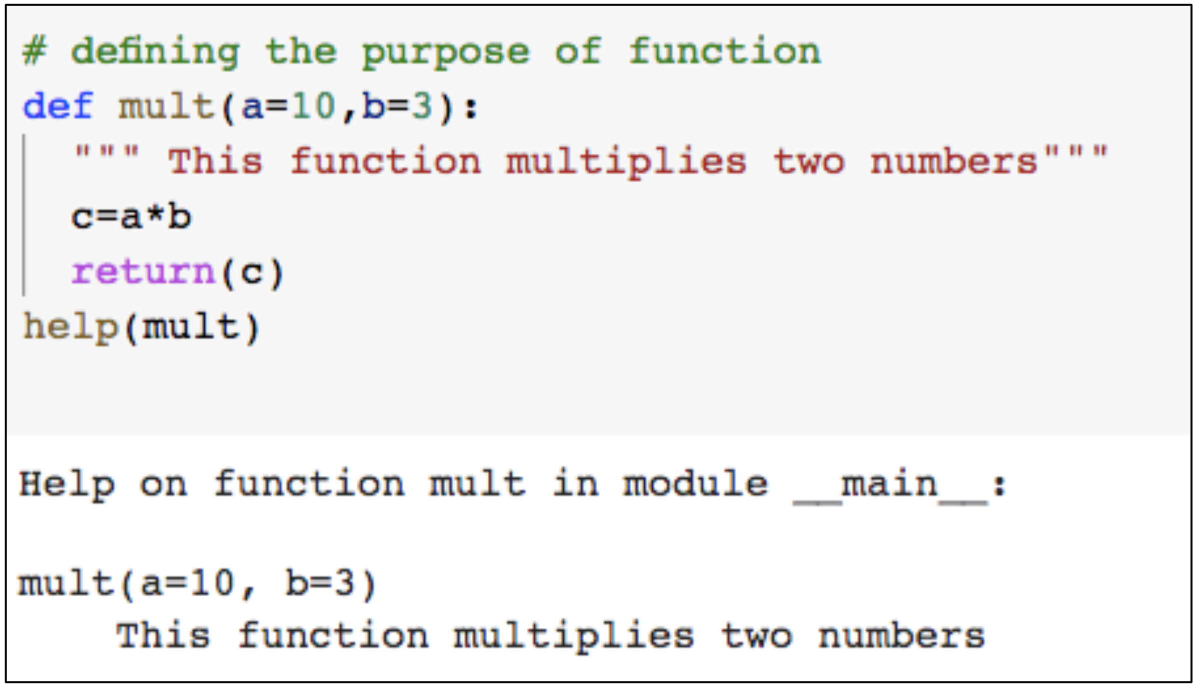
```python
def mult(a=10,b=3):
    c=a*b
    print(c)
mult()
mult(9,10)
mult(a=9,b=10)
```

Introduction to Python



```
# returing the value
def mult(a=10,b=3):
    c=a*b
    return(c)

p=mult()
print(p)
print(mult(8,5))
```

Function returns the value

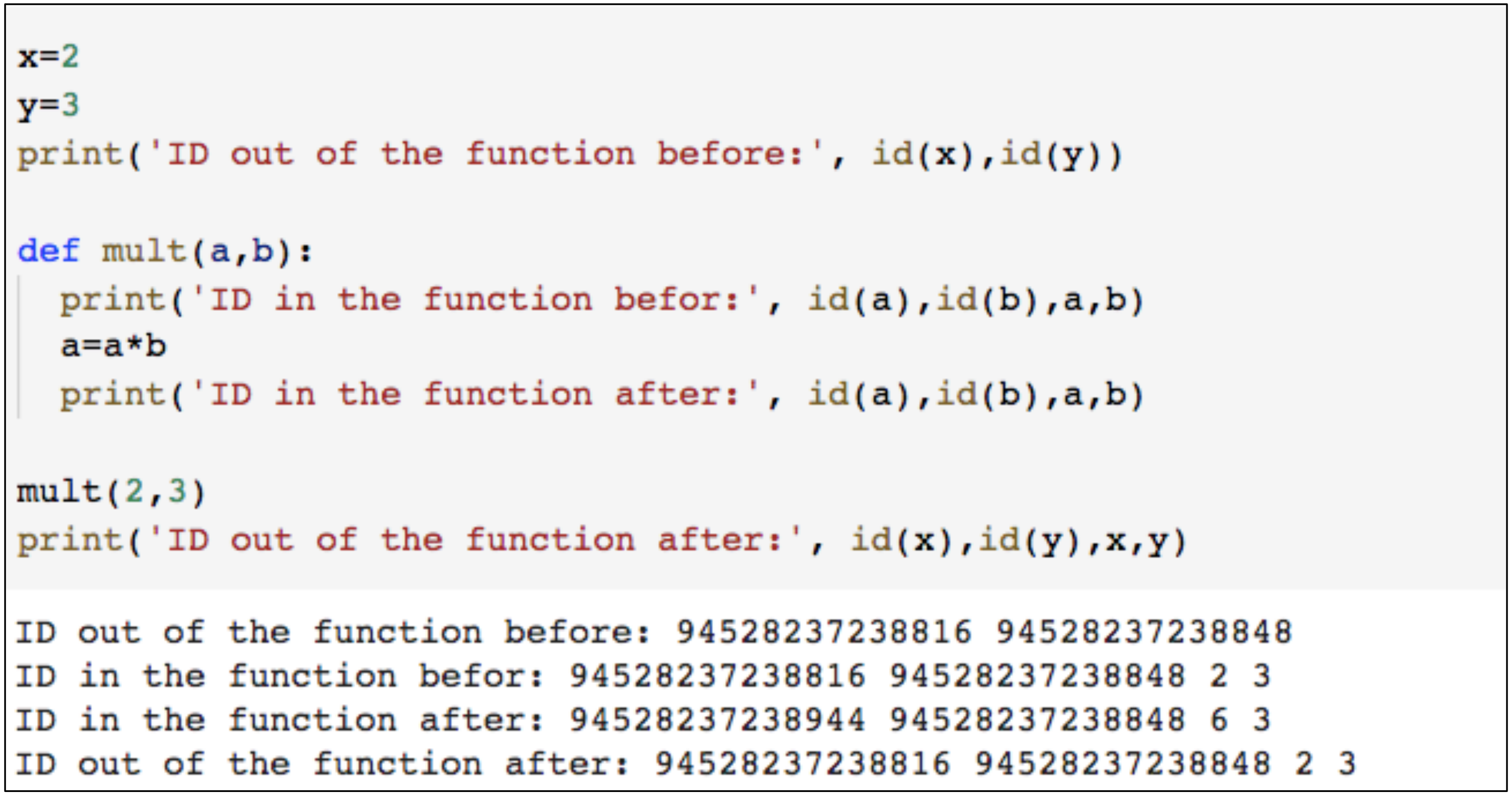


```
# defining the purpose of function
def mult(a=10,b=3):
    """ This function multiplies two numbers"""
    c=a*b
    return(c)
help(mult)
```

```
Help on function mult in module __main__:

mult(a=10, b=3)
    This function multiplies two numbers
```

Introduction to Python

Argument pass by objects reference



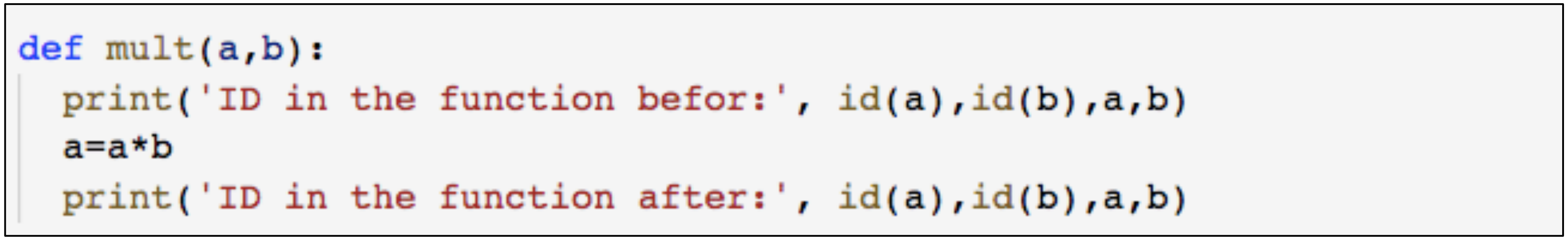
```python
x=2
y=3
print('ID out of the function before:', id(x),id(y))

def mult(a,b):
    print('ID in the function befor:', id(a),id(b),a,b)
    a=a*b
    print('ID in the function after:', id(a),id(b),a,b)

mult(2,3)
print('ID out of the function after:', id(x),id(y),x,y)
```

```
ID out of the function before: 94528237238816 94528237238848
ID in the function befor: 94528237238816 94528237238848 2 3
ID in the function after: 94528237238944 94528237238848 6 3
ID out of the function after: 94528237238816 94528237238848 2 3
```

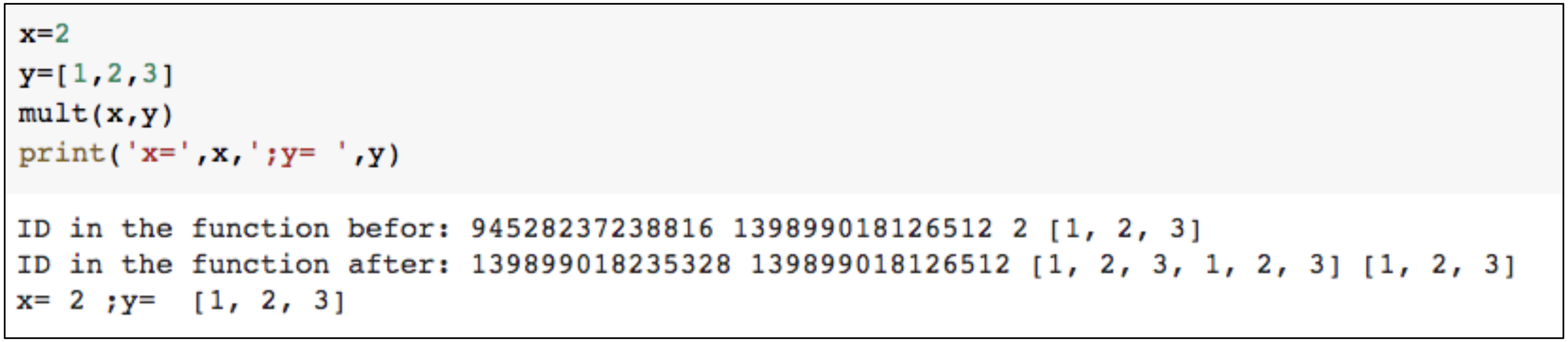
Pass-by-value, wherein the value of the input variables to the function remains unchanged even-if the variable is updates within the function.

Introduction to Python

Argument pass by objects reference



```
def mult(a,b):
    print('ID in the function befor:', id(a),id(b),a,b)
    a=a*b
    print('ID in the function after:', id(a),id(b),a,b)
```

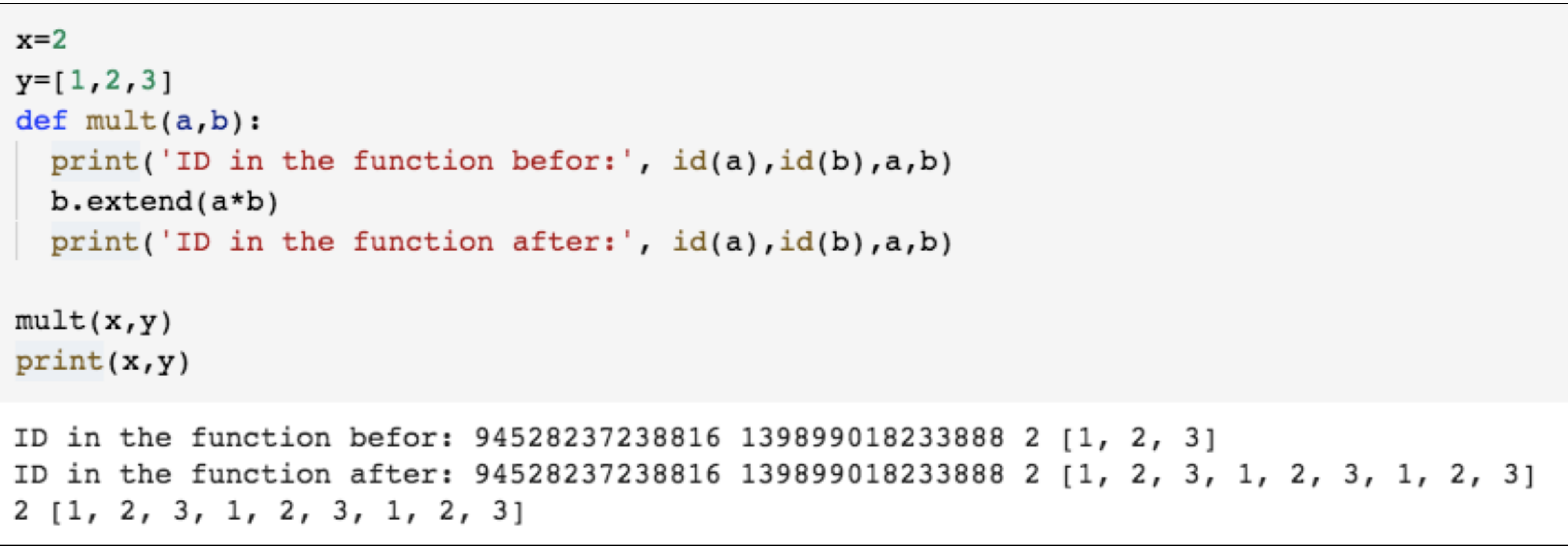


```
x=2
y=[1,2,3]
mult(x,y)
print('x=',x,';y= ',y)
```

```
ID in the function befor: 94528237238816 139899018126512 2 [1, 2, 3]
ID in the function after: 139899018235328 139899018126512 [1, 2, 3, 1, 2, 3] [1, 2, 3]
x= 2 ;y=  [1, 2, 3]
```

Introduction to Python

Argument pass by objects reference



```
x=2
y=[1,2,3]
def mult(a,b):
    print('ID in the function befor:', id(a),id(b),a,b)
    b.extend(a*b)
    print('ID in the function after:', id(a),id(b),a,b)

mult(x,y)
print(x,y)
```

```
ID in the function befor: 94528237238816 139899018233888 2 [1, 2, 3]
ID in the function after: 94528237238816 139899018233888 2 [1, 2, 3, 1, 2, 3, 1, 2, 3]
2 [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Pass-by-reference, wherein the value of the input variables to the function is updated if any in the calling function

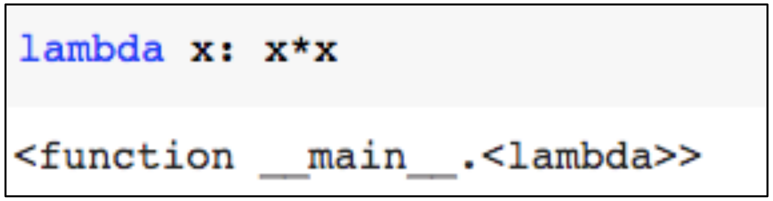
Introduction to Python

Lambda functions

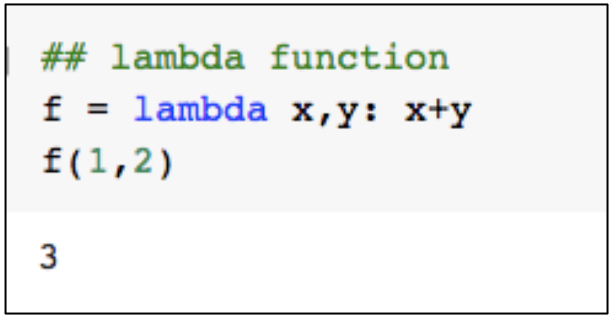
The Lambda function is anonymous function, i.e without a name. The lambda is used instead of function name!!. The syntax is

```
lambda arguments: expression
```

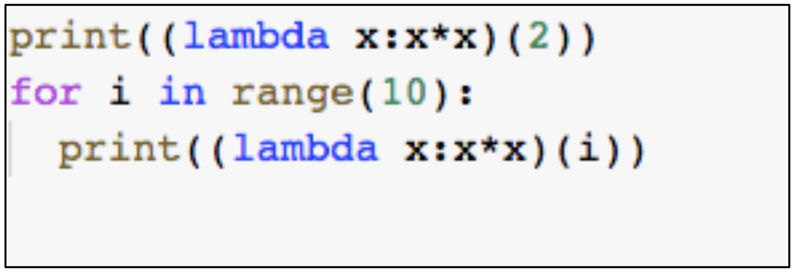
It can have any number of arguments but only one expression



```
lambda x: x*x

<function __main__.<lambda>>
```



```
## lambda function
f = lambda x,y: x+y
f(1,2)

3
```



```
print((lambda x:x*x)(2))
for i in range(10):
    print((lambda x:x*x)(i))
```

Python Modules

The python module is python program (functions, classes, variables), which can be imported into another program and various functions can be accessed. Many modules can be put in package.

The command *import* loads the module.

```
import random  ##
 The functions are called as random.somefunction

import random as rn
 The functions are called as rn.somefunction

from random import randint
   This statement imports the function randint, which can be called as
randint()
```

Introduction to Python

Python Modules

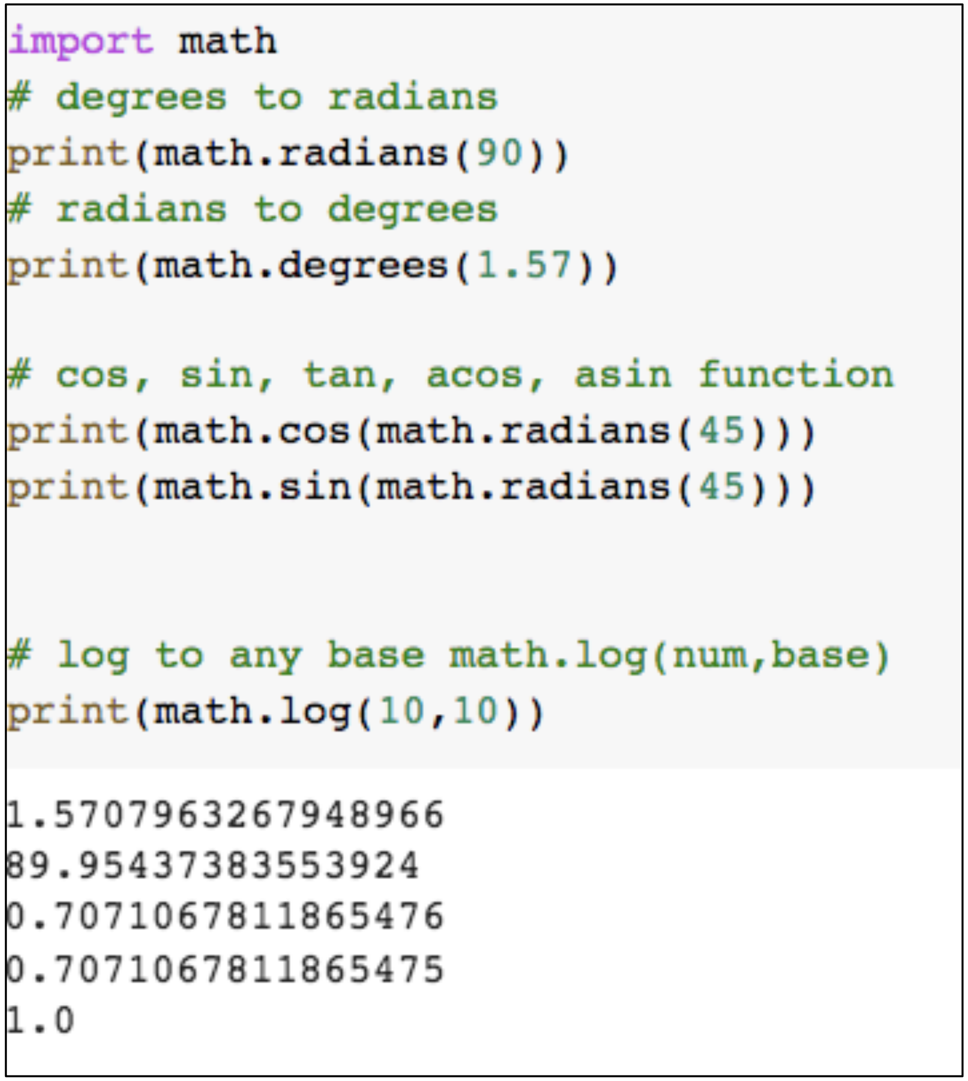
```
import random
print(random.random()) ## prints random number between 0-1
print(random.randint(1,100)) ## prints random number between 1 and 100

import random as rn
print(rn.random()) ## prints random number between 0-1
print(rn.randint(1,100)) ## prints random number between 1 and 100
```

```
from random import random
print(random())
```

```
0.8121944402537778
```

Introduction to Python

Python Modules



```python
import math
# degrees to radians
print(math.radians(90))
# radians to degrees
print(math.degrees(1.57))

# cos, sin, tan, acos, asin function
print(math.cos(math.radians(45)))
print(math.sin(math.radians(45)))


# log to any base math.log(num,base)
print(math.log(10,10))
```

```
1.5707963267948966
89.95437383553924
0.7071067811865476
0.7071067811865475
1.0
```

Introduction to Python

Variable scopes

The concept of scope rules on how variables and names are accessible in a code or variables visibility in a code. The python follows LEGB (local enclosing, global and builtin scopes).

local (function) scope: contains names defined within a function. These will be visible only within the function

Enclosing scope: exists for nested functions. Contains names defined in the enclosing function.

Global scope: names defined at top level of program

Built-in scope: contains reserved keywords

The LEGB rule can be assumed to be a name lookup procedure, which determines the order in which Python looks up names.