Functions in Python

Difference between *function* and *method*

Functions:

def hello\_world (n: int):

 for i in range(n):

 print("Hello World")

hello\_world(3)

Method:

a = [1, 2, 3]

a.insert(1, 4)

print(a)

The data type for the parameters CAN be omitted

def message(number):

print(type(number))

message(5)

Of course, the traditional way of passing parameters into functions can work, but also passing by the value

def introduction(first\_name, last\_name):

 print("Hello, my name is", first\_name, last\_name) introduction(first\_name = "James", last\_name = "Bond")

introduction(last\_name = "Skywalker", first\_name = "Luke")

We can also make some of the parameters to “default” so that it can be omitted during the calling of the function.

def introduction(first\_name, last\_name="Smith"):

 print("Hello, my name is", first\_name, last\_name) introduction("James", "Doe") # output: Hello, my name is James Doe

introduction("Henry") #output: Hello, my name is Henry Smith

All the above are *void* functions in C++, but what about functions with return values?



def boring\_function():

 print("'Boredom Mode' ON.")

 return 123

boring\_function() #does nothing with the return value

print(boring\_function()) #output 123

Null in python, which is equivalent to NULL in C++

value = None # NULL in C++

if value is None:

 print("Sorry, you don't carry any value")

If a function doesn’t have a return value, it is equal to return None

def strange\_function(n):

 if(n % 2 == 0):

 return True

print(strange\_function(2)) # output True

print(strange\_function(1)) # output None

Variables created in the functions are local, just like in C++

def my\_function():

 var = 2 # new local variable

 print(var) # output: 2

var = 1 # local variable

my\_function()

print(var) # output: 1

Lists are different, as explained in “Lists in Python”

def my\_function(my\_list\_1):

 print("Print #1:", my\_list\_1) # output: Print #1: [2, 3]

 print("Print #2:", my\_list\_2) # output: Print #2: [2, 3]

 del my\_list\_1[0] # Pay attention to this line.

 print("Print #3:", my\_list\_1) # output: Print #3: [3]

 print("Print #4:", my\_list\_2) # output: Print #4: [3]

my\_list\_2 = [2, 3]

my\_function(my\_list\_2)

print("Print #5:", my\_list\_2) # output: Print #5: [3]

‘\’ character can be used to tell Python to continue the line of the code in the next line

def bmi(weight, height):

 if height < 1.0 or height > 2.5 or \ # continues to next line

 weight < 20 or weight > 200:

 return None

 return weight / height \*\* 2

print(bmi(352.5, 1.65))