Functions ¶

If you have tasks you need to do many times over, don't code these tasks many times, but instea, use function
so that each time you need to execute the task, you can simply call the function. This brings simplicity, readability
and more importantly, easy software maintenance.

Created by: John C.S. Lui on May 25, 2018.

Important note: *If you want to use and modify this notebook file, please acknowledge the author.*

General syntax of defining a function

# Let's define a function
def function_name (argument_1, argument_2):
    write down the code of your function here
    write down the code of your function here

# Later on in your programe, you can use the *function_name* to call the function
....
function_name(value_1, value_2)
....

Examples

In [ ]:
```
# Example
def print_reverse_name(first_name, last_name):
      print("The person's name is %s %s" %(last_name, first_name))

f_name1 = "John C.S."
l_name1 = "Lui"
print_reverse_name(f_name1, l_name1)

f_name2 = "Frank"
l_name2 = "Kelly"
print_reverse_name(l_name2, f_name2)  # It is the "ordering" that is i
```

In [ ]:
```python
# Let say you need to send emails to the following students with the fo
print("Peter, you are doing great in CSCI2040.")
print("And please, give me a good coure evaluation because my job is o

print("Mary, you are doing great in CSCI2040.")
print("And please, give me a good coure evaluation because my job is o

print("Susan, you are doing great in CSCI2040.")
print("And please, give me a good coure evaluation because my job is o

print("Stanley, you are doing great in CSCI2040.")
print("And please, give me a good coure evaluation because my job is o
```

In [ ]:
```python
# The above code is UGLY, you can simplify it using function
def message_func (name):
    print("%s, you are doing great in CSCI2040." %name)
    print("And please, give me a good coure evaluation because my job 

message_func('Peter')
message_func('Mary')
message_func('Susan')
message_func('Stanley')
```

In [ ]:
```python
# Better yet, we can use list and function
def message_func (name):
    print("%s, you are doing great in CSCI2040." %name)
    print("And please, give me a good coure evaluation because my job 

name_list = ['Peter', 'Mary', 'Susan', 'Stanley']

for name in name_list:
    message_func(name)
```

In [ ]:
```python
# We can also use the list methods we learnt to do more fancy things
def message_func (name):
    print("%s, you are doing great in CSCI2040." %name)
    print("And please, give me a good coure evaluation because my job 

name_list = ['Peter', 'Mary', 'Susan', 'Stanley']

# sort the list
name_list.sort()

print("*****Display items in sorted order:")
for name in name_list:
    message_func(name)

# sort the list in reverse order
name_list.sort(reverse=True)
print("\n*****Display items in reverse sorted order:")
for name in name_list:
    message_func(name)
```

Why we want to use functions?

- You only write a set of code ONCE.
- Good software engineering practice. You can focus on a function to make sure it is correct.
- One can modify the function behavior, instead of modifying spreaded codes around your program

In [ ]:
```
# Let say you want to print a message, and give a list of students.
def message_students (students, message):
    print(message)
    for student in students:
        print('\t *', student)

name_list = ['Peter', 'Mary', 'Susan', 'Stanley']
message = "The following students got 'A' in CSCI2040"

name_list.sort()
message_students(name_list, message)

print("\n")
name_list.sort(reverse=True)
message = "The folloiwng students also hate John C.S. Lui"
message_students(name_list, message)
```

Values returned by a function

Each function can return some values to the caller, and it can be very useful.

In [ ]:
```
def message_students (students, message):
    print(message)
    for student in students:
        print('\t *', student)
    return len(students)

name_list = ['Peter', 'Mary', 'Susan', 'Stanley']
message = "The following students got 'A' in CSCI2040"

name_list.sort()
list_size = message_students(name_list, message)
print("\nThere are %d students in the list" %list_size)
```

Use globals

At times, we want to access some "global variables". We can use the **global** statement

In [ ]:

```python
# def a function
def scopetest (a):
    global b
    b = 4
    a = 12
    print('inside function scopetest, a =', a, ";", "b =", b)

a = 1
b = 2
print ('a=', a, ';', 'b=', b)
scopetest(a)
print ('now a=', a, ";", "b=", b)
```

Exercise

Write a function which takes:

- A list of 10 names
- Another list of 10 numbers which represents the age of the corresponding people in the first list
- Print out the name, the age, and a message that
  if a person is above 50, the message is "over the hill",
  else a message is "has a bright future"
- You may need to look up the syntax for "if... else"

Initialize the two lists, and then call the function.