Activity: function call and function docstring

In the provided file **A14-Function-Call.py** some functions have been defined.

Edit the provided script and for each function add a function docstring and function call.

Activity: function with return

Make a script called **A14-moon.py** and in it:

a. Make a function ***w_moon*** that takes weight on Earth (we) as parameter and returns the weight on the Moon (wm).

Define local variables **g_earth = 9.81** and **g_moon = 1.62,** i.e.
inside the function definition . These variables store gravitational constants

To calculate the weight on the Moon wm use this formula
wm= we *g_moon/g_earth

b. Run (call) this function on a weight of 100 Kg (on Earth) and store the result (the weight on Moon) in a variable called **wm_out**

c. Print a formatted statement like this

```
My object weights 16.51 Kg on the Moon
```

d. Use the function to calculate the weights on the Moon for the following weights
50, 51, 52 ,..,60 Kg on the Earth.
Results should be stored in a list type. Use list comprehension.

Activity: function returns multiple values

Make a script called **A14-moon-jupiter.py** and in it:

a. Make a function called ***w_moon_jupiter*** that takes weight on Earth (we) as parameter and returns both the weight on the Moon **wm** and the weight on Jupiter **wj.**

Define the following non-local variables that stores gravitational constants , i.e., define them outside the function

```
g_earth = 9.81
g_moon = 1.62
g_jupiter = 24.79
```

Calculate the weight on the Moon and on Jupiter by using the following formula

```
wm=we*g_moon/g_earth
wj=we*g_jupiter/g_earth
```

b. Call the function for a weight of 100 Kg and save the result in a variable called my_weight

c. Print variable my_weight

d. Make a formatted print statement reporting the weights on the Moon and Jupiter with 2 decimal precision and use variable my_weight.

Activity: if-statements in functions

In a script **A14-isc.py**

Make a function called *func_isc.*
This function should do the following*:*

- take no parameters.
- use the input function to ask the user this question:
  Do you enjoy this course? Yes or no?
  Save the user's input in a variable called **answer**

- use an **if-elif-else** statement to do the following:
  - if the user answers *yes* print a statement that says *Good!*
  - if the user answers no print a statement that says *Sorry!*
  - if the user answers neither *yes* nor *no*, print a statement saying
    You didn't pick yes or no! Try again.

Run the function *func_isc*

Optional- Recursive algorithm calls itself

Now do the following modifications to the **A14-isc.py** script

Add to the function ***func_isc*** a call to itself, reported in red below

Make a function called *func_isc.* This function should do the following*:*

- take no parameters
- use the input function to ask the user this question:
  Do you enjoy this course? Yes or no?
  Save the user's input in a variable called **answer**

- use an **if-elif-else** statement to do the following:
  - if the user answers *yes* print a statement that says *Good!*
  - if the user answers no print a statement that says *Sorry!*
  - if the user answers neither *yes* nor *no*, print a statement saying
    You didn't pick yes or no! Try again.
    call the function *func_isc*

Run the function *func_isc*

Activity: Functions can call other functions

Restaurant bill

You are in a restaurant, and you get a bill (bill). Write a script called **A14-bill.py** and in it:

- Write a function called ***tax*** that takes as a parameter a bill in dollars (i.e., a number) and calculates and returns the amount of the bill with 8% tax .
  To calculate the bill with tax you can multiply the bill amount by 1.08

- Write another function called ***tip*** that takes as a parameter a bill in dollars and calculates the bill with an 18% tip that is applied to the bill with tax. To calculate the bill with the tip you can multiply the amount of bill with tax by 1.18. Do this by calling the function ***tax***. The function returns the bill including tax and tip.

- Run the function ***tip*** to calculate how much you'd pay total for a 100$ bill. Print result to screen

while loops in functions

In a script called **A14-while.py**

```
import random
while True:
    rand_num = random.randint(1,6)
    if rand_num == 5:
        break
     print(rand_num)
```

a. Put the while loop into a function called *rand_not5(par).* The function takes the parameter *par*, which is an integer number. This parameter defines the range of integer random numbers in *random.randint(1,par).*
The function prints.
You should understand what the function prints, from the code reported above.

b. Run the function on one integer number of your choice

function with no return

Make a script called **A15-moon-argv.py** and in it:

a. Make a function ***w_moon*** that takes weight on Earth (we) as parameter and prints a formatted statement reporting the the weight on the Moon (wm).

If 100 Kg is the weight on Earth the function should print

```
My object weights 16.51 Kg on the Moon
```

Define local variables **g_earth = 9.81** and **g_moon = 1.62,** i.e. inside the function definition. These variables store gravitational constants

To calculate the weight on the Moon wm use this formula

wm= we *g_moon/g_earth

b. Run this function on a weight on Earth of your choice, e.g., 100.0
The weight is provided from the command line.
You should import the argv attribute of the sys module.

function with input function and no return

Make a scrip called **moon-noreturn.py**

a. Define global variables **g_earth = 9.81** and **g_moon = 1.62**

b. Make a function called ***w_moon*.** This function should do the following:

- take no parameters
- use the input function to ask the user to enter a weight and save the user's input in a variable called we
- calculate the weight on the Moon wm
  wm= we *g_moon/g_earth
- print the weight on the Moon

c. Run this function

Exercise

In **extra1.py** do the following:

Define a function called *input_rand* that does the following:

- takes no parameters
- uses the input function to ask the user to input an integer number (N).
- generates random numbers in range 1 to 10 (10 included) N times.
- the function calculates and returns two values: how many times randomly generated numbers were smaller than or equal to 5, and how many times they were larger than 5.

Call the function

Exercise

In **extra2.py** do the following:

Make a function *digit_sum* to sum digits in an integer number. The function should take one number (integer type) as parameter and return the sum of the digit of that number.

For example, for number 1234, the result should be 10 (1+2+3+4).

Example - run the function like this:

```
digit_sum(1234)
```

Exercise: function with two parameters and return

Write a script called ***myfunc1.py*** and in it do the following:

a. Make a function called *mysum* that takes two parameters and returns the sum of those two parameters. Write a doc string within the function describing what this function does

b. Call this function on two lists, and store the result in variable **L**. Print the variable

c. Would you be able to use that same function on two strings or two numbers? Answer in a comment line

Exercise: Taking input from the input function

Make a script called ***input1.py*** and in it:

a. Make a function *input_and_sum()* that does the following:

- takes no parameters
- uses the input function to ask the user to enter two numbers separated by a comma.
- returns the sum (arithmetic) of those two numbers.

b. Run the function on two numbers, and print the result to screen

Exercise: Taking input from the input function, and no return

Make a script called ***input2.py*** and in it:

a. Make a function *input_and_sum()* that does the following:

- takes no parameters
- uses the input function to ask the user to enter two numbers separated by a comma.
- **print** the sum (arithmetic) of those two numbers.

b. Run the function on two numbers. The result should be displayed to screen.
None should not appear in the output.

Exercise: function returns multiple values

Make a script ***myfunc2.py*** and in it:

a. Make a function called *dict_keys_values* that takes a dictionary as a parameter and returns both the list of keys and the list of values of that dictionary.

b. Run your function on this dictionary and store the result in variable d_out

```
d1 = {'a': 1, 'b': 2}
```

c. Print variable d_out

d. Access values in d_out and print the list of keys

e. Access values in d_out and print the list of values

f. Would you be able to run this function on another data type?
Answer with a comment line

Submit to A14:

- **A14-Function-Call.py**
- **A14-moon.py**
- **A14-moon-jupiter.py**
- **A14-isc.py**

The other exercises are optional, but a good practice!
If you have time, try to do A14-bill.py in class.