**Part 1: Answer following questions**

***Why should we use functions at all?***

We use function because it is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. It helps us organize programs into chunks that match how we think about the problem.

***How to define/declare a function?***

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

The syntax for a function definition is:

***def NAME( PARAMETERS ):***

***STATEMENTS***

* Function blocks begin with the keyword in the header - **def** followed by the function name and parentheses **( ( ) )**.
* Any **input parameters** or **arguments** should be placed within these parentheses. You can also define parameters inside these parentheses. The parameter list may be empty, or it may contain any number of parameters separated from one another by commas
* The first statement of a function can be an optional statement - the documentation string of the function or docstring.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

***How to call/use a function?***

Defining a new function does not make the function run. To do that we need a function call.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Function calls contain the name of the function being executed followed by a list of values, called arguments, which are assigned to the parameters in the function definition. Functions can call other functions

***What is return, why and how do we use it?***

A function that returns a value can be called a fruitful function.

A function can produce a value with the return statement, which ends the execution of the function call and optionally pass an expression back to the caller. So if you want to exit a function and then, hand back a value to its caller, use “return”.

Function bodies can contain one or more return statement. They can be situated anywhere in the function body.

***Do we have to use return in every function?***

No, a function can have one or many return statements and it even executes without a return statement but for one time only.

***What are function arguments/parameters, why and how we use it?***

The arguments of a function are defined within the def statement. Like all other variables in Python, there is no explicit type associated with the function arguments. This fact is important to consider when making assumptions about the types of data that your function will receive. Most functions require arguments: the arguments provide for generalization. For example, if we want to find the absolute value of a number, we have to indicate what the number is.

So far we have looked at functions with empty parentheses that do not take arguments, but we can define parameters in function definitions within their parentheses. A parameter is a named entity in a function definition, specifying an argument that the function can accept. Inside the function, the values that are passed get assigned to variables called parameters.

Parameters are arguments that are typically defined as variables within function definitions. They can be assigned values when you run the method, passing the arguments into the function.

***How to use function from a different file other than our currently working file?***

There isn't any need to add file.py while importing. Just write **from file (name of the file that define that function) import function (name of that function)**, and then call the function using **function(a, b)**. The reason why this may not work, is because file is one of Python's core modules, so I suggest you change the name of your file.

Note that if you're trying to import functions from a.py to a file called b.py, you will need to make sure that a.py and b.py are in the same directory.

**Part 2: Turtle exercises**

turtle\_exercises.py

**Part 3: Serious Exercises**

1. Writing codes

items = ['T-Shirt', 'Sweater']

print('Our items: ', *end*='')

print(\*items, *sep*=", ")

print()

items.append('Jeans')

print('Our items: ', *end*='')

print(\*items, *sep*=", ")

print()

items[1]='Skirts'

print('Our items: ', *end*='')

print(\*items, *sep*=", ")

print()

del items[2]

print('Our items: ', *end*='')

print(\*items, *sep*=", ")

1. Sheep.py