# Tutorial III - Functions

Programming with Python

# Introduction

Just like in the previous tutorial, you will likely find solutions to most exercises online. However, I still strongly encourage you to work on these exercises independently without searching for answers. Understanding someone else's solution is very different from developing your own.

Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities. If you encounter difficulties, review the lecture materials, experiment with different approaches, and don't hesitate to ask for clarification during class discussions. This was the last time I repeat this, I promise!

# Small Functions for various tasks

In this exercise, you will practice writing small functions for various tasks. You will also practice using the `return` statement to return a value from a function and the `global` keyword to modify a global variable.

```
# a) TODO: Write a function that takes two numbers as input and returns their squared sum.
# Your code here

# b) TODO: Implement a function that uses a global variable to keep track of how many times it has been
# Call the functions 10 times and print the result to the console.
# Your code here

# c) TODO: Create a function called password_strength that takes a password as input.
# It shoudl return "Weak", "Medium", or "Strong" based on the passwords length with the following crite
# - "Weak" if the password is less than 8 characters long
# - "Medium" if the password is between 8 and 15 characters long
# - "Strong" if the password is longer than 15 characters long
# Your code here

# d)TODO: Implement a function called `secret_number_game` that does the following:
# - Use a variable `secret_number` set to 42
# - The function should take one parameter `guess`
# - If the guess is correct, it should print "Correct!" and increment a global counter `correct_guesses
# - If the guess is incorrect, it should print "Wrong!" and increment a global counter `wrong_guesses`
# - The function should then return the total number of guesses made (correct + wrong)
# Your code here

# e) TODO: Write a recursive function (a function that calls itself) to calculate the sum of digits of a
# Your code here

# f) TODO: Implement a function is_palindrome that checks if a given string is a palindrome (reads the s
# Your code here
```

# Different classes for different tasks

In this exercise, you will practice creating different classes for different tasks. You will create a class for a bank account, a class for a car, and a class for a computer.

```
# a) # TODO: Create a class called 'Book' with the following specifications:
# - It should have attributes for 'title', 'author', and 'pages'
# - Include a method called 'display_info' that prints all the book's information
# - Add a method 'is_long' that returns True if the book has more than 400 pages, False otherwise
# Your code here

# b) TODO: Create your favorite book as an object to the class you just created. Check if it is a long
# Your code here

# c) TODO: Create a class called 'Student' with the following specifications:
# - It should have attributes for 'name', 'age', and 'current_grade'
# - Add a method 'is_excellent' that returns True if the student's grade is lower than 2.0
# - Add a method 'student_grade' that returns the current grade with the following printed statement:
# - If the grade is lower than 2.0: "The current grade of the student is: <grade>. This is a fantastic
# - If the grade is higher than 2.0 but lower than 4.0: "The current grade of the student is: <grade>.
# - If the grade is higher than 4.0: "The current grade of the student is: <grade>. This is a not so fa
# Your code here

# d) TODO: Create your yourself as an object to the class you just created. Check if you are excellent
# Your code here
```

# Rebuild the number guessing game

Remember the number guessing game from the previous tutorial? Well, it's time to rebuild it. This time, we will use functions to organize the code and make it more reusable. You will learn how to use functions to modularize your code and make it more readable.

```
# Start by looking at your code from the previous tutorial and think about the different steps that you
# a) TODO: Try to modularize each step of the game into a separate function.
# - Name the functions appropriately, to make clear what they do.
# b) TODO: Enhance the game by adding difficulty levels as described in the bonus task from the previous
# - The difference here is that the difficulty level should be given as a parameter to a function.
# - If no difficulty level is given, i.e. the game master just enters an empty input, the function shoul
# Your code here
```