Tutorial III - Building Reusable 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.

def squared_sum(a, b):
    pass # Your code here

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

functions_called = 0
# Your code here

# c) TODO: Create a function called password_strength that takes a password as input.
# It should return "weak", "medium", or "strong" based on the passwords length with the
 ↪  following criteria:
# - Return "weak" if the password is less than 8 characters long
# - Return "medium" if the password is between 8 and 15 characters long
# - Return "strong" if the password is longer than 15 characters long
# - The function should then be called as illustrated below.
# Your code here

password = input("Enter a password: ")
strength = password_strength(password)
print(f"The strength of the password is {strength}.")

# 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 be used in the while loop below to guess the secret number.
# Your code here

while True:
    guess = int(input("Enter a guess: "))
    secret_number_game(guess)
    if correct_guesses == 1:
```

```
        break

# e) TODO: Write a recursive function (a function that calls itself) to calculate the sum
 ↪  of digits of a positive integer.
# E.g. 1234 -> 1 + 2 + 3 + 4 = 10
# Hint: You can use a for loop to iterate over the characters in a string and convert
 ↪  them to integers.
# Your code here

# f) TODO: Implement a function is_palindrome that checks if a given string is a
 ↪  palindrome (reads the same forwards and backwards).
# Hint: Remember how we can use slicing to reverse a string.
# 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: Extend the class called 'Books' with the following specifications:
# - It should have attributes for 'title', 'author', and 'pages'
# - Use the `__init__` method to initialize the attributes
# - Include a method called 'display_info' that prints all the book's information
# - Add a method 'is_short' that returns True if the book has less than 100 pages, False
 ↪  otherwise

class Books:
    def __init__(self, title_name):
        self.title = title_name

    def display_info(self):
        print(f"Title: {self.title}")

    def is_long(self):
        return self.pages > 400

# b) TODO: Create your favorite book as an object to the class you just created. Check if
 ↪  it is a long book.
# 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 grade."
# - If the grade is higher than 2.0 but lower than 4.0: "The current grade of the student
 ↪  is: <grade>. This is still a fantastic grade.
# - If the grade is higher than 4.0: "The current grade of the student is: <grade>. This
 ↪  is a not so fantastic grade..."
# Your code here

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

That’s it!

You can find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them in next week’s tutorial. To access the solutions, click on the Github button on the lower right and search for the folder with today’s lecture and tutorial. Alternatively, you can ask ChatGPT or Claude to explain them to you. Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.