

Tutorial II - Control Structures

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. Don't worry, I won't repeat this section again and again.

Decoding secret messages with methods

In this exercise, we'll practice string manipulation and slicing. You'll work with a secret message encoded using various string operations and learn how to apply some new methods like `title()`, `replace()` and `count()`. By following a series of steps, you'll gradually decode the message. Let's begin with the encoded message and work through each decoding step:

```
# Decode a secret message by following a series of instructions.
# Each instruction requires you to use different operations and methods.
# The encoded message is:
secret_message = ".tnega terces a sa slliks ym sevorpmi erutcel oN"

# a) TODO: Reverse the string
# Hint: You can use slicing to reverse a string
# Your code here

# b) TODO: Remove the period at the end
# Your code here

# c) TODO: Replace 'No' with 'This'
# Your code here

# d) TODO: Convert the string to title case
# -> Capitalize the first letter of each word
# Your code here

# e) TODO: Add an exclamation mark at the end of the sentence
# Your code here

# f) TODO: Count how many times the letter 's' appears in the decoded message
# Your code here
```

Tip

Use the `help()` function to get more information about a method. For example, typing `help(str.replace)` in the shell will show the documentation for the `replace()` method. To exit the documentation, press `q`.

Classifying temperatures

In this exercise, we'll practice using conditional statements to classify temperatures into different categories. We'll create a program that takes a temperature input from the user and provides a classification based on the temperature range. This exercise will help you understand how to use if-elif-else statements, handle user input, and implement a simple loop for program repetition.

```
# Use if-elif-else statements to classify the temperature
# Below 0: "Freezing"
# 0-10: "Cold"
# 11-20: "Cool"
# 21-30: "Warm"
# Above 30: "Hot"

# a) TODO: Formulate a small program that asks the user for a temperature and then prints the corresponding classification
# Your code here

# b) TODO: Test your function with -5, 15 and 35 as temperature
# Your code here

# c) TODO: Add a feature that asks the user if they want to continue and then repeats the program if they answer yes
# Your code here
```

Number guessing game

In this exercise, we'll create an interactive number guessing game for two players. This game will help you practice using loops, conditional statements, and user input handling. You'll also learn how to implement a simple game logic and manage player turns. This exercise will reinforce your understanding of control structures and basic development concepts in Python.

```
# Create a number guessing game with 2 players. The first player is the game master and the second player is the player.  
  
# a) TODO: Your task list in more detail:  
# - Ask for the names of the game master and the player  
# - Implement the main game loop using a while loop  
# - Ask the game master for the secret number from 1-20  
# - Ask the player for a guess  
# - Compare the guess to the secret number  
# - Provide feedback (too high, too low, or correct)  
# - Keep track of the number of guesses  
# - End the game when the correct number is guessed  
# - Print a congratulatory message with the number of guesses taken  
# Your code here  
  
# b) TODO: Add a feature to allow both players to play again, this time switching the roles with the game master.  
# Your code here  
  
# c) TODO: Play the game with someone sitting next to you.
```

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.