Building a Multiple-Choice Quiz Game

Computer Science I

Learning Objectives

Today, after this lesson you will be able to:

1. Work as a software development pair to build a quiz game
2. Use conditions to check answers
3. Track scores using variables
4. Explain your code using Computer Science terminology

Code Recall — What you remember about this program?

```
attempts = 0
password = input("Enter password: ")

if password == "python":
    print("Access granted")
else:
    print("Access denied")
    attempts = attempts + 1

print("Failed attempts:", attempts)
```

Today's task

Description:

- Create a simple multiple-choice quiz game using Python. The program should ask the user a series of questions and check their answers. After all the questions have been answered, the program should display the final score.

Requirements:

1. The quiz should consist of at least 3 questions.
2. Each question should have multiple choices, and the user should input their answer.
3. The program should compare the user's answer with the correct answer using conditional branching (if statements).
4. Keep track of the user's score and display it at the end of the quiz.
5. You can assume the user will enter only lowercase letters (but for a challenge, have the program function correctly with uppercase letters as well!)
6. Use meaningful prompts for questions and provide feedback on whether the answer is correct or incorrect.

Let’s try together

Let’s try together

```
score = 0

print("Question 1: Which planet is closest to the Sun?")
print("a) Earth")
print("b) Mercury")
print("c) Venus")

answer = input("> ")

if answer == "b":
    print("Correct!")
    score = score + 1
else:
    print("Incorrect.")

print("Score:", score)
```

Today's task

Description:

- **Working in pair** create a simple multiple-choice quiz game using Python. The program should ask the user a series of questions and check their answers. After all the questions have been answered, the program should display the final score.

Requirements:

1. The quiz should consist of at least 3 questions.
2. Each question should have multiple choices, and the user should input their answer.
3. The program should compare the user's answer with the correct answer using conditional branching (if statements).
4. Keep track of the user's score and display it at the end of the quiz.
5. You can assume the user will enter only lowercase letters.
6. Use meaningful prompts for questions and provide feedback on whether the answer is correct or incorrect.

Challenge Tasks ⭐

If your pair finishes the core task, try one or more of these extensions:

- **Accept both uppercase and lowercase answers *(Hint: use .lower())***
- **Add more than 3 questions**
- **Create a bonus question worth 2 points**
- **Show a final message depending on the score**

  (e.g. "Excellent!", "Good effort!", "Keep practicing!")
- **Reduce repeated code using: a function or a loop**
- **Add an option to replay the quiz at the end**

Exit Ticket

Complete

Part 1 — Self-Assessment

and

Part 2 — Explaining Your Code

Choose ONE option:

- 🔶 A — Fill in the blanks
  🔷 B — Explain the code
  🚀 C — Improve the code
- **Use correct Computer Science vocabulary.**
- **Work independently.**