
PYTHON CONTROL FLOW (IF ELSE, ELIF, NESTED IF ELSE)
DEADLINE: 11TH FEB 2024
ASSIGNMENT NO # 11

Conceptual Explanation:
Control flow in Python refers to the order in which statements are executed based on certain conditions. Python provides several control flow statements such as if, else, elif (short for else if), and nested if-else, which allow programmers to control the flow of execution in their code.

- **if statement:** The if statement is used to execute a block of code only if a specified condition is true. If the condition is false, the block of code associated with the if statement is skipped.
- **else statement:** The else statement is used in conjunction with the if statement to execute a block of code if the condition specified in the if statement is false.
- **elif statement:** The elif statement allows you to check multiple conditions after the initial if statement. If the condition specified in the elif statement is true, the corresponding block of code is executed, and the rest of the elif and else blocks are skipped.
- **Nested if-else:** Nested if-else statements are if-else statements that are nested within each other. This allows for more complex conditional logic, where one condition depends on the outcome of another condition.

Logical Breakdowns:

1. **if statement:**
    - Check if a condition is true.
    - If true, execute a block of code associated with the if statement.
    - If false, skip the block of code and move to the next statement.
2. **else statement:**
    - Used in conjunction with the if statement.
    - Executes a block of code if the condition specified in the if statement is false.
3. **elif statement:**
    - Allows you to check additional conditions after the initial if statement.
    - If the condition specified in the elif statement is true, execute the corresponding block of code.
    - If false, move to the next elif statement or else block.
4. **Nested if-else:**
    - Used when there is a need for multiple levels of conditional logic.
    - Inner if-else statements are contained within outer if-else statements.
    - Each level of if-else is evaluated sequentially, and the corresponding block of code is executed based on the condition.

Understanding these concepts is crucial for writing efficient and structured code in Python. It enables you to implement conditional logic effectively, leading to more readable and maintainable code.

SOLVE THESE QUESTIONS

Conceptual Questions:

1. What is the purpose of using control flow statements like if, else, and elif in Python?
2. How does Python determine which block of code to execute in an if-else statement?
3. Explain the difference between the if-elif-else and nested if-else structures.
4. How can you use logical operators (and, or, not) with if-else statements in Python?
5. Describe scenarios where nested if-else statements are preferred over if-elif-else structures.
6. How does Python handle multiple conditions in an if-elif-else ladder?
7. Why is it important to indent properly when using control flow statements in Python?

Logical Problems:

1. Write a Python program to check if a given number is positive, negative, or zero.
2. Create a program that asks the user to enter their age and prints out whether they are a child, teenager, adult, or senior citizen.
3. Develop a program that prompts the user to enter two numbers and prints out the larger of the two.
4. Write a Python script to determine whether a given year is a leap year or not.
5. Implement a program that takes a user's input of three numbers and prints out the maximum and minimum among them.
6. Create a program that asks the user to enter their exam score and prints out their grade based on the following criteria: A (90-100), B (80-89), C (70-79), D (60-69), F (below 60).

Simple Practical Problems:

1. Design a Python program that calculates the total cost of items purchased by a customer based on the provided unit price and quantity, applying a discount of 10% if the total cost exceeds $1000.
2. Develop a program that prompts the user to enter their current temperature in Celsius and prints out whether they should wear a jacket (if temperature is below 20°C) or not.
3. Write a Python script that takes a user's input of three sides of a triangle and determines whether it is equilateral, isosceles, or scalene.
4. Create a program that simulates a simple ATM machine. It should ask the user for their PIN, verify it, and then allow them to withdraw money if the balance is sufficient.
5. Develop a Python script that takes a user's input of a month (as a number) and prints out the number of days in that month.
6. Implement a program that takes a user's input of a year and month and prints out the number of days in that month, considering leap years.

Healthy Lifestyle:

1. Calorie Counter: Write a program that asks the user for their age, weight, and activity level, then calculates their daily calorie goal based on recommended guidelines. Use if-else statements to adjust the goal based on the user's activity level.
2. Sleep Tracker: Create a program that asks the user for their bedtime and wake-up time, then calculates their total sleep duration. Use if statements to determine if they met the recommended sleep amount and provide feedback accordingly.
3. Hydration Helper: Design a program that prompts the user for their weight and desired level of hydration (e.g., light, moderate, intense exercise). Use nested if-else statements to suggest the amount of water they should drink throughout the day.

Time Management:

1. To-Do List: Develop a program that allows the user to add tasks to a to-do list. Use if statements to categorize tasks as urgent, important, or less important based on their due date and priority.
2. Pomodoro Timer: Create a program that implements the Pomodoro Technique (25 minutes work, 5 minutes break). Use a loop and if statements to track time intervals and notify the user when to switch between work and breaks.
3. Meeting Scheduler: Design a program that helps users find a common meeting time among a group. Use if-else statements to check for available time slots in each user's calendar and suggest suitable meeting times.

Bonus Challenges:

1. Combine multiple concepts from the above examples to create a more complex program related to healthy habits or time management.
2. Think own real-life examples and create programs to address them using control flow statements.