Homework Assignment: Control Structures – Conditionals and Logical Operators

2.1 Conditional Statements

If Statement

Explanation:

- The `if` statement is used to evaluate a condition. If the condition is `True`, the code block inside the `if` statement is executed.
- It's essential for making decisions in your programs based on specific criteria.

Elif and Else Statements

Explanation:

- `elif` (short for "else if") allows you to check multiple conditions sequentially.
- `else` executes a block of code if none of the previous conditions are `True`.
- Together, `if`, `elif`, and `else` provide a way to handle multiple scenarios and outcomes in your code.

Comparison Operators

Explanation:

- Comparison operators are used to compare two values.
- They return a boolean value (`True` or `False`) based on the comparison.
- Common comparison operators include:

- `==` : Equal to
- `!=` : Not equal to
- `>` : Greater than
- `<` : Less than
- `>=` : Greater than or equal to
- `<=` : Less than or equal to

2.2 Logical Operators

Explanation:

- Logical operators allow you to combine multiple conditions in your conditional statements.
- They help in creating more complex and precise conditions.
- The primary logical operators in Python are:

AND (`and`)

- **Purpose:** Both conditions must be `True` for the entire expression to be `True` .

OR (`or`)

- **Purpose:** At least one of the conditions must be `True` for the entire expression to be `True` .

NOT (`not`)

- **Purpose:** Inverts the boolean value of the condition. If the condition is `True`, `not` makes it `False`, and vice versa.

Exercises

Complete each exercise by writing a separate Python file (`exercise1.py`, `exercise2.py`, etc.). Ensure your code runs without errors and produces the expected output.

Exercise 1: Number Sign Checker

Task:

Write a program that asks the user to enter a number and prints whether the number is positive, negative, or zero.

Example Output:

Enter a number: 10

The number is positive.

Enter a number: -5

The number is negative.

Enter a number: 0

The number is zero.

Exercise 2: Divisibility Tester

Task:

Create a program that asks the user to enter a number and determines if it is divisible by both 4 and 6. Use the modulus operator (`%`) to perform the checks.

Example Output:

Enter a number: 24

The number is divisible by both 4 and 6.

Enter a number: 18

The number is not divisible by both 4 and 6.

Exercise 3: String Length Checker

Task:

Write a program that asks the user to enter a word and checks if the length of the word is greater than 5 characters using the `len()` function. Print an appropriate message based on the result.

Example Output:

Enter a word: Python

The word has more than 5 characters.

Enter a word: Code

The word has 5 or fewer characters.

Exercise 4: Multiple Condition Validator

Task:

Develop a program that asks the user to enter their age and whether they have a driver's license (`yes` or `no`). The program should print "Eligible to rent a car." only if the user is at least 21 years old **and** has a driver's license. Use logical operators to combine the conditions.

Example Output:

Enter your age: 25

Do you have a driver's license? yes

Eligible to rent a car.

Enter your age: 19

Do you have a driver's license? yes

Not eligible to rent a car.

Enter your age: 22

Do you have a driver's license? no

Not eligible to rent a car.

Submission Guidelines

1. **Complete All Exercises:**

Ensure each exercise is completed and saved in a separate Python file (`exercise1.py` , `exercise2.py` , etc.).

2. **Code Quality:**

- Use meaningful variable names.
- Include comments to explain your code where necessary.

- Follow proper indentation and coding standards.

3. **Testing:**

Run each script to verify that it works correctly and produces the expected output.

Good luck, and happy coding!