Python Control Structures Homework: Theory and Exercises

This assignment focuses on **control structures** in Python: `if-else` statements, the modulus operator (`%`), `for` loops, and the `range()` function. Below is a detailed explanation of each concept, followed by exercises to reinforce your understanding.

Theory Overview

If-Else Statements

In Python, **if-else statements** allow you to control the flow of a program based on conditions. Conditions are expressions that evaluate to `True` or `False`. Using if-else statements, you can execute different code blocks based on whether a condition is met.

Structure of If-Else Statements:

```
if condition:
```

Code to execute if condition is True

```
elif another_condition:
```

Code to execute if the first condition is False and this condition is True

```
else:
```

Code to execute if all previous conditions are False

• if checks the initial condition. If it's True, the code block under if is executed, and the program skips the other conditions.

• elif (short for "else if") allows you to add additional conditions. If the if condition is False but an elif condition is True, the code under elif runs.

• else provides a fallback; it runs only if all previous conditions are False.

Example of an If-Else Statement:

```
number = 10



if number > 0:

print("Positive")

elif number < 0:

print("Negative")

else:

print("Zero")

# Output: Positive
```

In this example, number is greater than 0, so the output is "Positive."

The Modulus Operator (%)

The **modulus operator** (%) returns the remainder after division of one number by another. This operator is essential for checking conditions like whether a number is even or odd or if it's a multiple of another number.

• When number % 2 == 0, it indicates that number is even because there's no remainder after dividing by 2.

• When number % 2 != 0, it indicates that number is odd because there's a remainder of 1 after dividing by 2.

Example of Using the Modulus Operator:

```
number = 8



if number % 2 == 0:

print("Even")

else:

print("Odd")

# Output: Even
```

In this case, 8 % 2 is 0, so the output is “Even.”

For Loops

A **for loop** is used for iterating over a sequence (like a list, tuple, string, or range of numbers). In this context, we’ll primarily use for loops with the range() function to repeat actions a specified number of times.

Basic Structure of a For Loop:

```
for variable in sequence:
# Code to execute for each item in the sequence
```

• variable takes on the value of each item in sequence in each loop iteration.

• The code inside the loop runs once for each item in the sequence.

For example, using a for loop with range(5) will cause the loop to run five times, with variable taking on values from 0 to 4.

Example of a For Loop:

```
for i in range(5):

print(i)

# Output:

# 0

# 1

# 2

# 3

# 4
```

In this case, i starts at 0 and goes up to 4 (one less than 5).

The Range Function

The range() function is often used with for loops to specify the range of numbers to iterate over. range() can take up to three parameters: start, stop, and step.

Basic Forms of range():

• range(stop) – Starts from 0 and goes up to (but doesn’t include) stop.

• range(start, stop) – Starts from start and goes up to (but doesn’t include) stop.

• range(start, stop, step) – Starts from start, goes up to (but doesn’t include) stop, and changes by step each time.

Example of Using range() with Different Parameters:

```
for i in range(2, 10, 2):

print(i)

# Output:

# 2

# 4

# 6

# 8
```

In this example, i starts at 2 and increases by 2 each time, stopping before it reaches 10.

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 11: Sum of Even Numbers

Theory:

In this exercise, we'll practice using a for loop and conditionals to add only even numbers within a specified range. The modulus operator (%) is essential here, as it allows us to check if a number is even.

Example of Theory in Code:

```
# Check if a number is even
number = 4



if number % 2 == 0:

print(number, "is even")  # Output: 4 is even

else:

print(number, "is odd")
```

Assignment:

Write a program that asks the user for a number and calculates the sum of all even numbers from 1 up to that number.

Tips and Tricks for Assignment:

• Initialize a variable like total_sum to keep track of the running total.

• Use range(1, user_number + 1) to include the user's specified number in the loop.

• Use an if statement to check if each number is even, and if it is, add it to total_sum.

Exercise 12: Count Down from a Number

Theory:

In this exercise, we need to count down from a specified number to 1. The range() function in Python can be used to count backward by specifying a negative step.

Example of Theory in Code:

```
# Count down from 5 to 1

for num in range(5, 0, -1):

print(num)

# Output:

# 5

# 4

# 3

# 2

# 1
```

Assignment:

Write a program that asks the user to enter a positive number and then counts down to 1, displaying each number.

Tips and Tricks for Assignment:

• Use range(start, stop, step) to set up the countdown. Set start to the user's number, stop to 0 (as it's exclusive and won't include 0), and step to -1.

• Test your range to verify it counts down as expected.

Exercise 13: Multiples of Three

Theory:

We'll use conditionals to find numbers that are multiples of 3 within a specific range. A number is a multiple of 3 if dividing it by 3 leaves no remainder (number % 3 == 0).

Example of Theory in Code:

```
# Check if a number is a multiple of 3

number = 9



if number % 3 == 0:

print(number, "is a multiple of 3") # Output: 9 is a multiple of 3

else:

print(number, "is not a multiple of 3")
```

Assignment:

Write a program that asks the user to enter a number and prints all multiples of 3 from 1 up to that number.

Tips and Tricks for Assignment:

• Use range(1, user_number + 1) to include the user's specified number in the range.

• Only print numbers that meet the condition of being a multiple of 3.

Exercise 14: Sum of Odd Numbers

Theory:

This exercise involves adding up all odd numbers in a specified range. An odd number has a remainder of 1 when divided by 2 (number % 2 != 0).

Example of Theory in Code:

```
# Check if a number is odd

number = 5



if number % 2 != 0:

print(number, "is odd")  # Output: 5 is odd

else:

print(number, "is even")
```

Assignment:

Write a program that asks the user for a number and calculates the sum of all odd numbers from 1 up to that number.

Tips and Tricks for Assignment:

• Initialize a variable like total_sum to keep track of the sum.

• Include a conditional in the loop to check if each number is odd before adding it to the sum.

Exercise 15: Positive, Negative, or Zero Counter

Theory:

This exercise combines loops and conditional statements. You'll ask the user for a series of inputs, and for each input, check if the number is positive, negative, or zero.

Example of Theory in Code:

```
# Classify a single number

number = -3



if number > 0:

print("Positive")

elif number < 0:

print("Negative")

else:

print("Zero")

# Output: Negative
```

Assignment:

Write a program that asks the user to enter 5 numbers, one at a time. Count how many of those numbers are positive, negative, or zero, and print out the results.

Tips and Tricks for Assignment:

• Create counters for positive, negative, and zero values.

• Use if, elif, and else statements to classify each number and update the appropriate counter.

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!