Theory Overview

---

range() Function

The `range()` function is useful for generating sequences of numbers. It can take up to three parameters:

1. `range(stop)` - Starts at 0 and goes up to (but doesn't include) stop.
2. `range(start, stop)` - Starts at start and goes up to (but doesn't include) stop.
3. `range(start, stop, step)` - Starts at start, stops before stop, and increments (or decrements) by step.

If-Else Statements

If-else statements help execute code based on conditions, making decisions within your program. Use `if`, `elif`, and `else` to check multiple conditions in sequence.

Modulus Operator (%)

The modulus operator `%` gives the remainder of division. It's useful for checking even/odd numbers and identifying multiples:

- `number % 2 == 0`: True if the number is even.
- `number % 3 == 0`: True if the number is a multiple of 3.

Exponentiation Operator (**)

The `**` operator raises a number to the power of another. For instance, `2 ** 3` equals 8 (2 raised to the power of 3).

Exercises

Exercise 1: Sum of Numbers in a Range

Theory:

Use `range(start, stop)` to calculate the sum of all numbers between two values. This requires a “running total” to accumulate each value in the range.

Assignment:

Write a program that asks the user for two numbers, start and end. Calculate and print the sum of all numbers from start to end (inclusive).

Example Input-Output:

```
Enter start value: 3
Enter end value: 7
Output: Sum: 25
```

Tips and Tricks:

- Use `range(start, end + 1)` to include the end value.
- Initialize a variable like `total_sum` to keep track of the sum.

Exercise 2: Display Every Nth Number

Theory:

This exercise uses `range(start, stop, step)` to display every `step` -th number, allowing you to skip numbers by a specific interval.

Assignment:

Write a program that asks the user for three values: start, stop, and step. Print every `step` -th number from start to stop.

Example Input-Output:

```
Enter start value: 2
Enter stop value: 15
Enter step value: 3
Output:
```

```
2
5
8
11
14
```

Tips and Tricks:

- Use `range(start, stop + 1, step)` to include the stop value.
- Ensure `step` is a positive value for counting up.

Exercise 3: Count Down from a Given Number

Theory:

This exercise uses `range(start, stop, step)` with a negative step to count down. Negative step values allow you to count backward.

Assignment:

Ask the user for a positive integer, then display a countdown from that number to 1.

Example Input-Output:

```
Enter a positive integer: 5
Output:
5
4
3
2
1
```

Tips and Tricks:

- Set `step` to -1 to count down.
- Use `range(start, 0, -1)` to stop at 1.

Exercise 4: Debugging Task – Finding Multiples

Theory:

This exercise helps identify multiples of a number using if statements with the modulus operator `%`.

Assignment:

The code below should ask for a number and print all multiples of 5 from 1 up to that number. Identify and fix the errors.

Code to Debug:

```
user_number = int(input("Enter a number: "))
for i in range(1, user_number + 1):
    if i % 5 = 0:
        print(i)
```

Example Input-Output:

```
Enter a number: 20
Output:
5
10
15
20
```

Tips for Debugging:

- Ensure `if` is indented properly and uses `==` for comparison.
- Test the corrected code with different values for `user_number`.

Exercise 5: Sum of Squares

Theory:

This exercise involves calculating the squares of numbers in a range. Squares can be calculated with the exponentiation operator (`**`), e.g., `number ** 2`.

Assignment:

Write a program that asks the user for start and end values and calculates the sum of squares for all numbers between start and end.

Example Input-Output:

```
Enter start value: 1
Enter end value: 4
Output: Sum of squares: 30
```

Tips and Tricks:

- Use `range(start, end + 1)` to include the end value.
- Square each number before adding it to the total.

Exercise 6: Debugging Task – Sum of Even Numbers

Theory:
This exercise involves fixing code to calculate the sum of even numbers in a range. Even numbers have no remainder when divided by 2 (`number % 2 == 0`).

Assignment:
Identify and fix errors in the following code, which should print the sum of all even numbers between start and end.

Code to Debug:

```
start = int(input("Enter start value: "))
end = int(input("Enter end value: "))
sum = 0
for i in range(start, end):
    if i % 2 = 0
        sum + i
print("Sum of even numbers:", sum)
```

Example Input-Output:

```
Enter start value: 1
Enter end value: 10
Output: Sum of even numbers: 30
```

Tips for Debugging:

- Use `==` in `if` statements and `+=` for adding to `sum`.
- Ensure the end value is included by using `range(start, end + 1)`.

Exercise 7: Print All Odd Numbers in Reverse Order

Theory:

This exercise uses `range(start, stop, step)` with a negative step to print numbers in reverse order, focusing only on odd numbers.

Assignment:

Ask the user for a positive odd integer as the starting point. Print all odd numbers from that number down to 1.

Example Input-Output:

```
Enter a positive odd integer: 9
Output:
9
7
5
3
1
```

Tips and Tricks:

- Confirm the starting number is odd; if not, ask for another input.
- Use `range(start, 0, -2)` to include only odd numbers in reverse.

Exercise 8: Print Multiplication Table Using Range

Theory:

This exercise demonstrates the use of loops with `range()` to display multiplication tables. Multiply a given number by each value in a range.

Assignment:

Ask the user for a number, then print its multiplication table up to 10.

Example Input-Output:

```
Enter a number: 3
Output:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
...
3 x 10 = 30
```

Tips and Tricks:

- Use `range(1, 11)` to iterate from 1 to 10.
- For each loop iteration, multiply the user's number by the loop variable and display the result.

Exercise 9: Debugging Task – Product of Numbers

Theory:
This exercise involves calculating the product of numbers within a range by multiplying each number in sequence. Debugging helps identify logical errors in multiplication.

Assignment:
The code below should calculate the product of numbers from start to end, but it contains errors. Identify and correct them.

Code to Debug:

```python
start = int(input("Enter start value: "))
end = int(input("Enter end value: "))
product = 1
for i in range(start, end):
    product * i
print("Product:", product)
```

Example Input-Output:

```
Enter start value: 1
Enter end value: 4
Output: Product: 24
```

Tips for Debugging:

- Use `*=` to accumulate the product in `product`.
- Make sure `range(start, end + 1)` includes the end value.
- Test with simple input values to confirm accuracy.