For Loops

Key Concepts:

- For Loops with Data Structures
- For Loops with `range`

Iteration

While Loop:

- The condition determines the number of times the loop is executed (conditional loop).

For Loop:

- If the number of iterations is known in advance, use the `for` loop (unconditional loop).

For Loop with Data Structures

Purpose:

- To iterate over data structures such as lists of numbers, objects, or characters in a string.

Examples:

```
# Iterating over a list of numbers
for e in [1, 4, 5, 0, 9, 1]:
    print(e)

# Iterating over a list of characters
for e in ["a", "e", "i", "o", "u", "y"]:
    print(e)

# Iterating over a string
for e in "python":
    print(e)
```

In these examples, the variable `e` takes on each value from the list or string in each iteration.

For Loop with `range`

Syntax:

```
for i in range(start, stop, step):
    # instructions
```

- `range(start, stop, step)` generates a sequence of integers from `start` to `stop-1` with a step of `step`.

Examples:

```
# Basic range example
for i in range(1, 6):
    print(i, end=",")  # Output: 1,2,3,4,5,

# Range with different parameters
for i in range(4):
    print(i)  # Output: 0, 1, 2, 3

for j in range(2, 5):
    print(j)  # Output: 2, 3, 4

for k in range(3, 12, 3):
    print(k)  # Output: 3, 6, 9

for l in range(12, 3, -2):
    print(l)  # Output: 12, 10, 8, 6, 4
```

Important Notes

- If there is an inconsistency in the range parameters, the loop is ignored, and the program continues with the next instructions.

Example:

```
# Loop with inconsistent range
for k in range(200, 210, -2):
    print(k)  # This loop is ignored
```

```
for k in range(110, 100, -2):
    print(k)  # Output: 110, 108, 106, 104, 102
```

- Regardless of what happens in the loop body, the loop variable takes the next value from the range or list at each new step.

Example:

```
for i in range(1, 5):
    print(i)
    i = i * 2  # This modification does not affect the loop's
progression
```

Exercises

For the following cases, indicate the successive values printed on the console.

Exercise 1:

```
for i in range(4):
    print(i)  # Output: 0, 1, 2, 3
```

Exercise 2:

```
for j in range(2, 5):
    print(j)  # Output: 2, 3, 4
```

Exercise 3:

```
for k in range(3, 12, 3):
    print(k)  # Output: 3, 6, 9
```

Exercise 4:

```
for l in range(12, 3):
    print(l)  # No output, as the loop range is inconsistent
```

Exercise 5:

```
for m in range(12, 3, -2):
    print(m)  # Output: 12, 10, 8, 6, 4
```

Exercise 6:

Calculate the sum of positive numbers in a list.

```
def sum_of_positives(lst):
    s = 0
    for e in lst:
        if e > 0:
            s += e
    return s

# Main program
my_list = [2, -4, 6, 0, -5, 1]
for e in my_list:
    print(e + 1)

total = sum_of_positives(my_list)
print(total)  # Output: 9
```