Loops

1. For Loops

For loops iterate over sequences like lists, strings, or ranges.

1.1 Iterating Over Lists

```
fruits = ["apple", "banana", "cherry"]

print("Fruits:")
for fruit in fruits:
    print(f"  - {fruit}")
```

```
Fruits:
  - apple
  - banana
  - cherry
```

1.2 Using Range

The range() function generates a sequence of numbers.

```
# Range(n) generates 0 to n-1
print("Range(5):")
for i in range(5):
    print(i, end=" ")
print()

# Range(start, end)
print("Range(2, 6):")
for i in range(2, 6):
    print(i, end=" ")
print()

# Range(start, end, step)
print("Range(0, 10, 2):")
for i in range(0, 10, 2):
    print(i, end=" ")
print()
```

```
Range(5):
0 1 2 3 4
Range(2, 6):
2 3 4 5
Range(0, 10, 2):
0 2 4 6 8
```

1.3 Enumerate

Enumerate provides both index and value when iterating.

```
fruits = ["apple", "banana", "cherry"]

print("With enumerate:")
for idx, fruit in enumerate(fruits):
    print(f"  {idx}: {fruit}")

# Start index at 1
print("\nStarting at 1:")
for idx, fruit in enumerate(fruits, start=1):
    print(f"  {idx}: {fruit}")
```

```
With enumerate:
  0: apple
  1: banana
  2: cherry

Starting at 1:
  1: apple
  2: banana
  3: cherry
```

2. While Loops

While loops continue executing as long as a condition is true.

```
count = 0
print("While loop:")
while count < 5:
    print(count, end=" ")
    count += 1
print()

# Be careful of infinite loops!
# while True:  # This would run forever
#     print("Forever!")
```

```
While loop:
0 1 2 3 4
```

3. Loop Control

Use break to exit a loop early, continue to skip to the next iteration.

```
print("Break and Continue:")
for i in range(10):
    if i == 3:
        continue  # Skip 3
    if i == 7:
        break     # Stop at 7
    print(i, end=" ")
print()
```

```
Break and Continue:
0 1 2 4 5 6
```

4. Nested Loops

You can place loops inside other loops for multidimensional iteration.

```
print("Multiplication table:")
for i in range(1, 4):
    for j in range(1, 4):
        print(f"{i}x{j}={i*j}", end="  ")
    print()  # New line after each row
```

```
Multiplication table:
1x1=1  1x2=2  1x3=3
2x1=2  2x2=4  2x3=6
3x1=3  3x2=6  3x3=9
```

5. Loop with Else

Python loops can have an else clause that executes when the loop completes normally.

```
# Else executes if loop completes without break
for i in range(5):
    print(i, end=" ")
else:
    print("\nLoop completed!")

# Else doesn't execute if break is used
for i in range(5):
    if i == 3:
        break
    print(i, end=" ")
else:
    print("\nThis won't print")
```

```
0 1 2 3 4
Loop completed!
0 1 2
```

Key Takeaways

• for loops for iterating over sequences
• range() for number sequences
• enumerate() for index and value
• while loops for conditional iteration
• break exits loop, continue skips to next iteration
• else clause executes when loop completes normally