**Control Structures in Python**

Control structures are fundamental in programming as they determine the flow of execution in a program. In Python, control structures include conditional statements and loops.

**1. Conditional Statements (if, elif, else)**

Conditional statements allow you to execute different blocks of code based on conditions.

**Syntax:**

if condition:

# Code block executed if condition is True

elif another\_condition:

# Code block executed if another\_condition is True

else:

# Code block executed if no condition is True

**Example 1: Basic if statement**

age = 18

if age >= 18:

print("You are eligible to vote.")

**Output:**

You are eligible to vote.

**Example 2: Using if and else**

number = 10

if number % 2 == 0:

print("Even number")

else:

print("Odd number")

**Output:**

Even number

**Example 3: Using if, elif, and else**

marks = 75

if marks >= 90:

print("Grade: A")

elif marks >= 75:

print("Grade: B")

elif marks >= 50:

print("Grade: C")

else:

print("Grade: F")

**Output:**

Grade: B

**Example 4: Nested if statements**

You can also nest if statements inside another if.

x = 10

if x > 5:

print("x is greater than 5")

if x < 20:

print("x is also less than 20")

**Output:**

x is greater than 5

x is also less than 20

**2. Loops in Python**

Loops allow us to execute a block of code multiple times until a condition is met. Python provides two types of loops:

1. **for loop** (used for iteration over sequences)
2. **while loop** (executes as long as a condition is True)

**2.1 for Loop**

The for loop in Python is used to iterate over a sequence such as a list, tuple, string, or range.

**Syntax:**

for variable in sequence:

# Code block to execute

**Example 1: Iterating over a list**

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

for fruit in fruits:

print(fruit)

**Output:**

apple

banana

cherry

**Example 2: Iterating over a string**

for letter in "Python":

print(letter)

**Output:**

P

y

t

h

o

n

**2.2 The range() Function in for Loops**

The range() function generates a sequence of numbers and is commonly used with for loops.

**Syntax:**

range(start, stop, step)

* **start**: Optional; default is 0. Specifies the starting number.
* **stop**: Required. Specifies the end of the range (not included).
* **step**: Optional; default is 1. Specifies the increment.

**Example 1: Using range() in a loop**

for i in range(5): # Equivalent to range(0, 5, 1)

print(i)

**Output:**

0

1

2

3

4

**Example 2: Using range(start, stop)**

for i in range(2, 7):

print(i)

**Output:**

2

3

4

5

6

**Example 3: Using range(start, stop, step)**

for i in range(1, 10, 2):

print(i)

**Output:**

1

3

5

7

9

**2.3 while Loop**

A while loop runs as long as a given condition is True.

**Syntax:**

while condition:

# Code block to execute

**Example 1: Basic while loop**

count = 1

while count <= 5:

print(count)

count += 1 # Increment count

**Output:**

1

2

3

4

5

**Example 2: Using while with user input**

password = ""

while password != "secret":

password = input("Enter password: ")

print("Access granted!")

**Output:** (User enters incorrect password)

Enter password: hello

Enter password: python

Enter password: secret

Access granted!

**Example 3: Infinite loop (use with caution!)**

while True:

print("This is an infinite loop") # Will run forever unless stopped

To stop an infinite loop, use **Ctrl + C** or include a break statement.

**2.4 The in Operator in Loops**

The in operator is used to check if an item exists in a sequence (like a list, string, or tuple).

**Example 1: Checking if an element exists in a list**

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

if "banana" in fruits:

print("Banana is in the list")

**Output:**

Banana is in the list

**Example 2: Using in with a loop**

for letter in "Python":

print(letter)

**Output:**

P

y

t

h

o

n

**3. break and continue Statements in Loops**

* **break**: Exits the loop immediately.
* **continue**: Skips the rest of the current iteration and moves to the next iteration.

**Example 1: Using break**

for num in range(10):

if num == 5:

break # Stops the loop when num is 5

print(num)

**Output:**

0

1

2

3

4

**Example 2: Using continue**

for num in range(5):

if num == 2:

continue # Skips printing 2

print(num)

**Output:**

0

1

3

4

**Summary**

| **Concept** | **Explanation** |
| --- | --- |
| if/elif/else | Used for decision-making based on conditions. |
| for loop | Iterates over a sequence (list, tuple, string, range). |
| while loop | Repeats a block of code while a condition is True. |
| range() | Generates sequences of numbers for loops. |
| in operator | Checks for membership in a sequence. |
| break | Exits a loop early. |
| continue | Skips the current iteration and moves to the next one. |