**Conditional Statements**

**Conditional Statements in Python**

Conditional statements allow your program to perform different actions depending on whether a condition is true or false.

**Types of Conditional** Statements

1. if Statement
2. if-else Statement
3. if-elif-else Statement
4. Nested if Statements
5. Ternary Conditional Operator

**1. if Statement**

It checks a condition, and if the condition is true, it executes a block of code.

**Syntax:**

if condition:

# Code to be executed if condition is true

**Ex:**

temperature = 30

if temperature > 25:

print(“High Temp”)

**2. if-else Statement**

This adds an alternative block of code to execute if the condition is false.

**Syntax:**

if condition:

# Code if condition is true

else:

# Code if condition is false

**Ex:**

age = 17

if age >= 18:

print(“eligible to vote”)

else:

print(“not eligible to vote”)

**3. if-elif-else Statement**

Checks multiple conditions in sequence. The first condition that is true gets executed, and the other are ignored.

**Syntax:**

if condition1:

# Code for condition1

elif condition2:

# Code for condition2

else:

# Code if none of the conditions are true

**Ex:**

marks = 85

if marks >= 90:

print(“Grade A”)

elif marks >= 75:

print(“Grade B”)

elif marks >= 50:

print(“Grade C”)

else:

print(“Fail”)

**4. Nested if Statements**

An if statement inside another if statement for more complex conditions.

**Syntax:**

if condition1:

if condition2:

# Code for both conditions true

**Ex:**

age = 20

income = 30000

if age >= 18:

if income >= 25000:

print(“You are eligible for a bank account”)

**5. Ternary Conditional Operator**

A easy way to write if-else in one line.

**Syntax:**

value\_if\_true if condition else value\_if\_false

**Ex:**

age = 16

status = "Adult" if age >= 18 else "Minor"

print(status) # Output: Minor

**Conditional Break and Continue:**

The break and continue statements in Python are used to control the flow of loops based on specific conditions. They allow us to make loops more flexible and efficient, which is helpful in real-world scenarios.

**Break:**

The break statement is used to immediately terminate a loop, regardless of the loop condition. When Python encounters a break, it exits the loop and moves to the next statement outside the loop.

**Syntax:**

if condition:

break

**Ex 1: Breaking a for Loop**

for number in range(1, 11):

if number == 5: # Condition to stop the loop

break

print(number)

# Output:

# 1

# 2

# 3

# 4

**Ex 2: Breaking a while Loop**

count = 0

while count < 10:

print(count)

if count == 5: # Exit when count reaches 5

break

count += 1

# Output:

# 0

# 1

# 2

# 3

# 4

# 5

**Continue:**

The continue statement is used to skip the rest of the code in the current iteration and proceed to the next iteration of the loop. It doesn't exit the loop but skips over the remaining code for the current iteration.

**Syntax:**

if condition:

continue

**Ex 1: Skipping Values in a for Loop**

for number in range(1, 11):

if number % 2 == 0: # Skip even numbers

continue

print(number)

# Output:

# 1

# 3

# 5

# 7

# 9

**Ex 2: Skipping Values in a while Loop**

count = 0

while count < 10:

count += 1

if count == 5: # Skip the number 5

continue

print(count)

# Output:

# 1

# 2

# 3

# 4

# 6

# 7

# 8

# 9

# 10