DAY - 02

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

Conditional statements allow you to execute a block of code based on a condition.

python

Copy

* if: Checks the condition. If it evaluates to True, the associated block of code is executed.
* elif: Stands for "else if." It's used to check additional conditions if the previous if conditions are false.
* else: Executes the block of code when all preceding conditions are false.

### **2. Loops (for, while)**

Loops are used to repeat a block of code multiple times.

#### **for loop:**

Used for iterating over a sequence (like a list, tuple, or string).

python

Copy

for i in range(5):

print(i)

* range(5) generates numbers from 0 to 4, and the loop will execute the print statement five times.

#### **while loop:**

Executes a block of code as long as the given condition is True.

python

Copy

count = 0

while count < 5:

print(count)

count += 1

* The loop will continue until count becomes 5.

### **3. Break and Continue**

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

python

Copy

for i in range(5):

if i == 3:

break

print(i)

Output:

Copy

0

1

2

The loop breaks when i == 3.

python

Copy

for i in range(5):

if i == 3:

continue

print(i)

Output:

Copy

0

1

2

4

The loop skips printing 3 and continues.

### **4. Pass**

The pass statement is a placeholder that does nothing. It can be used when a statement is syntactically required but you don't want to execute any code.

python

Copy

if condition:

pass # Nothing happens here

### **5. Try and Except (Exception Handling)**

Python provides exception handling tools to catch and handle errors without terminating the program.

python

Copy

try:

num = int(input("Enter a number: "))

except ValueError:

print("Invalid input! Not a number.")

* try: Executes the code within it.
* except: Catches and handles exceptions that occur during the execution of the try block.

### **6. Functions (Defining and Calling)**

Functions help control the flow by allowing you to define reusable blocks of code.

def greet(name):

print(f"Hello, {name}!")

greet("Alice")

### **7. Lambda Functions (Anonymous Functions)**

These are small, one-line functions defined with the lambda keyword.

python

Copy

add = lambda x, y: x + y

print(add(3, 5))

### **8. Return in Functions**

The return statement is used to exit a function and optionally return a value.

python

Copy

def square(x):

return x \* x

result = square(4)

print(result)

These tools help manage how your program executes, responds to user input, or handles data.

* if: Checks the condition. If it evaluates to True, the associated block of code is executed.
* elif: Stands for "else if." It's used to check additional conditions if the previous if conditions are false.
* else: Executes the block of code when all preceding conditions are false.

### **2. Loops (for, while)**

Loops are used to repeat a block of code multiple times.

#### **for loop:**

Used for iterating over a sequence (like a list, tuple, or string).

python

Copy

for i in range(5):

print(i)

* range(5) generates numbers from 0 to 4, and the loop will execute the print statement five times.

#### **while loop:**

Executes a block of code as long as the given condition is True.

count = 0

while count < 5:

print(count)

count += 1

* The loop will continue until count becomes 5.

### **3. Break and Continue**

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

python

for i in range(5):

if i == 3:

break

print(i)

Output:

0

1

2

The loop breaks when i == 3.

python

Copy

for i in range(5):

if i == 3:

continue

print(i)

Output:

0

1

2

4

The loop skips printing 3 and continues.

### **4. Pass**

The pass statement is a placeholder that does nothing. It can be used when a statement is syntactically required but you don't want to execute any code.

if condition:

pass # Nothing happens here

### **5. Try and Except (Exception Handling)**

Python provides exception handling tools to catch and handle errors without terminating the program.

num = int(input("Enter a number: "))

except ValueError:

print("Invalid input! Not a number.")

### **6. Functions (Defining and Calling)**

Functions help control the flow by allowing you to define reusable blocks of code.

def greet(name):

print(f"Hello, {name}!")

greet("Alice")

### **7. Lambda Functions (Anonymous Functions)**

These are small, one-line functions defined with the lambda keyword.

python

Copy

add = lambda x, y: x + y

print(add(3, 5))

### **8. Return in Functions**

The return statement is used to exit a function and optionally return a value.

python

Copy

def square(x):

return x \* x

result = square(4)

print(result)

These tools help manage how your program executes, responds to user input, or handles data.

#even or odd

number = int(input("Enter a number: "))

if number % 2 == 0:

print(f"{number} is even.")

else:

print(f"{number} is odd.")

Enter a number: 4

4 is even.

a=int(input("enter start range"))

b=int(input("enter end range"))

sum =0

for i in range(a,b+1):

sum=sum+i

print(sum)

enter start range1

enter end range4

1

3

6

10

number = int(input("Enter a number to find its factorial: "))

factorial = 1

for i in range(1, number + 1):

factorial \*= i

print(f"The factorial of {number} is {factorial}.")

Enter a number to find its factorial: 5

The factorial of 5 is 120.

n = int(input("Enter the number of terms in the Fibonacci sequence: "))

a, b = 0, 1

print("Fibonacci sequence:")

for i in range(n):

print(a, end=" ")

a, b = b, a + b

Enter the number of terms in the Fibonacci sequence: 5

Fibonacci sequence:

0 1 1 2 3

num1=float(input("Enter first number:"))

num2=float(input("Enter second number:"))

operation =input("Enter operation(+,-,\*,/):")

if operation == '+':

print("Result:",num1+num2)

elif operation == '-':

print("Result:", num1-num2)

elif operation =='\*':

print("Result:",num1\*num2)

elif operation =='/':

print("reslut:",num1/num2)

else

print("Invalid Operation")

year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

print(f"{year} is a leap year.")

else:

print(f"{year} is not a leap year.")

Enter a year: 2009

2009 is not a leap year.

9s

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num3 = float(input("Enter the third number: "))

if num1 >= num2 and num1 >= num3:

largest = num1

elif num2 >= num1 and num2 >= num3:

largest = num2

else:

largest = num3

print(f"The largest number is {largest}.")

Enter the first number: 34

Enter the second number: 12

Enter the third number: 90

The largest number is 90.0.

number = float(input("Enter a number: "))

if number > 0:

print(f"{number} is a positive number.")

elif number < 0:

print(f"{number} is a negative number.")

else:

print(f"{number} is zero.")

Enter a number: -5

-5.0 is a negative number.

start = int(input("Enter the start of the range: "))

end = int(input("Enter the end of the range: "))

sum\_even = 0

for num in range(start, end + 1):

if num % 2 == 0:

sum\_even += num

print(f"The sum of even numbers between {start} and {end} is {sum\_even}.")

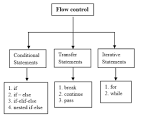
Enter the start of the range: 2

Enter the end of the range: 10

The sum of even numbers between 2 and 10 is 30.

What are control flow statements in Python?

AI Overview



Control flow statements in Python are used to control the order of a program's execution. They allow programmers to create logical pathways and make decisions in their code.

Types of control flow statements

* **Conditional statements**: Use if, elif, and else to run different code blocks based on conditions
* **Looping statements**: Use for and while to repeatedly run code
* **Jumping statements**: Use break, continue, and pass to skip iterations or stop execution

How control flow statements work

* Control flow statements break up the flow of execution by using decision making, looping, and branching
* They allow programs to conditionally execute particular blocks of code
* They enable programs to respond dynamically to different situations

Examples of control flow statements

* **If statement**: The most fundamental form of a conditional statement
* **Elif statement**: Used to test multiple conditions
* **For loop**: A looping statement that allows you to repeatedly run code
* **While loop**: A looping statement that allows you to repeatedly run code

Python provides three primary control statements: continue, break, and pass