Control flow statements in Programming

Control flow refers to the order in which statements within a program execute. While programs typically follow a **sequential flow** from top to bottom, there are scenarios where we need more flexibility. This article provides a clear understanding about everything you need to know about Control Flow Statements.

What are Control Flow Statements in Programming?

Control flow statements *are fundamental components of programming languages that allow developers to control the order in which instructions are executed in a program. They enable execution of a block of code multiple times, execute a block of code based on conditions, terminate or skip the execution of certain lines of code, etc.*

Types of Control Flow statements in Programming:

Control Flow Statements Type	Control Flow Statement	Description
Conditional Statements	if-else	Executes a block of code if a specified condition is true, and another block if the condition is false.
Looping Statements	for	Executes a block of code a specified number of times, typically iterating over a range of values.
	while	Executes a block of code as long as a specified condition is true.
Jump Statements	break	Terminates the loop or switch statement and transfers control to the statement immediately following the loop or switch.
	continue	Skips the current iteration of a loop and continues with the next iteration.
	return	Exits a function and returns a value to the caller.

Conditional Statements in Programming:

Conditional statements in programming are used to execute certain blocks of code based on specified conditions. They are fundamental to decision-making in programs. Here are some common types of conditional statements:

1. If Statement in Programming:

The if statement is used to execute a block of code if a specified condition is true.

C++CJavaC#JavaScriptPython3

```
a = 5

if a == 5:

    print("a is equal to 5")
```

Output

a is equal to 5

2. if-else Statement in Programming:

The if-else statement is used to execute one block of code if a specified condition is true, and another block of code if the condition is false.

C++CJavaC#JavaScriptPython3

```
a = 10

if a == 5:
    print("a is equal to 5")
else:
    print("a is not equal to 5")
```

Output

a is not equal to 5

3. if-else-if Statement in Programming:

The if-else-if statement is used to execute one block of code if a specified condition is true, another block of code if another condition is true, and a default block of code if none of the conditions are true.

Python3

```
a = 15

if a == 5:
    print("a is equal to 5")
elif a == 10:
    print("a is equal to 10")
else:
    print("a is not equal to 5 or 10")
```

Output

a is not equal to 5 or 10

4. Ternary Operator or Conditional Operator in Programming:

In some programming languages, a ternary operator is used to assign a value to a variable based on a condition.

Python3

```
a = 10

print("a is equal to 5" if a == 5 else "a is not equal to 5")
```

Output

a is not equal to 5

Each programming language may have its own syntax and specific variations of these conditional statements.

Looping Statements in Programming:

Looping statements, also known as iteration or repetition statements, are used in programming to repeatedly execute a block of code. They are essential for performing tasks such as iterating over elements in a list, reading data from a file, or executing a set of instructions a specific number of times. Here are some common types of looping statements:

1. For Loop in Programming:

The for loop is used to iterate over a sequence (e.g., a list, tuple, string, or range) and execute a block of code for each item in the sequence.

```
for i in range(5):
    print(i)
```

Output

0

1

2

3

4

2. While Loop in Programming:

The while loop is used to repeatedly execute a block of code as long as a specified condition is true.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Output

0

1

2

3

4

4. Nested Loops in Programming:

Loops can be nested within one another to perform more complex iterations. For example, a for loop can be nested inside another for loop to create a two-dimensional iteration.

```
for i in range(2):
    for j in range(2):
```

```
    print(f"i={i} j={j}")
```

Output

```
i=0 j=0
i=0 j=1
i=1 j=0
i=1 j=1
```

Each programming language may have its own syntax and specific variations of these looping statements.

Jump Statements in Programming:

Jump statements in programming are used to change the flow of control within a program. They allow the programmer to transfer program control to different parts of the code based on certain conditions or requirements. Here are common types of jump statements:

1. Break Statement in Programming:

The break statement is primarily used to exit from loops prematurely. When encountered inside a loop, it terminates the loop's execution and transfers control to the statement immediately following the loop.

```
for i in range(10):
    if i == 5:
        break
    print(f"{i} ", end="")
```

Output

```
0 1 2 3 4
```

2. Continue Statement in Programming:

The continue statement is used to skip the current iteration of a loop and proceed to the next iteration.

```
for i in range(10):
    if i % 2 == 1:
        continue
    print(f"{i} ", end="")
```

Output

```
0 2 4 6 8
```

3. Return Statement in Programming:

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

Python is a widely used high-level, interpreted programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

What can we do with Python?

Python is used for:

- **Web Development:** Frameworks like Django, Flask.
- **Data Science and Analysis:** Libraries like Pandas, NumPy, Matplotlib.
- **Machine Learning and AI:** TensorFlow, PyTorch, Scikit-learn.
- **Automation and Scripting:** Automate repetitive tasks.
- **Game Development:** Libraries like Pygame.
- **Web Scraping:** Tools like BeautifulSoup, Scrapy.
- **Desktop Applications:** GUI frameworks like Tkinter, PyQt.
- **Scientific Computing:** SciPy, SymPy.
- **Internet of Things (IoT):** MicroPython, Raspberry Pi.
- **DevOps and Cloud:** Automation scripts and APIs.
- **Cybersecurity:** Penetration testing and ethical hacking tools.

Key Features of Python

- **Easy to Learn and Use:** Python's simple and readable syntax makes it beginner-friendly.
- **Cross-Platform Compatibility:** Python runs seamlessly on Windows, macOS, and Linux.
- **Extensive Libraries:** Includes robust libraries for tasks like web development, data analysis, and machine learning.
- **Dynamic Typing:** Variable types are determined automatically at runtime, simplifying code writing.
- **Versatile:** Supports multiple programming paradigms, including object-oriented, functional, and procedural programming.
- **Open Source:** Python is free to use, distribute, and modify.

Why Learn Python?

Whether you are a beginner or an experienced developer, both have their own benefits.

For Beginners:

- **Easy Syntax:** Python syntax is like plain English, which allows you to focus on logic instead of worrying about complex rules.
- **Built-in Libraries for Beginners:** Python has beginner friendly libraries like random, re, os etc, which can be used while learning fundamentals.
- **Error Friendly:** Python's error messages are easy to understand and debug.
- **Project Oriented Learning:** You can start making simple projects while learning the Python basics.

For Experienced:

- **Easy Career Transition:** If you know any other programming language, moving to Python is super easy.
- **Great for Upskilling:** Moving to Python expands your skill sets and gives opportunity to work in areas like AI, Data Science, web development etc.
- **High Demand of Python in Emerging tech:** Python is widely used in trending domains, like Data Science, Machine Learning, Cloud Computing etc.
- **Bridge Between Roles:** For software developers working with different language, learning Python can help you integrate advanced features like AI in your projects.

Hello World in Python

Hello, World! in python is the first python program which we learn when we start learning any program. It's a simple program that displays the message "Hello, World!" on the screen.

Here's the "Hello World" program:

```
print("Hello, World!")
```

Output

```
Hello, World!
```

How does this work:

- print() is a built-in function in Python that tells the program to display something on the screen. We need to add the string in parenthesis of print() function that we are displaying on the screen.

- “Hello, World!” is a string text that we want to display. Strings are always enclosed in quotation marks.

Python Comments

Comments in Python are the lines in the code that are ignored by the interpreter during the execution of the program.

- Comments enhance the readability of the code.
- Comment can be used to identify functionality or structure the code-base.
- Comment can help understanding unusual or tricky scenarios handled by the code to prevent accidental removal or changes.
- Comments can be used to prevent executing any specific part of your code, while making changes or testing.

Example:

 # I am single line comment

""" Multi-line comment used

print("Python Comments") """

Explanation:

- In Python, single line comments starts with hashtag **symbol #.**
- Python ignores the string literals that are not assigned to a variable. So, we can use these string literals as Python Comments.

Indentation in Python

In Python, indentation is used to define blocks of code. It tells the Python interpreter that a group of statements belongs to a specific block. All statements with the same level of indentation are considered part of the same block. Indentation is achieved using whitespace (spaces or tabs) at the beginning of each line.

Example:

```
if 10 > 5:
    print("This is true!")
    print("I am tab indentation")
```

```
print("I have no indentation")
```

Explanation:

- The first two print statements are indented by 4 spaces, so they belong to the if block.
- The third print statement is not indented, so it is outside the if block.

Famous Application Built using Python

- **YouTube:** World's largest video-sharing platform uses Python for features like video streaming and backend services.
- **Instagram:** This popular social media app relies on Python's simplicity for scaling and handling millions of users.
- **Spotify:** Python is used for backend services and machine learning to personalize music recommendations**.**
- **Dropbox:** The file hosting service uses Python for both its desktop client and server-side operations.
- **Netflix:** Python powers key components of Netflix's recommendation engine and content delivery systems (CDN).
- **Google:** Python is one of the key languages used in Google for web crawling, testing, and data analysis.
- **Uber:** Python helps Uber handle dynamic pricing and route optimization using machine learning.
- **Pinterest:** Python is used to process and store huge amounts of image data efficiently.

Input and Output in Python

With the print() function, we can display output in various formats, while the input() function enables interaction with users by gathering input during program execution.

Printing Output using print() in Python

At its core, printing output in Python is straightforward, thanks to the print() function. This function allows us to display text, variables and expressions on the console. Let's begin with the basic usage of the print() function:

In this example, "Hello, World!" is a string literal enclosed within double quotes. When executed, this statement will output the text to the console.

```
print("Hello, World!")
```

Output

Hello, World!

Printing Variables

We can use the print() function to print single and multiple variables. We can print multiple variables by separating them with commas. **Example:**

 # Single variable

s = "Akash"

print(s)

 # Multiple Variables

s = "Adi"

age = 25

city = "Haridwar"

print(s, age, city)

Output

Akash

Adi 25 Haridwar

Taking input in Python

Python input() function is used to take user input. By default, it returns the user input in form of a string.

Syntax: *input(prompt)*

How to Take Input in Python

The code prompts the user to input their name, stores it in the variable “name”, and then prints a greeting message addressing the user by their entered name.

name = input("Enter your name: ")

print("Hello,", name, "! Welcome!")

Output

Enter your name: BSACET
Hello, BSACET! Welcome!

How to Change the Type of Input in Python

By default input() function helps in taking user input as string. If any user wants to take input as int or float, we just need to typecast it.

How to Print Names in Python

The code prompts the user to input a string (the color of a rose), assigns it to the variable color, and then prints the inputted color.

 # Taking input as string

color = input("What color is rose?: ")

print(color)

Output

What color is rose?: Red
Red

How to Print Numbers in Python

The code prompts the user to input an integer representing the number of roses, converts the input to an integer using typecasting, and then prints the integer value.

 # Taking input as int

 # Typecasting to int

n = int(input("How many roses?: "))

print(n)

Output

How many roses?: 8
8

How to Print Float/Decimal Number in Python

The code prompts the user to input the price of each rose as a floating-point number, converts the input to a float using typecasting, and then prints the price.

 # Taking input as float

 # Typecasting to float

```
price = float(input("Price of each rose?: "))

print(price)
```

Output

```
Price of each rose?: 50.30
50.3
```

Python Syntax

Python syntax is like grammar for this programming language. Syntax refers to the set of rules that defines how to write and organize code so that the Python interpreter can understand and run it correctly. These rules ensure that your code is structured, formatted, and error-free.

Here are some basic Python syntax:

Indentation in Python

Python Indentation refers to the use of whitespace (spaces or tabs) at the beginning of code line. It is used to define the code blocks. Indentation is crucial in Python because, unlike many other programming languages that use braces "{}" to define blocks, Python uses indentation. It improves the readability of Python code, but on other hand it became difficult to rectify indentation errors. Even one extra or less space can leads to indentation error.

Python keywords

Keywords in Python are reserved words that have special meanings and serve specific purposes in the language syntax. They cannot be used as identifiers (names for variables, functions, classes, etc.). For instance, "for", "while", "if", and "else" are keywords and cannot be used as identifiers.

Below is the list of keywords in Python:

False	await	else	import	pass	
None	break	except	in	raise	
True	class	finally	is	return	
and	continue	for	lambda	try	
as	def	from	nonlocal	while	
assert	del	global	not	with	
async	elif	if	or	yield	

Comments in Python

Comments in Python are statements written within the code. They are meant to explain, clarify, or give context about specific parts of the code. The purpose of comments is to explain the working of a code, they have no impact on the execution or outcome of a program.

Python Single Line Comment

Single line comments are preceded by the "#" symbol. Everything after this symbol on the same line is considered a comment.

Python Multi-line Comment

Python doesn't have a specific syntax for multi-line comments. However, programmers often use multiple single-line comments, one after the other, or sometimes triple quotes (either ''' or """), even though they're technically string literals. Below is the example of multiline comment.

```
'''
Multi Line comment.
Code will print name.
```

'''

Taking Input from User in Python

The input() function in Python is used to take user input from the console. The program execution halts until the user provides input and presses "Enter". The entered data is then returned as a string. We can also provide an optional prompt as an argument to guide the user on what to input.

name = input("Please enter your name: ")