# FLOW OF CONTROL IN PYTHON

BRIDGE COURSE

GRADE-8

## PRIMARY PREVIEW

- Python Statements-Flow of Control
- Ternary Operator in Python
- Jump Statements
- Python Iterative Statements
- Some More Programs
- Python Selection Statements
- The Infinite Loop

## PYTHON STATEMENTS-FLOW OF CONTROL

The order in which program statements are executed is called the **flow of control**. It can be **sequential**, **selective** or **iterative**.

In all the programs you have written so far, the statements execute sequentially—meaning one after the other, from top to bottom in the program. This type of flow is called **sequential flow**.

In **selective** flow of control, also called **Decision Making**, the interpreter chooses which statement to execute above others, whereas in **iterative** flow, some statements are executed multiple times. You have already learnt about sequential statements, let's learn about the other flow of control statements.

## PYTHON SELECTION STATEMENTS

You make many decisions every day—like "What should I eat today?" or "Which shirt should I wear?" Each of these decisions is based on a certain criterion. For example, your food choice might depend on your mood or whether you're following a specific diet. Once you've made the decision, you follow it with an action. Therefore, decision-making is a two-step process: first, deciding what to do based on a criterion and second, taking an action.

### IF STATEMENT

Decision-making by a computer is based on the same two-step process. In Python, decisions are made using the `if` statement, also known as the **selection statement**. When an `if` statement is executed, the computer evaluates a condition (a logical expression). If the condition is `True`, the action inside the `if` block is performed. If not, it will be skipped.

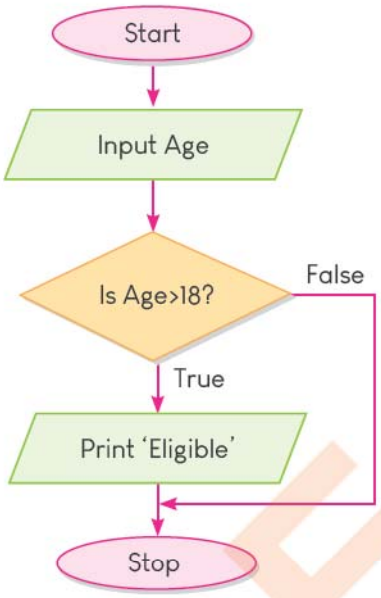
Syntax of the `if` statement:

```
if (condition):
    Statement(s)
```

The `if` statement starts with the keyword `if`, followed by a condition and ends with a colon(:). The condition is essentially an expression (comparison) that gives `True` or `False` as an answer.

After the `if` condition, on the next line, write an indented block of statements. When the condition evaluates to true, the block of statements given below it is executed; otherwise, it is skipped and control flows to the statement after `if`.

Here, if the age of the user is more than or equal to 18, 'Eligible' will be printed, otherwise nothing will be printed.




**Program 1** To demonstrate the use of the 'if' statement.

Program1.py

File Edit Format Run Options Window Help

```
age=int(input("Enter Age: "))
if(age>=18):
    print("You are eligible to cast a vote")
```

Output

```
Enter Age: 20
You are eligible to cast a vote
Enter Age: 16
```

**Program 2** To greet the user based on the time entered.

Program2.py

File Edit Format Run Options Window Help

```
time=int(input("Enter Time in 24-hour format: "))
if (time<12):
    print("Good Morning")
```

```
Output
Enter Time in 24-hour format: 6
Good Morning
Enter Time in 24-hour format: 18
```

**LIVE LEARNING**

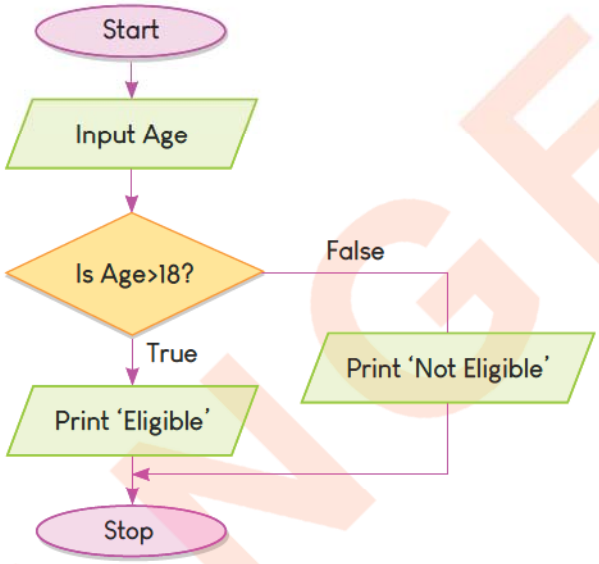
Write a Python program to enter a number. If it is less than 0, display: 'Negative numbers are not allowed'.

## IF ... ELSE ... STATEMENT

The **if...else... statement** lets your program choose between two options. Python checks the condition; if it's true, it executes statement block 1 after the `if` statement—otherwise, it executes statement block 2 after the `else` statement.

Syntax of the `if ... else ...` statement:

```
if (condition):
    Statement block 1
else:
    Statement block 2
```




Note that both statement blocks 1 and 2 are indented. Block 1 is under `if`, so it will be executed if the condition mentioned is True. Block 2 is under `else`, so it will be executed if the condition is False.

Here, if the age of the user is more than 18, 'Eligible' will be printed, otherwise 'Not Eligible' will be printed.

**Program 3** To show a message if the user is 'Eligible' or 'Not Eligible' to cast a vote.

Program3.py

File Edit Format Run Options Window Help

```
age=int(input("Enter Age: "))
if ( age>=18):
    print("You are eligible to cast a vote.")
else:
    print("You are not eligible to cast a vote.")
```

```
Output
Enter Age: 25
You are eligible to cast a vote.

Enter Age: 15
You are not eligible to cast a vote.
```

**Program 4** To check whether a number is odd or even.

Program4.py

File Edit Format Run Options Window Help

```
Num=int(input("Enter a Number: "))
if Num%2==0:
  print("Number entered is an even number.")
else:
  print("Number entered is an odd number.")
```

Output

```
Enter a Number: 12
Number entered is an even number.

Enter a Number: 7
Number entered is an odd number.
```

This program takes an input Num as an integer, divides it by 2 and compares the remainder with 0. If it is found equal, the number is even, otherwise the program prints that the number is odd.

## NESTED IF STATEMENT

The **nested if** structure allows you to place one if statement inside another. This means the program first checks a main condition and if that condition is true, it checks another condition within it.

**Program 5** To check if a number is positive or negative and, if so, check whether it is even or odd.

Program5.py

File Edit Format Run Options Window Help

```
num = int(input("Enter a number: "))
if num > 0:
    print("You have entered a positive number.")
    if num % 2 == 0:
        print("Even")
    else:
        print("Odd")
else:
    print("You have entered a negative number.")
    if num % 2 == 0:
        print("Even")
    else:
        print("Odd")
```

```
Output
Enter a number:9
You have entered a positive number.
Odd

Enter a number:-4
You have entered a negative number.
Even
```

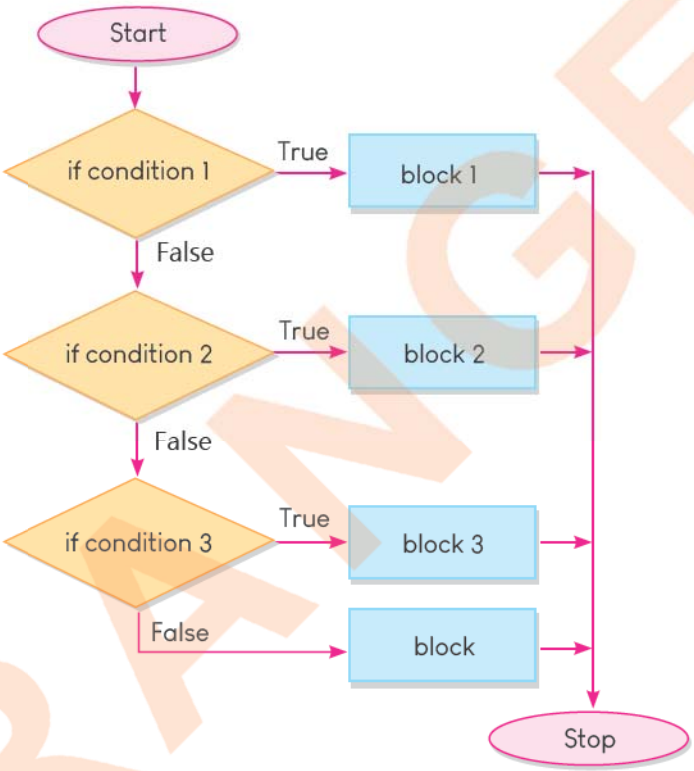
## IF..ELIF....ELSE STATEMENT

Python provides the `elif`(else-if) statement to check for multiple conditions and execute the code block within if any of the conditions are evaluated to be true.

The `elif` statement is optional like the else statement but multiple `elif` statements can be used in a block of code following an if statement.

Syntax of the `if..elif....else` statement:

```
if condition1:
    Statement block 1
elif condition2:
    Statement block 2
elif condition3:
    Statement block 3
    ..
    ..
else:
    Statement block n
```




The interpreter starts at the top, testing condition1. If it is true, statement block 1 is executed and the rest of the constructs are ignored. If it is false, the control moves down and checks condition2, if it is true, statement block 2 will be executed and if it is false, it moves next to condition3 and so on. The process repeats till a true condition is found. In case none of the conditions evaluate to true, the block after the `else` is executed.

**Program 6** To take marks of a student as input and display a message based on the marks:

- If the marks are equal or more than 90, show "Excellent"
- If equal or more than 70, show "Very Good"

- If equal or more than 50, show "Fairly Good"
- Otherwise, show "Need more effort"

Program6.py

File Edit Format Run Options Window Help

```
marks =int(input("Enter your marks:  "))
if marks>=90:
    print("Excellent")
elif marks>=70:
    print("Very Good")
elif marks>=50:
    print("Fairly Good")
else:
    print("Need more effort")
```

Output

```
Enter your marks: 90
Excellent
Enter your marks: 75
Very Good
Enter your marks: 51
Fairly Good
Enter your marks: 40
Need more effort
```

The number of `elif` statements used in this construct can vary depending on the problem and there is no fixed upper limit for it. However, `else` can be used only once for each `if` statement.

## EXPERIENTIAL LEARNING

21st Century Skills #Critical Thinking

Create a Discount Calculator program that calculates the discount based on the purchase amount. The discount rate changes based on the amount spent and the program should display the discount amount and the final total. Students will use if-else statements to apply the correct discount and perform mathematical operations like multiplication and subtraction based on the purchase value.

## RAPID RECALL

Tick (✓) if you know this.

1. The if...else... statement lets your program choose between two options.
2. The nested if structure allows you to place one if statement inside another.

## TERNARY OPERATOR IN PYTHON

The **ternary operator** allows you to perform conditional checks and assign values or execute expressions in a single line. It is also called a **conditional expression** because it evaluates a condition and returns one value if the condition is True and another value if the condition is False.

Syntax of the `ternary operator`:

```
<value_if_true> if (condition) else <value_if_false>
```

**Program 7** To print "Eligible to vote" if the age is greater than or equal to 18, else "Not eligible to vote"

Program7.py

File Edit Format Run Options Window Help

```
age = int(input("Enter your age: "))
print("Eligible to vote" if age >= 18 else "Not eligible to vote")
```

Output

```
Enter your age: 15
Not eligible to vote
```

## PYTHON ITERATIVE STATEMENTS

**Iteration** means repeating a set of instructions again and again. It is a flow of control that allows a part of the code to run multiple times without writing it again each time. In many programs, we often need to repeat certain actions. Instead of copying the same code many times, we use loops to do it automatically.

Using loops helps save time, reduce mistakes and make the code shorter and cleaner.

Python provides two main types of loops: the **for loop**, which repeats a block of code for each item in a sequence or for a specific number of iterations and the **while loop**, which repeats a block of code as long as a condition is true.

**range() function**

The `range()` function is a built-in function in Python that generates a sequence of numbers.

The `range()` function is most commonly used with the `for` loop to repeat tasks a specific number of times.

Syntax of the `range()` function:

```
range(start, stop, step)
```

where,

- **`start`:** This is the optional value where the sequence begins. If you don't specify it, Python will automatically start from 0.

  For example, range(5) is treated as range(0, 5) and generates: [0, 1, 2, 3, 4].

- **`stop`:** This is the mandatory value at which the sequence **ends**, but it is **not included** in the result.

  For example, range(2, 6) gives: [2, 3, 4, 5] (6 is excluded).

**FACT File**

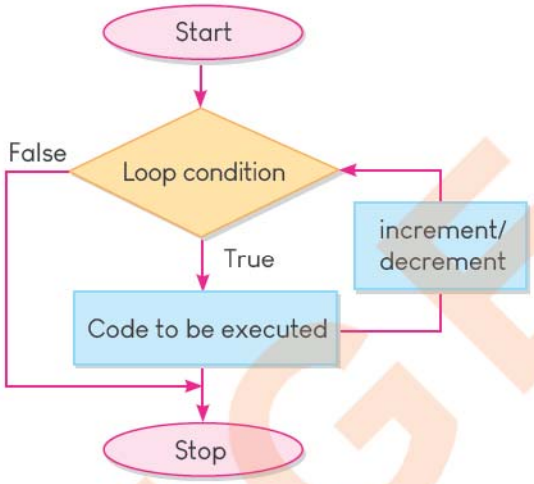
Python has no limit on variable name length, but line lengths should be kept under **79** characters for better readability.

- **step:** This optional value defines how much the value should increase or decrease each time. The default step value is 1, which means the numbers go up one at a time. You can also use a negative step value to create a descending sequence.

  For example: range(10, 2, -2) gives [10, 8, 6, 4].

  range(2, 8, 3) gives [2, 5].

## FOR LOOP IN PYTHON



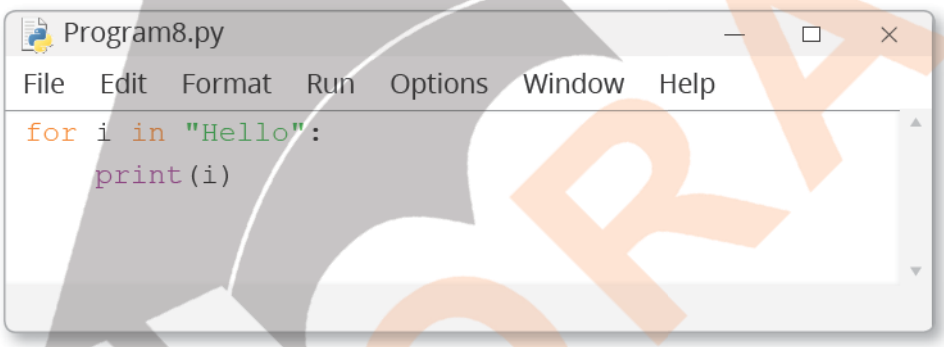

The `for` loop repeats a block of statements a fixed number of times. It is commonly used to go through (iterate over) a sequence like a list, a string or a range of numbers. It can be specified using a sequence or a range of values.

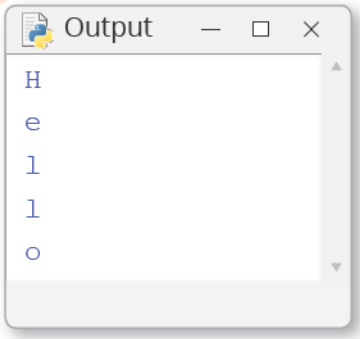
Syntax of the `for` loop:

```
for variable in sequence:
    Statement block to be repeated
```

The `for` loop variable iterates over the set of values given in the sequence, repeating the loop body once for each item. Therefore, the loop header itself defines how many times the loop is going to repeat. For each iteration, the loop variable takes up the next value from the set and executes the loop body one more time. The process goes on till the last item of the set has been used.

**Program 8** To print each character of a string.




Syntax of the `for` loop with the range():

```
for variable in range(start, stop, step):
    Statement block to be repeated
```

The `for` loop initialises the `variable` with the `start` value. With each iteration, the variable changes based on the `step` value—increment or decrement. This continues until it reaches the stop value (which is not included). The code inside the loop runs once for each value. This makes it easy to repeat tasks in a clear and controlled way when you know how many times you want to repeat them.

**Program 9** To print the squares of the first five natural numbers.

Program9.py

File Edit Format Run Options Window Help

```
for i in range(1,6):
    print(i*i)
```

Output

```
1
4
9
16
25
```

**Program 10** To find the sum of the first 10 natural numbers.

Program10.py

File Edit Format Run Options Window Help

```
sum=0
for i in range(0,11):
    sum+=i
print(sum)
```

Output

```
55
```

**Program 11** To print the first five multiples of a number.

Program11.py

File Edit Format Run Options Window Help

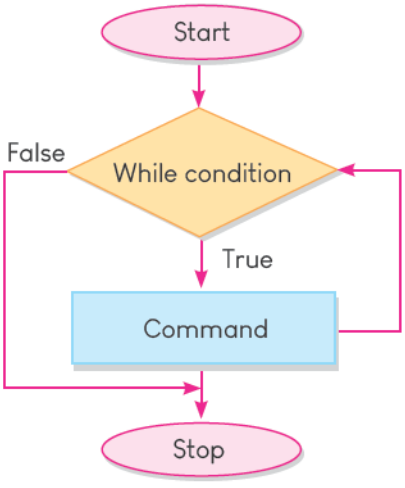
```
num = int(input("Enter a number: "))
for i in range(1, 6):
    print(num, "x", i, "=", num * i)
```

Output

```
Enter a number: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
```

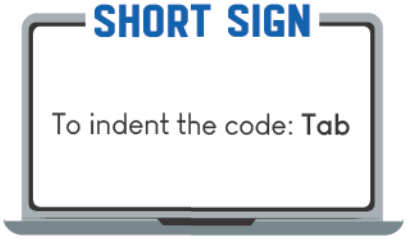
## WHILE LOOP IN PYTHON

A `while` loop repeats a set of instructions as long as a given condition is true. It checks the condition before each repetition. If the condition is true, the loop executes; if it's false, the loop stops and the control moves to the statement outside the loop. It is often used when the number of repetitions is not known in advance.




Syntax of `while` statement:

```
while <condition>:
    Statement block to be
    repeated
```



Let us look at an example:

**Program 12** To print the first five natural numbers.

Program12.py

File Edit Format Run Options Window Help

```
i=1
while i<=5:
    print(i)
    i+=1
```

Output

```
1
2
3
4
5
```

## THE INFINITE LOOP

An **infinite loop** is a loop that never ends because its condition is always true and never becomes false. For example, if a loop is supposed to stop when a certain variable reaches a value but that variable is never changed inside the loop, the condition remains true forever.

**Program 13** To demonstrate the execution of an infinite loop.

Program13.py

File Edit Format Run Options Window Help

```
i=1
while(i<10):
    print("Orange Education")
```

Output

```
Orange Education
Orange Education
Orange Education
Orange Education
Orange Education
```

Here, the value of i is not changed and it will always be less than 10, so Orange Education will keep on printing infinitely.

To exit the infinite loop, you can close the program window. This will break the execution of the program.

### RAPID RECALL

Tick (✓) if you know this.

1. Iteration means repeating a set of instructions again and again.
2. The for loop repeats a block of statements a fixed number of times.

## JUMP STATEMENTS

Sometimes, you may want to stop a loop early or skip certain steps inside a loop. In such cases, Python provides **jump statements** to control the flow of the loop. Python offers two jump statements–**break** and **continue**, which are used within loops.

### THE BREAK STATEMENT

The **break** statement is used to exit a loop before it has completed all its iterations. When Python encounters break, it immediately stops the loop, exits it and move on to the code after the loop.

**HINTS & HACKS**

The **break** statement only exits the immediate loop in which it is written; it does not affect any outer loops.

**Program 14** To accept 5 even numbers from the user. If an odd number is entered, display a message and exit the loop, showing how many even numbers are entered.

Program14.py

File Edit Format Run Options Window Help

```
count = 0
while count < 5:
    num = int(input("Enter an even number: "))
    if num % 2 != 0:
        print("Odd number entered. Exiting.")
        break  # stop the loop if number is odd
    count += 1
print("You entered",count,"even numbers.")
```

Output

```
Enter an even number: 2
Enter an even number: 6
Enter an even number: 8
Enter an even number: 5
Odd number entered. Exiting.
You entered 3 even numbers.
```

### THE CONTINUE STATEMENT

The continue statement is used to skip the current iteration of the loop and move to the next one. When Python encounters continue, it goes back to the beginning of the loop, without running the rest of the code in that iteration.

**HINTS & HACKS**

The **continue** statement only skips the immediate iteration in which it is written.

**Program 15** To print only odd numbers, skip any even number.

Program15.py

File Edit Format Run Options Window Help

```
for i in range(1, 11):
    if i % 2 == 0:
        continue  # skip even numbers
    print("Odd number:", i)
print("Out of for loop")
```

Output

```
Odd number: 1
Odd number: 3
Odd number: 5
Odd number: 7
Odd number: 9
Out of for loop
```



## SOME MORE PROGRAMS

**Program 16** To find the average of the first 10 natural numbers.

Program16.py

File Edit Format Run Options Window Help

```
total = 0
for i in range(1, 11):
    total += i
average = total / 10
print("Average of first 10 natural numbers is:", average)
```

Output

```
Average of first 10 natural numbers is: 5.5
```

**Program 17** To know the amount of discount based on how much money you spent.

| Amount spent | Discount% |
|---|---|
| ₹5000 or more | 20% off |
| ₹3000 to ₹4999 | 10% off |
| ₹1000 to ₹2999 | 5% off |
| Less than ₹1000 | No discount |

Show the final amount of discount and amount to be paid.

Program17.py

File Edit Format Run Options Window Help

```
amount = float(input("Enter the total purchase amount: "))
if amount >= 5000:
    discount = 0.20  # 20% discount
    print("You get a 20% discount.")
elif amount >= 3000:
    discount = 0.10  # 10% discount
    print("You get a 10% discount.")
elif amount >= 1000:
    discount = 0.05  # 5% discount
    print("You get a 5% discount.")
else:
    discount = 0     # No discount
    print("No discount available.")
discount_amount=amount*discount
print("Discount amount is : ", discount_amount)
final_amount = amount - (amount * discount)
print("Amount to be paid after discount:", final_amount)
```

Output

```
Enter the total purchase amount: 2500
You get a 5% discount.
Discount amount is: 125.0
Amount to be paid after discount: 2375.0
```

**Program 18** To print a right-angled triangle using the pattern of stars.

```
*
**
***
****
*****
```

Program18.py

File Edit Format Run Options Window Help

```
for i in range(1, 6):
    print("*" * i)
```

Output

```
*
**
***
****
*****
```

**TECH TERMS**

- **Expression:** A combination of values, variables, operators and functions that Python evaluates to produce a result.
- **Interpreter:** A program that reads and executes Python code line by line, converting it to machine code.
- **Execution:** The process of running code, where the interpreter executes instructions to produce results.

## REWIND RUN

- The order in which program statements are executed is called the flow of control.
- The if...else... statement lets your program choose between two options.
- The nested if structure allows you to place one if statement inside another.
- The ternary operator allows you to perform conditional checks and assign values or execute expressions in a single line.
- The for loop repeats a block of statements a fixed number of times.
- A while loop repeats a set of instructions as long as a given condition is true.
- Python offers two jump statements—break and continue, which are used within loops.