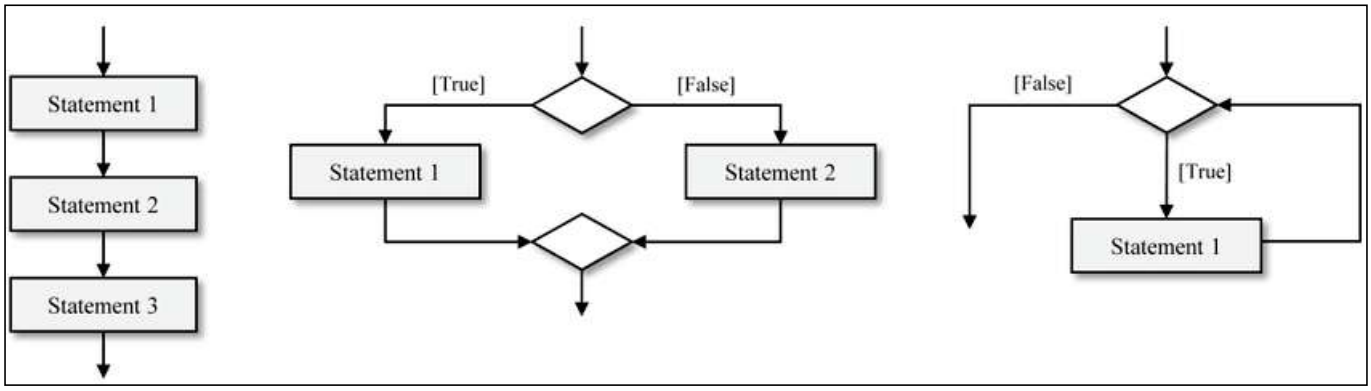
Chapter-5
PYTHON CONTROL STRUCTURES

Familiar



A B C

What if I say that you are very familiar with control structures

- In fact you use these structures in your daily life
- Also you have been using them for the last few classes

Take a guess which among the above diagram is a

- Loop (Iteration)
- Sequence
- Condition (Selection)

A is .....

B is .....

C is .....

Let's understand

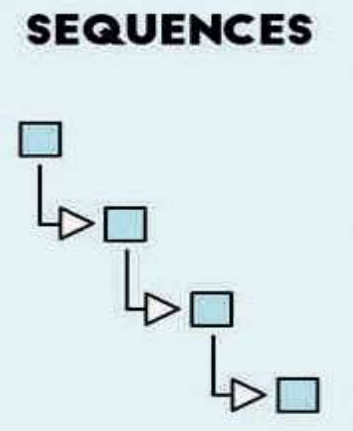
When we are using software or carrying out our general day-to-day tasks. We subconsciously make use of these structures.

- **Its Raining** - If it is raining we wear a raincoat. If it is not raining we do not wear it. This is basically a **Condition (Selection)** structure where you are taking decision based on a condition.

- **To brush teeth**, you first pick your tooth brush then you open the lid of tooth paste, then you squeeze tooth paste and so on.... These are basically steps performed one after another to carry out a task. This is basically a **Sequence structure.**

- Assume you are **bowling in a cricket match**. You have to bowl 6 balls to complete one over. You will first take a ball and then run towards a batsman and you will bowl it. After completing one ball, you will again bowl the next ball. Again after second ball, you will bowl a third ball. You are basically repeating the same step for 6 times to complete a task. This is basically an **Iterative or Loop structure.**

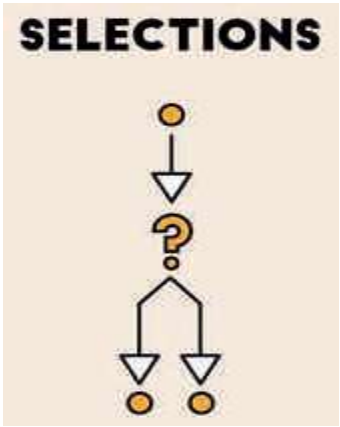
Definitions – Now that we understand these control structures. Let us look into the definitions



Sequence Structure –

A sequence is a series of actions that are completed in a specific order. Action 1 is performed then Action 2 and then Action 3 and so on. This process continues until all the actions in a sequence are carried out.

Conditional (Selection) Structure –

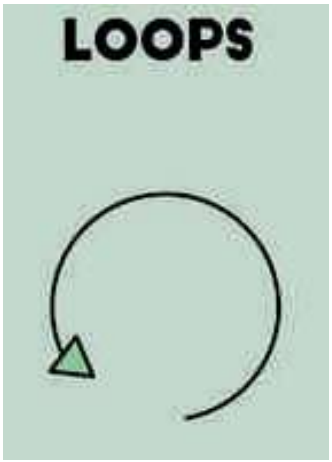


Selection structure is bit different. Instead of following one predefined order, the flow is defined by the output of a condition.

In other words, if condition A is met then take Path 1, but if condition A is not met then take Path 2.

Loop (Iteration) Structure –

In this structure also the flow is defined by the output of a condition. However, the difference is that the condition is asked over and over and over again until a certain task is complete. In other words, a certain task will be executed as long as a condition is TRUE.



PYTHON CONDITIONS

In this topic we will look **Python Conditions** in more detail. We will focus on

- if
- if ... else
- if ... elif ... else

Computers make decisions based on these IF and ELSE conditions of a software.

IF is "अगर" and **ELSE** is "वरना" in Hindi Language.

In a common form, these conditional statements in Coding or Programming is a way to tell the computer that

- if the condition is true
  - do this
- else, if the condition is false
  - do a different thing

Important Note – In a computer program, a **condition** has to be something that a computer can decide whether True or False.

True Examples	False Examples
5 > 2	5 * 2 == 12
"S" in "Saaransh"	A = 10, B = 20, A == B
English has 26 alphabets	Week has 8 days
6 – 2 == 4	6 + 2 == 6

IF Condition

When you are just checking for a condition to be **True** or **False** and take decision based on it then you use **IF** condition.

Definition - IF a condition (_____) is TRUE then do this (_____) task

Definition - अगर ये (_____) सच है तो ये (_____) करो

If statement: The "if" statement allows you to execute a block of code only if a certain condition is true. The basic syntax is as follows:

```
if condition:
    # Code block to execute if the condition is true
```

Example : Checking if a number is positive

```
num = 5

if num > 0:
    print("The number is positive.")
```

Code explanation:

- In this example, we have a variable `num` initialized with the value 5.
- The if statement checks if num is **greater** than 0. If the condition is true, it **executes** the indented block of code under the if statement.
- The indented code block simply prints "The number is positive."

Output

```
The number is positive.
```

Explanation of output: Since the value of num is 5, which is greater than 0, the **condition** in the if statement is true.

Hence, the code block within the if statement is executed, resulting in the message "The number is positive." being printed as the output.

Indentation & Block of Code

Indentation is used to **define** the grouping or block of **code**. It is a way of visually organizing the code to indicate which statements belong to a specific control structure, such as an if statement

In simple terms, indentation refers to the spaces or tabs added at the beginning of lines of code.

Here's an example to illustrate the concept of indentation:

```python
if x > 0:
    print("x is positive")
    print("This statement is inside the if block")
print("This statement is outside the if block")
```

In the above code snippet, notice the indentation used before the print statements.

The lines of code indented under the if statement (`print("x is positive")` and `print("This statement is inside the if block"))` are considered part of the if block. These statements will only be executed **if the condition** `x > 0` is **true**.

On the other hand, the line of code that is not indented (`print("This statement is outside the if block"))` is outside the if block. **It will be executed regardless of whether the condition is true or false.**

Output if x = -5 (i.e x < 0)

```
This statement is outside the if block
```

Output if x = 5 (i.e x > 0)

```
x is positive
This statement is inside the if block
This statement is outside the if block
```

IF .. ELSE Condition

IF .. ELSE condition is expansion to IF condition where along with "What to do" when condition is TRUE, we also mention "What to do" when condition is FALSE.

Definition - IF a condition (_____) is TRUE then do this (_____) task ELSE do that (_____)

Definition - अगर ये (_____) सच है तो ये (_____) करो वरना ये (_____) करो

If-else statement: The "if-else" statement adds an alternative block of code to execute when the condition is false. The syntax is as follows:

```
if condition:
    # Code block to execute if the condition is true
else:
    # Code block to execute if the condition is false
```

Example: Checking if a number is a positive or negative

```
num = 5

if num >= 0:
    print("The number is positive or zero.")
else:
    print("The number is negative.")
```

Code snippet explanation:

- In this example, we have a variable `num` initialized with the value `5`.
- The `if` statement checks if `num` is **greater** than or **equal** to `0`. If the condition is **true**, it **executes** the indented block of and prints the message "The number is positive or zero."
- If the condition in the `if` statement is **false**, the program moves to the `else` block and **executes** the message "The number is negative."

IF .. ELSE IF (ELIF) .. ELSE Condition

IF .. ELSE IF .. ELSE condition is expansion to IF .. ELSE condition where we can mention "What to do" in various different conditions.

Note: In python "else if" is written as "elif"

This is called **Nested if-else statements**. They are nothing but a series of ifs and else combined together and used within another.

Definition –
IF a condition (_____) is TRUE then do this (_____) task
ELSE IF this condition (_____) is TRUE then do this (_____) task
ELSE IF this condition (_____) is TRUE then do this (_____) task
ELSE do this (_____) task

Definition –
अगर ये (_____) सच है तो ये (_____) करो वरना ये (_____) करो
वरना ये (_____) सच है तो ये (_____) करो
वरना ये (_____) सच है तो ये (_____) करो
वरना ये (_____) करो

If-elif-else statement: The "if-elif-else" statement is useful when you have multiple conditions to check. It allows you to test multiple conditions and execute different blocks of code based on which condition is true. The syntax is as follows:

python Copy code

```python
if condition1:
    # Code block to execute if condition1 is true
elif condition2:
    # Code block to execute if condition2 is true
else:
    # Code block to execute if all conditions are false
```

Example : Checking the range of the number.

```
if num < 0:
    print("The number is negative.")
elif num == 0:
    print("The number is zero.")
else:
    print("The number is positive.")
```

Code snippet explanation:

- In this example, we have a variable `num` used to represent a number whose range we want to check.
- The `if` statement checks if `num` is **less** than 0. If the condition is **true**, it executes the indented block of code under the `if` statement.
- If the condition in the `if` statement is **false**, the program moves to the `elif` statement.
- The `elif` statement checks if num is **equal** to 0. If the condition is **true**, it executes the indented block of code under the `elif` statement.
- If both the `if` and `elif` conditions are **false**, the program moves to the `else` block and executes the code within it.
- In this case, the code within the `else` block is **executed**, which prints the message "The number is positive."

Output When `num` is 50:

```
The number is positive.
```

Since the value of `num` (50) is **greater** than 0, the conditions in both the `if` and `elif` statements are **false**. Hence, the code block within the `else` statement is **executed**, resulting in the message "The number is positive." being printed as the output.

Output When `num` is -25:

```
The number is negative.
```

Since the value of `num` (-25) is **less** than 0, the condition in the `if` statement is **true**. Therefore, the code block within the `if` statement is **executed**, resulting in the message "The number is negative." being printed as the output.

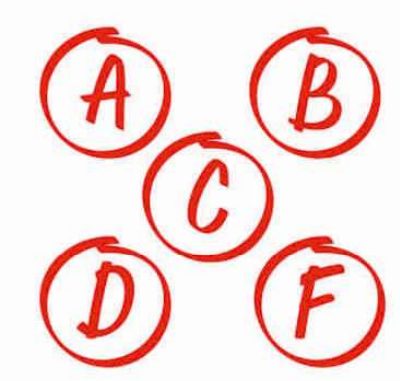
Output When `num` is 0:

```
The number is zero.
```

Since the value of `num` is 0, the condition in the `elif` statement (num == 0) is **true**. Therefore, the code block within the `elif` statement is executed, resulting in the message "The number is zero." being printed as the output.

In summary, the code uses the `if`, `elif`, and `else` statements to check different conditions and execute the corresponding code block based on the **true** condition. The specific output depends on the value assigned to `num`.

More with Grade Example



Program to determine the grade based on the student's score

```python
# Program to determine the grade based on the student's score

# Input the student's score
score = int(input("Enter the student's score: "))

# Determine the grade based on the score
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

# Print the grade
print("The student's grade is:", grade)
```

Code snippet explanation:

- In this example, we begin by providing a comment (#) to describe the purpose of the program.
- The program **prompts** the user to **enter the student's score** using the `input()` function. The `int()` function is used to **convert** the input to an integer.
- The program then uses `if`, `elif`, and `else` statements to determine the grade based on the score. Each condition checks the score against a specific range of values and **assigns** the appropriate grade to the variable `grade`.
- Finally, the program **prints** the grade using the `print()` statement.

Output and explanation: Let's consider different input values and their corresponding outputs:

Output When the input score is 95:

```
The student's grade is: A
```

Since the score (95) is **greater** than or **equal** to 90, the condition in the **first** if statement is **true**. Therefore, the grade "**A**" is assigned to the variable grade, and it is **printed** as the output.

Output When the input score is 82:

```
The student's grade is: B
```

Since the score (82) is **greater** than or **equal** to 80 but **less** than 90, the condition in the **second** if statement is **true**. Therefore, the grade "**B**" is assigned to the variable grade, and it is **printed** as the output.

Output When the input score is 70:

```
The student's grade is: C
```

Since the score (70) is **greater** than or **equal** to 70 but **less** than 80, the condition in the **third** if statement is **true**. Therefore, the grade "**C**" is assigned to the variable grade, and it is **printed** as the output.

Output When the input score is 59:

```
The student's grade is: F
```

Since the score (59) is **less** than 60, **none** of the conditions in the if and elif statements are **true**. Hence, the code within the else block is **executed**, and the grade "**F**" is assigned to the variable grade, which is then **printed** as the output.

Practice Program

1. Program to check if a given year is a leap year or not. Any year is leap year
    a. The year must be divisible by 4.
    b. If the year is divisible by 100, it must also be divisible by 400.
        i. For example – 2020 is a leap year
        ii. For example – 2000 is a leap year
        iii. For example – 2016 is a leap year
        iv. For example – 2017 is not a leap year

2. Program to determine the eligibility to drive based on the user's age.
    a. Valid age for driving a car is 18 years

3. Program to check if a given number is prime or not.
    a. Prime number is a number which is only divisible by 1 and itself
        i. For example – 2 & 3 are prime numbers
        ii. But 4, 6 & 9 are not prime numbers since they are divisible by 2 or 3.

4. Program to find the largest among three numbers.
    a. Ask user to enter 3 numbers and return which is the largest.
        i. For example, if user enters 11, 5 & 19. It should return 19.

5. Program to check if a given number is even or odd.
    a. If a given number is fully divisible by 2 then it's even else it's an odd number
    b. For example – 4 is even number and 11 is odd number