Day 4- Facilitation Guide
Conditional Constructs

Index

(1.5 hrs) ILT

I. Recap

In our last session we learned:

- **Programming Statements:**Programming statements, also known as code statements or simply statements, are the fundamental building blocks of computer programs. They are the individual instructions or commands that make up a program's logic.
- **User Input:**How to accept the value from the user
- **Comments in Program:** What is comments and when and how to use comments. Also we try to understand single line and multiline comments

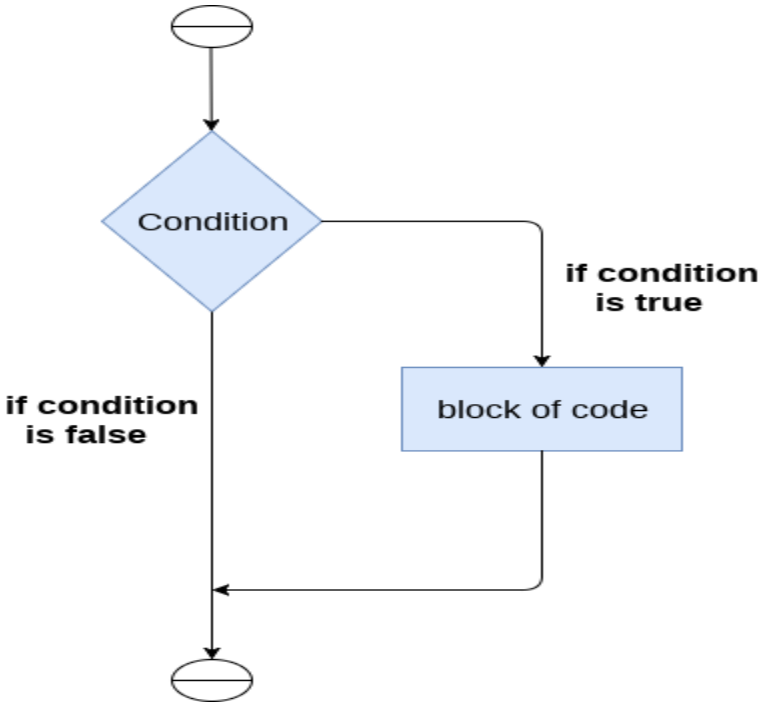
In this session we are going to learn about Conditional Constructs,if Statement,Ladder if else and Nested conditions and their uses.

II. Conditional Constructs

- There are situations where an action needs to be performed based on a condition. This is known as conditional execution.
- The various conditional constructs are implemented using
    - if statement
    - if…else statement
    - Ladder conditions
    - Nested conditions

III. if statement:

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



The syntax of the if-statement is given below.

if expression:

    # statement

Example:

```
x = 10
if x > 5:
    print("x is greater than 5")
```

Another example:

Accept a number from the user and check the number is even

You can use the input() function to accept a number from the user and then check if it's even by using the modulo operator %.

```
# Accept a number from the user
user_input = input("Enter a number: ")

# Convert the user input to an integer
number = int(user_input)

# Check if the number is even
if number % 2 == 0:
    print(f"{number} is an even number.")
```

In this code:

We use the input() function to get user input as a string.

We convert the user input to an integer using int() so that we can perform mathematical operations on it.

We use the modulo operator % to check if the number is even. If number % 2 equals 0, it means the number is even.

We display the result accordingly using print().

if-else statement:

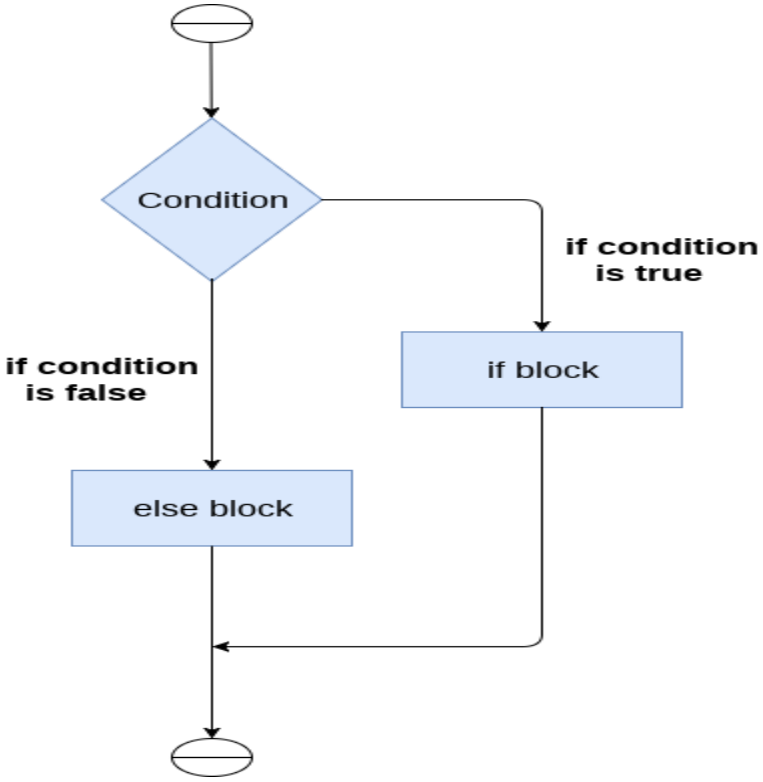
The if-else statement in Python is used to make decisions in your code. It allows you to specify two blocks of code: one to execute if a certain condition is true (if block), and another to execute if the condition is false (else block).

Here's the basic syntax:

if condition:

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

- The if clause contains a condition that evaluates to either True or False.
- If the condition is True, the code within the if block is executed.
- If the condition is False, the code within the else block is executed.



Here's an example:

```
age = 20
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")
```

Another example:

Accept a number from the user and check the number is even or odd

You can use the input() function to accept a number from the user and then check if it's even by using the modulo operator %.

```
# Accept a number from the user
user_input = input("Enter a number: ")

# Convert the user input to an integer
number = int(user_input)

# Check if the number is even
if number % 2 == 0:
    print(f"{number} is an even number.")
else:
    print(f"{number} is not an even number.")
```

In this code:

We use the input() function to get user input as a string.

We convert the user input to an integer using int() so that we can perform mathematical operations on it.

We use the modulo operator % to check if the number is even. If number % 2 equals 0, it means the number is even; otherwise, it's not.

We display the result accordingly using print().

Another example:

Accept a alphabet from the user and check that alphabet is vowel or consonant

```
# Accept a single alphabet character from the user
alphabet = input("Enter a single alphabet character: ")
    # Check if the alphabet is a vowel or a consonant using a ladder of if-elif-else
statements
```

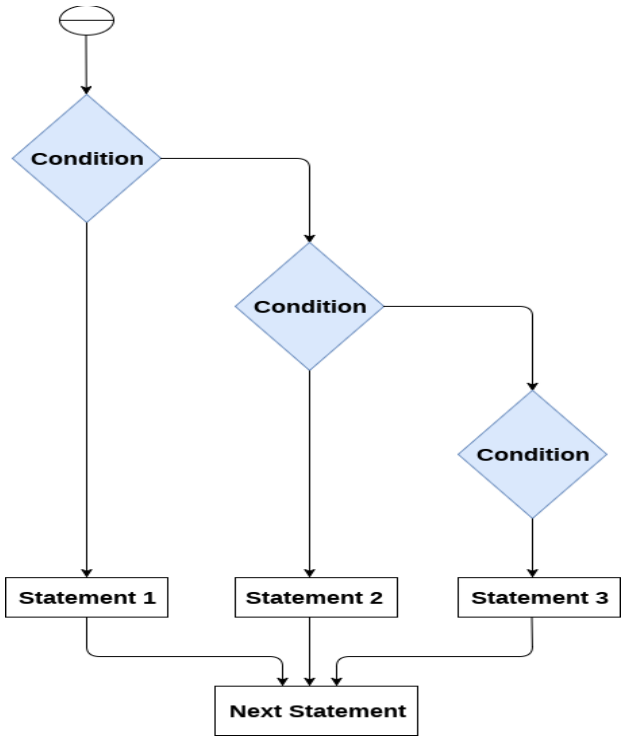
```
if alphabet == 'a' or alphabet == 'e' or alphabet == 'i' or alphabet == 'o' or alphabet == 'u':
    print(f"{alphabet} is a vowel.")
else:
     print(f"{alphabet} is a consonant.")
```

In this code, we first check if the user entered a single alphabet character as explained in the previous example. Then, we use if-else statements to check if the alphabet is a vowel ('a', 'e', 'i', 'o', 'u') or a consonant.

IV. Ladder if else:

A "ladder" of if-elif-else statements is a programming construct used when you have multiple conditions to check sequentially. Each condition is checked in order, and the first condition that evaluates to True triggers the corresponding block of code. If none of the conditions is True, the else block (if provided) is executed. Here's the basic structure:

```
if condition1:
    # Code to execute if condition1 is true
elif condition2:
    # Code to execute if condition1 is false and condition2 is true
elif condition3:
    # Code to execute if condition1 and condition2 are false, and condition is true
else:
    # Code to execute if all conditions are false
```



You can have any number of elif clauses, and they are checked in order from top to bottom. Once a condition is met, the corresponding block is executed, and the remaining conditions are not evaluated.

Here's an example:

```
x = 10
if x < 5:
    print("x is less than 5.")
elif x < 10:
    print("x is less than 10 but not 5.")
elif x == 10:
    print("x is exactly 10.")
else:
    print("x is greater than 10.")
```

In this example, Python checks the conditions sequentially. Since **x** is equal to 10, the third condition (x == 10) is True, and the corresponding block is executed. The other conditions are not evaluated.

Another example:

A school has following rules for grading system:
a. Below 25 - F
b. 25 to 45 - E
c. 45 to 50 - D
d. 50 to 60 - C
e. 60 to 80 - B
f. Above 80 - A
Ask users to enter marks and print the corresponding grade.

```
marks=int(input("Enter marks"))
if marks<25:
  print("F")
elif marks>=25 and marks<45:
  print("E")
elif marks>=45 and marks<50:
  print("D")
elif marks>=50 and marks<60:
  print("C")
elif marks>=60 and marks<80:
  print("B")
else:
  print("A")
```

In this code:

We use the input() function to get the user's input as a string.

We use a series of if-elif-else statements to determine the grade based on the score. You can adjust the grade cutoffs as needed.

Finally, we display the determined grade to the user.

V. Nested conditions:

Nested conditions refer to situations where you have conditional statements (e.g., if, elif, else) inside other conditional statements. This allows you to create complex decision-making logic that depends on multiple conditions. In Python, you can nest conditional statements by placing one inside another. Here's an example:

```
x = 10
y = 5
if x > 5:
    print("x is greater than 5.")
    if y > 2:
        print("y is greater than 2.")
    else:
        print("y is not greater than 2.")
else:
    print("x is not greater than 5.")
```

In this example:

The outer if statement checks if x is greater than 5.

If x is greater than 5, it executes the first block of code and proceeds to the inner if statement.

If x is not greater than 5, it executes the else block.

The inner if statement checks if y is greater than 2.

If y is greater than 2, it prints "y is greater than 2."

If y is not greater than 2, it prints "y is not greater than 2."

The key point is that the execution of nested conditional statements depends on the conditions of both the outer and inner statements. You can nest conditions to create

more intricate decision-making processes, and you can nest multiple levels deep if needed.

Here's a more complex example:

```
grade = 75
if grade >= 90:
    print("A")
elif grade >= 80:
    print("B")
else:
    print("C")
    if grade >= 70:
        print("Pass")
    else:
        print("Fail")
```

In this example:

The first if statement checks for an "A" grade.

The first elif statement checks for a "B" grade if the "A" condition is not met.

The else block prints "C" and contains a nested if-else block that checks for a passing or failing grade.

Nested conditions are useful for handling complex decision trees where the outcome depends on a combination of conditions. However, as the level of nesting increases, code readability can become a concern, so it's important to strike a balance between complexity and maintainability.

Exercise ChatGPT

1. A company decided to give a bonus of 5% to an employee if his/her year of service is more than 5 years .Ask users for their salary and year of service and print the net bonus amount.
2. A shop will give a discount of 10% if the cost of purchased quantity is more than 1000.Ask user for quantity Suppose, one unit will cost 100.Judge and print total cost for user.