CBS Department

Selection Control:
IF Statement,
MATCH CASE Statement

Soma Soleiman Zadeh

Object-Oriented Programming (CBS 215)

Fall 2025 - 2026

Week 2

October 15-16, 2025

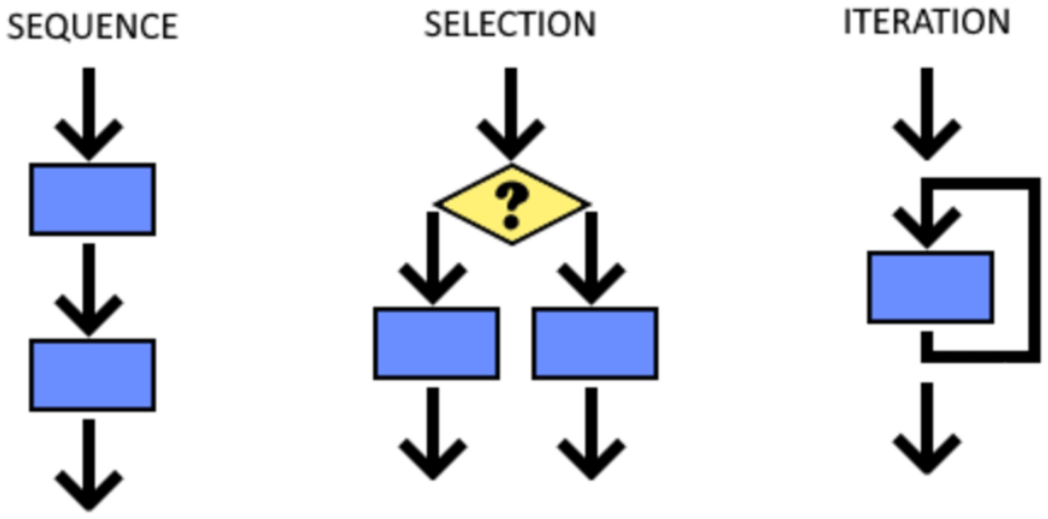
Outline

- Control Statements
- Boolean Expressions and Relational Operators
- Membership Operators
- **Logical** Operators
- **IF-ELSE** Statement (Two-Way Decision)
- **IF-ELIF-ELSE** Statement (Multi-Way Decision)
- **MATCH-CASE** Statement

Control Statements

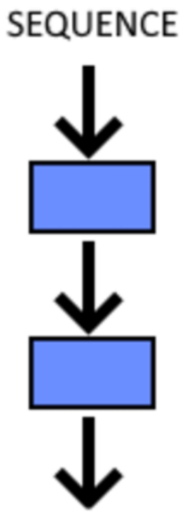
- A **control statement** is a statement that determines the control flow of a set of instructions.
- There are **three forms of control** that programming languages provide:
    - **Sequential** control
    - **Selection** control
    - **Iterative** control

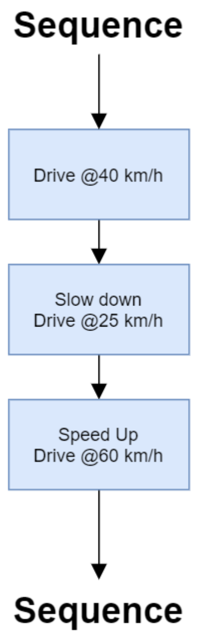
Tree Forms of Control in Programming



Sequential Control

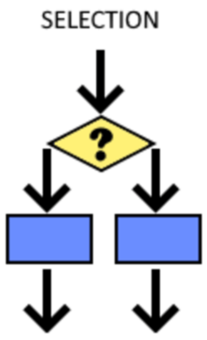
- **Sequential** means "in sequence" or "one-after-the-other".
- All statements are in the order that we want them to be executed, and the program executes them in sequence from the **Start** statement to the **End** statement.

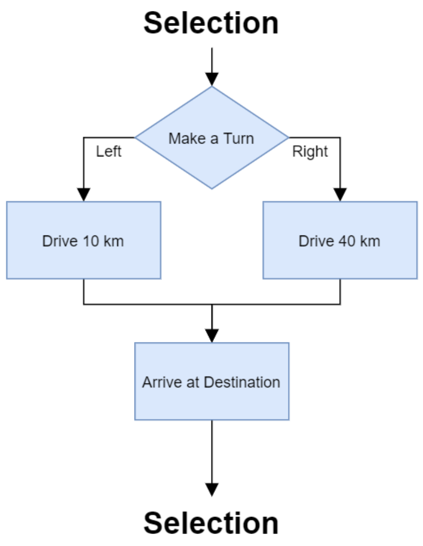




Selection Control

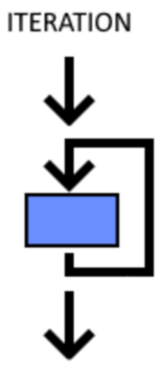
- A **selection control** allows you to make decisions in your code about the current state of your program, and then to choose one of two choices leading to the next statement.

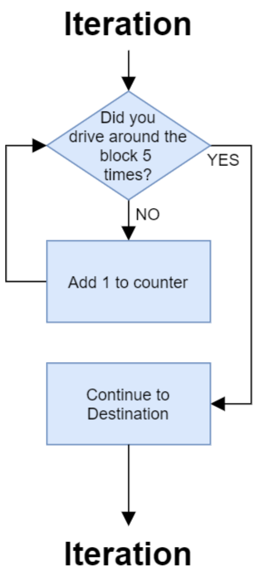




Iterative Control

- An **iterative control** statement executes a sequence of statements multiple times, based on a condition or set of conditions.





Boolean Expressions

- We already mentioned the **Boolean** data type as one of the basic data types. The **Boolean** data type **contains two Boolean values**, denoted as **True** and **False** in Python.

- A **Boolean expression** is an expression **that has a Boolean value**.
- **Boolean expressions** are used to represent the conditions for **selection** and **iterative** control statements.

Relational Operators in Python

- The **relational operators (comparison operators)** in Python perform the usual **comparison operations**.

- Relational expressions are a type of **Boolean expression** since they have a **Boolean** result.

Relational Operators (Comparison Operators)

Operator	Description	Example	Example's Result
<	Less than	6 < 4	False
>	Greater than	9 > 5	True
<=	Less than or equal to	8 <= 12	True
>=	Greater than or equal	10 >= 15	False
= =	Equal to	7 == 9	False
!=	Not equal to	7 != 9	True

Let's Try it!

- What is the result value of each relational expression?

10 == 20 → False

10 != 20 → True

10 <= 20 → True

Membership Operators

- **Membership operators** are used to check whether a value or variable exists in a sequence or not.
- There are two membership operators: **in , not in**

Membership Operators	Examples	Result
in	10 **in** (10, 20, 30)	True
	'red' **in** ('red', 'green', 'blue')	True
not in	10 **not in** (10, 20, 30)	False

Membership Operators

- The membership operators can also be used to check if a given character or string occurs within another string,

‘Good’ **in** ‘Good Morning’ → True
‘M’ **in** ‘Good Morning’ → True
‘m’ in ‘Good Morning’ → **False**

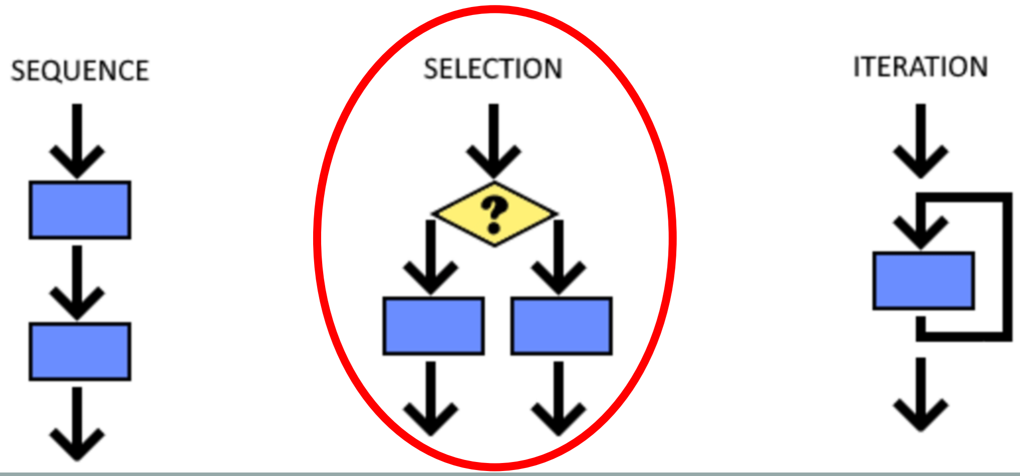
- The **membership operators**, like relational operators, can be used to construct Boolean expressions.

Let’s Try it!

- What is the result value of each expression?

10 **in** (40 , 20 , 10) → True
10 **not in** (40 , 20 , 10) → **False**
grade = ‘A’
grade **in** (‘A’ , ‘B’ , ‘C’ , ‘D’) → True
city = ‘Zaxo’
city **not in** (‘Erbil’ , ‘Sulaymaniah’ , ‘Duhok’) → True

SELECTION Control Structure



SELECTION Statements

- There are two main **SELECTION** statements in Python:
  - **IF** Statement
    - ➢ **If**
    - ➢ **If-Else**
    - ➢ **If-Elif-…-Else**
  - **Match Case** Statement
- Both **IF** and **Match Case** are **<u>conditional statements</u>** in Python.

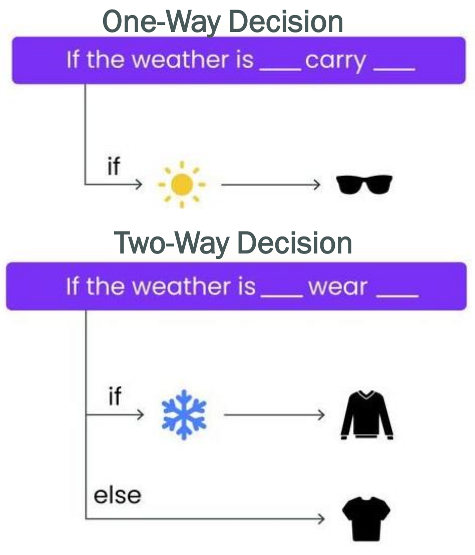
IF Statement

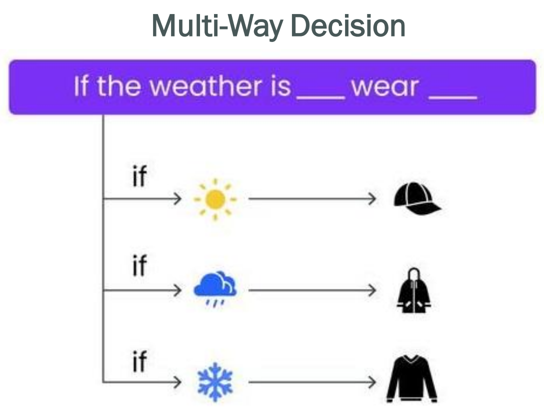
- IF statement is used to **make a decision**.
- **IF** statement: If a condition (or conditions) is True, it executes a set of commands (The If block statements).
  - Otherwise, the set of commands is skipped.

```
if    condition :
          execute statements
```

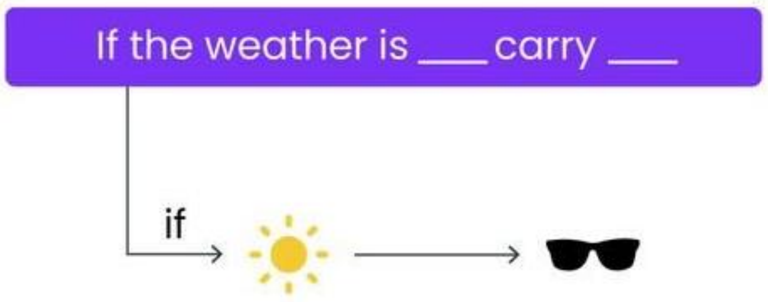
Indentation

One-Way, Two-Way, Multi-Way Decision


One-Way Decision
If the weather is ___ carry ___
if
Two-Way Decision
If the weather is ___ wear ___
if
else



IF Statement (One-Way Decision)



IF Statement

```
temperature = -10

if (temperature<0) :
    print("It is below freezing point!")
```

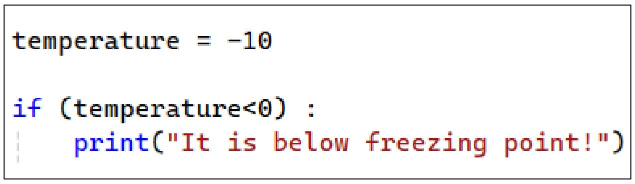
Output

```
It is below freezing point!
```

Indentation

- **Indentation** is whitespaces at the beginning of a line in a Python code.
- **Indentation** defines scope in code. For example, in **if** statement, we use indentation to define the scope of **if** statement.
- Other programming languages may use curly brackets for this purpose.

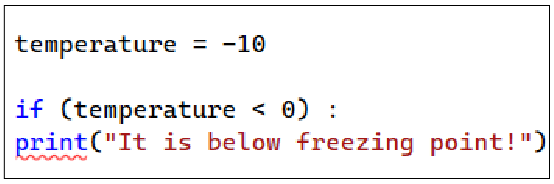
✓



```
temperature = -10

if (temperature<0) :
    print("It is below freezing point!")
```

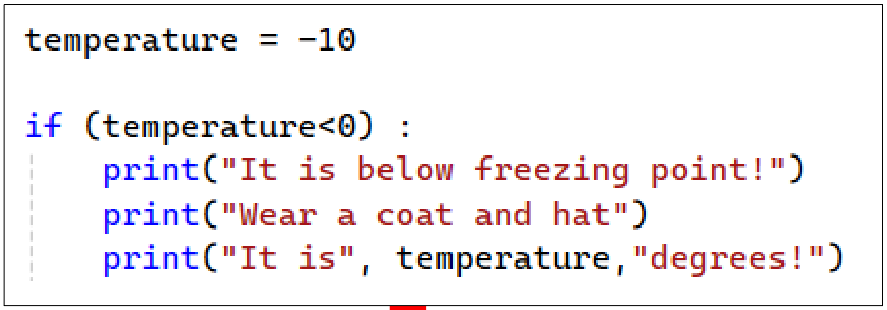
✗



```
temperature = -10

if (temperature < 0) :
print("It is below freezing point!")
```

IF Statement



```
temperature = -10

if (temperature<0) :
    print("It is below freezing point!")
    print("Wear a coat and hat")
    print("It is", temperature,"degrees!")
```

Output

```
It is below freezing point!
Wear a coat and hat
It is -10 degrees!
```

Multiple IF Statements

```
temperature = 45
if (temperature>40) :
    print("Temperature is > 40")

if (temperature>20) :
    print("Temperature is > 20")

if (temperature>0) :
    print("Temperature is above freezing point!")
```

Output

```
Temperature is >40
Temperature is >20
Temperature is above freezing point!
```

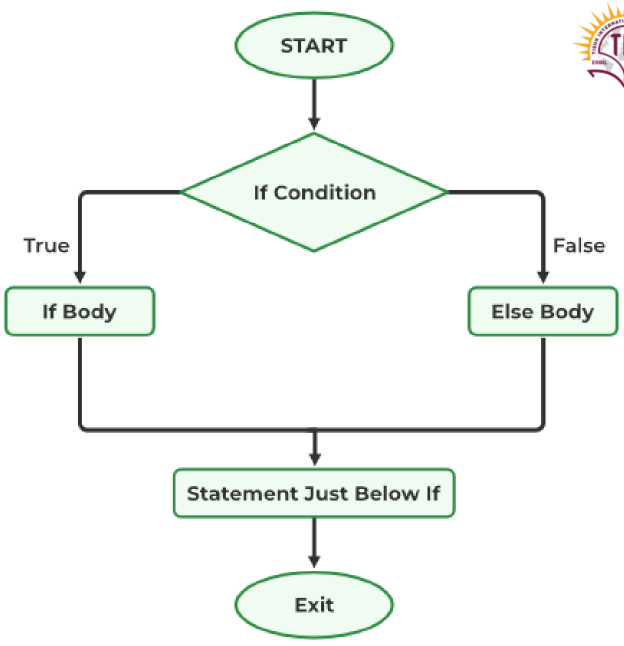
Forming Simple Conditions

- Relational expressions using relational operators are used as conditions in **IF** statement.
  - Equals → a = = b
  - Not Equals → a **!=** b
  - Less than → a **<** b
  - Less than or Equal to → a **<=** b
  - Greater than → a **>** b
  - Greater than or Equal to → a **>=** b

```
a = 7
b = 10
if a < b :
    print('a is less than b')
```

IF Flowchart

(Two-Way Decisions)



IF-ELSE Statement (Two Way Decisions)

- A two-way decision can be implemented by attaching an **else** clause to an **if** clause.
- The **else** keyword is **optional**.
- The **else** keyword is used to decide what to do if the condition is **False**.

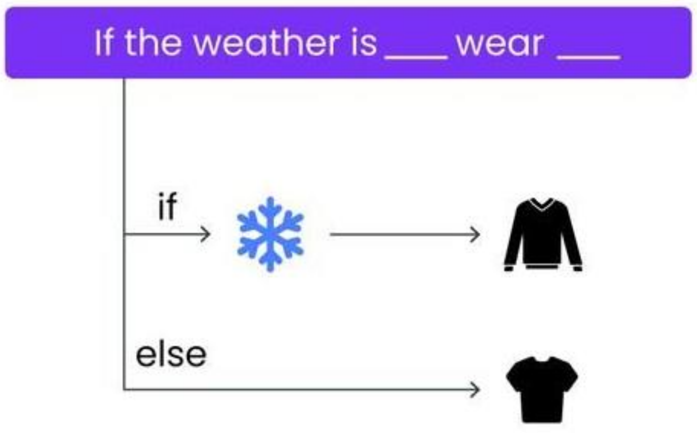
```
if   condition :
          execute statements
else :
          execute statements
```

IF-ELSE Statement (Two Way Decisions)

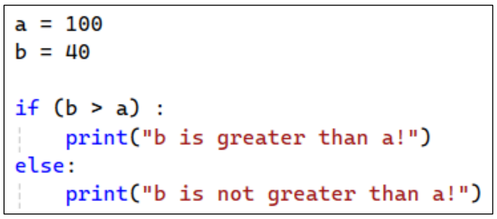
- If the condition is **True**, the **if block is executed**, and the else block is skipped.
- If the condition is **False**, the **else block is executed**, and the if block is skipped.

```
if    condition :
          statement1
          statement2        <------    if Block
          ...
else :
          statement1
          statement2        <------    else Block
          ...
```

If-Else Statement (Two-Way Decision)



IF-ELSE Statement

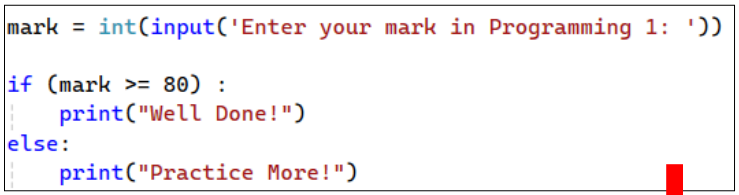


```
a = 100
b = 40

if (b > a) :
    print("b is greater than a!")
else:
    print("b is not greater than a!")
```

Output

```
b is not greater than a!
```



```
mark = int(input('Enter your mark in Programming 1: '))

if (mark >= 80) :
    print("Well Done!")
else:
    print("Practice More!")
```

Output if the user enters 60 as their mark

```
Practice More!
```

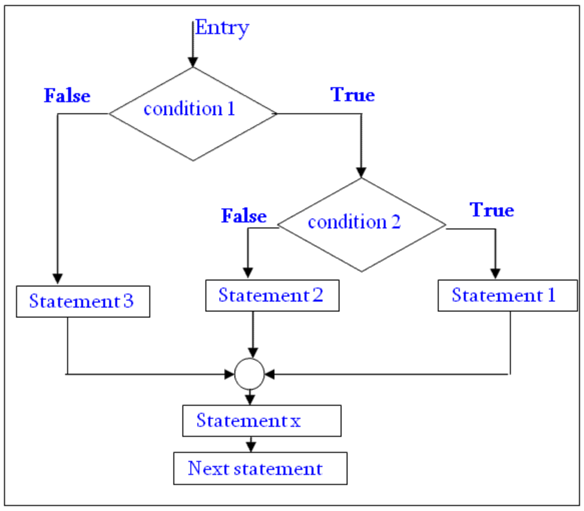
Nested IF Statement

◦ You can have **IF** statement inside another **IF** statement, which is called **Nested IF** statement.

```
if condition :
    if condition :
        statement1
    else:
        statement2
else :
    statement3
```

Nested **if-else** statement

Flowchart of Nested IF



```
if  condition1 :
        if  condition2 :
                Statement1
        else :
                Statement2
else :
        Statement3
```

Examples of Nested IF Statements

```python
num = 50

if (num > 10) :
    print("Above ten, ")

    if (num > 30) :
        print("and also above 30!")
    else:
        print("but not above 30!")
```

Output

```
Above ten,
and also above 30!
```

```python
quiz1 = int(input('Enter mark of first quiz: '))
quiz2 = int(input('Enter mark of second quiz: '))
quiz3 = int(input('Enter mark of third quiz: '))

average = (quiz1 + quiz2 + quiz3) / 3

if (average < 50 ):
    print('Failed!')
else:
    if (average < 80):
        print ('Nice! but you need to practice more!')
    else:
        print ('Well Done!')
```

Output if the user enters 70, 90, 80 as their quiz marks

```
Well Done!
```

```python
a = 5
b = 10

if(a >= 5):
    if(b != 10):
        print ("Option A")
    else:
        print ("Option B")

else:
    if(b < 11):
        print ("Option C")
    else:
        print ("Option D")
```

Output →

```
Option B
```

- First **if** statement is evaluated → **if (a >=5)**
- The condition **(a >=5)** is **True**, so the <u>if block is executed</u> and the else block is skipped.
- Now the second **if** statement is evaluated
  → if (b != 10)
- The condition **(b != 10)** is **False**, so <u>the else block is executed</u> and the if block is skipped.

```python
a = 4
b = 10

if(a >= 5):
    if(b != 10):
        print ("Option A")
    else:
        print ("Option B")

else:
    if(b < 11):
        print ("Option C")
    else:
        print ("Option D")
```

Output →

```
Option C
```

- First **if** statement is evaluated → **if (a >=5)**
- The condition **(a >=5)** is **False**, so the <u>else block is executed</u> and the if block is skipped.
- Now the second **if** statement is evaluated
  → if (b < 11)
- The condition **(b < 11)** is **True**, so <u>the if block is executed</u> and the else block is skipped.

Logical Operators

- Logical operators connect two or more conditions (relational expressions) into one or reverse the logic of a condition.

Operator	Description	Example	Example's Result
AND	Returns **True** <u>if both statements are **True**</u>, otherwise it returns **False**.	2 <= 5 AND 9 > 3	True
		10 > 20 AND 20 < 30	False
OR	Returns **True** if at least one of the statements is **True**. It returns **False** only if both statements are **False**.	2 < 5 OR 6 >= 9	True
		8 <= 5 OR 7 > 9	False
NOT	Reverse the result. It returns **False** if the statement is **True**, and returns **True** if the statement is **False**.	NOT (10 > = 5)	False
		NOT (10 < 6)	True

Let's Try it!

- What is the result value of each expression?

(5 < 2) AND (5 > 3)	→	False
(5 < 9) AND (5 > 3)	→	True
(5 < 2) OR (5 >= 3)	→	True
(5 < 2) OR (5 > 10)	→	False
NOT (5 > 10)	→	True
NOT (5 <= 10)	→	False

IF Statement Using Logical Operators

```python
a = 100
b = 50
c = 400

if (a > b) and (c > a):
    print('Both conditions are True!')
else:
    print("At least one of the conditions is False!")
```

Output

```
Both conditions are True!
```

IF Statement Using Logical Operators

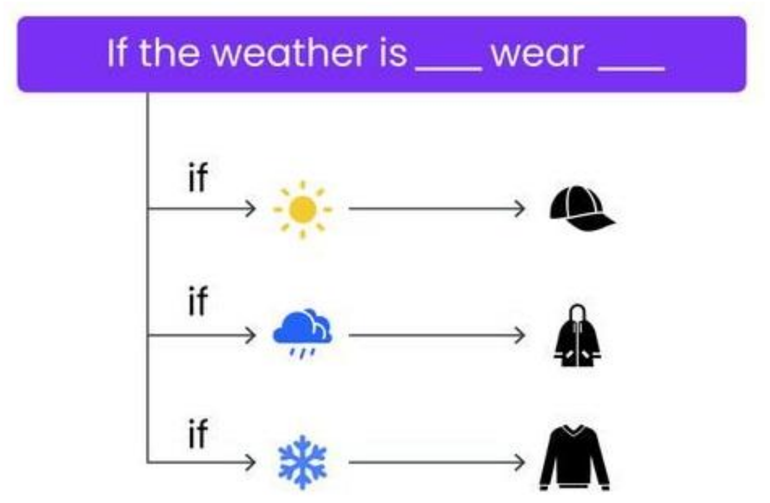
```python
a = 100
b = 50
c = 400

if (a < b) or not(c >= b):
    print('At least one of the conditions is True!')
else:
    print("Both conditions are False!")
```

Output

```
Both conditions are False!
```

IF-ELIF-ELSE Statement (Multi-Way Decision)



Multi-Way Decisions

- Imagine if you need to make a four-way. Which way is better?

Nested IF

```
if  (condition1) :
        statements1
else :
        if  (condition2) :
                statements2
        else :
                if  (condition3) :
                        statements3
                else :
                        statements4
```

IF-ELIF-ELSE

```
if    (condition1) :
          statements1
elif  (condition2) :
          statements2
elif  (condition3) :
          statements3
else :
          statements4
```

if-elif-else Statement

- **if...else** statement is used when two-way decision.
- If we must choose between more than two selections, we use the **if...elif...else** statement.

```
if      (condition1) :
                execute statements
elif    (condition2) :
                execute statements
else :
                execute statements
```

if-elif-else Statement (Examples)

```
x = 10
y = 20


if x < y:
    print ('x is less than y')

elif x > y:
    print ('x is greater than y')

else:
    print ('x and y are equal')
```

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

if x > 0:
    print (x, 'is positive.')
elif x < 0:
    print (x, 'is negative.')
else:
    print (x, 'is zero.')
```

if-elif-else

- You can have as many **elif** clauses as you need.

```
if      (condition1) :
                execute statements
elif    (condition2) :
                execute statements
elif    (condition3) :
                execute statements
...

else :
                execute statements
```

if-elif-else Statement

- Is it possible to have **if** and **elif** clauses without **else** clause?
    - Yes, but it is recommended to use an **else** clause to handle the cases when none of the conditions are True.

```
if      (condition1) :
                execute statements
elif    (condition2) :
                execute statements
elif    (condition3) :
                execute statements
```

if-elif-else Statement (Example)

- A program to check the **temperature** and decide on the **weather**.

```python
temperature = 40

if temperature < 0:
    status = "freezing"
elif temperature < 10:
    status = "cold"
elif temperature < 20:
    status = "mild"
elif temperature < 30:
    status = "warm"
else:
    status = "hot"

print("Today's weather is", status)
```

Output

```
Today's weather is hot
```

What is the Difference?

```python
temperature = 45

if (temperature > 40):
    print("Weather is hot.")

if (temperature > 20):
    print("Weather is mild.")

if (temperature > 0):
    print("Weather is cold.")
```

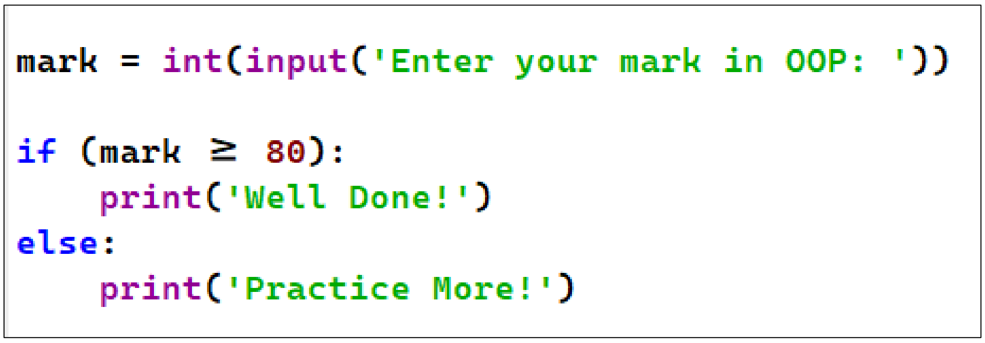
```python
temperature = 45

if (temperature > 40):
    print("Weather is hot.")

elif (temperature > 20):
    print("Weather is mild.")

elif (temperature > 0):
    print("Weather is cold.")
```

IF-ELSE Statement



```
mark = int(input('Enter your mark in OOP: '))

if (mark ≥ 80):
    print('Well Done!')
else:
    print('Practice More!')
```

Output if the user enters 60 as their mark

```
Practice More!
```

Let’s Try it More!

Change the code in a way that:

- If the student’s **mark is between 80 and 100**,
  - the program prints **“Well Done!”**
- The If student’s **mark is between 70 and 80**,
  - the program prints **“Good! But you need more practice!”**
- If the student’s **mark is between 50 and 70**,
  - the program prints **“Study harder!”**
- If the student’s **mark is between 0 and 50**,
  - the program prints **“Failed!”**

Code!

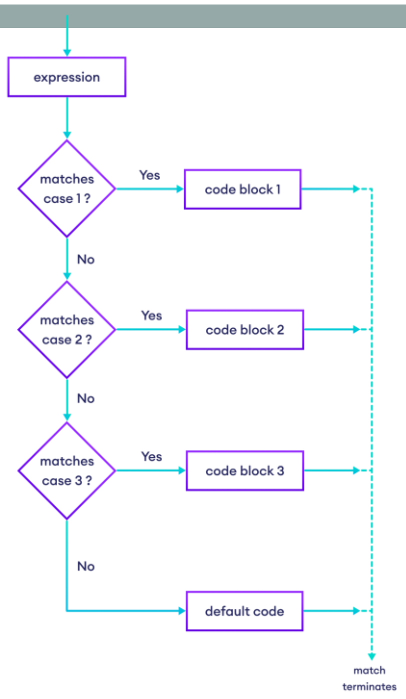
```
mark = int(input('Enter your mark in OOP: '))

if (80 <= mark <= 100):
    print('Well Done!')
elif (70 <= mark < 80):
    print('Good! But you need more practice!')
elif (50 <= mark < 70):
    print('Study Harder!')
elif (0 <= mark < 50):
    print('Failed!')
else:
    print('The entered mark is not valid!')
```

MATCH-CASE Statement

- A **MATCH CASE** statement is a conditional statement that **compares the result of an expression with different patterns** and runs the code linked to the first pattern that fits.
- With the **MATCH CASE** statement, you control what parts of code are executed if conditions are met.
- **MATCH CASE** is similar to the **SWITCH-CASE** statement in other programming languages, but with enhanced capabilities.

Flow Chart of Match-Case



MATCH-CASE Statement Syntax

```
match   expression :
        case   value1 :
               code to execute for value1
        case   value2 :
               code to execute for value2
        case   value3 :
               code to execute for value3
        case   _ :
               default code to execute
```

IF-ELIF-ELSE

```
day = “Thursday”
if   (day == “Friday”) :
        print(“Today is weekend!”)
elif   (day == “Saturday”) :
        print(“Today is weekend!”)
else :
        print(“Today is a weekday!”)
```

MATCH-CASE

```
day = “Thursday”
match   day :
        case   “Friday” :
                print(“Today is weekend!”)
        case   “Saturday” :
                print(“Today is weekend!”)
        case   _ :
                print(“Today is a weekday!”)
```

Let’s Change the Code!

IF Statement

```
day = “Thursday”
if   (day == “Friday”) or (day == “Saturday”)
:
        print(“Today is weekend!”)
else :
        print(“Today is a weekday!”)
```

MATCH-CASE Statement

```
day = “Thursday”
match   day :
        case   “Friday” | “Saturday” :
                print(“Today is weekend!”)
        case   _ :
                print(“Today is a weekday!”)
```

Classwork – OCT 16, 2025

Ask the user to enter a **country name**, and the code outputs the **capital city**.

- By using an **IF** statement,
- By using the **MATCH CASE** statement.

Country	Capital
Netherlands	Amsterdam
France	Paris
UK	London
Germany	Berlin