**5. Membership Operators:**

in : Checks if a value is present in a sequence (e.g., list, tuple, string)

not in : Checks if a value is not present in a sequence

**6. Identity Operators:**

is : Checks if two objects are the same object (memory location)

is not : Checks if two objects are not the same object



**Python has several built-in data type:**

**Numeric Types:**

int: Represents integers (whole numbers), e.g., 1, 2, -10.

float: Represents floating-point numbers (numbers with decimal points), e.g., 3.14, -2.7.

complex: Represents complex numbers, e.g., 2 + 3j.

# Type of if else statement



Python offers several variations of the if-else statement:

**1.If condition:**

```
    # code to execute if the condition is true
```

**2. if-else statement:**

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

**3. if-elif-else statement:**

```
If condition1:
    # code to execute if condition1 is true
elif condition2:
    # code to execute if condition2 is true
else:
    # code to execute if none of the above conditions are true
```

# Explanation



The outer if statement checks if x is greater than 5.
If the first condition is true, the inner if statement checks if x is greater than 8.
If the second condition is also true, the code inside the inner if block executes.
If the second condition is false, the code inside the inner else block executes.

**Another example with elif:**

```
Num = 15


if num > 0:
    print(“Number is positive”)
    if num % 2 == 0:
        print(“Number is even”)
    else:
        print(“Number is odd”)
else:
    print(“Number is not positive”)
```

**Example**

```
X = 10


if x > 15:
    print("x is greater than 15")
elif x > 5:
    print("x is greater than 5 but less than or equal to 15")
else:
    print("x is less than or equal to 5")
```

**Output**

```
x is greater than 5 but less than or equal to 15
```

# Key points



You can have multiple elif statements following an if statement.
Python will evaluate the conditions in order, and execute the code block associated with the first true condition.
If none of the conditions are true, the else block will execute, if present.
If you only need to check one condition, you can use a simple if statement.
If you need to check multiple conditions, and execute different code for each, elif is a powerful tool.

**Key Points:**

Iterable:

The object you want to loop through.

Item:

A variable that takes on the value of each element in the sequence during the loop.

Code Block:

The indented code that is executed for each item in the sequence.

# Nested for loop statement



In Python, a nested for loop is a loop that is placed inside another loop. The outer loop iterates over a sequence, and for each iteration, the inner loop executes completely.

**Syntex:**

```
For outer_variable in outer_sequence:
    # Code to execute in the outer loop
    for inner_variable in inner_sequence:
        # Code to execute in the inner loop
```

**Example**

```
For i in range(1, 4):  # Outer loop
    for j in range(1, 4):  # Inner loop
        print(i, j)
```

**Explanation**:

def: The keyword used to define a function.

Function_name: A unique name to identify the function.

Parameters (optional): Input values that the function can accept.

Docstring (optional): A brief description of the function's purpose.

Code to be executed: The body of the function containing the instructions to perform the task.

Return (optional): The keyword used to return a value from the function.



**Example:**

```
Def greet(name):
    """This function greets the user."""
    print("Hello,", name)


greet("Alice")  # Output: Hello, Alice
```

**Built-in Polymorphism:**
Many built-in functions and operators in Python exhibit polymorphic behavior. For example, the len() function works on strings, lists, tuples, and other sequence types.

**Example of polymorphism using method overriding:**

```
Class Animal:
    def speak(self):
        pass


class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"


 def animal_sound(animal):
    print(animal.speak())


dog = Dog()
cat = Cat()


 animal_sound(dog)  # Output: Woof!
 Animal_sound(cat)  # Output: Meow!
```

Thank you