

# LAB#02



Artificial Intelligence  
Abdul Haseeb

# Lab Objectives

if -else

if-else-if

Logical Operators

Loops

- For loop
- While loop

Lists in Python



# if-else

```
if condition:  
    # Code to execute if condition is True  
else:  
    # Code to execute if condition is False
```

# if-else

python

```
temperature = 30
```

```
if temperature > 25:
```

```
    print("It's hot outside.")
```

```
else:
```

```
    print("It's not too hot outside.")
```

# if-else-if

```
if condition1:
    # Code to execute if condition1 is True
elif condition2:
    # Code to execute if condition2 is True
elif condition3:
    # Code to execute if condition3 is True
else:
    # Code to execute if none of the above conditions are True
```


# if-else-if

```
score = 85

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: D or lower")
```

# Logical Operators

Logical operators are used to combine conditional statements and evaluate multiple conditions



Python provides three main logical operators:

**and**

**or**

**not**

# And Operator

- The and operator returns True if both operands (conditions) are true. If either operand is false, it returns False

```
condition1 and condition2
```



# And Operator

```
age = 25
has_ticket = True

if age >= 18 and has_ticket:
    print("You can enter the movie.")
else:
    print("You cannot enter the movie.")
```

# Or Operator

- The or operator returns True if at least one of the operands (conditions) is true. If both operands are false, it returns False

```
condition1 or condition2
```

# Or Operator

```
rainy = False
snowy = True

if rainy or snowy:
    print("It's a bad weather day.")
else:
    print("The weather is fine.")
```

# Not Operator

- The not operator inverts the Boolean value of its operand.

```
not condition
```

# Not Operator

```
is_open = False

if not is_open:
    print("The store is closed.")
else:
    print("The store is open.")
```



# Key points

and: Returns True only if all conditions are True.

or: Returns True if at least one condition is True.

not: Returns the inverse Boolean value of the condition (negates the condition)

# While Loop

---



A while loop in Python repeatedly executes a block of code as long as a specified condition remains true.



It checks the condition before each iteration, and if the condition evaluates to True, it runs the code inside the loop.



The loop terminates when the condition evaluates to False.

# While Loop

```
while condition:  
    # Code to execute  
    # Update condition
```

# While Loop

```
count = 0
while count < 5:
    print(count)
    count += 1
```

# For Loop

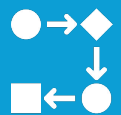
---



A for loop in Python iterates over a sequence of elements, such as a list, tuple, string, or range object



It automatically retrieves each element from the sequence and executes the block of code for each element.



It is generally used when the number of iterations is known or when iterating over a collection of items.



# For Loop

```
for variable in sequence:  
    # Code to execute
```

# For Loop

```
for i in range(5):  
    print(i)
```

# Lists in Python

a list is a built-in data type

Used to store a collection of items.

Lists are ordered,

changeable (mutable)

allow duplicate elements.

Each item in a list is called an element and can be of any data type, including other lists (heterogenous).

Lists are defined by enclosing elements in square brackets ([]), with elements separated by commas.

# List in Python

```
fruits = ["apple", "banana", "cherry", "date"]
```

# Slicing

**Slicing** is a technique used to extract a subset of elements from a list. The syntax for slicing is:

```
list[start:stop:step]
```



# Basic Slicing

- Start from 1 but doesn't include 4<sup>th</sup> element

```
fruits = ["apple", "banana", "cherry", "date", "elderberry"]  
print(fruits[1:4]) # Output: ['banana', 'cherry', 'date']
```

# Slicing with step

```
print(fruits[0:5:2]) # Output: ['apple', 'cherry', 'elderberry']
```

# Omitting Indices

```
print(fruits[:3]) # Output: ['apple', 'banana', 'cherry']  
print(fruits[2:]) # Output: ['cherry', 'date', 'elderberry']  
print(fruits[::2]) # Output: ['apple', 'cherry', 'elderberry']
```

# Adding Elements to a list

- `append()`:
  - Adds element to the end of the list

```
fruits = ["apple", "banana", "cherry"]  
fruits.append("date")  
print(fruits) # Output: ['apple', 'banana', 'cherry', 'date']
```

# Adding Elements to a list

- `insert()`:
  - Adds an element at a specified index.

```
fruits = ["apple", "banana", "cherry"]  
fruits.insert(1, "blueberry")  
print(fruits) # Output: ['apple', 'blueberry', 'banana', 'cherry']
```



# Adding elements to a list

- `extend()`
  - Adds another list to end of current list

```
fruits = ["apple", "banana"]  
more_fruits = ["cherry", "date"]  
fruits.extend(more_fruits)  
print(fruits) # Output: ['apple', 'banana', 'cherry', 'date']
```

# Removing Elements from a list

- `remove()`:
  - Removes the first occurrence of a specified value.

```
fruits = ["apple", "banana", "cherry"]  
fruits.remove("banana")  
print(fruits) # Output: ['apple', 'cherry']
```

# Removing Elements from a list

- `pop()`:
  - Removes and returns the element at a specified index (default is the last item).

```
fruits = ["apple", "banana", "cherry"]  
last_fruit = fruits.pop()  
print(last_fruit)  # Output: 'cherry'  
print(fruits)      # Output: ['apple', 'banana']
```

# Removing Elements from a list

- del:
  - Removes an element at a specified index or deletes the entire list.

```
fruits = ["apple", "banana", "cherry"]  
del fruits[1]  
print(fruits) # Output: ['apple', 'cherry']  
  
del fruits      # This will delete the entire list
```

# Removing elements from a list

- `clear()`:
  - Removes all elements from the list, making it empty.

```
fruits = ["apple", "banana", "cherry"]  
fruits.clear()  
print(fruits) # Output: []
```

# Code for Traversing List

```
fruits = ["apple", "banana", "cherry", "date"]  
  
# Traverse the list and print each element  
for fruit in fruits:  
    print(fruit)
```

# Code for Traversing List (Starts from Zero)

```
fruits = ["apple", "banana", "cherry", "date"]

# Traverse the list with index and print each element with its index
for i in range(len(fruits)):
    print(f"Index {i}: {fruits[i]}")
```

# Code for Traversing List (Print both index and a value)

```
fruits = ["apple", "banana", "cherry", "date"]  
  
# Traverse the list and print each element with its index using enumerate  
for index, fruit in enumerate(fruits):
```



# LAB TASKS

## Task 1: Number Checking

- Prompt the user to enter a number.
- Check if the number is positive or non-positive.
  - If the number is positive, print "The number is positive."
  - Otherwise, print "The number is zero or negative."

## Task 2: Input a Number:

- If the score is 90 or above, it prints "Grade: A".
- If the score is between 80 and 89, it prints "Grade: B".
- If the score is between 70 and 79, it prints "Grade: C".
- If the score is between 60 and 69, it prints "Grade: D".
- If the score is below 60, it prints "Grade: F".

# LAB TASKS

## Task 3:

- Using a for loop sum the given series:
  - 1,3,5,7,9,11,13

## Task 4:

- Using while loop compute factorial of a number

# Lab Tasks

- Task 5: Working with Lists:
  - Create a list of your five favorite movies. Print the list.
  - Create a list containing the numbers from 1 to 10 (inclusive). Print the list.
  - Given a list of numbers [1, 3, 5, 7], add the number 4 at the position between 3 and 5. Print the updated list.
  - Add the item "Python" to the end of the list ["Java", "C++", "JavaScript"]. Print the updated list.
  - From the list [10, 20, 30, 40, 50], remove the item at index 2
  - Delete the first item from the list ["apple", "banana", "cherry"] using the del statement and print the updated list.

# Lab Tasks

- Given the list [5, 2, 9, 1], sort the list in ascending order and print the result.
- Create a list of strings ["banana", "apple", "cherry"]. Sort the list in descending order and print it.
- Given the list ['a', 'b', 'c', 'd', 'e', 'f'], create a sublist containing only the middle three elements. Print the result.
- Slice the list [10, 20, 30, 40, 50, 60, 70] to get every other element starting from index 1. Print the result.