Lab 11: INTRODUCTION TO PYTHON

Python is a high-level, interpreted, interactive, and object-oriented programming language developed by Guido van Rossum between 1985 and 1990. It is designed to be simple, readable, and easy to learn, using English-like keywords instead of complex syntax. Python's source code is freely available under the GNU General Public License (GPL).

Key Features of Python:

- **Interpreted:** Python code is executed line by line, so there's no need for compilation.
- **Interactive:** You can write and test code directly using the Python prompt.
- **Object-Oriented:** Python supports classes and objects to organize code efficiently.
- **Beginner-Friendly:** Its simplicity makes it ideal for new programmers.
- **Versatile:** Used for web development, data analysis, automation, artificial intelligence, and more.

Python is one of the most popular programming languages today, suitable for both beginners and professionals who want to build everything from small scripts to large-scale applications.

Python is an easy-to-learn programming language that uses simple syntax similar to English. It is widely used for web development, data analysis, automation, and more. Below are some very basic concepts every beginner should know before starting Python programming.

The following table summarizes some of the most fundamental Python concepts, providing a brief description and an example for each. These basics form the foundation for writing efficient and effective Python programs.

Concept	Description	Example
Print Statement	Used to display output on the screen	print("Hello, World!")
Input Statement	Used to take input from the user	name = input("Enter your name: ")
Comments	Used to write notes in code, ignored by Python	# This is a comment
Indentation	Used to define blocks of code instead of braces	if a > b:\n print("a is greater")
Variables	Used to store data values	x = 10
Constants	Used to store fixed values	PI = 3.14

Data Types	Different types of data in Python	x = 10 # int\ny = 3.5 # float\nname = "Amna" # str
Dictionaries	Key-value pairs	student = {"name": "Amna", "age": 20}
Loops	Used to execute a block of code multiple times	for i in range(5):\n print(i)
Conditional Statements	Used to perform actions based on conditions	if x > 0:\n print("Positive")
Functions	Reusable block of code	def greet():\n print("Hello")

VARIABLES IN PYTHON

A variable in Python is a name used to store a value in the computer's memory. It acts as a container that holds data for processing. Each variable has a data type such as number, string, list, tuple, or dictionary. You can create a variable simply by assigning a value.

For example: a = 10 or name = "Ali".

TYPES OF VARIABLES IN PYTHON:

Variable Name	Description	Example
x	Integer variable	x = 10
name	String variable	name = "Ali"
price	Float variable	price = 9.99
string	Text value or characters	string = "Hello"
array	Stores multiple values (via list)	array = [1, 2, 3]
list	Ordered collection of items	list1 = ["apple", "banana", "cherry"]
tuple	Immutable ordered collection	tuple1 = (10, 20, 30)

Sample Code:

```
name = “John"
age = 21
print("Name:", name)
print("Age:", age)
```

MAIN OPERATORS IN PYTHON

Operators are symbols used to perform operations on variables and values. Python supports several types of operators.

Operator Type	Symbol / Keyword	Example	Result
Arithmetic	+, -, *, /, %	10 + 5	15
Comparison	==, !=, <, >, <=, >=	5 < 10	True
Logical	and, or, not	a < 5 and b > 2	True/False
Assignment	=, +=, -=	a += 5	Adds 5 to a

Sample Code:

```
a = 10
b = 20
print(a + b)
print(a > b)
print(a < b and b > 10)
```

Example 1: Add Two Numbers

```
# This program adds two numbers
num1 = 1.5
num2 = 6.3
sum = num1 + num2
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

Output

```
The sum of 1.5 and 6.3 is 7.8
```

Example 2: Add Two Numbers With User Input

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')
sum = float(num1) + float(num2)
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

Output

```
Enter first number: 1.5
Enter second number: 6.3
The sum of 1.5 and 6.3 is 7.8
```

STRINGS IN PYTHON

A string is a collection of characters inside quotes. Strings can be single or double quoted.

Function	Description	Example
len()	Gives length of string	len("Python") → 6
upper()	Converts string to uppercase	"hi".upper() → HI
lower()	Converts string to lowercase	"HI".lower() → hi
split()	Splits string into list	"a b c".split() → ["a","b","c"]
replace()	Replaces word or character	"hi".replace("h","b") → bi

Sample Code:

```
text = "Hello World"
print(text[0])
print(text[2:5])
print(text.upper())
print(len(text))
```

ARRAYS / LISTS IN PYTHON

Python does not have a built-in array type, but lists are used instead. Lists can store multiple items in a single variable.

Operators	Description	Example
append()	Add item to the end	fruits.append("orange")
remove()	Remove an item	fruits.remove("apple")
len()	Find length of list	len(fruits)
sort()	Sort items	fruits.sort()
clear()	Remove all items	fruits.clear()

Sample Code:

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)
fruits.remove("banana")
print(fruits)
```

Access List Items in Python

```
a = [10, 20, 30, 40, 50]
# Accessing the first item
print(a[0])
# Accessing the last item
print(a[-1])
```

Output

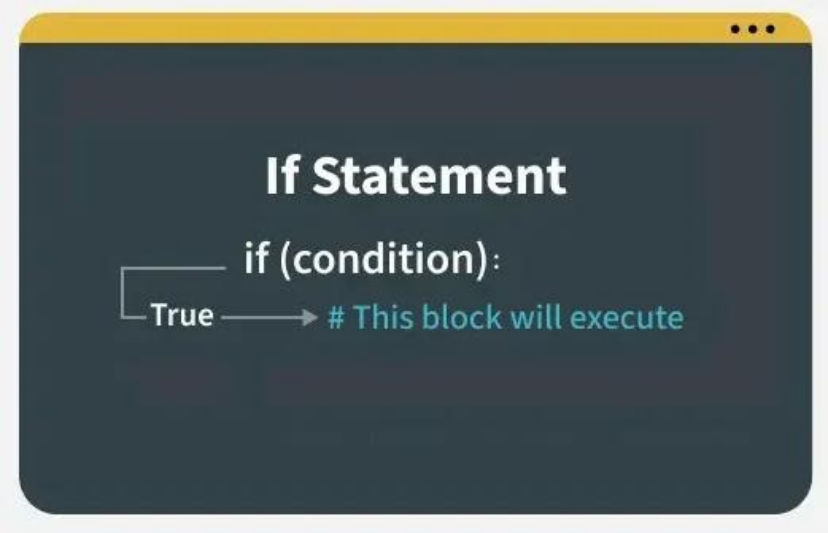
```
10
50
```

CONDITIONAL STATEMENTS IN PYTHON

Conditional statements in Python are used to execute certain blocks of code based on specific conditions. These statements help control the flow of a program, making it behave differently in different situations.

If Conditional Statement

If statement is the simplest form of a conditional statement. It executes a block of code if the given condition is true.



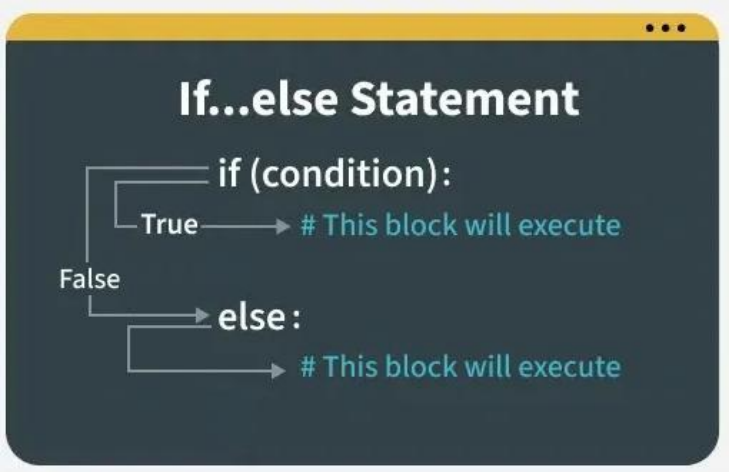

```
age = 20
if age >= 18:
print("Eligible to vote.")
```

Output

```
Eligible to vote.
```

If else Conditional Statement

If Else allows us to specify a block of code that will execute if the condition(s) associated with an if statement evaluates to False. Else block provides a way to handle all other cases that don't meet the specified conditions.




```
age = 10
if age <= 12:
    print("Travel for free.")
else:
    print("Pay for ticket.")
```

Output

```
Travel for free.
```

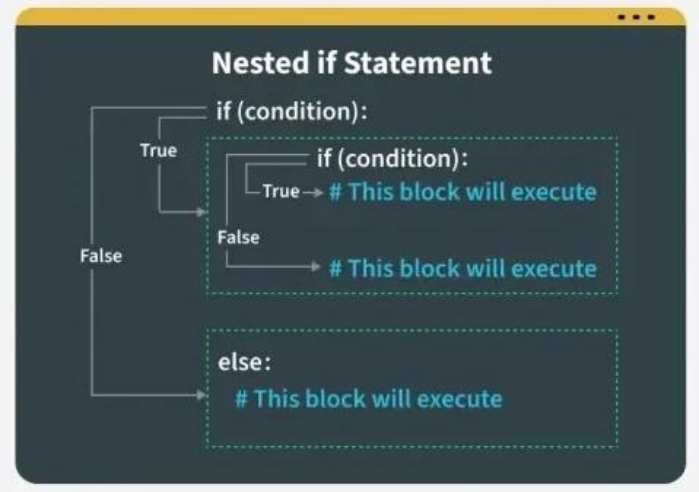
Nested if..else Conditional Statement

Nested if..else means an if-else statement inside another if statement. We can use nested if statements to check conditions within conditions.

SAMPLE CODE:

```
age = 70
is_member = True

if age >= 60:
    if is_member:
        print("30% senior discount!")
    else:
        print("20% senior discount.")
else:
    print("Not eligible for a senior discount.")
```



Output

```
30% senior discount!
```

Ternary Conditional Statement

A ternary conditional statement is a compact way to write an if-else condition in a single line. It's sometimes called a "conditional expression."

```
# Assign a value based on a condition
age = 20
s = "Adult" if age >= 18 else "Minor"
print(s)
```

Output

```
Adult
```

Here:

- **If age >= 18 is True, status is assigned "Adult".**
- **Otherwise, status is assigned "Minor".**

EXERCISE:

1. Given two numbers **a** and **b**. You need to perform basic mathematical operations on them. You will be provided an integer named as **operator.**

- If the operator equals to **1** add **a** and **b**, then print the result.
- If the operator equals to **2** subtract **b** from **a,** then print the result.
- If the operator equals to **3** multiply **a** and **b,** then print the result.
- If the operator equals to any other number**,** print "Invalid Input"(without quotes).

Note: Do not add a new line at the end.

Examples:

Input: a = 1, b = 2, operator = 3
Output: 2
Input: a = 2, b = 2, operator = 2
Output: 0

2. You are given a list that contains integers. You need to return the length of the list.

Examples:

Input: arr = [54, 43, 2, 1, 5]
Output: 5

Input: arr = [324, 5, 2, 2]
Output: 4

3. You are given a list that contains integers. You need to return the sum of the list.

Examples:

Input: arr = [54, 43, 2, 1, 5]
Output: 105

Input: arr = [324, 5, 2, 2]
Output: 333

4. Check if a String Contains a Specific Word. Take a sentence from the user and check if it contains the word "Python".

Example:

Input: Enter a sentence: I am learning Python programming.

Output: The sentence contains the word 'Python'.

5. Calculate Average of Three Numbers.Take three numbers as input and calculate their average.

Example:

Input: Enter three numbers: 5 10 15

Output: Average = 10.0

6. Create a list of five fruits. Ask the user for an index (0-4) and display the fruit at that position.

Example:

Input: Enter an index (0-4)

Output: Fruit at index 2 is: Mangoes

7. Find the Smallest of Three Numbers Problem: Take three numbers and print the smallest using if-else statements.

Example:

Input: Enter three numbers: 8 4 12

Output: The smallest number is 4.

8. Take an integer input from the user and check whether it is positive, negative, or zero.

Example:

Input: Enter a number: -5

Output: The number is negative.