1) Explain Programming and Python in detail

• Definition: Programming refers to a technological process for telling a computer which tasks to perform in order to solve problems. To put it simply, computer programming is the process of humans communicating with computers to make them perform specific tasks. It is a like a collaboration between humans and computers.

• Purpose of Programming: The purpose of programming is to make human life better and easier. It does so because the computer programs can perform a set of tasks once we define them clearly. Hence we can use computer to solve our day to day problems by providing it clear instructions rather than solving it by ourself. This saves a lot of human time and energy.

• Characteristics of Python:

1) Python's simple and readable syntax makes it beginner-friendly.
2) Python runs seamlessly on Windows, MacOS and Linux.
3) Includes libraries for tasks like web development, data analysis and machine learning.
4) Variable types are determined automatically at runtime, simplifying code writing.
5) Supports multiple programming paradigms, included object-oriented, functional and procedural programming.
6) Python is free to use, distribute and modify.

• Applications of Python:

1) Web-development: Django, Flask and Pyramid are popular web frameworks in python used for building applications ranging from small websites to large enterprise applications.
2) Data science and analytics: Python is highly demanded language in data science and machine learning, with popular libraries like

Numpy, Pandas, Matplotlib.

3) Automation and scripting: Python extensively used for automation tasks, such as scripting, testing and deployment, providing an efficient way to automate repetitive tasks.

4) Desktop GUI Application: Python provide several GUI toolkits like Tkinter, PyQt and Wxpython, allowing developers to create desktop applications.

5) Game development: Python can be used to develop games with libraries like Pygame, which provides graphics, sound and input handling tools for game development.

- Types of comments in Python:

Comments are a critical aspect of coding that can help to explain what your code does and why you wrote it a certain way. In python, comments are signified by a '#'. It is put at the starting of a line, everything after it is considered a comment. There are two types of comment in Python.

1) Single-line comment in python: Single line comments in python add a short description or note about a single line of code. They start with '#' symbol.

Syntax: # This is a single-line comment
This is another single-line comment

2) Multi-line comment in python: Multi-line comments in python add longer description or notes about multiple lines of code.

Syntax: """ This is a multi-line comment.
It can span multiple lines..."

- Importance of python in software development: Python has earned its place in modern software development because it blends simplicity with power for building a web development, experimenting with AI. Python gives us flexibility to grow and innovate without unnecessary complexity. From startups to global enterprise, it continues to shape the way software is planned, built and improved.

2) Describe Data Types and Operators in Python

• Built-in Data types in Python:-

1) Numeric Data types: A numeric value can be an integer, a floating number, or even a complex number.

• Integer: This value is represented by int class. It contains positive or negative whole numbers. Ex: 10, -5, 0, a = 10

• Float: This value is represented by float class. It contain real number with a decimal point. Ex: 3.14, -0.99, 0.0, b = 3.14

• Complex: A complex number is represented by a complex class. It is specified as (real part) + (imaginary part). Ex: 2+3j

2) Sequence Datatypes: There are several sequence data types in python.

• String: It is represented by str class. Strings in python can be created using single quotes, double quotes or even triple quotes.
Ex: "Hello", 'Python', name = "Python"

• List: Lists are just like arrays, declared in other languages which is an ordered collection of data. They are represented in square brackets
Ex: [1, 2, 3], ["a", "b"], my_list = [1, 2, 3]

• Tuple: Just like a list, a tuple is an ordered collection of python object. The only difference is that tuple are immutable whereas lists are mutable. Ex: my_tuple = (1, 2, 3)

3) Dictionary Datatype: A dictionary in python is a collection of data values, used to store data value like a map, unlike other datatypes that hold only single value, a dictionary holds a key: value pair. Ex: my_dict = {"a": 1, "b": 2}

4) Set Datatype: Set is collection of multiple items having different datatypes. Sets are mutable, unindexed and donot contain duplicates.
Ex: S = {1, 2, 3}

5) Boolean: It gives logical value (true/false). Ex: flag = True

• Type identification using type(): The type() function in Python is used to determine the type of an object. It helps in identifying data types such as integers, strings, lists, dictionaries. This function is used for dynamic checking. Syntax: type(object)

Ex:- x = 10
y = "Hello"
print(type(x)) # <class 'int'>
print(type(y)) # <class 'str'>

• Various python operators and Real-World usage of operators

1) Arithmetic Operators: Used for mathematical calculations

Operator	Name	Example
+	Addition	10+20 = 30
−	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus	22%10 = 20
**	Exponent	4**2 = 16
//	Floor Division	9//2 = 4

Real-world usage: calculating total bill amount, finding average marks...
Ex: price = 200
quantity = 3
total = price * quantity.

2) Comparision Operators: Used to compare two values and returns (T/F)

Operator	Name	Example
==	Equal	4==5 is False
!=	Not Equal	4!=5 is True
>	Greater than	4>5 is False
<	less than	4<5 is True
>=	Greater than(or) Equal to	4>=5 is False
<=	less than(or) equal to	4<=5 is True

Real-World usage: checking exam pass/fail, age eligibility for voting...

Ex: marks = 75

marks >= 40

3) Assignment operators: used to assign values to variable.

Operator	Name	Example
=	Assignment Operator	a = 10
+=	Addition Assignment	a += 5
-=	Subtraction Assignment	a -= 5
*=	Multiplication Assignment	a *= 5
/=	Division Assignment	a /= 5
%=	Remainder Assignment	a %= 5
**=	Exponent Assignment	a **= 2
//=	Floor Division Assignment	a //= 3

Real-World usage: Updating bank balance, Increasing score in a game...

Ex: balance = 1000

balance += 500

4) Logical operators used to combine multiple conditions.

Operator	Description	Example
and Logical AND	If both of the operands are true then the condition is True.	(a and b) is true
or Logical OR	If any one of the condition is non-zero, then the condition is True.	(a or b) is true
not Logical NOT	Used to reverse the logical state of its operand	Not (a and b) is false

Real-World usage: Login systems, Loan approval conditions

Ex: age = 20

citizen = True

age >= 18 and citizen

5) Membership Operators: Used to check if a value exists inside a sequence

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it doesn't find a variable in the specified sequence and false otherwise	x not in y, here not in results in a 1 if x is not a member of sequence y.

Real-world usage: Searching items in a list, checking enrolled students

Python Ex:

```
fruits = ["apple", "banana", "mango"]
"apple" in fruits
```

6) Identity Operators: Used to check if two variables refer to the same object.

Operator	Description	Example
is	Evaluates to true if the variable on either side of the operator point to the same object and false otherwise	x is y, here a1 results in 1, if id(x) equals id(y).
is not	Evaluates to false if the variable on either side of the operator point to the same object and true otherwise	x is not y, there are no results in 1 if id(x) is not equal to id(y)

Real-World usage: Memory comparision, checking Object references in programs, debugging.

Ex: a = [1, 2, 3]

b = a

a is b

3) Explain Python input and output operations:

- **Input() function and it's dyfault data type :**

The input() python function takes user input and evaluates it. Python reads the input as either text or numbers based on how we handle it in the code. By default, the input function always returns the input as a string. No matter what value we enter as an input, the input() function in python will automatically convert it into a string.

Syntax: Variable_name = input("Your message here:")

Ex: name = input("Enter your name:") #output Enter your name: Sadia
print("Hello", name) Hello, Sadia

- **Type conversion while taking input :** When we write code, we may need to change data from one type to another, like turning a string into an integer or a float into a string. This process is called "Python Type conversion". There are two types

1) Implicit conversion - In this type of conversion, Python changes the type automatically.

2) Explicit conversion - In this, Python is when we manually change the datatype of a value using built-in functions like int(), float().

Example:

int	float	tuple	complex
x = int("1010", 2)	x = float("10")	print(tuple("abc"))	print(complex(3,4))
print(x)	print(x)	print(tuple())	output - (3+4j)
output - 10	output - 10	output - ('a', 'b', 'c')	
		()	

list	set
print(list("abc")	print(set([1, 2, 2, 3]))
print(list())	output - {1, 2, 3}
output - ['a', 'b', 'c']	
[]	

• Multiple input : When we need multiple inputs, Python makes it simpler. Instead of asking inputs one by one, we can have multiple values at once using different methods.

1) Using input().split() : The split() function separates the values entered by the user based on space (or) delimiter you specify

Example:

```
x,y,z = input("Enter three numbers: ").split(',')
print(x,y,z)
```

2) Converting input type with map() : By default, input are strings. If you want integers and float, you can combine map() with split().

Example

```
numbers = list(map(int,(input("Enter numbers: ")).split()))
print("Numbers:", numbers)
```

3) Using list comprehension : List comprehension gives you a compact way to store input in a list.

Example:

```
values = [int(x) for x in input("Enter values:").split()]
print(values)
```

4) Taking inputs across Multiple lines : Sometimes you may want each input on a new line. You can use a loop for this.

Example:

```
a = input("Enter first value:")
b = input("Enter second value:")
```

• Formatted output using print(), separators and format specifiers :

The print() function allows you to print out the value of a variable and strings in parantheses.

Syntax: print() (or) print(value, sep=' ', end='\n') where.

Sep - The optional parameter sep is a separator between the output values. We can use a character, integer or a string as a separator. The default separator is space.

end - This also optional and it allow us to specify any string to be appended after the last value. The default is a new line.

formatting strings:

f-strings allows you to format selected parts of a string.

To specify a string as an f-string, simply put an 'f' infront of the string literal.

example

```
txt = f"The price is 49 dollars"
print(txt)
```

Example of print() function:

```
N1 = 20
N2 = 30
Sum = N1+N2
Print(sum) # output = Sum = 30
```

Example of sep:

```
print('H','e',l','l','o', sep = '-')   #output: H-e-l-l-o
```

Example of end:

```
print("My name is stephen", end = ' ')
print("and learn python")  #output: My name is stephen and
                                    learn python
```

4) Discuss control statements and Decision-Making statements in Python

- Meaning and importance of control statements:

Control statements are fundamental components of programming that enable us to dictate the flow of execution within a program.

In python, these statements allows us to make decisions, repeat actions, and manage the sequence of operations based on specific conditions.

• Importance of control statements :

1) Decision Making - They allow programs to make decisions based on user input or other variables, enable more interactive applications.

2) Code Efficiency - By using loops and conditional statements, we can avoid redundant code, leads to cleaner and efficient programs.

3) Flexibility - control statements provide the ability to adapt the behaviour of a program in response to changing conditions.

4) Error Handling - exception handling is vital for managing unexpected errors and controlling program flow during these situations.

• Types of control statements :-

1) Conditional control statements : These statements includes if, elif, else allow us to execute certain code based on whether a specific condition is true or false.

2) Looping control statement : The for and while loops enable us to repeat a block of code multiple times, making it easier to handle repetitive tasks.

3) control flow altering statements : statements like break, continue can alter the flow within loops, providing further control over how iterations are executed.

4) Exception handling control statements : Using try, except and finally, these statements allow us to manage errors and exception that may occur during program execution.

• Decision-making statements and syntax flow with examples :

1) if-statement : The simplest form of a conditional statement is if statement. It checks the condition and executes the associated block of the code if the condition is true.

Syntax :

```
if condition:
    statement
```

Example : age = 18

```
if age >= 18:
    print ("You are eligible to vote.")
```

#output : You are eligible to vote

1) If - else statement: If - else statement extends the if statement by providing an alternative block of code to execute when the condition is false.

Syntax:

```
if condition:
    statement true
else:
    statement false
```

Example:

```
age = 14
if age >= 18:
    print ("You are eligible to vote")
else:
    print ("You are not eligible to vote")
```

output:

You are not eligible to vote

2) If - elif - else statement: The if - elif - else statement allows us to check multiple conditions in sequence. If the first condition is false, it evaluates the next condition in elif block and continues until it find a true condition or reaches the else block.

Syntax:

```
if condition 1:
    statement 1
elif condition 2:
    statement 2
else:
    statement false.
```

Example

```
num = 10
if num > 0:
    print ("Positive number")
elif num == 0:
    print ("zero")
else:
    print ("Negative number")
```

output:

Positive number

5) Write an essay on Python Programming Fundamentals

Python is a high level, interpreted and general-purpose programming language that is widely used in various fields such as web development, data science, artifical intelligence, automation and software engineering. Created by Guido Van Russom in late 1980s, Python was designed with an emphasis on simplicity, readability and ease of learning. Due to these features, Python has become one of the most popular programming languages in the world, especially for beginners.

Another important fundamental of python is it interpreted nature. Python code is interpreted line by line by an interpreter, which makes debugging easier. Error can identified quickly without compiling the entire program.

Python support dynamic typing, meaning that vaviables doesnot need explicit data type declarations. The type of a variable is declared at runtime based on the values assigned to it. Python provides built-in data-types such as integers, floats, strings, lists and tuples, sets and dictionaries. These data structures allow efficient storage and manipulation of data and are essential for writing effective programs.

Input and output operations are also basic fundamental of python programming. The `input()` function is used to take input from the user and the `print()` function is used to display output. Python allows formatted output using separators, format specifiers and f-strings which makes output representation flexible and user-friendly.

Python includes control structures such as conditional statements and loops. conditional statements like `if`, `elif` and `else` are used to make decisions based on conditions. Looping statements such as `for` and `while` allow repeated execution of code blocks. These control structures

help implement logic and automate repetitive tasks.

Functions are another core concept in Python. A function is a block of reusable code that performs a specific task. Functions improve the code reusability, modularity, reduce redundancy and enhance readability. Python also supports built-in functions as well as user-defined functions.

Python follows an object-oriented programming (OOP) approach, supporting concepts such as classes, objects, inheritance, polymorphism and encapsulation. This make python suitable for developing large and complex applications by organizing code into reusable components.

In addition to these fundamentals, Python has a vast standard library and strong community support. Libraries and framework like Numpy, Pandas, Django and Flask extend Python's functionality and make it powerful for real-world applications.

In conclusion, the fundamentals of python programming - simple syntax, dynamic typing, interpreted execution, rich data types control structures, functions and object-oriented features - make it an ideal language for beginners and professional alike. Python's versatility and ease of use have established it as a cornerstone of modern programming.

Python Programming:

1) Movie Ticket Pricing:

```
age = int(input("enter your age: "))
x = int(input("3d movie: Yes(1) or No(0):"))
if age<13:
    ticket=150
elif age==13 or age<=59:
    ticket = 250
else:
    ticket=200
if x==1:
    ticket=ticket+30
elif x==0:
    ticket=ticket
print("ticket",ticket)
```

Output

```
enter your age:9
3d movie: Yes(1)or No(0):1
ticket : 180
```

2) College Attendance Rule

```
atp=int(input("enter attendance percentage:"))
med=int(input("medical certificate: yes(1)or no(0):"))
if atp>=75:
    print("Allowed")
elif atp>=60 and med==1:
    print("Allowed")
else:
    print("Not Allowed")
```

Output

```
enter attendance percentage: 70
medical certificate: Yes(1) or no(0): 1
Allowed
```

3) E-commerce Discount

```
bill = float(input("enter bill amount:"))
prime = int(input("prime member:
        Yes(1) or No(0): "))
if bill>=5000:
    bill=bill-bill/100*20
elif bill==2000 or bill<=4999:
    bill=bill-bill/100*10
else:
    print("No discount")
if prime==1:
    bill=bill-bill/100*5
elif prime==0:
    bill=bill
print("Total bill:",bill)
```

Output

```
enter bill amount: 4932.67
prime member: Yes(1) or NO(0): 1
Total bill: 4217.43985
```

4) Smartphone Battery Warning:

```
battery = int(input("enter battery:"))
charge = int(input("is it charging:"))
if charge == 1:
    print("charging")
else:
    if battery <= 20:
        print("Low battery")
    elif battery == 21 or battery <= 80:
        print("Normal")
    else:
        print("Full")
```

output

enter battery: 52
is it charging: 1
charging

5) Driving License check:

```
age = int(input("enter age:"))
if age >= 60:
    print("eligible")
else:
    test = int(input("passed driving
                test? Y(1) or N(0):"))
    if age >= 18 and test == 1:
        print("eligible")
    elif test == 0:
        print("Not eligible")
```

Output

enter age: 34
passed driving test: Y(1) or No(0): 1
eligible

6) Online Food delivery

```
amount = float(input("amount:"))
isGold = int(input("gold member?:"))
dist = int(input("enter distance:"))
if dist <= 10 and (isGold == 1 or
                amount > 500):
    print("Free Delivery")
else:
    print("Delivery charged")
```

Output

amount = 789.00
gold member? : 1
enter distance: 12
Delivery charged

7) Bank loan Approval

```
salary = float(input("enter salary:"))
if salary >= 50000:
    print("loan accepted")
else:
    cscore = int(input("enter credit score:"))
    if salary >= 30000 and score >= 700:
        print("loan accepted")
```

```
else:
    print("loan rejected")
```

Output

enter salary: 48000

enter credit score: 800

loan accepted

7) Bank Loan Approved

8) Electricity Bill :

```
unit = float(input("enter units consumed:"))
if unit<=100:
    unit = unit*2
    print("bill:", unit)
elif unit<=200:
    unit = 200+(unit-100)*3
    print("bill:", unit)
else:
    unit = 500+(unit-200)*5
    print("final bill:", unit)
```

output

enter units consumed: 950

final bill: 750.0

9) Student Scholarship :

```
marks = int(input("enter marks:"))
s = int(input("are you single parent?
        y(1) or n(0):"))
if marks>=85:
    if s==1:
        print("you will get a scholarship")
    else:
        family = int(input("enter family income:"))
        if family<=500000 and marks>=85:
            print("you will get a scholarship")
        else:
            print("Not eligible for scholarship")
else:
    print("Not eligible for scholarship")
```

Output

enter marks: 90

are you single parent? y(1) or N(0): 0

enter family income: 400000

Not eligible for scholarship

10) Online Exam Result :

```
theory = int(input("enter theory marks:"))
practical = int(input("enter practical marks:"))
total = theory+practical
if (theory>=40 or practical>=40) and
        total>=100:
    print("passed")
elif theory>=40 and practical>=40:
    print("passed")
else:
    print("failed")
```

Output

enter theory marks : 78

enter practical marks : 23

passed

11) Hotel Room pricing :

```
day = int(input("enter number of days stayed:"))
weekend = int(input("enter weekend(1) or normal day(0):"))
if weekend == 1:
    amount = 6000
    print("total bill:", amount)
else:
    amount = 3000
    print("total bill:", amount)
total = amount * day
if day > 3:
    total *= 0.85
    print("Total bill amount:", total)
```

Output

enter number of days stayed : 3

Enter weekend(1) or normal day(0) : 0

Total bill : 3000.

12) Gaming zone Entry :

```
score = int(input("enter a score:"))
premium = int(input("premium pass: y(1) or n(0)"))
weekdeat = int(input("weekdeat: y(1) or n(0)"))
if score >= 100 or premium == 1:
    print("Next level")
elif weekdeat == 1:
    print("access denied")
else:
    print("your score is low")
```

Output

enter a score : 90

premium pass : y(1) or no (0) : 0

weekdeat : y(1) or n(0) : 1

access denied

13) Mobile data usage :

```
data = int(input("enter a daily usage in mb:"))
unlimited = int(input("unlimited plan: y(1) or n(0):"))
roaming = int(input("is roaming: y(1) or n(0):"))
if roaming == 1:
    print("unlimited plan is not working")
```

```
elif data <= 3048 or unlimited == 1:
    print ("unlimited data available")
else:
    print ("High daily usage data")
```

Output

```
enter a daily usage in mb : 3001
unlimited plan: y(1) or n(0): 1
is roaming: y(1) or n(0): 0
unlimited data available
```

14) Office Entry System:

```
id = int(input("id: y(1) or n(0):"))
holiday = int(input("is today Holiday?
                    y(1) or n(0):"))
facescan = int(input("facescan: y(1) or
                     no(0):"))
faceid = int(input("faceid: y(1) or n(0): "))
if holiday == 1:
    print ("access denied")
elif id == 1 and (facescan == 1 or faceid == 1):
    print ("access accepted")
else:
    print ("id doesn't match")
```

Output:

```
id: y(1) or n(0): 1
is today Holiday: y(1) or n(0): 0
facescan: y(1) or n(0): 1
faceid: y(1) or n(0): 0
access accepted
```

15) Movie Rating Display

```
rating = float(input("enter average
                     rating:"))
editor = int(input(" is editor choice:
                   y(1) or n(0): "))
if editor == 1:
    print("Recommended")
elif rating >= 8.5:
    print("excellent")
elif rating >= 6.0 or rating <= 8.4:
    print("good")
else:
    print("Average")
```

Output

```
enter average rating: 8.6
is editor choice: y(1) or n(0): 0
excellent
```