PYTHON ASSIGNMENT

1. Explain the key features of Python that make it a popular choice for programming
   - python is the most popular choice because it is most accessible open source coding language as it has a simple syntex to code. Because of its simplicity of learning and utilization, python codes can be handily composed and executed much more quickly than other programming dialects.

   Key features of python are:
   *__Easy to learn__: Python's syntax is similar to English and doesn't use brackets or semicolons

   __Open-source__: Python's source code is free to use, modify, and share

   __Large standard library__: Python's built-in modules help with common coding tasks.

   __Cross-platform__: Python can run on many different platforms without needing to be modified.

   __Community support__: Python has a large user community with developers of all skill levels.

   __Versatile__: Python is a general-purpose language that can be used for many different applications.

   __Easy to read__: Python uses English keywords and has a simple syntax.

   __Portable__: Python is not designed for specific applications, so it can be used for many different types of applications.

Other features

.Database support

Object-oriented language

PY

Extensible

GUI programming support

High-level language

Dynamically-typed language

Dynamic memory allocation

Q2. Describe the role of predefined keywords in Python and provide examples of how they are used in a program.

- Predefined keywords in Python are reserved words that have specific meanings and purposes within the language. They form the core syntax and structure of Python code and cannot be used as identifiers (e.g., variable names, function names). Keywords enable the interpreter to understand and execute instructions, control program flow, and perform specific operations.

Examples include if,none,else,for,while,def,class,import,true,false.

.if:Starts a conditional statement, executing code based on a condition.

```
x = 10
if x > 5:
    print("x is greater than 5")
```

.elif:Provides an additional condition to be checked if the previous `if` condition is false.

```
x = 7

if x > 10:

    print("x is greater than 10")

elif x > 5:
```

PY

```
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

Q3. Compare and contrast mutable and immutable objects in Python with examples.

Python data type is categorized into two types:

- Mutable Data Type – A mutable data type is one whose values can be changed.
    - Example: List, Dictionaries, and Set
- Immutable Data Type – An immutable data type is one in which the values can't be changed or altered.
    - Example: String and Tuples

Difference Between Mutable and Immutable Data Type: Mutable vs Immutable

	Mutable	Immutable
Definition	Data type whose values can be changed after creation.	Data types whose values can't be changed or altered.
Memory Location	Retains the same memory location even after the content is modified.	Any modification results in a new object and new memory location
Example	List, Dictionaries, Set	Strings, Types, Integer
Performance	It is memory-efficient, as no new objects are created for frequent changes.	It might be faster in some scenarios as there's no need to track changes.

Thread-Safety	Not inherently thread-safe. Concurrent modification can lead to unpredictable results.	They are inherently thread-safe due to their unchangeable nature.
Use-cases	When you need to modify, add, or remove existing data frequently.	When you want to ensure data remains consistent and unaltered.

Q4. Discuss the different types of operators in Python and provide examples of how they are used.

Python offers several types of operators, broadly classified as arithmetic,comparison/relational,logical,assignment,membership and identity operators. These operators are used to perform calculations, compare values, combine conditions, assign values to variables, check for membership in sequences, and compare object identities.

1. Arithmetic Operators: These operators perform basic mathematical calculations:

```
# Addition

print(5 + 3)  # Output: 8

# Subtraction

print(10 - 4)  # Output: 6

# Multiplication

print(2 * 6)  # Output: 12

# Division

print(15 / 3)  # Output: 5.0

# Modulus (remainder)
```

```
print(10 % 3)  # Output: 1

# Floor division (integer division)

print(15 // 2)  # Output: 7

# Exponentiation

print(2 ** 3)  # Output: 8
```

Q5. Explain the concept of type casting in Python with examples.

- in Python, is the process of changing a variables data type from one type to another. It involves converting a value from its original type (e.g., int) to a desired type (e.g., float or str). Python offers both implicit (automatic) and explicit (manual) type casting.

Implicit Type Casting:

Implicit type casting occurs automatically when Python performs operations that require different data types to be compatible. For example, when adding an integer and a float, Python automatically converts the integer to a float to ensure the result is also a float.

Example:

```
x = 5

y = 2.5

z = x + y  # x is implicitly converted to float before addition

print(z)  # Output: 7.5

print(type(z))  # Output: <class 'float'>
```

Explicit Type Casting:-Explicit type casting, or type casting, is when you

manually convert a variable's data type using built-in functions like int(), float(), or str(). This is often necessary when you want to perform operations that expect a specific data type or when you need to change the representation of a value.

Examples:

```
# Converting a string to an integer

num_str = "10"

num_int = int(num_str)

print(num_int)  # Output: 10

print(type(num_int))  # Output: <class 'int'>



# Converting a float to an integer (truncates decimal part)

num_float = 3.14

num_int = int(num_float)

print(num_int)  # Output: 3

print(type(num_int))  # Output: <class 'int'>
```

Q6. How do conditional statements work in Python? Illustrate with examples

- Conditional statements in Python allow for different blocks of code to be executed depending on whether a condition is true or false. The primary conditional statements are `if`, `elif`, and `else`

if statement

The if statement evaluates a condition. If the condition is true, the block of code indented below the if statement is executed. If the condition is

false, the block of code is skipped.
Example:

```
x = 10

if x > 5:
    print("x is greater than 5")
```

`elif` statement

The `elif` (else if) statement is used to check multiple conditions in sequence. It is placed after an `if` statement and before an optional `else` statement. If the `if` condition is false, the `elif` condition is evaluated. If the `elif` condition is true, its corresponding code block is executed

```
x = 3

if x > 5:
    print("x is greater than 5")

elif x < 5:
    print("x is less than 5")
```

Q7. Describe the different types of loops in Python and their use cases with examples

Types of Loops in Python

While Loops in Python

1. **For Loops in Python**
2. **Nested Loops in Python**

PY

Name of the Python loop	Description
While loop	Repeats the given statement or group of statements until the given condition is true. The while loop programs in Python test the condition before executing the loop body.
For loop	Executes a code block multiple times. It abbreviates the code managing the loop variable.
Nested loop	Allows to iterate a loop inside another loop.

1. While Loops in Python

Starting with the most common query, “What is a while loop in Python?” It is used in Python programming to execute a block of statements until the given condition is true. Once the condition is evaluated as false, the program executes the line immediately after the loop.

While Loop in Python Syntax

```
while expression:

statement(s)
```

Python indentation is a way of grouping statements. The statements indented using the same number of spaces are considered part of the same code block.

Let’s take a look at examples of While Loops in Python in different scenarios.

For Loops in Python

We use the for loop program in Python for sequential traversal. It is designed to iterate over a sequence, such as a list, dictionary, tuple, set, and string. Iteration is the process of traversing a sequence.

It works more like an iterator method in object-oriented programming than the for loop in other programming languages. The for-in loop in Python is similar to the for-each loop in other programming languages.

3. Nested Loops in Python

In Python, the nested loop refers to using a loop within another loop. Every single iteration of the outer loop has an inner loop executing its iterations. For example, if the outer loop has a and the inner loop has b iterations, the total iterations will be a x b. This means that for every outer loop iteration, the inner loop executes b times.

PY