1.What is Python, and why is it popular?

Python is a high-level, interpreted programming language known for its clean syntax and ease of learning. It's popular because of its versatility, vast libraries, community support, and use in fields like web development, data science, AI, and automation.

2. What is an interpreter in Python?

An interpreter in Python reads and executes the code line by line. It translates Python code into machine-readable instructions during runtime, making debugging easier.

3. What are pre-defined keywords in Python?

Pre-defined keywords are reserved words in Python that have specific meanings and uses, like if, else, for, while, def, and return. They are part of the Python syntax.

4. Can keywords be used as variable names?

No, keywords cannot be used as variable names in Python because they are reserved by the language for specific functions and commands.

5. What is mutability in Python?

Mutability refers to an object’s ability to change its content after it has been created. Mutable objects can be changed in place, while immutable objects cannot.

6. Why are lists mutable, but tuples are immutable?

Lists are mutable to allow dynamic changes like adding or removing items. Tuples are immutable to provide data safety, faster access, and use as keys in dictionaries.

7. What is the difference between “==” and “is” operators in Python?

== checks if two variables have the same value, while is checks if they refer to the same object in memory.

8. What are logical operators in Python?

Logical operators (and, or, not) are used to combine or reverse conditional statements. They help in decision-making based on multiple conditions.

9. What is type casting in Python?

Type casting is the process of converting a variable from one data type to another, such as converting a string to an integer using int("5").

10. What is the difference between implicit and explicit type casting?

Implicit type casting is automatically done by Python (e.g., int to float). Explicit type casting is done manually using functions like int(), float(), or str().

11. What is the purpose of conditional statements in Python?

Conditional statements like if, elif, and else control the flow of a program by executing certain blocks of code only when specific conditions are met.

12. How does the elif statement work?

The elif (else if) statement checks another condition if the previous if condition is false. It allows multiple conditions to be evaluated in order.

13. What is the difference between for and while loops?

A for loop is used when the number of iterations is known (e.g., looping over a list). A while loop is used when looping should continue until a condition becomes false.

14. Describe a scenario where a while loop is more suitable than a for loop.

A while loop is better when waiting for user input to meet a condition, like asking for a valid password until the correct one is entered.

1. Write a Python program to print "Hello, World!

```
print("Hello World")
```

```
hello world
```

2. Write a Python program that displays your name and age

```
name=input("Enter your name: ")
age=int(input("Enter your age: "))
print("Name:",name)
print("Age:",age)
```

```
Enter your name: anshika
Enter your age: 22
Name: anshika
Age: 22
```

3.Write code to print all the pre-defined keywords in Python using the keyword library

```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'co
```

4.Write a program that checks if a given word is a Python keyword.Write a program that checks if a given word is a Python keyword. m

```
a=input("Enter a word: ")
if a in keyword.kwlist:
  print("It is a keyword")
else:
  print("It is not a keyword")
```

```
Enter a word: anshika
It is not a keyword
```

5.Create a list and tuple in Python, and demonstrate how attempting to change an element works differently for each

```
# Create a list and a tuple
my_list = [1, 2, 3, 4, 5]
my_tuple = (1, 2, 3, 4, 5)

print("Original list:", my_list)
print("Original tuple:", my_tuple)

# Attempt to change an element in the list (mutable)
print("\nAttempting to change the first element of the list...")
my_list[0] = 10
print("List after attempting change:", my_list)

# Attempt to change an element in the tuple (immutable)
print("\nAttempting to change the first element of the tuple...")
try:
  my_tuple[0] = 10
except TypeError as e:
  print("Error when attempting to change the tuple:", e)

print("Tuple after attempting change:", my_tuple) # The tuple remains unchanged
```

```
Original list: [1, 2, 3, 4, 5]
Original tuple: (1, 2, 3, 4, 5)

Attempting to change the first element of the list...
List after attempting change: [10, 2, 3, 4, 5]

Attempting to change the first element of the tuple...
Error when attempting to change the tuple: 'tuple' object does not support item assignme
Tuple after attempting change: (1, 2, 3, 4, 5)
```

6.Write a function to demonstrate the behavior of mutable and immutable arguments

```
def demonstrate_mutability(mutable_list, immutable_int):
  print("Inside the function:")
  print("Original mutable_list:", mutable_list)
  print("Original immutable_int:", immutable_int)

  # Modify the mutable argument
  mutable_list.append(4)
  print("Modified mutable_list:", mutable_list)

  # Attempt to modify the immutable argument (creates a new object)
  immutable_int += 1
  print("Modified immutable_int:", immutable_int)

# Demonstrate with a mutable list and an immutable integer
my_list = [1, 2, 3]
my_int = 10

print("Before calling the function:")
print("my_list:", my_list)
print("my_int:", my_int)

demonstrate_mutability(my_list, my_int)

print("After calling the function:")
print("my_list:", my_list) # The list is changed
print("my_int:", my_int)   # The integer is unchanged
```

```
Before calling the function:
my_list: [1, 2, 3]
my_int: 10
Inside the function:
Original mutable_list: [1, 2, 3]
Original immutable_int: 10
Modified mutable_list: [1, 2, 3, 4]
Modified immutable_int: 11
After calling the function:
my_list: [1, 2, 3, 4]
my_int: 10
```

7.Write a program that performs basic arithmetic operations on two user-input numbers

```
A=input("Enter first number: ")
B=input("Enter second number: ")
A=int(A)
B=int(B)
print("Addition:",A+B)
print("Subtraction:",A-B)
print("Multiplication:",A*B)
```

```
Enter first number: 18
Enter second number: 64
Addition: 82
Subtraction: -46
Multiplication: 1152
```

8. Write a program to demonstrate the use of logical operators.

```
a=10
b=20
print(a&b)
print(a/b)
print(a*b)
print(~a)
```

```
0
0.5
200
-11
```

9 .Write a Python program to convert user input from string to integer, float, and boolean types

```
a=input("Enter a String: ")
print(type(a))
print(int(a))
print(float(a))
print(bool(a))
```

```
Enter a String: 24
<class 'str'>
24
24.0
True
```

10.Write code to demonstrate type casting with list elements.

```
str_list=["1","2","3","4"]
int_list=[int(element)for element in str_list]
print(int_list)
```

```
[1, 2, 3, 4]
```

11.Write a program that checks if a number is positive, negative, or zero

```
num = float(input("Enter a number: "))
```

```
if num > 0:
  print("The number is positive.")
elif num < 0:
  print("The number is negative.")
else:
  print("The number is zero.")
```

```
Enter a number: 24
The number is positive.
```

12.Write a for loop to print numbers from 1 to 10.

```
for i in range(1, 11):
  print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

13. Write a Python program to find the sum of all even numbers between 1 and 50

```
total_sum = 0
for number in range(2, 51, 2):
  total_sum += number
print("The sum of all even numbers between 1 and 50 is:", total_sum)
```

```
The sum of all even numbers between 1 and 50 is: 650
```

14. Write a program to reverse a string using a while loop.

```
input_string = input("Enter a string to reverse: ")
reversed_string = ""
index = len(input_string) - 1

while index >= 0:
  reversed_string += input_string[index]
  index -= 1

print("Reversed string:", reversed_string)
```

```
Enter a string to reverse: anshika
Reversed string: akihsna
```

15.. Write a Python program to calculate the factorial of a number provided by the user using a while loop.

```
num = int(input("Enter a non-negative integer to calculate its factorial: "))

if num < 0:
  print("Factorial is not defined for negative numbers.")
elif num == 0:
  print("The factorial of 0 is 1.")
else:
  factorial = 1
  i = 1
  while i <= num:
    factorial *= i
    i += 1
  print(f"The factorial of {num} is {factorial}")
```

```
Enter a non-negative integer to calculate its factorial: 12
The factorial of 12 is 479001600
```