What is Python, and why is it popular?

Python is a high-level, interpreted programming language known for its simple syntax, readability, and versatility. It's popular because it supports multiple paradigms (object-oriented, procedural, functional), has a vast library ecosystem, and is easy to learn for beginners.

Double-click (or enter) to edit

2.What is an interpreter in Python?

An interpreter is the program that reads and executes Python code line by line, translating it into machine-readable instructions.

3.What are pre-defined keywords in Python?

**Pre-defined keywords are reserved words with special meaning in Python (e.g., if, else, for, def, True, None). They cannot be used as variable names.

4.Can keywords be used as variable names? **No, keywords cannot be used as variable names because they are reserved for specific language functions

5.What is mutability in Python?

**Mutability refers to whether an object's value can be changed after creation. Mutable objects can be modified (e.g., lists), while immutable objects cannot (e.g., tuples, strings).

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

**Lists are designed to allow changes like adding, removing, or modifying elements. Tuples are immutable to ensure data integrity and faster performance when used as fixed data sets or keys in dictionaries

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

**== checks for value equality (whether two objects have the same value).

is checks for identity equality (whether two references point to the same object in memory).

8.What are logical operators in Python? The logical operators are:

**and (True if both conditions are True),

or (True if at least one condition is True),

not (negates a condition). 9.What is type casting in Python?

*"Type casting is converting one data type into another, such as using int(), float(), or str() to convert values.

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

**Implicit casting (Type conversion): Python automatically converts types when safe (e.g., adding 2 + 3.0 converts 2 to 2.0).

Explicit casting (Type casting): The programmer manually converts a type using functions like int(), float(), or str().

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

**Conditional statements (if, elif, else) allow the program to execute different blocks of code based on specific conditions, enabling decision-making.

12.How does the elif statement work? **elif (else-if) checks another condition if the previous if or elif was False. It prevents multiple independent if statements by chaining conditions efficiently

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

**for loop: Iterates over a sequence (like a list or range) for a predetermined number of times.

while loop: Repeats as long as a condition remains True, useful when the number of iterations is unknown.

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

**When the number of iterations isn't known in advance—e.g., reading user input until they type "exit" or processing data until a specific condition is met—while is more suitable.

[ ]

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

```
print("Hello, World!")
```

[2]

Write a Python program that displays your name and age

```
name = "Saika Shabir"
age = 25
print("My name is", name, "and I am", age, "years old.")
```

[ ]

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

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

[ ]

Write a program that checks if a given word is a Python keyword

```
import keyword

word = input("Enter a word: ")
if keyword.iskeyword(word):
    print(f"{word} is a Python keyword.")
else:
    print(f"{word} is not a Python keyword.")
```

[ ]

Create a list and tuple in Python, and demonstrate how attempting to change an element works dif

```
# List (mutable)
my_list = [1, 2, 3]
my_list[0] = 100  # Allowed
print("List after modification:", my_list)

# Tuple (immutable)
my_tuple = (1, 2, 3)
try:
    my_tuple[0] = 100  # Will raise an error
except TypeError as e:
    print("Error when trying to modify tuple:", e)
```

[3]
0s

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

```
def modify_list(lst):
    lst.append(100)

def modify_number(n):
    n += 100
    return n

my_list = [1, 2, 3]
modify_list(my_list)
print("List after modification:", my_list)

my_number = 10
new_number = modify_number(my_number)
print("Original number:", my_number)
print("Modified number:", new_number)
```

[ ]

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

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

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b if b != 0 else "Cannot divide by zero")
```

[ ]

Write a program to demonstrate the use of logical operators

```
a = True
b = False

print("a and b:", a and b)
print("a or b:", a or b)
print("not a:", not a)
print("not b:", not b)
```

[ ]

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

```
user_input = input("Enter a value: ")

int_value = int(user_input)
float_value = float(user_input)
bool_value = bool(user_input)

print("Integer:", int_value)
print("Float:", float_value)
print("Boolean:", bool_value)
```

[ ]

Write code to demonstrate type casting with list elements

```
list_of_strings = ["1", "2", "3"]
list_of_integers = list(map(int, list_of_strings))
print("List of integers:", list_of_integers)
```

[ ]

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

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

if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    int("Zero")
```

[ ]

Write a for loop to print numbers from 1 to 10

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

[ ]

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

```
sum_even = sum(i for i in range(2, 51, 2))
print("Sum of even numbers between 1 and 50:", sum_even)
```

[ ]

Write a program to reverse a string using a while loop

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

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

[ ]

Start coding or generate with AI.

[ ]

Start coding or generate with AI.