Que- What is python, and why it is popular?

Ans- Python is a high level programming language for data science due to its simplicity, verstility, extensive ecosystem of open source libraries. There is 137000 libraries in python. It was first developed in 1980's but launched in 1991 by Guido Van Rossum.

Popular:

It is go to programming language for data science due to its simplicity , verstility, extensive ecosystem of open source libraries. Easy to learn and use -> helps to focus on problem solving Extensive libraries for DS->>

- Data Manipulation-> libraries(Panda, Numpy,etc.)
- Data Visualization->seaborn, matplotlib, Plotpy, Scalability & Versatility,

  Strong community,

  Integration Capabilities->Flask, Fast API,

  Open source and cost effective,

  Huge active community

Que.What is an interpreter in Python?

Ans.In Python, an interpreter is the program that reads and executes the code line by line, converting it into a form the computer can understand and run.

Que.What are pre-define keywords in Python?

Ans.Pre-define keywords hold a special meaning and have specific purpose.

Here is the List of keywords in Python:

False class from or None continue global pass True def if raise and del import return as elif in try assert else is while async except lambda with await finally nonlocal yield break for not

```
#Que. Can keyword be used as variable names?

#Ans. No keywords cannot be used as variables
#Ex. If a=print
# and we give a commond >> print("hello world") it will not print hello world b
```

Que. What is mutability in Python?

Ans. Mutability :- It means that the given data can be changed if anything can be edited.

Here []>>square bracket is used.

Objects/container whose state or value can be changed after they are created are called mutable/container.

```
#Ex.. List comes under mutable object/ supports item assignment.
```

Que.Why are lists mutable and tuples immutable in Python?

Ans. Lists are mutable because they are designed for dynamic data manipulation, while tuples are immutable to ensure data integrity and enable performance optimizations.

Ex. my_list=[1,2,3]

my_list.append(4)

my_list becomes [1,2,3,4]

Why TUPLES are Immutable

Tuples are designed for fixed, constant collections of items—things that shouldn't change once created.

Major reasons:

1. Tuples Enable Hashability

A tuple can be a dictionary key, as long as all its contents are also immutable:

coords = {(10, 20): "point"}

If tuples were mutable, using them as keys would break the dictionary:

Dictionary hashes must NEVER change.

A mutable tuple could change its hash ⟶ chaos.

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

Ans. "==" operators--> Value Equality (Checks if two variable have the same value)

Ex. a=[1,2,3] b=[1,2,3] print(a==b) True

#"is" Operators-->>Identity Equality(Checks if two variables refers to the same object in the memory)

Ex a=(1,2,3,) b=(1,2,3) print(a is b) #False

Que.What are logical operators in Python?

Ans.Logical operators in Python are used to combine conditional statements and control the flow of logic in your code.

✅ Practical Examples x = 5 y = 10

and

print(x > 0 and y > 0) # True

or

print(x > 10 or y > 5) # True

Que.What is type casting in Python?

Ans. Type casting in Python means converting a value from one data type to another. Python provides built-in functions to perform these conversions.

Que.What is the difference between implicit and explicit type casting? Ans. Implicit Type Casting (Automatic Conversion)

- Python automatically converts one data type to another during operations when needed.
- Use: To avoid data loss and ensure smooth execution.

Example x = 5 # int y = 2.5 # float result = x + y # Python converts x to float print(result) # 7.5 print(type(result)) # <class 'float'>

Explicit Type Casting (Manual Conversion)

- You manually convert a value from one type to another using functions like int(), float(), str(), etc.
- Use: When you need control over the data type, especially with user input or mixed types.

Example a = "10" b = int(a) # Convert string to int print(b + 5) # 15

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

Ans.The purpose of conditional statements in Python is to control the flow of your program by making decisions based on conditions. They allow your code to respond dynamically to different inputs or situations.

- They help your program choose between actions.
- They make your code interactive and intelligent.
- They're essential for loops, functions, and real-world logic

x = 10

if x > 0: print("Positive number") elif x == 0: print("Zero") else: print("Negative number")

Que.How does the elif statement work?

Ans.The elif statement in Python stands for "else if" and is used to check multiple conditions after an initial if. It helps your code make decisions by testing one condition at a time, in order.

How elif works:

- Python checks conditions top to bottom.
- As soon as it finds one that's True, it runs that block and skips the rest.

- You can use multiple elif statements between if and else.

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

Ans.A for loop is used when you know how many times you want to repeat something or when you are iterating over a collection.

A while loop is used when you want to repeat something until a condition changes, and you don’t necessarily know in advance how many repetitions are needed.

For Loop

Iterates over a sequence (like a list, range, or string).

Number of iterations is usually predetermined or tied to the length of the sequence.

Example (Python):

for i in range(5): print(i)

This will run exactly 5 times.

While Loop

Continues to run as long as a condition is true.

Number of iterations may be unknown and depends on logic inside the loop.

Example:

i = 0 while i < 5: print(i) i += 1

This stops only when the condition becomes false.

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

Ans.A while loop is more suitable when you don’t know in advance how many times the loop will need to run and the repetition depends on something changing over time. Example Scenario: Waiting for User Input Imagine you’re writing a program that keeps asking a user to enter a password until they enter the correct one. You can’t know how many times they’ll get it wrong — it might be once, or it might be 20 times. So a while loop is ideal: password = "" correct = "OpenSesame"

while password != correct: password = input("Enter password: ")

print("Access granted!")

Why a while loop fits:

The loop continues until a condition becomes true (password == correct).

Practical Questions

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

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

```
Hello,World!
```

```
# Write a Python program that displays your name and age.
name=("Priya")
print(name)
age=26
print(age)
```

```
Priya
26
```

```
#Write code to print all the pre-defined keywords in Python using the keyword l
import keyword
keywords= keyword.kwlist
print(keywords)
```

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

```
#Write a program that checks if a given word is a Python keyword.

import keyword

# Input from the user
word = input("Enter a word: ")

# Check if it's a Python keyword
if keyword.iskeyword(word):
    print(f"'{word}' is a Python keyword.")
else:
    print(f"'{word}' is NOT a Python keyword.")
```

```
Enter a word: True
'True' is a Python keyword.
```

```
# Create a list and tuple in Python,and demonstrate how attempting to change ar

#creating a list and tuple

# List (mutable)
```

```
my_list = [1, 2, 3]
print("Original List:", my_list)
my_list[0] = 10
print("Modified List:", my_list)

# Tuple (immutable)
my_tuple = (1, 2, 3)
print("\nOriginal Tuple:", my_tuple)
my_tuple[0] = 10
```

```
Original List: [1, 2, 3]
Modified List: [10, 2, 3]

Original Tuple: (1, 2, 3)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2316172013.py in <cell line: 0>()
     12 my_tuple = (1, 2, 3)
     13 print("\nOriginal Tuple:", my_tuple)
---> 14 my_tuple[0] = 10

TypeError: 'tuple' object does not support item assignment
```

```
# Write a function to demonstrate the behavior of mutable and immutable argumer
def modify_values(a, b):
    a = a + 10          # int is immutable
    b.append(10)        # list is mutable
    print("Inside function: a =", a, ", b =", b)

num = 5
lst = [1, 2, 3]

print("Before function call:", num, lst)
modify_values(num, lst)
print("After function call:", num, lst)
```

```
Before function call: 5 [1, 2, 3]
Inside function: a = 15 , b = [1, 2, 3, 10]
After function call: 5 [1, 2, 3, 10]
```

```
# Write a program that performs basic arithmetic operations on two user-input r
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

print("add:", a + b)
print("sub:", a - b)
print("mul:", a * b)
print("div:", a / b)
```

```
Enter first number: 89
Enter second number: 53
add: 142
sub: 36
mul: 4717
div: 1.679245283018868
```

```
# Write a program to demonstrate the use of logical operators.
x = True
y = False

print("x and y:", x and y)
print("x or y:", x or y)
print("not x:", not x)
```

```
x and y: False
x or y: True
not x: False
```

```
# Write a Python program to convert user input from string to integer, float, a
user_input = input("Enter a value: ")

print("As string:", user_input)
print("As integer:", int(user_input))
print("As float:", float(user_input))
print("As boolean:", bool(user_input))
```

```
Enter a value: 30
As string: 30
As integer: 30
As float: 30.0
As boolean: True
```

```
#Write code to demonstrate type casting with list elements.
my_list = ["1", "2", "3", "4"]

int_list = [int(x) for x in my_list]
float_list = [float(x) for x in my_list]

print("Original:", my_list)
print("As integers:", int_list)
print("As floats:", float_list)
```

```
Original: ['1', '2', '3', '4']
As integers: [1, 2, 3, 4]
As floats: [1.0, 2.0, 3.0, 4.0]
```

```
# 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:
    print("Zero")
```

```
Enter a number: 0
Zero
```

```
# 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
```

```
#Write a Python program to find the sum of all even numbers between 1 and 50.
total = 0
for i in range(1, 51):
    if i % 2 == 0:
        total += i
print("Sum of even numbers:", total)
```

```
Sum of even numbers: 650
```

```
#Write a program to reverse a string using a while loop.
text = input("Enter a string: ")
reverse = ""
i = len(text) - 1
while i >= 0:
    reverse += text[i]
    i -= 1
print("Reversed string:", reverse)
```

```
Enter a string: hari
Reversed string: irah
```

```
#Write a Python program to calculate the factorial of a number provided by the
num = int(input("Enter a number: "))
fact = 1
```

```
i = 1
while i <= num:
    fact *= i
    i += 1
print("Factorial of", num, "is", fact)
```

```
Enter a number: 8
Factorial of 8 is 40320
```

Start coding or generate with AI.