Strings, lists and loops

Strings in Python

Strings in Python are sequences of characters enclosed in quotes, immutable and they are one of the most commonly used data types.

```
s1 = 'single quotes'
s2 = "double quotes"
s3 = """triple quotes
     for multi-line"""
```

1. Concatenation

```
first = "Data"
second = "Engineer"
full = first + " " + second
print(full)  # Output: Data Engineer
```

2. Length

```
s = "Hello"
print(len(s))  # Output: 5
```

3. Access Characters

```
s = "Python"
print(s[0])   # Output: P
print(s[-1])  # Output: n
```

4. Slicing

```
s = "DataEngineer"
print(s[0:4])     # Output: Data
print(s[4:])      # Output: Engineer
print(s[::-1])    # Output: reenignEataD (reversed)
```

name = 'Ankit'

name[startindex : endindex : stepsize]

For reverse slicing give negative value of stepsize -> negative indexing

5. String methods

```
s = "  hello world  "

print(s.upper())       # HELLO WORLD
print(s.strip())       # hello world
print(s.replace("world", "Python"))  # hello Python
print(s.startswith("h"))  # False (due to leading spaces)
print("world" in s)    # True
```

Strings are immutable

Name*2 ->

Questions : **Check if the first half of the string is the reverse of the second half.**

```
s = "abccba"
# Output: True
# First half = "abc", Second half = "cba" → reverse of "abc"
```

Number functions

** for power

`Pow()`

Abs

Round

Import math module and use math functions

Function	Description
math.sqrt(x)	Square root of x
math.ceil(x)	Rounds **up** to the nearest integer
math.floor(x)	Rounds **down** to the nearest integer
math.trunc(x)	Truncates decimal part (like int(x))
math.log(x)	Natural log (base e)
math.log10(x)	Logarithm base 10
math.exp(x)	e raised to the power of x
math.fabs(x)	Absolute value as float
math.isfinite(x)	Checks if x is neither inf nor NaN
math.pi	The constant π ≈ 3.14159
math.e	The constant e ≈ 2.71828

List in Python

A list in Python is an ordered, mutable collection of items. Lists can contain elements of any data type—integers, strings, other lists, even functions.

☑ Creating a List

```
my_list = [1, 2, 3, 4]
mixed_list = [1, "two", 3.0, [4, 5]]
empty_list = []
```

🔁 Accessing Elements

```
nums = [10, 20, 30, 40]

print(nums[0])    # 10
print(nums[-1])   # 40

# slicing
print(nums[1:3])  # [20, 30]
```

Modifying a List

```
 # add items
nums.append(50)        # Add at end
nums.insert(1, 15)     # Insert at index 1

 # remove items
nums.remove(20)        # Removes first occurrence of 20
popped = nums.pop()    # Removes and returns last item
popped = nums.pop(i)    # Removes element at index i
del nums[0]            # Deletes item at index 0

 # updating the element
nums = [10, 20, 30, 40]
nums[1] = 99
print(nums)        # [10, 99, 30, 40]

nums[2:4] = [300, 400]
print(nums)        # [10, 99, 300, 400]
```

➕ Combining Lists

```
a = [1, 2]
b = [3, 4]
c = a + b               # [1, 2, 3, 4]
a.extend(b)             # a becomes [1, 2, 3, 4]
```

■ Multi-dimensional Lists (Nested Lists)

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

print(matrix[0][1])      # 2
matrix[1][2] = 60        # Update middle row, last element
print(matrix)
# Output: [[1, 2, 3], [4, 5, 60], [7, 8, 9]]
```

📊 Useful List Methods

```
list1 = [3, 1, 4, 2]

list1.sort()            # Sort ascending
list1.reverse()         # Reverse order
print(len(list1))       # Length of list
print(list1.count(3))   # Count occurrences of 3
print(list1.index(4))   # Find index of 4
```

◇ split() Method in Python

The split() method is used to **split a string into a list** based on a **separator** (default is space).

```
string.split(separator, maxsplit)
```

- separator → (Optional) Delimiter to split by (default is space).
- maxsplit → (Optional) Maximum number of splits.

```
text = "Data Engineering Rocks"
words = text.split()
print(words)
# Output: ['Data', 'Engineering', 'Rocks']

line = "apple,banana,cherry"
fruits = line.split(",")
print(fruits)
# Output: ['apple', 'banana', 'cherry']
```

Split method -> email

sort

Assigments : reverse indexing eg get something from a url

Get question name from coding problem url

Join Method

The .join() method in Python is used to combine elements of a list (or any iterable) into a single string, using a specified separator.

```
separator.join(iterable)
```

- separator: The string you want to insert between elements (like space, comma, newline, etc.)
- iterable: A list or tuple of strings to join

Example: Join list of words with a space

```
words = ["Data", "Engineering", "is", "awesome"]
sentence = " ".join(words)
print(sentence)
# Output: Data Engineering is awesome
```

Example: Join with comma

```
items = ["apple", "banana", "cherry"]
result = ", ".join(items)
print(result)
# Output: apple, banana, cherry
```

Join with newline (for multi-line string)

```
lines = ["Line 1", "Line 2", "Line 3"]
output = "\n".join(lines)
print(output)
# Output:
# Line 1
# Line 2
# Line 3
```

⚠ Important Notes:

- All items in the iterable must be strings. If not, convert them first:

Question: You are given a sentence with extra spaces between words.
Clean up the sentence so that:

- There's only one space between words.
- No leading or trailing spaces.

```
s = "   Data    Engineering   is   awesome   "

s= " ".join(s.split())
# Output: "Data Engineering is awesome"
```

for loop

A for loop in Python is used to iterate over a sequence (like a list, tuple, string, or range) and execute a block of code for each element.

```
for item in sequence:
    # do something with item
```

Example

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Looping with range

```
for i in range(5):  # from 0 to 4
    print("i is", i)
```

Courses : ['sql','python']

Course 1 is sql

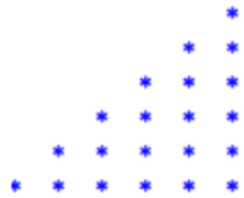
Course 2 is python

For course in courses

Range method

Create right angle triangle

Print all even number till 100



How many time SQL is there in this paragraph

SQL is a powerful language used to manage and query relational databases. With SQL you can easily retrieve specific data using SELECT statements, update records with UPDATE, or remove unwanted entries using DELETE. Learning sql is essential for data analysts, as most data stored in companies resides in SQL based systems. Advanced SQL techniques like window functions and common table expressions allow for more complex data manipulation. Whether you're building dashboards or running reports, SQL is the foundation of effective data analysis.

Insert 100 in a sorted list

```
a = [10,40,97,110,200]
```

After insert list should remain sorted: with and without lists insert function

Without insert

```
a.append(None)
            #(5,3,-1)
for j in range(len(a)-1,i,-1):
    if j>i :
        a[j]=a[j-1]

a[i]=100

print(a)
```

Talk about edge cases : if first element > 100 , or last element less than 100 or if list is empty

```
a = [10, 40, 97, 110, 200]
x = 100

if not a:
   a.append(x)
elif x <= a[0]:
   a.insert(0, x)
elif x >= a[-1]:
   a.append(x)
else:
   for i in range(len(a)):
      if x < a[i]:
         a.insert(i, x)
         break

print(a)
```

Loop on string:

You can **iterate over each character** in a string using a for loop.

☑ Example: Loop Through Characters

```
word = "Python"

for char in word:
    print(char)

#output
P
y
t
h
o
n
```

☑ Example: Count Vowels in a String

```
text = "Data Engineer"
vowels = "aeiouAEIOU"
count = 0

for char in text:
    if char in vowels:
        count += 1

print("Number of vowels:", count)
```

while loop

A while loop in Python repeatedly executes a block of code **as long as a given condition is True**.

```
while condition:
    # Code block to execute
```

Example

```
count = 1
while count <= 5:
    print("Count is:", count)
    count += 1
```

Output

```
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
```

🗀 Scenario: Wait until a file is uploaded (e.g., by another process or user)

```python
import os
import time

file_path = "/path/to/your/file.csv"

print("⏳ Waiting for file to appear...")

while not os.path.exists(file_path):
    print("File not found. Checking again in 5 seconds...")
    time.sleep(5)

print("☑ File found! Proceeding with processing.")
```

🔍 Why while is ideal instead of For loop:

- You don't know **when** the file will appear.
- The loop should continue **as long as the file is missing**.
- You need to check the condition dynamically, not for a fixed number of times.

List Comprehension :

List comprehensions in Python are a concise way to create lists by applying an expression to each item in an iterable, optionally including a condition.

```python
a = [10,40,97,110,200]
new_list= [number for number in a if number%2 ==0 ]
new_list = [ "Even" if number%2==0 else "odd" for number in a  ]
```