**Day 5 – (10-06-2025)**

**Python Refresher & Advanced Python for Data Engineering**

* **Core Python Refresher**

**Python Basics**

* **Features:** Easy syntax, interpreted, dynamic typing, huge libraries.
* **First Program:**
* print("Hello, Data Engineers!")
* **Variables & Data Types**
* name = "Laks" # String
* age = 21 # Integer
* score = 94.5 # Float
* passed = True # Boolean

**Control Structures**

for i in range(3):

print("Looping", i)

if score > 90:

print("Excellent!")

elif score >= 75:

print("Good job")

else:

print("Keep learning")

**Collections Overview**

| **Type** | **Example** |
| --- | --- |
| List | marks = [85, 78, 92] |
| Dictionary | {"name": "Sakshi", "score": 95} |
| Set | {1, 2, 3} |
| Tuple | ("Alice", [85, 78, 92]) |

**Functions & Lambda**

def greet(name="Friend"):

return f"Hello, {name}!"

add = lambda x, y: x + y

print(add(10, 5)) # Output: 15

**OOP in Python**

class Student:

def \_\_init\_\_(self, name, marks):

self.name = name

self.marks = marks

def average(self):

return sum(self.marks) / len(self.marks)

s = Student("Sakshi", [85, 90, 95])

print(s.average()) # Output: 90.0

**File Handling & Exception**

try:

with open("data.csv", "r") as f:

content = f.read()

except FileNotFoundError:

print("File not found!")

**Modules and Packages**

# mymodule.py

def add(a, b):

return a + b

# main.py

import mymodule

print(mymodule.add(2, 3)) # Output: 5

**Data Engineering Essentials**

**CSV / JSON Handling with Pandas**

import pandas as pd

df = pd.read\_csv("students.csv")

print(df.head())

df.to\_json("students.json")

**MySQL with Python**

import mysql.connector

conn = mysql.connector.connect(

host="localhost", user="root", password="1234", database="school"

)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students")

for row in cursor.fetchall():

print(row)

**Data Cleaning with NumPy & Pandas**

import numpy as np

import pandas as pd

arr = np.array([[1, np.nan, 3], [4, 5, np.nan]])

df = pd.DataFrame(arr)

df.fillna(0, inplace=True)

**Task 1 – Students Marks Management System**

students = [

("lakshan",[88,89,90]),

("haley",[89,88,90]),

("harish",[90,91,85])

]

print("Average Marks of Each Student: ")

topper\_name = ""

highest\_avg = 0

for name, marks in students:

avg = sum(marks)/len(marks)

print(f"{name} - Average: {avg:.2f}")

if avg > highest\_avg:

highest\_avg = avg

topper\_name = name

print("\nTopper:",topper\_name)

students.append(("Rock", [80,89,95]))

for i in range(len(students)):

if students[i][0] == "Jennifer":

students[i][1][1] = 89

break

print("\n Updated Student's List: ")

for name, marks in students:

print(f"{name}: {marks}")

student\_stats = []

for name, marks in students:

total = sum(marks)

avg = total/len(marks)

student\_stats.append((name, total, round(avg, 2)))

student\_stats.sort(key=lambda x: x[2], reverse=True)

print("\n Students sorted by Average Marks: ")

for name, total, avg in student\_stats:

print(f"{name} - Total: {total}, Average: {avg}")

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 2 – Smart City Project**

import numpy as np

temps = np.array([

[28, 30, 29, 31, 32, 33, 34],

[35, 34, 36, 37, 38, 36, 35],

[22, 23, 24, 25, 26, 27, 28],

[31, 30, 29, 28, 27, 26, 25],

[39, 38, 40, 41, 42, 40, 39]

])

days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

#1. Function: Average temperature of a city for the week

def average\_temperature(city\_index):

return np.mean(temps[city\_index])

# 2. Function: Hottest day index for a city

def hottest\_day(city\_index):

return np.argmax(temps[city\_index])

# 3. Loop through all cities and print average temperature and hottest day

print("City-wise Average Temperature and Hottest Day:")

for i in range(5):

avg\_temp = average\_temperature(i)

hot\_day\_index = hottest\_day(i)

hot\_day\_name = days[hot\_day\_index]

print(f"City {i+1}: Avg Temp = {avg\_temp:.2f}°C, Hottest Day = {hot\_day\_name} (Index: {hot\_day\_index})")

print()

#4. Function: Day-wise average temperature across all cities

def day\_wise\_avg():

return np.mean(temps, axis=0)

# Print day-wise average

day\_avg = day\_wise\_avg()

print("Day-wise Average Temperatures Across All Cities:")

for i in range(7):

print(f"{days[i]}: {day\_avg[i]:.2f}°C")

print()

#5. Loop to find:

# City with highest weekly average

city\_avgs = [average\_temperature(i) for i in range(5)]

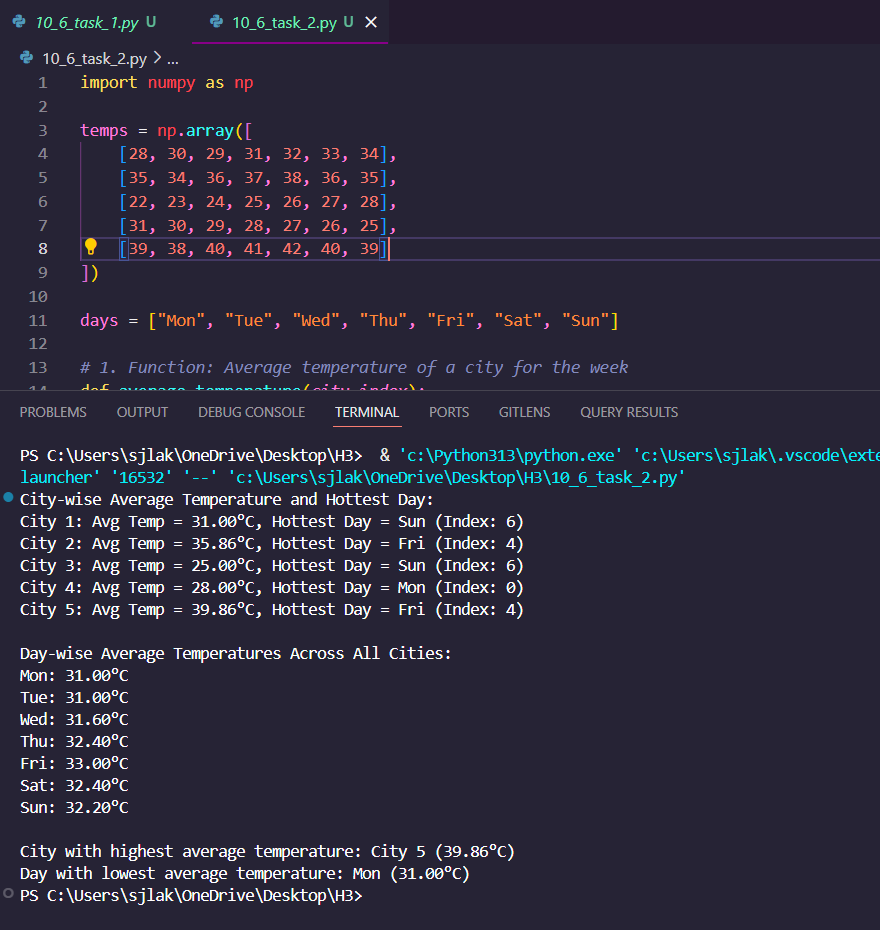
highest\_city\_index = np.argmax(city\_avgs)

# Day with lowest average temperature

lowest\_day\_index = np.argmin(day\_avg)

print(f"City with highest average temperature: City {highest\_city\_index + 1} ({city\_avgs[highest\_city\_index]:.2f}°C)")

print(f” Day with lowest average temperature: {days[lowest\_day\_index]} ({day\_avg[lowest\_day\_index]:.2f}°C)")



**Python Lists**

In Python, **lists** are one of the most used data structures. A list stores a collection of items in a specific order. You can think of it like a container that holds different types of values.

**📦 What is a List?**

A list is a collection of items separated by commas and enclosed in square brackets [ ].

fruits = ['apple', 'banana', 'cherry']

numbers = [10, 20, 30, 40]

mixed = ['hello', 123, True]

* Lists can store different data types.
* Indexing starts from **0** (just like string indexing).

**🔍 Accessing List Elements**

Use square brackets with the index number to get an item from the list.

colors = ['red', 'green', 'blue', 'yellow']

print(colors[0]) # red

print(colors[2]) # blue

print(colors[-1]) # yellow (last item)

You can also access a **range** of items using **slicing**:

print(colors[1:3]) # ['green', 'blue']

**✏️ Updating List Elements**

You can change any item in the list by assigning a new value at a specific index.

animals = ['cat', 'dog', 'rabbit']

animals[1] = 'parrot'

print(animals) # ['cat', 'parrot', 'rabbit']

**➕ Adding Items to a List**

Use the append() method to add an item to the end:

languages = ['Python', 'Java']

languages.append('C++')

print(languages) # ['Python', 'Java', 'C++']

You can also use insert(index, value) to add an item at a specific position:

languages.insert(1, 'JavaScript')

print(languages) # ['Python', 'JavaScript', 'Java', 'C++']

**❌ Removing Items from a List**

**Using del:**

numbers = [10, 20, 30, 40]

del numbers[2]

print(numbers) # [10, 20, 40]

**Using remove():**

names = ['Alice', 'Bob', 'Charlie']

names.remove('Bob')

print(names) # ['Alice', 'Charlie']

**Using pop():**

scores = [100, 90, 80]

scores.pop() # removes last item

scores.pop(0) # removes item at index 0

print(scores) # [90]

**📏 Useful List Functions**

**1. len(): Count of items**

print(len(['a', 'b', 'c'])) # 3

**2. max() / min(): Highest / Lowest value**

marks = [55, 89, 73]

print(max(marks)) # 89

print(min(marks)) # 55

**3. list(): Convert other types to list**

text = "hello"

print(list(text)) # ['h', 'e', 'l', 'l', 'o']

**🔁 List Operations**

| **Operation** | **Example** | **Result** |
| --- | --- | --- |
| Length | len([1, 2, 3]) | 3 |
| Concatenation | [1, 2] + [3, 4] | [1, 2, 3, 4] |
| Repetition | ['hi'] \* 3 | ['hi', 'hi', 'hi'] |
| Membership | 'apple' in ['apple', 'mango'] | True |
| Iteration | for x in [1, 2, 3]: print(x) | prints 1 2 3 |

**🧰 More List Methods**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| append(x) | Adds x to end | mylist.append(5) |
| extend(seq) | Adds all items from seq | mylist.extend([6, 7]) |
| count(x) | Counts how many times x appears | mylist.count(2) |
| index(x) | Finds the first index of x | mylist.index('apple') |
| reverse() | Reverses the list in place | mylist.reverse() |
| sort() | Sorts the list | mylist.sort() |

**✅ Examples to Try**

shopping = ['milk', 'bread', 'eggs']

shopping.append('butter')

shopping.remove('bread')

shopping.insert(1, 'juice')

print(shopping)

**Output:**

['milk', 'juice', 'eggs', 'butter']