CODE AND SUMMARY OF CLASS 3

Sahir Ahmed Sheikh

Saturday (2 – 5)

Teachers:

Muhammad Bilal And Ali Aftab Sheikh

Code And Summary Of Class 3 – Saturday (2 – 5) | Quarter 3

Assalamu Alaikum! Hope you all are doing well. First of all, Ramadan Mubarak to everyone. Today marks our first class in Ramadan. Due to Ramadan, our class timing has been reduced, and now our classes will be held from 11 AM to 1 PM during Ramadan.

Our instructors emphasized the importance of being on time since we previously started classes 10-15 minutes late. With our reduced time, punctuality is necessary.

Before proceeding, we **reviewed the last class**. We discussed multiple topics and assignments, and sir also introduced a form for submitting assignments. The class repository and form were shown again for reference.

Last Class Assignments:

1. **Unit Converter**
2. **Presentation on Machine Learning, Deep Learning, and Generative AI**

Many students had not submitted their assignments, so everyone was reminded to **complete and submit them**. Assignment details and the submission form are available in **Discord (Announcements Section).**

Important Updates:

- Students will be filtered based on their **assignment submissions** at the end of the quarter.
- **Active students** are those who submit all assignments.
- If you **cannot see the Discord channel**, check **settings** and **select your class timing** to access relevant sections.
- **All queries should be posted on Discord**, where teachers are active and can assist.

Topics Covered Today

📌 **Class Code:** Class Notebook

1. Lists in Python
2. Tuples in Python
3. Sets and Frozen Sets
4. Input Handling in Python
5. Implicit vs Explicit Type Casting
6. Modules in Python
7. Functions and Return Statements
8. Using the Random Module
9. String Formatting and f-strings
10. Function with Named Parameters

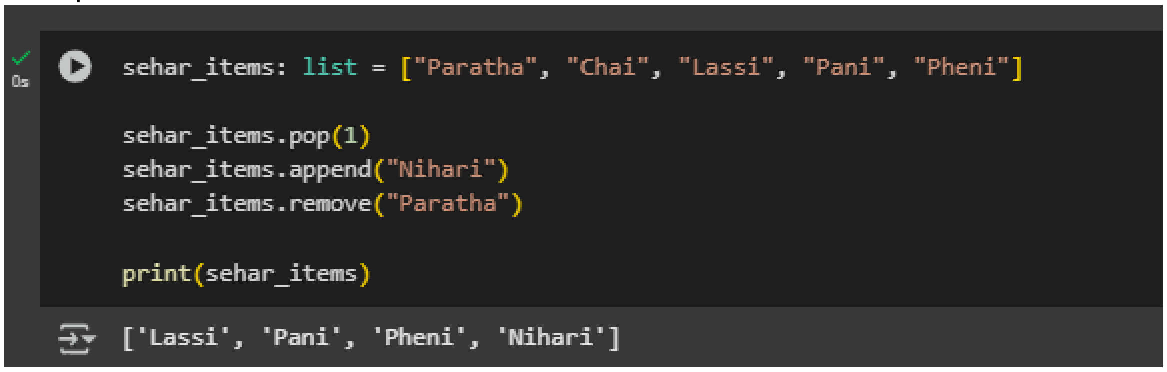
1. Lists in Python

- Lists are **ordered collections** that store multiple data types.
- Lists are **mutable**, meaning we can modify their elements.
- Indexing in lists starts from **0**.

Basic List Operations

- Adding an element using .append()
- Removing an element using .remove()
- Removing the last element using .pop()

Example:



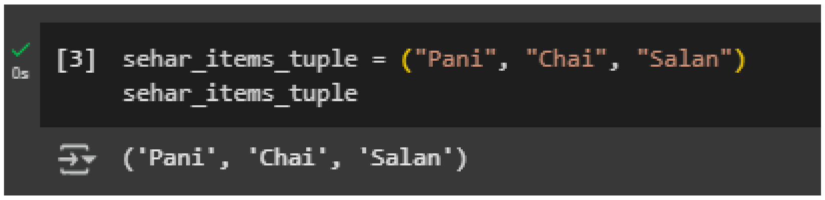
```
sehar_items: list = ["Paratha", "Chai", "Lassi", "Pani", "Pheni"]

sehar_items.pop(1)
sehar_items.append("Nihari")
sehar_items.remove("Paratha")

print(sehar_items)
```

```
['Lassi', 'Pani', 'Pheni', 'Nihari']
```

2. Tuples in Python

- Tuples are similar to lists but **immutable**.
- Used for storing fixed collections of data.

Example:



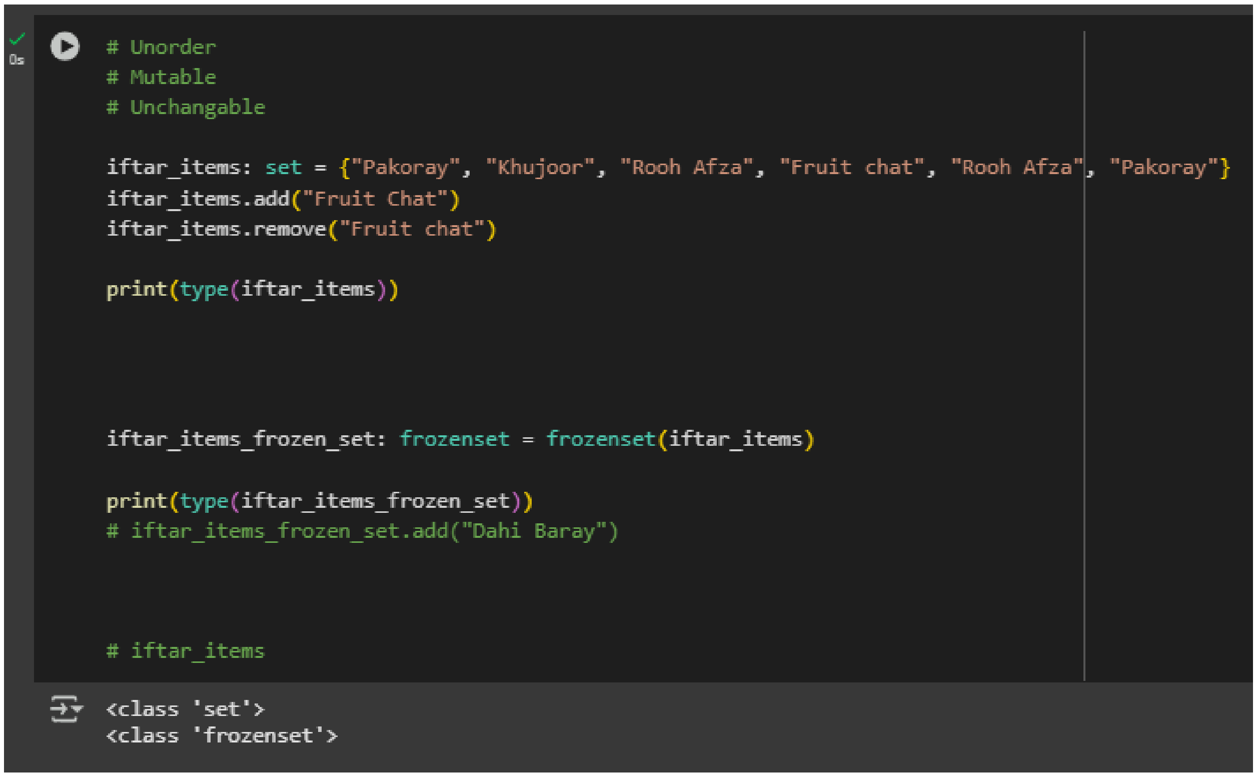
```
[3] sehar_items_tuple = ("Pani", "Chai", "Salan")
    sehar_items_tuple
```

```
('Pani', 'Chai', 'Salan')
```

3. Sets and Frozen Sets

- **Sets** store unique values and do not follow an order.
- **Frozen sets** are immutable versions of sets.
- We can add or remove values in a set but not in a frozen set.

Example:



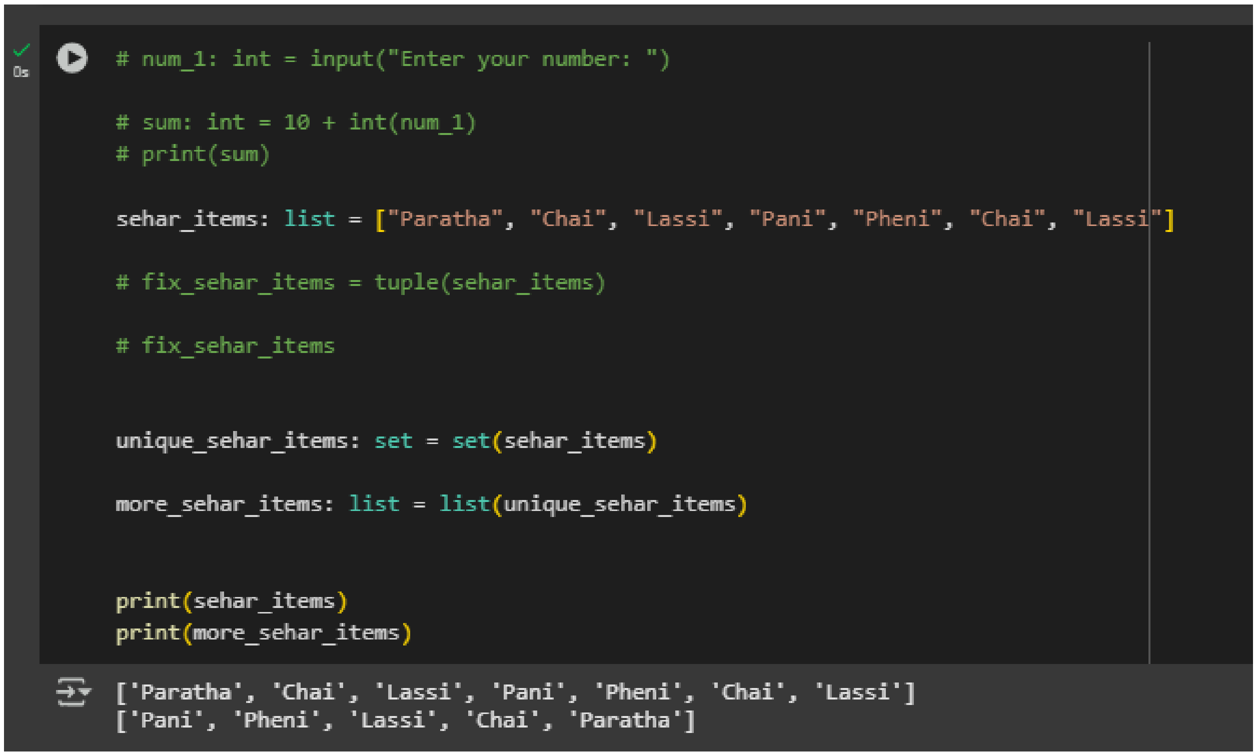
```
# Unorder
# Mutable
# Unchangable

iftar_items: set = {"Pakoray", "Khujoor", "Rooh Afza", "Fruit chat", "Rooh Afza", "Pakoray"}
iftar_items.add("Fruit Chat")
iftar_items.remove("Fruit chat")

print(type(iftar_items))



iftar_items_frozen_set: frozenset = frozenset(iftar_items)

print(type(iftar_items_frozen_set))
# iftar_items_frozen_set.add("Dahi Baray")



# iftar_items
```

```
<class 'set'>
<class 'frozenset'>
```

4. Input Handling in Python

- We take input using **input()**.
- By default, Python stores input as a **string**.
- We must convert the type using **int(), float(), etc.** for calculations.

Example:



```python
# num_1: int = input("Enter your number: ")

# sum: int = 10 + int(num_1)
# print(sum)

sehar_items: list = ["Paratha", "Chai", "Lassi", "Pani", "Pheni", "Chai", "Lassi"]

# fix_sehar_items = tuple(sehar_items)

# fix_sehar_items


unique_sehar_items: set = set(sehar_items)

more_sehar_items: list = list(unique_sehar_items)


print(sehar_items)
print(more_sehar_items)
```

```
['Paratha', 'Chai', 'Lassi', 'Pani', 'Pheni', 'Chai', 'Lassi']
['Pani', 'Pheni', 'Lassi', 'Chai', 'Paratha']
```

5. Implicit vs Explicit Type Casting

- **Implicit Casting**: Python automatically converts types when necessary.
- **Explicit Casting**: We manually change the type.

Implicit Type Casting

Python automatically converts one data type to another when needed.

```python
num1 = 10   # Integer
num2 = 2.0  # Float

sum_value = num1 / num2  # Python converts result to float
print(type(sum_value))   # Output: <class 'float'>
```

Explicit Type Casting

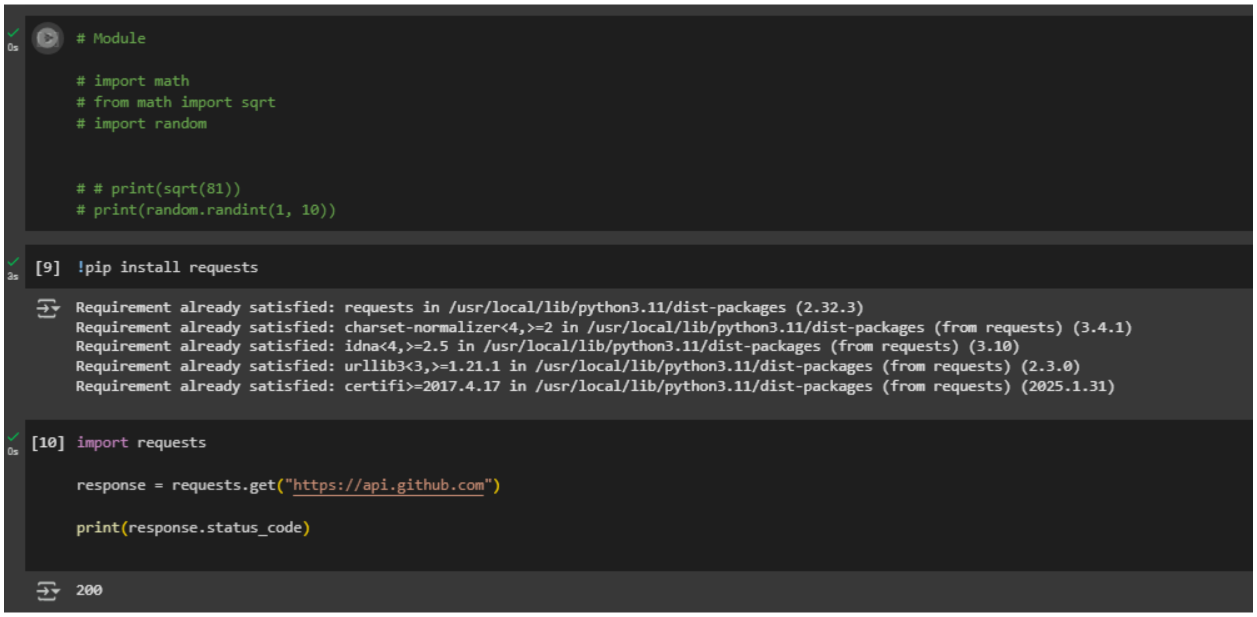
We manually change the type using functions like int(), float(), or str().

```python
num = "10"  # String type

converted_num = int(num)  # Explicitly converting string to integer
print(type(converted_num))  # Output: <class 'int'>
```

6. Modules in Python

- A module is a **.py file** containing **functions, classes, and variables**.
- Types of modules:
    - **Built-in** (math, random, os, etc.)
    - **User-defined** (created by users)
    - **Third-party** (installed using pip)

Example:



```
# Module

# import math
# from math import sqrt
# import random


# # print(sqrt(81))
# print(random.randint(1, 10))
```

```
[9] !pip install requests
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.1.31)
```

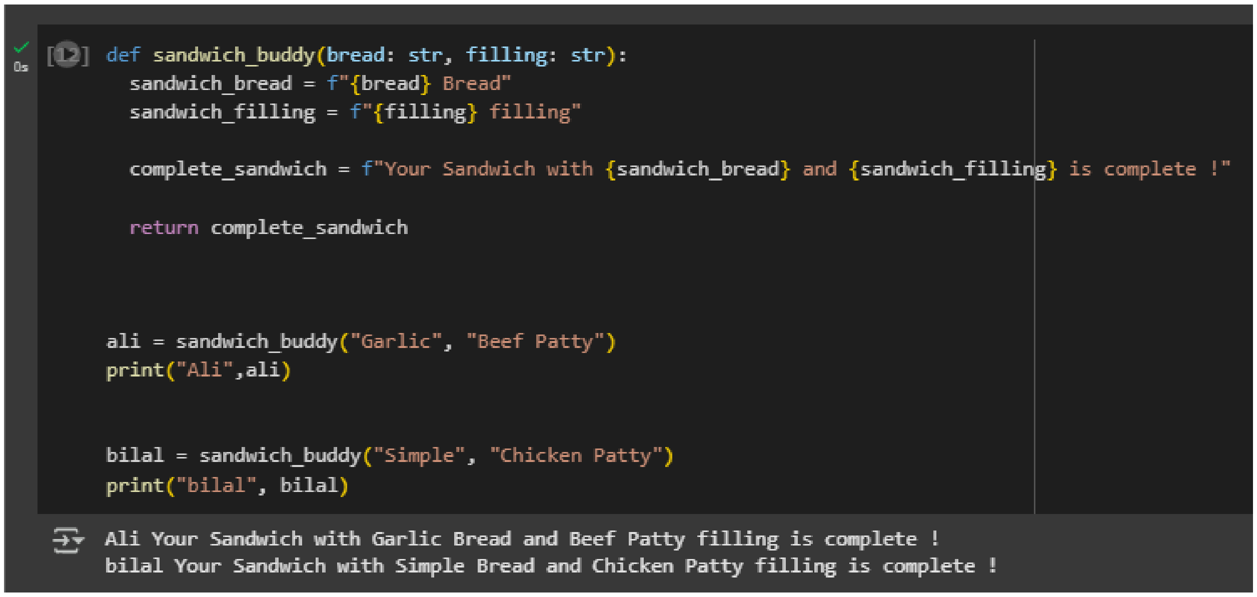
```
[10] import requests

     response = requests.get("https://api.github.com")

     print(response.status_code)
```

```
200
```

7. Functions and Return Statements

- **Functions** are reusable blocks of code that perform a specific task.
- The **return** statement sends a value back to the function caller.

Example:



```python
def sandwich_buddy(bread: str, filling: str):
  sandwich_bread = f"{bread} Bread"
  sandwich_filling = f"{filling} filling"

  complete_sandwich = f"Your Sandwich with {sandwich_bread} and {sandwich_filling} is complete !"

  return complete_sandwich



ali = sandwich_buddy("Garlic", "Beef Patty")
print("Ali",ali)


bilal = sandwich_buddy("Simple", "Chicken Patty")
print("bilal", bilal)
```

```
Ali Your Sandwich with Garlic Bread and Beef Patty filling is complete !
bilal Your Sandwich with Simple Bread and Chicken Patty filling is complete !
```

8. Using the Random Module

- The **random** module generates random numbers.
- Useful for simulations, testing, and games.

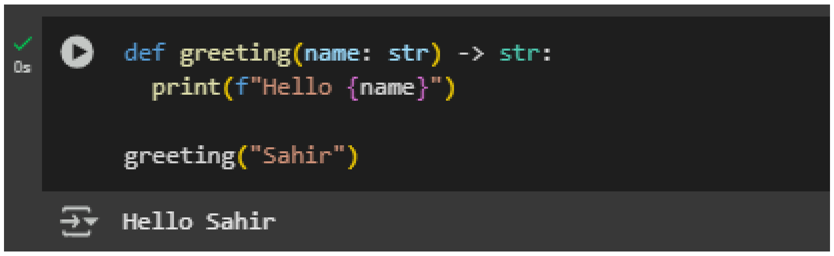
Example: Generating a Random Number

```python
import random

print(random.randint(1, 10))  # Generates a random number between 1 and 10
```

9. String Formatting and f-strings

- **f-strings** allow embedding expressions inside string literals.
- More readable and efficient than concatenation.

Example: Using f-strings for String Formatting



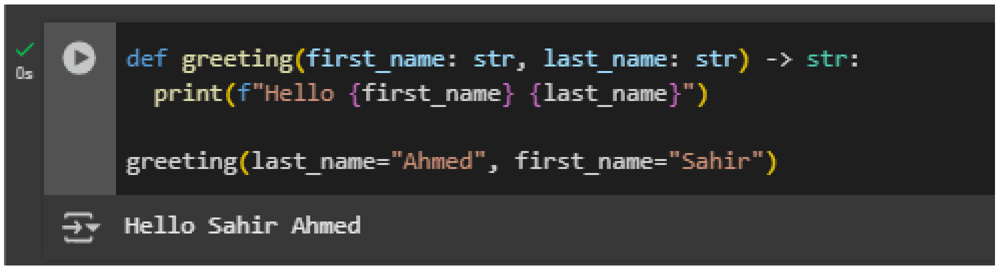
```
def greeting(name: str) -> str:
  print(f"Hello {name}")

greeting("Sahir")
```

```
Hello Sahir
```

10. Function with Named Parameters

- We passed arguments using named parameters for better clarity.

Example: Using Named Parameters in a Function



```
def greeting(first_name: str, last_name: str) -> str:
  print(f"Hello {first_name} {last_name}")

greeting(last_name="Ahmed", first_name="Sahir")
```

```
Hello Sahir Ahmed
```

Engaging Classroom Experience

During the class, there were multiple discussions and interactive activities to help students grasp the concepts more effectively. The session included:

- A recap of previous assignments and submissions, ensuring that everyone was up to date.
- Hands-on coding practice where students experimented with Python concepts in real time.
- An engaging Q&A segment where the instructor asked thought-provoking questions to keep the class engaged.
- Conceptual breakdowns using relatable examples, such as making a sandwich to explain function returns and parameters.

This session provided a great opportunity to strengthen foundational Python concepts through practical implementation and peer interaction.

Assignments

✓ **Assignment 1: Modern AI Python Steps**
Revise the first eight steps and learn step nine from Sir Zia's repository:
Learn Modern AI with Python - GitHub Repo

✓ **Assignment 2: Password Strength Meter**
Work on the Password Strength Meter project from the repository:
Password Strength Meter - GitHub Repo

✓ **Assignment 3: Complete Previous Assignments**
Ensure all previous assignments are completed and submitted on LinkedIn and the class submission form.

📅 **Deadline:** Before the next class
📌 **Submission:**

- Upload all assignments on LinkedIn
- Submit all class assignments using this form: Assignment Submission Form

Stay consistent and keep learning!

Thank You for Reading!

Hope you understood Class 3 well.

"Learning never exhausts the mind" – *Leonardo da Vinci*