Unit:3; Class:7

# Unit 3 Review and Preparation for new Unit

Instructor: Musfique Ahmed

# What is Object-Oriented Programming (OOP)?



- A programming paradigm based on **objects** and **classes**.
- Helps in writing **modular, reusable, and maintainable** code.
- Used in **game development, AI, GUI apps, databases, and more**.

## 4 Key OOP Concepts:

1. **Encapsulation** – Protects data inside a class.
2. **Inheritance** – Allows a class to acquire properties from another.
3. **Polymorphism** – Uses the same method for different types.
4. **Abstraction** – Hides complex implementation details.

# Classes and Objects



- **Class:** A blueprint for objects.
- **Object:** An instance of a class.

💡 **Example Code:**

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display_info(self):
        print(f"Car: {self.brand} {self.model}")

my_car = Car("Toyota", "Corolla")
my_car.display_info()
```

# Encapsulation



- Data protection using **private** and **public** attributes.

💡 **Example Code:**

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # Private variable

    def get_balance(self):
        return self.__balance

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount

account = BankAccount(1000)
account.deposit(500)
print(account.get_balance())  # Output: 1500
```

# Inheritance



- Creating a new class from an existing class.

💡 **Example Code:**

```
class Animal:
    def sound(self):
        print("Some generic sound")

class Dog(Animal):
    def sound(self):
        print("Bark!")

dog = Dog()
dog.sound()  # Output: Bark!
```

# Polymorphism



- **Method Overriding:** Redefining a parent method in the child class.

💡 **Example Code:**

```
class Shape:
    def area(self):
        pass  # Placeholder method

class Circle(Shape):
    def area(self, radius):
        return 3.14 * radius * radius

circle = Circle()
print(circle.area(5))  # Output: 78.5
```

# Abstraction



- Hiding implementation details using **abstract base classes**.

💡 **Example Code:**

```
from abc import ABC, abstractmethod

class Vehicle(ABC):
    @abstractmethod
    def start_engine(self):
        pass

class Car(Vehicle):
    def start_engine(self):
        print("Car engine started!")

my_car = Car()
my_car.start_engine()  # Output: Car engine started!
```

# GUI Libraries and Modules in Python

# 1️⃣ Tkinter (Standard Python GUI Library)

Tkinter Project Playlist



✅ **Used for:** Building desktop applications
✅ **Key Widgets:** Button, Label, Entry, Frame, Canvas

💡 **Example Code:**

```
import tkinter as tk

root = tk.Tk()
root.title("Tkinter Example")

label = tk.Label(root, text="Hello, Tkinter!")
label.pack()

root.mainloop()
```

# 2 Streamlit (Web-based Python Apps)



Streamlit Project Playlist

✅ **Used for:** Interactive web applications for data science
✅ **Key Widgets:** st.button(), st.slider(), st.text_input(), st.file_uploader()

💡 **Example Code:**

```
import streamlit as st

st.title("My First Streamlit App")
st.write("This is a simple web app!")

if st.button("Click Me!"):
    st.write("Button clicked!")
```

# 3️⃣ Pygame (Game Development Library)



Pygame Project Playlist

✅ **Used for:** Making 2D games in Python

✅ **Key Features:** Sprites, Animations, Sounds, Events

💡 **Example Code:**

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 400))
pygame.display.set_caption("Pygame Example")

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

# 4 Turtle (Graphics & Animation)



Turtle Project Playlist

✅ **Used for:** Drawing and animation

✅ **Key Functions:** forward(), left(), right(), penup(), pendown()

💡 **Example Code:**

```
import turtle

t = turtle.Turtle()
t.pensize(3)

for _ in range(4):
    t.forward(100)
    t.right(90)

turtle.done()
```

# Project Ideas (Using OOP + GUI Modules)



🟢 **Tkinter Projects:**

1️⃣ **To-Do List App** (Encapsulation for task management)

2️⃣ **Expense Tracker** (Polymorphism for different expense categories)

🟠 **Streamlit Projects:**

3️⃣ **Data Dashboard** (Abstraction for data processing)

4️⃣ **Portfolio Website** (Encapsulation for user details)

🔵 **Pygame Projects:**

5️⃣ **Flappy Bird Clone** (Inheritance for different game objects)

6️⃣ **Brick Breaker Game** (Polymorphism for ball & paddle interactions)

🔴 **Turtle Projects:**

7️⃣ **Drawing App** (Using classes to handle pen movements)

8️⃣ **Maze Solver** (OOP to design paths and obstacles)

# Final Challenge – Pick One Project!



🚀 **Assignment:**

- Select one project from the list.
- Implement **OOP concepts** in it.
- Submit code + short explanation.
- Integrate a **GUI library** with OOP in your project!

# Installation



1. **Install Jupyter** from VS Code extensions.
2. **Install python karnal** (if needed)[the vs code will show to install it]
3. Use the jupyter to see if it works properly
4. Install Pandas [ **pip install pandas** ]
5. Install Numpy [ **pip install numpy** ]
6. Install Matplotlib [ **pip install matplotlib** ]
7. Goto vscode make a file with name prac.ipynb (.ipynb is must)
8. See if everything was installed successfully by importing them-
    - **import pandas as pd**
    - **import numpy as np**
    - **import matplotlib.pyplot as plt**

# Thank You

Do the Quiz Please, you have 10 minutes to do that!