1. Decorators are functions that modify the behaviour of other functions or methods. They are often used to add functionality without modifying the original code structure.

code:-

.def my\_decorator(func):

def wrapper():

print("Something is happening before the function is called.")

func()

print("Something is happening after the function is called.")

return wrapper

@my\_decorator

def say\_whee():

print("Whee!")

say\_whee()

1. Method Overriding: Changing the implementation of a method in a subclass.

Method Overloading: Python doesn't support traditional method overloading (same method name with different parameters), but you can achieve similar behaviour using default arguments or variable-length arguments.

1. Hybrid inheritance

Hybrid inheritance combines multiple types of inheritance to form a more complex hierarchy.

Code:-

class Class1:

def method\_A(self):

print("Method A from class 1")

class Class2(Class1):

def method\_B(self):

print("Method B from class 2")

class Class3(Class1):

def method\_C(self):

print("Method C from class 3")

class Class4(Class2, Class3):

def method\_D(self):

print("Method D from class 4")

obj\_D = Class4()

obj\_D.method\_A()

obj\_D.method\_B()

obj\_D.method\_C()

obj\_D.method\_D()

4.

5. Widgets in streamlit

st.text\_input() - allows the user to input the text

st.slider() - allows the user to select from a range of values

st.checkbox - provide a checkbox

6. Session is an instance where the app's state is controlled. Using the code st.session\_state you can store and maintain variables even when app reruns due to user interaction.

7. To run python file in streamlit use “streamlit run”

8.code:-

import matplotlib.pyplot as plt

Bar Chart

plt.figure(figsize=(8, 4))

plt.bar(labels, values, color=['blue', 'orange', 'green'])

plt.title("Bar Chart Example")

plt.xlabel("Categories")

plt.ylabel("Values")

plt.show()

Pie Chart

plt.figure(figsize=(6, 6))

plt.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)

plt.title("Pie Chart Example")

plt.show()