1. Write a program to count word frequencies in a given text.

```python
text = input("Enter a text: ")

words = text.lower().split()

freq = {}

for word in words:

    freq[word] = freq.get(word, 0) + 1

print("Word Frequencies:")

for word, count in freq.items():

    print(f"{word}: {count}")
```

---

2. Palindrome Checker – Write a program that checks if a given word is a palindrome.

```python
word = input("Enter a word: ").lower()

if word == word[::-1]:

    print("Yes, it's a palindrome!")

else:

    print("No, it's not a palindrome.")
```

---

3. List Manipulation – Create a list of numbers, then write a program that prints the square of each number in the list.

```python
numbers = [2, 4, 6, 8, 10]

squares = [n**2 for n in numbers]

print("Numbers:", numbers)

print("Squares:", squares)
```

OOPs in Python

By Vijay

Introduction to OOPs

- - OOP = Object Oriented Programming
- - Organizes code into objects and classes
- - Increases reusability, scalability, modularity

Four Pillars of OOP

- 1. Encapsulation
- 2. Abstraction
- 3. Inheritance
- 4. Polymorphism

Classes & Objects in Python

- - Class → Blueprint
- - Object → Instance of class

- Example:
- class Student:
-     def __init__(self, name):
-         self.name = name

- s1 = Student('Vijay')

Inheritance

- - One class inherits properties of another

- Example:
- class Parent:
-     def show(self):
-         print('Parent class')

- class Child(Parent):
-     pass

Polymorphism

- - Same function name, different behaviors

- Example:
- print(len('Python')) # string
- print(len([1,2,3]))   # list

Encapsulation & Abstraction

- - Encapsulation → Hiding data using private/protected variables
- - Abstraction → Hiding implementation using abstract classes

Real-World Applications of OOP

- - GUI Applications (Tkinter, PyQt)
- - Web Frameworks (Django, Flask)
- - Machine Learning Models (Scikit-learn, TensorFlow)

Advantages of OOP in Python

- - Code Reusability

- - Modularity

- - Easy to Debug & Maintain

- - Scalable

Conclusion

- - OOP makes Python powerful
- - Helps in building real-world applications
- - Thank You
- - By Vijay