# Assignment 2

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

```
def count_word_frequencies(text):

      words = text.lower().split()
      freq = {}
      for word in words:
            word = word.strip('.,!?";:')
            freq[word] = freq.get(word, 0) + 1
      return freq

text ="How are you,what are you doing? I hope you are doing
       well."

frequencies = count_word_frequencies(text)
print(frequencies)
```

**Output:**

```
{'how': 1, 'are': 3, 'you,what': 1, 'you': 2, 'doing': 2, 'i': 1,
'hope': 1, 'well': 1}
```

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

```
def is_palindrome(word):
    if word == word[::-1]:
        return True
    else:
        return False
# Example usage
word = "madam"
if is_palindrome(word)==True:
    print(f"{word} is a palindrome")
else:
    print(f"{word} is not a palindrome")
```

**Output:**

**madam is a palindrome**

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

```
# List Manipulation – Square Each Number
def square_list(numbers):
    return [x ** 2 for x in numbers]

# Example usage
nums = [1, 2, 3, 4, 5]
squared = square_list(nums)
print("Original list:", nums)
print("Squared list:", squared)
```

Output:

**Original list: [1, 2, 3, 4, 5]**

**Squared list: [1, 4, 9, 16, 25]**

# Object-Oriented Programming in Python

**Presented by : Pandya Shaunak Rajnikant**

# What is OOP?

- A programming paradigm based on the concept of "objects"
- Objects contain both data and methods that operate on that data
- Helps structure complex programs into simple, reusable pieces
- Commonly used in many modern programming languages

# Key Concepts in OOP

- **Class**: A blueprint for creating objects
- **Object**: An instance of a class
- **Encapsulation**: Bundling data with methods that operate on that data
- **Abstraction**: Hiding internal details and showing only essential features
- **Inheritance**: A way to form new classes using classes that have already been defined
- **Polymorphism**: The ability to present the same interface for different data types

# Classes and Objects

- **Class**: A blueprint or template for creating objects
- **Object**: An instance of a class with its own data and behavior
- Objects interact with each other through methods
- Classes define the structure and behavior that the objects will have

# Inheritance and Polymorphism

- **Inheritance** promotes code reusability by allowing a new class to take on the properties of an existing class
- **Polymorphism** allows objects of different classes to be treated as objects of a common superclass
- These features support scalability and flexibility in program design

# Advantages of Using OOP

- Improves code organization and readability
- Encourages modular design for complex systems
- Makes maintenance and debugging easier
- Enhances code reusability through inheritance
- Supports abstraction and data protection

# Real-World Applications of OOP

- Used in GUI-based applications, web frameworks, and games
- Common in design of enterprise software and databases
- Python's OOP is widely used in data science, automation, and AI projects
- Frameworks like Django and Flask follow OOP principles

# Thank You