1. Write a program to print all even numbers from 1 to 50 using a loop.z
2. Use a loop to count and print the number of vowels in a string.
3. Write a program to print the factorial of a given number using a loop.
4. Write a program to find the average of 5 numbers.
5. Write a program to print the below pattern  
   \*  
   \* \*  
   \* \* \*  
   \* \* \* \*  
   \* \* \* \* \*
6. Write a program to display the multiplication table of a number entered by the user.
7. Use a loop to calculate the sum of the first 100 natural numbers.
8. Write a program to check if a given number is divisible by 2 and 4.
9. Check if a given number is odd or even.
10. Write a program to check if a string contains the character ‘t’ in it.
11. Write a program to check if a number is divisible by 5 and 11.
12. Write a program to check if a string contains the character ‘a’ in it.
13. Convert a list into a tuple using type-cast.
14. Take two sets and find their intersection.
15. Write a Python program to find the largest and smallest number in a list.
16. Take a list of numbers and print the sum of all even numbers.
17. Create a program to sort a list of integers in descending order.
18. Take a list of words and print the words that start with a vowel.
19. Accept a string and print it in reverse using slicing.
20. Accept a string and replace all vowels with a \*.
21. Create a dictionary with 5 elements where the value of each is the squared value of the key.
22. Create a tuple and demonstrate unpacking it into variables.
23. CHAPTER-III
24. Write a program to create a new text file and write your name into it.
25. Write a program to append user input to an existing file.
26. Create a program that reads a file line by line and prints each line.
27. Write a program to count the number of lines in a file.
28. Write a program to count the total number of words in a file.
29. Write a program that overwrites a file with new content.
30. Check if a file exists before reading or writing using the os module.
31. CHAPTER-IV
32. Create a class Person with attributes name and age. Write a method to display the person's details.
33. Write a class Rectangle with methods to compute area and perimeter.
34. Define a class Circle with a method to calculate area. Accept radius as input from the user.
35. Define a class Book with attributes title, author, and price. Write a method to display book details.
36. Create a base class Animal and a subclass Dog that adds a method bark().
37. Create a class BankAccount with attributes account\_holder and balance. Initialize them through a constructor.
38. Showcase the use of the super() method in OOPs by incorporating in a program.
39. Write a class Patient that stores name and disease. Use methods to access and update data securely.

# 1. Print all even numbers from 1 to 50

for i in range(1, 51):

if i % 2 == 0:

print(i, end=' ')

# 2. Count vowels in a string

string = "Hello World"

vowels = 'aeiouAEIOU'

count = 0

for char in string:

if char in vowels:

count += 1

print("\nNumber of vowels:", count)

# 3. Factorial of a number

num = 5

factorial = 1

for i in range(1, num + 1):

factorial \*= i

print("Factorial:", factorial)

# 4. Average of 5 numbers

nums = [10, 20, 30, 40, 50]

average = sum(nums) / len(nums)

print("Average:", average)

# 5. Pattern printing

for i in range(1, 6):

print("\* " \* i)

# 6. Multiplication table of a number

n = int(input("Enter a number: "))

for i in range(1, 11):

print(f"{n} x {i} = {n\*i}")

# 7. Sum of first 100 natural numbers

print("Sum of first 100 natural numbers:", sum(range(1, 101)))

# 8. Divisibility check by 2 and 4

num = 8

if num % 2 == 0 and num % 4 == 0:

print(f"{num} is divisible by 2 and 4")

# 9. Odd or even

n = 7

print("Even" if n % 2 == 0 else "Odd")

# 10. Check if string contains 't'

text = "Python"

print("Contains 't'" if 't' in text else "Does not contain 't'")

# 11. Divisibility by 5 and 11

num = 55

if num % 5 == 0 and num % 11 == 0:

print("Divisible by 5 and 11")

# 12. Check if string contains 'a'

text = "banana"

print("Contains 'a'" if 'a' in text else "Does not contain 'a'")

# 13. List to tuple

lst = [1, 2, 3, 4]

tpl = tuple(lst)

print(tpl)

# 14. Intersection of sets

s1 = {1, 2, 3, 4}

s2 = {3, 4, 5, 6}

print(s1 & s2)

# 15. Largest and smallest in list

lst = [3, 5, 1, 9, 7]

print("Max:", max(lst), "Min:", min(lst))

# 16. Sum of even numbers in list

lst = [1, 2, 3, 4, 5, 6]

print("Sum of evens:", sum(x for x in lst if x % 2 == 0))

# 17. Sort list in descending order

lst = [10, 4, 2, 8]

lst.sort(reverse=True)

print(lst)

# 18. Words starting with vowels

words = ["apple", "banana", "orange", "umbrella", "cat"]

for word in words:

if word[0].lower() in 'aeiou':

print(word)

# 19. Reverse string using slicing

s = "hello"

print(s[::-1])

# 20. Replace vowels with '\*'

s = "education"

for vowel in 'aeiouAEIOU':

s = s.replace(vowel, '\*')

print(s)

# 21. Dictionary with square values

d = {i: i\*\*2 for i in range(1, 6)}

print(d)

# 22. Tuple unpacking

tpl = ("Alice", 25, "Engineer")

name, age, profession = tpl

print(name, age, profession)

# 24. Create and write to a new file

with open("newfile.txt", "w") as f:

f.write("Your Name")

# 25. Append user input to file

with open("newfile.txt", "a") as f:

data = input("Enter text to append: ")

f.write("\n" + data)

# 26. Read file line by line

with open("newfile.txt", "r") as f:

for line in f:

print(line.strip())

# 27. Count lines in file

with open("newfile.txt", "r") as f:

print("Number of lines:", len(f.readlines()))

# 28. Count words in file

with open("newfile.txt", "r") as f:

print("Number of words:", len(f.read().split()))

# 29. Overwrite file content

with open("newfile.txt", "w") as f:

f.write("This is new content")

# 30. Check if file exists

import os

filename = "newfile.txt"

if os.path.exists(filename):

print("File exists")

else:

print("File does not exist")

# 32. Class Person

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display(self):

print(f"Name: {self.name}, Age: {self.age}")

person1 = Person("John", 30)

person1.display()

# 33. Class Rectangle

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

def perimeter(self):

return 2 \* (self.length + self.width)

rect1 = Rectangle(5, 3)

print("Area:", rect1.area())

print("Perimeter:", rect1.perimeter())

# 34. Class Circle

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14 \* self.radius \*\* 2

circle1 = Circle(7)

print("Circle Area:", circle1.area())

# 35. Class Book

class Book:

def \_\_init\_\_(self, title, author, price):

self.title = title

self.author = author

self.price = price

def display(self):

print(f"{self.title} by {self.author}, Price: {self.price}")

book1 = Book("Python Basics", "Alice", 299)

book1.display()

# 36. Animal and Dog classes

class Animal:

def \_\_init\_\_(self, species):

self.species = species

def sound(self):

print(f"{self.species} makes a sound")

class Dog(Animal):

def \_\_init\_\_(self, species, breed):

super().\_\_init\_\_(species)

self.breed = breed

def bark(self):

print(f"{self.breed} says Woof!")

dog1 = Dog("Mammal", "Labrador")

dog1.sound()

dog1.bark()

# 37. Class BankAccount

class BankAccount:

def \_\_init\_\_(self, account\_holder, balance):

self.account\_holder = account\_holder

self.balance = balance

def display(self):

print(f"Account Holder: {self.account\_holder}, Balance: {self.balance}")

account1 = BankAccount("Emma", 10000)

account1.display()

# 38. Use of super()

class Vehicle:

def \_\_init\_\_(self, brand):

self.brand = brand

class Car(Vehicle):

def \_\_init\_\_(self, brand, model):

super().\_\_init\_\_(brand)

self.model = model

def display(self):

print(f"Brand: {self.brand}, Model: {self.model}")

car1 = Car("Toyota", "Corolla")

car1.display()

# 39. Class Patient

class Patient:

def \_\_init\_\_(self, name, disease):

self.\_\_name = name

self.\_\_disease = disease

def get\_details(self):

return self.\_\_name, self.\_\_disease

def update\_disease(self, new\_disease):

self.\_\_disease = new\_disease

patient1 = Patient("David", "Flu")

print("Patient Details:", patient1.get\_details())

patient1.update\_disease("Recovered")

print("Updated Details:", patient1.get\_details())