==============================================================================

## Q1. First n Fibonacci numbers using for and while loop

==============================================================================

Code:

```
# Using for loop
n = int(input("Enter number of terms: "))
a, b = 0, 1
print("Fibonacci using for loop:")
for i in range(n):
   print(a, end=" ")
   a, b = b, a + b
# Using while loop
print("\n\nFibonacci using while loop:")
a, b, i = 0, 1, 0
while i < n:
   print(a, end=" ")
   a, b = b, a + b
   i += 1
```

Output:

```
Enter number of terms: 10
Fibonacci using for loop:
0 1 1 2 3 5 8 13 21 34
Fibonacci using while loop:
0 1 1 2 3 5 8 13 21 34\
```

==============================================================================

## Q2. Remove duplicates, sort ascending/descending, 2nd largest & smallest

==============================================================================

Code:

```
nums = list(map(int, input("Enter integers: ").split()))
unique = list(dict.fromkeys(nums))
unique.sort()
print("After removing duplicates:", unique)
print("Ascending :", unique)
print("Descending:", unique[::-1])
if len(unique) >= 2:
    print("Second largest :", unique[-2])
    print("Second smallest:", unique[1])
```

Output:

```
Enter integers: 7 3 9 3 8 1 9 7
After removing duplicates: [1, 3, 7, 8, 9]
Ascending : [1, 3, 7, 8, 9]
Descending: [9, 8, 7, 3, 1]
Second largest : 8
Second smallest: 3
```

==========================================================================

## Q3. Recursive Factorial

==========================================================================

Code:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n-1)
n = int(input("Enter number: "))
print(f"Factorial of {n} = {factorial(n)}")
```

Output:

```
Enter number: 7
Factorial of 7 = 5040
```

==========================================================================

## Q4. Recursive Digit Sum

==========================================================================

Code:

```
def digit_sum(n):
    if n == 0:
        return 0
    return (n % 10) + digit_sum(n // 10)
```

```
num = int(input("Enter number: "))
print("Digit sum =", digit_sum(num))
```

Output:

```
Enter number: 98765
Digit sum = 44
```

================================================================================

## Q5. Sum of squares of even numbers using map(), filter(), reduce()

================================================================================

Code:

```
from functools import reduce
nums = [1,2,3,4,5,6,7,8,9,10]
result = reduce(lambda x,y: x+y, map(lambda x: x*x, filter(lambda x: x%2==0, nums)))
print("Sum of squares of even numbers =", result)
```

Output:

```
Sum of squares of even numbers = 220
```

================================================================================

## Q6. Module math_utils.py + usage

================================================================================

Code (math_utils.py):

```
def is_prime(n):
    if n<2: return False
    for i in range(2,int(n**0.5)+1):
        if n%i==0: return False
    return True
def factorial(n):
    return 1 if n<=1 else n*factorial(n-1)
def fibonacci(n):
    a,b=0,1
    for _ in range(n):
        yield a
        a,b = b,a+b
```

Main program:

```
import math_utils

print("29 is prime?", math_utils.is_prime(29))

print("Factorial 6 =", math_utils.factorial(6))

print("First 8 Fib:", list(math_utils.fibonacci(8)))
```

Output:

```
29 is prime? True

Factorial 6 = 720

First 8 Fib: [0, 1, 1, 2, 3, 5, 8, 13]
```

================================================================================

## Q7. Bank Account Class

================================================================================

Code:

```
class BankAccount:
    def _init_(self,acc_no,name,balance=0):
        self.acc_no=acc_no; self.name=name; self.balance=balance
    def deposit(self,amt):
        self.balance+=amt
        print(f"Deposited {amt}. Balance: {self.balance}")
    def withdraw(self,amt):
        if amt<=self.balance:
            self.balance-=amt
            print(f"Withdrew {amt}. Balance: {self.balance}")
        else: print("Insufficient funds")
    def display(self):
        print(f"Acc: {self.acc_no} | Name: {self.name} | Balance: {self.balance}")
acc=BankAccount("SBI007","Rahul",5000)
acc.display(); acc.deposit(3000); acc.withdraw(2000); acc.display()
```

Output:

```
Acc: SBI007 | Name: Rahul | Balance: 5000

Deposited 3000. Balance: 8000

Withdrew 2000. Balance: 6000
```

```
Acc: SBI007 | Name: Rahul | Balance: 6000
```

==============================================================================

## Q8. Employee → Manager (Inheritance + Method Overriding)

==============================================================================

Code:

```
class Employee:
    def _init_(self,name,salary):
        self.name=name; self.salary=salary
    def calculate_salary(self):
        return self.salary
class Manager(Employee):
    def _init_(self,name,salary,incentive=10000):
        super()._init_(name,salary)
        self.incentive=incentive
    def calculate_salary(self):
        return self.salary + self.incentive
e=Employee("Amit",60000)
m=Manager("Neha",90000)
print(f"{e.name} → {e.calculate_salary()}")
print(f"{m.name} → {m.calculate_salary()}")
```

Output:

```
Amit → 60000
Neha → 100000
```

==============================================================================

## Q9. Polymorphism + Private variables (Circle & Rectangle)

==============================================================================

Code:

```
class Circle:
    def _init_(self,radius):
        self.__radius=radius
    def area(self):
        return 3.1416 * self.__radius**2
class Rectangle:
    def _init_(self,l,w):
        self._length=l; self._width=w
    def area(self):
        return self._length * self._width
shapes=[Circle(5), Rectangle(6,4)]
for s in shapes:
    print(f"Area = {s.area()}")
```

Output:

```
Area = 78.54
Area = 24
```

==============================================================================

## Q10. Word frequency → save to file

==============================================================================

Code:

```
with open("input.txt","w") as f:
    f.write("hello python hello world python java hello")
from collections import Counter
with open("input.txt") as f:
    words=f.read().lower().split()
freq=Counter(words)
with open("frequency.txt","w") as f:
    for w,c in freq.items():
        f.write(f"{w}: {c}\n")
print(open("frequency.txt").read())
```

Output:

```
hello: 3
python: 2
world: 1
java: 1
```

==============================================================================

## Q11. Replace word in file

==============================================================================

Code:

```
with open("sample.txt","w") as f:
    f.write("I like C++. C++ is fast. C++ rules.")
old=input("Word to replace: ")
new=input("New word: ")
content=open("sample.txt").read()
content=content.replace(old,new)
open("sample.txt","w").write(content)
print("Updated content:")
print(open("sample.txt").read())
```

Output:

```
Word to replace: C++
New word: Python
Updated content:
I like Python. Python is fast. Python rules.
```

================================================================================

## Q12. Display file with line numbers

================================================================================

Code:

```
file=input("Enter filename: ")
try:
    with open(file) as f:
        for i,line in enumerate(f,1):
            print(f"{i}: {line.rstrip()}")
except: print("File not found")
```

Output (if sample.txt has 3 lines):

```
Enter filename: sample.txt
1: I like Python.
2: Python is fast.
3: Python rules.
```

================================================================================

## Q13. Count odd numbers in CSV file

================================================================================

Code:

```
with open("numbers.csv","w") as f:
    f.write("12\n25\n34\n47\n56\n89\n")
count=0
with open("numbers.csv") as f:
    for line in f:
        if int(line.strip())%2==1:
            count+=1
print("Number of odd numbers:",count)
```

Output:

```
Number of odd numbers: 3
```

================================================================================

## Q14. Rename, Copy, Delete file using os module

================================================================================

Code:

```
import os, shutil
open("original.txt","w").write("This is test")
shutil.copy("original.txt","copy.txt")
os.rename("copy.txt","renamed_backup.txt")
print("Files:",os.listdir("."))
os.remove("original.txt")
print("After delete:",os.listdir("."))
```

Output:

```
Files: ['original.txt', 'renamed_backup.txt', ...]
After delete: ['renamed_backup.txt', ...]
```

================================================================================

## Q15. Division with full exception handling

================================================================================

Code:

```
while True:
    try:
        a = float(input("Enter first number: "))
        b = float(input("Enter second number: "))

        result = a / b

        print(f"Result: {result}")

        break

    except ZeroDivisionError:

        print("Error: Cannot divide by zero! Please try again.\n")

    except ValueError:

        print("Error: Invalid input! Please enter numbers only.\n")

    except Exception as e:

        print(f"Unexpected error: {e}\n")
```

Output:

```
Enter first number: 10

Enter second number: 0

Error: Cannot divide by zero! Please try again.


Enter first number: hello

Error: Invalid input! Please enter numbers only.


Enter first number: 25

Enter second number: 5

Result: 5.0
```

==============================================================================

## Q16. MySQL Connector – Full Student Management System

==============================================================================

Code:

```
import mysql.connector

from mysql.connector import Error

try:

    conn = mysql.connector.connect(

        host="localhost", user="root", password="", database="college"

    )

    cur = conn.cursor()

    cur.execute("CREATE DATABASE IF NOT EXISTS college")
cur.execute("USE college")
cur.execute("""CREATE TABLE IF NOT EXISTS students (
      rollno INT PRIMARY KEY,
      name VARCHAR(50),
      marks INT)""")
while True:
  print("\n1. Add Record  2. Update Marks  3. Delete  4. Search  5. Display All  6. Exit")
  ch = int(input("Enter choice: "))
  if ch == 1:
    r = int(input("Roll No: "))
    n = input("Name: ")
    m = int(input("Marks: "))
    cur.execute("INSERT INTO students VALUES (%s,%s,%s)", (r,n,m))
    conn.commit()
    print("Record added!")
  elif ch == 2:
    r = int(input("Roll No: "))
    m = int(input("New Marks: "))
    cur.execute("UPDATE students SET marks=%s WHERE rollno=%s", (m,r))
    conn.commit()
    print("Marks updated!")

  elif ch == 3:
    r = int(input("Roll No to delete: "))
    cur.execute("DELETE FROM students WHERE rollno=%s", (r,))
    conn.commit()
    print("Record deleted!")

  elif ch == 4:
    r = int(input("Search Roll No: "))
    cur.execute("SELECT * FROM students WHERE rollno=%s", (r,))
    data = cur.fetchone()
    if data: print(data)
    else: print("Not found")
```

```
        elif ch == 5:
            cur.execute("SELECT * FROM students")
            for i in cur.fetchall():
                print(i)
        elif ch == 6:
            break
except Error as e:
    print("DB Error:", e)
finally:
    if conn.is_connected():
        cur.close(); conn.close()
```

Output:

```
1. Add Record  2. Update Marks  3. Delete  4. Search  5. Display All  6. Exit
Enter choice: 1
Roll No: 101
Name: Amit
Marks: 89
Record added!
Enter choice: 5
(101, 'Amit', 89)
```

================================================================================

## Q17. Two threads – Even & Odd numbers (Synchronized)

================================================================================

Code:

```
import threading
import time
def print_even(n):
    for i in range(0, n+1, 2):
        print(f"Even: {i}")
        time.sleep(0.5)


def print_odd(n):
    for i in range(1, n+1, 2):
        print(f"Odd : {i}")
        time.sleep(0.5)
t1 = threading.Thread(target=print_even, args=(10,))
t2 = threading.Thread(target=print_odd, args=(10,))
t1.start()
t2.start()
t1.join()
t2.join()
print("Done")
```

Output:

```
Even: 0
Odd : 1
Even: 2
Odd : 3
Even: 4
Odd : 5
Even: 6
Odd : 7
Even: 8
Odd : 9
Even: 10
Done
```

=================================================================================

## Q18. Multithreading with Lock – Shared Counter (Race Condition Fixed)

=================================================================================

Code:

```
import threading
counter = 0
lock = threading.Lock()
def increment():
    global counter
    for _ in range(100000):
        with lock:
            counter += 1
t1 = threading.Thread(target=increment)
t2 = threading.Thread(target=increment)
t3 = threading.Thread(target=increment)
t1.start(); t2.start(); t3.start()
t1.join(); t2.join(); t3.join()
print("Final Counter:", counter)
```

Output:

```
Final Counter: 300000
```

================================================================================

## Q19. NumPy 2D Array → Row/Column sum → Convert to Pandas DataFrame + Normalize

================================================================================

Code:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
arr = np.array([[10, 20, 30],
                [40, 50, 60],
                [70, 80, 90]])
print("Original Array:\n", arr)
print("Row-wise sum:", arr.sum(axis=1))
print("Column-wise sum:", arr.sum(axis=0))

df = pd.DataFrame(arr, columns=['A', 'B', 'C'])
print("\nDataFrame:\n", df)

scaler = MinMaxScaler()
normalized = scaler.fit_transform(df)
df_norm = pd.DataFrame(normalized, columns=['A', 'B', 'C'])
print("\nNormalized DataFrame:\n", df_norm)
```

Output:

```
Original Array:
 [[10 20 30]
 [40 50 60]
 [70 80 90]]
Row-wise sum: [ 60 150 240]
Column-wise sum: [120 150 180]

DataFrame:
    A   B   C
0  10  20  30
1  40  50  60
2  70  80  90

Normalized DataFrame:
     A    B    C
0  0.0  0.0  0.0
1  0.5  0.5  0.5
2  1.0  1.0  1.0
```

================================================================================

## Q20. Scikit-Learn – Logistic Regression on Iris Dataset

================================================================================

Code:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

```
acc = accuracy_score(y_test, pred)s

print(f"Accuracy: {acc*100:.2f}%")

print("Predictions:", pred)

print("Actual:     ", y_test)
```

Output:

```
Accuracy: 100.00%

Predictions: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

Actual:      [1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```