

1. Write a Python program to calculate the sum of three given numbers. If the values are equal, then return three times of their sum.

```
def calculate_sum(num1, num2, num3):  
    total = num1 + num2 + num3
```

```
    if num1 == num2 and num2 == num3:  
        return total * 3
```

```
    else:  
        return total
```

```
print("Enter three numbers:")  
a = float(input("First number: "))  
b = float(input("Second number: "))  
c = float(input("Third number: "))
```

```
final_value = calculate_sum(a, b, c)
```

```
print(f"The result is: {final_value}")
```

2. Write a Python program to find the factorial of a non-negative integer using decision control statements.

```
num = int(input("Enter a non-negative integer: "))
```

```
factorial = 1
```

```
if num < 0:  
    print("Factorial does not exist for negative numbers.")
```

```
elif num == 0:  
    print("The factorial of 0 is 1")
```

```
else:  
    for i in range(1, num + 1):  
        factorial = factorial * i  
    print(f"The factorial of {num} is {factorial}")
```

3. To accept student's five courses marks and compute his/her result. Student is passing if he/she scores marks equal to and above 40 in each course.

- **If student scores aggregate greater than 75%, then the grade is Distinction.**
- **If aggregate is $60 \geq$ and < 75 , then the grade is First Division.**
- **If aggregate is $50 \geq$ and < 60 , then the grade is Second Division.**
- **If aggregate is $40 \geq$ and < 50 , then the grade is Third Division.**

```

marks = []
is_failed = False

print("Enter marks for 5 subjects:")
for i in range(5):
    m = float(input(f"Subject {i+1}: "))
    marks.append(m)
    if m < 40:
        is_failed = True

if is_failed:
    print("Result: Fail (Scored less than 40 in one or more subjects)")
else:
    total = sum(marks)
    percentage = total / 5

    print(f"Total Marks: {total}")
    print(f"Percentage: {percentage}%")

    if percentage > 75:
        print("Grade: Distinction")
    elif percentage >= 60:
        print("Grade: First Division")
    elif percentage >= 50:
        print("Grade: Second Division")
    elif percentage >= 40:
        print("Grade: Third Division")

```

4. To check whether input number is Armstrong number or not. An Armstrong number is an integer with three digits such that the sum of the cubes of its digits is equal to the number itself.

```

num = int(input("Enter a number: "))
temp = num
sum_of_cubes = 0

while temp > 0:
    digit = temp % 10
    sum_of_cubes += digit ** 3
    temp //= 10

if num == sum_of_cubes:
    print(f"{num} is an Armstrong number")
else:
    print(f"{num} is not an Armstrong number")

```

5. Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.

```
values = input("Input some comma seprated numbers: ")
list_data = values.split(",")
tuple_data = tuple(list_data)

print("List :", list_data)
print("Tuple :", tuple_data)
```

6. Write a Python program to get a directory listing, sorted by creation date.

```
import os
import time

def get_files_sorted_by_date(path):
    files = [os.path.join(path, f) for f in os.listdir(path) if os.path.isfile(os.path.join(path, f))]
    files.sort(key=os.path.getctime)
    return files

path = "."
sorted_files = get_files_sorted_by_date(path)

for file in sorted_files:
    print(f'{os.path.basename(file)} - {time.ctime(os.path.getctime(file))}')
```

7. Write a Python program using Lambda functions to manipulate strings:

- **Convert a string to uppercase.**
- **Check if a string starts with a specific character.**

```
to_upper = lambda s: s.upper()
starts_with_a = lambda s: s.startswith('A')

text = "Apple"
print(f"Uppercase: {to_upper(text)}")
print(f"Starts with 'A': {starts_with_a(text)}")

text2 = "banana"
print(f"Uppercase: {to_upper(text2)}")
print(f"Starts with 'A': {starts_with_a(text2)}")
```

8. Using concepts of OOP, write a Python program for banking application which performs following operations:

- **Create an account (Account_no, Name, Account_type, Balance).**
- **Perform transactions - Deposit money, Withdraw money.**
- **Check the balance of the account.**
- **Display account information.**

```
class BankAccount:
    def __init__(self, acc_no, name, acc_type, balance):
        self.acc_no = acc_no
        self.name = name
        self.acc_type = acc_type
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f'Deposited {amount}. New Balance: {self.balance}')

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance.")
        else:
            self.balance -= amount
            print(f'Withdrawn {amount}. New Balance: {self.balance}')

    def check_balance(self):
        print(f'Current Balance: {self.balance}')

    def display_info(self):
        print(f'Account No: {self.acc_no}')
        print(f'Name: {self.name}')
        print(f'Type: {self.acc_type}')
        print(f'Balance: {self.balance}')

account = BankAccount(101, "Manoj", "Savings", 5000)
account.display_info()
account.deposit(1000)
account.withdraw(2000)
account.check_balance()
```

9. Create class EMPLOYEE for storing details (Name, Designation, gender, Date of Joining and Salary). Define function members to compute:

- Total number of employees in an organization.**
- Count of male and female employees.**

c) Employee with salary more than 10,000.

d) Employee with designation "Asst Manager".

```
class Employee:
    total_emp = 0
    male_emp = 0
    female_emp = 0

    def __init__(self, name, designation, gender, doj, salary):
        self.name = name
        self.designation = designation
        self.gender = gender
        self.doj = doj
        self.salary = salary

    Employee.total_emp = Employee.total_emp + 1
    if self.gender == "Male":
        Employee.male_emp = Employee.male_emp + 1
    elif self.gender == "Female":
        Employee.female_emp = Employee.female_emp + 1

    def show_counts():
        print("Total Employees:", Employee.total_emp)
        print("Male Employees:", Employee.male_emp)
        print("Female Employees:", Employee.female_emp)

    def check_salary(self):
        if self.salary > 10000:
            print(self.name, self.salary)

    def check_designation(self):
        if self.designation == "Asst Manager":
            print(self.name, self.designation)

e1 = Employee("Raj", "Manager", "Male", "10-01-2022", 50000)
e2 = Employee("Simran", "Asst Manager", "Female", "12-02-2023", 15000)
e3 = Employee("Amit", "Clerk", "Male", "01-03-2023", 8000)
e4 = Employee("Neha", "Asst Manager", "Female", "05-04-2023", 12000)

print("\n--- Employee Counts ---")
Employee.show_counts()

print("\n--- Salary > 10000 ---")
e1.check_salary()
e2.check_salary()
e3.check_salary()
e4.check_salary()

print("\n--- Designation Asst Manager ---")
```

```
e1.check_designation()
e2.check_designation()
e3.check_designation()
e4.check_designation()
```

10. Write a Python program to calculate the Mean, Standard Deviation, Mode, Median, Minimum, and Maximum values of an array using appropriate NumPy functions.

```
import numpy as np
from scipy import stats

data = np.array([10, 20, 20, 30, 40, 50, 60, 70])

mean_val = np.mean(data)
std_dev = np.std(data)
median_val = np.median(data)
min_val = np.min(data)
max_val = np.max(data)
mode_val = stats.mode(data, keepdims=True).mode[0]

print(f"Data: {data}")
print(f"Mean: {mean_val}")
print(f"Standard Deviation: {std_dev}")
print(f"Median: {median_val}")
print(f"Mode: {mode_val}")
print(f"Minimum: {min_val}")
print(f"Maximum: {max_val}")
```

11. Download "Vehicle Sales" dataset from Kaggle website and write a Python program to perform following operations:

- Read the dataset file.**
- Show top ten sales records from the dataset.**
- Accept vehicle model and plot the Line graph based on sale date and selling price.**

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('vehicle_sales.csv')

print("Top 10 Sales Records:")
print(df.head(10))

model_name = input("Enter Vehicle Model to plot: ")
model_data = df[df['Model'] == model_name]
```

```
if not model_data.empty:
    plt.plot(model_data['SaleDate'], model_data['SellingPrice'])
    plt.title(f"Sales Trend for {model_name}")
    plt.xlabel("Sale Date")
    plt.ylabel("Selling Price")
    plt.show()
else:
    print("Model not found.")
```

12. Write a Python program to create a simple plot using Matplotlib and utilize NumPy arrays to store and manipulate data points.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 20, 25, 30, 50])

plt.plot(x, y, marker='o')
plt.title("Simple Plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.grid(True)
plt.show()
```