Vivekananda Institute of Professional Studies - Technical Campus

(Affiliated to GGSIP University, Delhi, Recognized by AICTE)

Vivekananda School of Information Technology

योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

ज्योतिर्वृणीत तमसो विजानन

MASTERS OF COMPUTER APPLICATIONS
2023-2025

MCA - 106

PRACTICAL FILE

OF

Python Programming

Submitted To:
Dr. Mamta Madan
Professor
VSIT, VIPS

Submitted By:
Muhammed Raihan P. S.
13017704423
MCA 2C

INDEX

S No	Topic	Date	Sign
1	Any number is input through keyboard. WAP ti find it is an odd or an even number		
2	WAP to check if the year is a leap year		
3	WAP to check if a triangle is valid or not. When the three angle is entered through keyboard		
4	WAP to print the absolute value of a number entered through the keyboard		
5	Calculate the area of Circle and a Rectangle		
6	WAP that recieves 3 sets of p, n, r and calculates the simple interest.		
7	WAP that prints the number from 1 to 10 all on the same line.		
8	WAP to calculate the factorial of any number.		
9	WAP to print the prime number from 1 to 300.		
10	WAP to print the multiplication table of any number entered by the user		
11	Calculate the salary of the person by asking various salary components like DA, HRA, TA		
12	WAP to input three integers from the keyboard and get their sum and product calculated		
13	Define a function leap and implement it		
14	WAP to calculate the largest among the three numbers		
15	WAP to calculate the LCM of few numbers		
16	WAP to check if the number is odd or even		

S No	Topic	Date	Sign
17	WAP a recursive function for factorial		
18	Practice use of global variables		
19	Apply recursion in one more problem fibonacci series		
20	WAP to print the digit at one's place of a number		
21	WAP to calculate the bill amount for an item given in quantity sold, value, discount and tax (use default argument function, with default value of discount and tax)		
22	WAP to calculate students results based on two examination, one sports event and three activities conducted. The weightage of activities = 30%, sports = 20% and examination = 50%		
23	WAP to print the ASCII value of character (ord('a'), char(65))		
24	WAP to read a character in uppercase and then print it in lowercase using lower() and upper() string methods		
25	Income Tax for individual is computer on slab rates as follows		
26	WAP to check a number entered is perfect or not		
27	WAP to check a number entered is circular or not (prime or not)		
28	WAP to check a number entered is palindrome or not		
29	Check a number for Armstrong number eg: 153 is an Armstrong number (1x1x1+5x5x5+3x3x3 = 153)		
30	WAP to swap two numbers entered by the user. You are not allowed to use a third variable		

S No	Topic	Date	Sign
31	WAP that has a user defined function to accept the coefficient of quadratic equation in variable and calculate its determinant		
32	Make a calculator.py file and import various functions into another file		
33	Implement command line arguments (display all the arguments)		
34	Implement command line arguments (add all the numbers passed on command line)		
35	Implement a calculator using command line arguments using parser		
36	From the string “Bamboozled”. WAP to obtain the following output		
37	Find all the occurrence of 'T' int the string 'The Terrible Tiger Tore The Towel'. Count them and print also replace them with 't'		
38	WAP to reverse a string		
39	WAP to check if the string is palindrome		
40	WAP to calculate the length of the string		
41	WAP to count the number of characters (character frequency) in a string		
42	WAP to get a single string from two strings separated by a space and swap the first two characters of each string		
43	WAP to accept a string and count the total number of vowels		
44	WAP to search a character in the string		
45	WAP to read the email id of the person in the format expected		

S No	Topic	Date	Sign
46	WAP to perform following on the list		
47	WAP to implement stack data structure which is LIFO		
48	Suppose a list has 20 numbers. WAP that removes all duplicates from the list		
49	WAP to obtain the median value of the list of numbers, without disturbing the order of numbers in the list		
50	WAP to count the occurrence of each character entered by the user and store them in a dictionary		
51	WAP to create two dictionaries, concatenate them and create a third dictionary		
52	WAP to check weather a given key exists in the dictionary		
53	WAP to remove a particular key from the dictionary		
54	WAP to display the minimum and maximum in a dictionary		
55	WAP to reverse a tuple		
56	WAP to calculate the product by multiplying all the numbers of a tuple		
57	WAP to check if the number exists in the tuple		
58	WAP to check if two sets have any elements in common. if yes display the common elements		
59	Update set1 by adding items from set2, except common elements		
60	Remove items from set1 that are not common to both set1 and set2		
61	WAP to print tuples which are not empty		

S No	Topic	Date	Sign
62	WAP to open a file try.txt, add some text into it and close it		
63	WAP to read the above file and print that on the screen		
64	WAP that writes four integers to a file called numbers Go to the following position and return a) 10 positions from the beginning b) 2 position to the right of current position c) 10 position to the left from the end		
65	WAP to implement inheritance using shape class and also make rectangle as a derived class. declare a function area to calculate the area of rectangle and also show the colour of the rectangle. Also make required constructor		
66	WAP to implement Multiple Inheritance using the example of employee class, waged employee and salaried employee. Write Appropriate constructors and functions to calculate pay.		
67	Implement operator overloading for + operator		
68	Implement operator overloading for less than equal to operator for user defined class		
69	Implement operator overloading for - operator		
70	Implement operator overloading for unary operator ++ and -- operator		
71	Implement exception handling for divide by 0 exception		
72	Implement exception handling using raise statement		
73	Implement threading		

S No	Topic	Date	Sign
74	WAP to construct a 3d array of dimension 4x2x3. Initialize the array to some value. Find the maximum along each axis (k,413)		
75	Create a program to achieve the following Create an array of size 10 with each element of it set to a value 3 Find the memory size of this array and its individual element Create an array b of size 10 with values ranging from 0 to 90 evenly spaced. Reverse elements of array b Add arrays a and b and store the result in c		
76	Declare a matrix o 3x3 and calculate its transpose		
77	Write a menu driven program to perform all the matrix operation		
78	WAP to count all the odd numbers in 1d array		
79	WAP to check if none of the elements is zero in the array		
80	WAP to extract all the prime numbers from an array		
81	Pre-process and perform different operations on data using Numpy and Pandas library		
82	Analyze the sales data and visualize it using different plots		

1) Any number is input through keyboard. WAP ti find it is an odd or an even number

```
a = int(input("Enter a number : "))
if a%2==0:
    print("Entered number is even")
else:
    print("Entered number is odd")
```

```
Enter a number : 4
Entered number is even
```

2) WAP to check if the year is a leap year

```
year = int(input("Enter a year : "))
if (year%400==0) and (year%100==0):
    print("Enteredyear is a leap year")
elif (year%4==0) and (year%100!=0):
    print("Entered year is a leap year")
else:
    print("Entered year is not a leap year")
```

```
Enter a year : 2040
Entered year is a leap year
```

3) WAP to check if a triangle is valid or not. When the three angle is entered through keyboard

```
angle1 = int(input("Enter the first angle : "))
angle2 = int(input("Enter the second angle : "))
angle3 = int(input("Enter the third angle : "))
sum_of_angle = angle1 + angle2 + angle3
if (sum_of_angle == 180):
    print("Triangle is valid")
else:
    print("Triangle is not valid")
```

```
Enter the first angle : 60
Enter the second angle : 60
Enter the third angle : 60
Triangle is valid
```

4) WAP to print the absolute value of a number entered through the keyboard

```
a = int(input("Enter a number : "))
print(f"Absolute value {a} is {abs(a)}")
```

```
Enter a number : -500
Absolute value -500 is 500
```

5) Calculate the area of Circle and a Rectangle

```python
def circle():
    pi = 3.14
    radius = int(input("Enter the radius : "))
    area = pi*(radius**2)
    print(f"Area of circle is {area}")

def rectangle():
    l = int(input("Enter the length : "))
    b = int(input("Enter the breadth : "))
    area = l*b
    print(f"Area of rectangle is {area}")

circle()
rectangle()
```

```
Enter the radius : 5
Area of circle is 78.5
Enter the length : 5
Enter the breadth : 10
Area of rectangle is 50
```

6) WAP that recieves 3 sets of p, n, r and calculates the simple interest.

```
i = 1
while(i<=3):
    p = int(input("Enter p : "))
    n = int(input("Enter n : "))
    r = int(input("Enter r : "))
    si=(p*n*r)/100;
    print(f"Simple interest for {i} set of values : {si}")
    i = i + 1
```

```
Enter p : 2000
Enter n : 3
Enter r : 23
Simple interest for 1 set of values : 1380.0
Enter p : 3000
Enter n : 2
Enter r : 12
Simple interest for 2 set of values : 720.0
Enter p : 5000
Enter n : 4
Enter r : 20
Simple interest for 3 set of values : 4000.0
```

7) WAP that prints the number from 1 to 10 all on the same line.

```
for i in range(1,11):
    print(i, end= " ")
```

```
1 2 3 4 5 6 7 8 9 10
```

8) WAP to calculate the factorial of any number.

```
num = int(input("Enter a number : "))
factorial = 1
if (num==1 or num==0):
    print(f"Factorial of {num} is {1}")
else:
    for i in range(1,num+1):
        factorial = factorial*i
    print(f"Factorial of {num} is {factorial}")
```

```
Enter a number : 5
Factorial of 5 is 120
```

9) WAP to print the prime number from 1 to 300.

```python
def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2
    return True

print("Prime numbers from 1 to 300:")
for num in range(1, 301):
    if is_prime(num):
        print(num, end=" ")
```

```
Prime numbers from 1 to 300:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167
173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293
```

10) WAP to print the multiplication table of any number entered by the user

```
num = int(input("Enter a number : "))
i = 1
while(i<=10):
    print(f"{num}x{i} = {i*num}")
    i = i + 1
```

```
Enter a number : 5
1x5 = 5
2x5 = 10
3x5 = 15
4x5 = 20
5x5 = 25
6x5 = 30
7x5 = 35
8x5 = 40
9x5 = 45
10x5 = 50
```

```
num = int(input("Enter a number : "))
for i in range(1,11):
    print(f"{num}x{i} = {i*num}")
```

```
Enter a number : 5
1x5 = 5
2x5 = 10
3x5 = 15
4x5 = 20
5x5 = 25
6x5 = 30
7x5 = 35
8x5 = 40
9x5 = 45
10x5 = 50
```

11) Calculate the salary of the person by asking various salary components like DA, HRA, TA

```
def gross_salary():
    basic_salary = float(input("Enter basic salary of the employee : "))
    DA = basic_salary*(25/100)
    print(f"Daily allowance of the employee is : {DA}")
    HRA = basic_salary*(15/100)
    print(f"House rent allowance of the employee is : {HRA}")
    TA = basic_salary*(5/100)
    print(f"Travelling allowance of the employee is : {TA}")
    gross_salary = basic_salary + DA + HRA + TA
    print(f"Gross salary of the employee is : {gross_salary}")

gross_salary()
```

```
Enter basic salary of the employee : 50000
Daily allowance of the employee is : 12500.0
House rent allowance of the employee is : 7500.0
Travelling allowance of the employee is : 2500.0
Gross salary of the employee is : 72500.0
```

12) WAP to input three integers from the keyboard and get their sum and product calculated

```python
def userinput():
    a = int(input("Enter the 1st number : "))
    b = int(input("Enter the 2nd number : "))
    c = int(input("Enter the 2rd number : "))
    return a, b, c

def calculate_sum(a, b, c):
    Sum = a + b + c
    print(f"Sum of three integers is : {Sum}")

def calculate_product(a, b, c):
    Product = a * b * c
    print(f"Product of three integers is : {Product}")

integers = userinput()
calculate_sum(*integers)
calculate_product(*integers)
```

```
Enter the 1st number : 5
Enter the 2nd number : 10
Enter the 2rd number : 15
Sum of three integers is : 30
Product of three integers is : 750
```

13) Define a function leap and implement it

```
def leap():
    year = int(input("Enter a year : "))
    if (year%400==0) and (year%100==0):
        print("Enteredyear is a leap year")
    elif (year%4==0) and (year%100!=0):
        print("Entered year is a leap year")
    else:
        print("Entered year is not a leap year")

leap()
```

```
Enter a year : 2040
Entered year is a leap year
```

14) WAP to calculate the largest among the three numbers

```
def largest():
    a = int(input("Enter the 1st number : "))
    b = int(input("Enter the 2nd number : "))
    c = int(input("Enter the 2rd number : "))
    if a>b and a>c:
        print(f"{a} is the largest number")
    elif b>a and b>c:
        print(f"{b} is the largest number")
    else:
        print(f"{c} is the largest number")

largest()
```

```
Enter the 1st number : 5
Enter the 2nd number : 4
Enter the 2rd number : 10
10 is the largest number
```

15) WAP to calculate the LCM of few numbers

```
def lcm():
    x = int(input("Enter 1st number : "))
    y = int(input("Enter 2nd number : "))

    if x>y:
        g=x
    else:
        g=y
    while(1):
        if (g%x==0) and (g%y==0):
            lcm = g
            break
        g+=1
    print(f"LCM : {g}")

lcm()
```

```
Enter 1st number : 5
Enter 2nd number : 6
LCM : 30
```

16) WAP to check if the number is odd or even

```
def oddeven():
    a = int(input("Enter a number : "))
    if a%2==0:
        print("Entered number is even")
    else:
        print("Entered number is odd")

oddeven()
```

```
Enter a number : 5
Entered number is odd
```

17) WAP a recursive function for factorial

```python
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n * factorial(n-1)

num = int(input("Enter the number : "))
f = factorial(num)
print(f"Factorial of {num} is {f}")
```

```
Enter the number : 5
Factorial of 5 is 120
```

18) Practice use of global variables

```
PI = 3.14
print(f"Value of {PI} before updation")

def perimeter(r):
    return 2*PI*r

def area(r):
    return PI*r*r

def updatePI():
    global PI
    PI = 3.14159
    print(f"Value of {PI} after updation")

per = perimeter(5)
print("Perimeter of Circle is : ", per)
area = area(5)
print(f"Area of Circle is : ", area)
updatePI()
```

```
Value of 3.14 before updation
Perimeter of Circle is :  31.400000000000002
Area of Circle is :  78.5
Value of 3.14159 after updation
```

19) Apply recursion in one more problem fibonacci series

```python
def fibonacci(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

def printfibonacci(n):
    for i in range(0, n+1):
        print(fibonacci(i))

printfibonacci(8)
```

```
0
1
1
2
3
5
8
13
21
```

20) WAP to print the digit at one's place of a number

```
n=int(input("Enter a number: "))
d=n%10
print("\n The digit at one's place of",n,"is",d)
```

```
Enter a number: 3452786

 The digit at one's place of 3452786 is 6
```

21) WAP to calculate the bill amount for an item given in quantity sold, value, discout and tax (use default argument function, with default value of discount and tax)

```python
def bill(discount = 0.15, tax = 0.2):
    quantity = int(input("Enter quantity of the item : "))
    value = int(input("Enter value of the item : "))
    total_amount = quantity*value
    bill_amount = (total_amount) - (total_amount*discount) + (total_amount*tax)
    print(f"Total amount payable after discount is : {bill_amount}")

bill()
bill(0.5,0.5)
```

```
Enter quantity of the item : 5
Enter value of the item : 100
Total amount payable after discount is : 525.0
Enter quantity of the item : 5
Enter value of the item : 100
Total amount payable after discount is : 500.0
```

22) WAP to calculate students results based on two examination, one sports event and three activities conducted. The weightage of activities = 30%, sports = 20% and examination = 50%

```python
def calculate_result(exam1_score, exam2_score, sports_score, activity_scores):
    # Calculate weighted scores
    exam_weight = 0.5
    sports_weight = 0.2
    activity_weight = 0.3

    total_activity_score = sum(activity_scores)

    weighted_exam_score = (exam1_score + exam2_score) / 2 * exam_weight
    weighted_sports_score = sports_score * sports_weight
    weighted_activity_score = total_activity_score / len(activity_scores) * activity_weight

    total_score = weighted_exam_score + weighted_sports_score + weighted_activity_score
    return total_score

def main():
    # Input scores
    exam1_score = float(input("Enter score for exam 1: "))
    exam2_score = float(input("Enter score for exam 2: "))
    sports_score = float(input("Enter score for sports event: "))

    activity_scores = []
    for i in range(3):
        score = float(input(f"Enter score for activity {i+1}: "))
        activity_scores.append(score)

    # Calculate result
    result = calculate_result(exam1_score, exam2_score, sports_score, activity_scores)
    print("Total Result:", result)

if __name__ == "__main__":
    main()
```

```
Enter score for exam 1: 50
Enter score for exam 2: 60
Enter score for sports event: 70
Enter score for activity 1: 80
Enter score for activity 2: 75
Enter score for activity 3: 60
Total Result: 63.0
```

23) WAP to print the ASCII value of character (ord('a'), char(65))

```python
# Print ASCII value of a character
alphabet = 'a'
ascii_value = ord(alphabet)
print(f"The ASCII value of '{alphabet}' is : {ascii_value}")

# Print character from ASCII value
number = 65
ascii_value = chr(number)
print(f"The character for ASCII value {number} is : {ascii_value}")
```

```
The ASCII value of 'a' is : 97
The character for ASCII value 65 is : A
```

24) WAP to read a character in uppercase and then print it in lowercase using lower() and upper() string methods

```
character = input("Enter a character in upper case (ie K) : ")
print("Character in lower case : ",character.lower())
```

```
Enter a character in upper case (ie K) : R
Character in lower case :  r
```

25) Income Tax for individual is computer on slab rates as follows

```
def income_tax():
    salary = int(input("Enter your salary : "))
    print(f"Salary of the user is : {salary}")
    tax = 0
    if (salary<=50000):
        print("Incone Tax Payable is 0");
    elif (salary>50000 and salary<=60000):
        tax = (salary-50000)*10/100
        print(f"Income Tax payable is {tax}")
    elif (salary>60000 and salary<=100000):
        tax = (salary-60000)*20/100
        print(f"Income Tax payable is {tax}")
    elif (salary>100000):
        tax = (salary-100000)*30/100
        print(f"Income Tax payable is {tax}")
    else:
        print("Invalid Input")

income_tax()
```

```
Enter your salary : 150000
Salary of the user is : 150000
Income Tax payable is 15000.0
```

26) WAP to check a number entered is perfect or not

```
num = int(input("Enter a number : "))
sum = 0
for i in range(1,num):
    if num %i == 0:
        sum += i

if sum == num:
    print(f"{num} is a perfect number")
else:
    print(f"{num} is not a perfect number")
```

```
Enter a number : 6
6 is a perfect number
```

27) WAP to check a number entered is circular or not

```
number = int(input("Enter a number : "))
flag = False

# prime number is greater than 1
if number > 1:
    for i in range(2,number):
        if (number%i)==0:
            # if factor is found set flag to True
            flag = True
            break

# check if flag is True
if flag:
    print(f"{number} is not a prime number")
else:
    print(f"{number} is a prime number")
```

```
Enter a number : 13
13 is a prime number
```

28) WAP to check a number entered is palindrome or not

```
number = int(input("Enter a number : "))
temp = number
reverse = 0

while(number>0):
    digit = number%10
    reverse = reverse*10 + digit
    number = number//10

print(f"Reverse of number : {reverse}")

if temp == reverse:
    print("Number is a palindrome")
else:
    print("Number is not a palindrome")
```

```
Enter a number : 64246
Reverse of number : 64246
Number is a palindrome
```

29) Check a number for Armstrong number eg: 153 is an Armstrong number (1x1x1+5x5x5+3x3x3 = 153)

```python
num = int(input("Enter a number : "))
order = len(str(num))
sum = 0
temp = num

while(temp>0):
    digit = temp % 10
    sum += digit ** order
    temp //= 10

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

```
Enter a number : 153
153 is an Armstrong number
```

30) WAP to swap two numbers entered by the user. You are not allowed to use a third variable

```python
a = int(input("Enter first number : "))
b = int(input("Enter second number : "))

print ("Before swapping: ")
print(f"Value of x : {a}")
print(f"value of y : {b}")

# code to swap 'x' and 'y'
a, b = b, a

print ("After swapping: ")
print(f"Value of x : {a}")
print(f"value of y : {b}")
```

```
Enter first number : 5
Enter second number : 10
Before swapping:
Value of x : 5
value of y : 10
After swapping:
Value of x : 10
value of y : 5
```

31) WAP that has a user defined function to accept the coefficient of quadratic equation in variable and calculate its determinant

```
def determinent():
    a = int(input("Enter the value of a : "))
    b = int(input("Enter the value of b : "))
    c = int(input("Enter the value of c : "))
    d = (b**2)-(4*a*c)
    print(f"Determinent is : {d}")
    if (d>0):
        print("Quadratic Equation has real roots")
    else:
        print("Quadratic Equation has no real roots")

determinent()
```

```
Enter the value of a : 5
Enter the value of b : 8
Enter the value of c : 2
Determinent is : 24
Quadratic Equation has real roots
```

32) Make a calculator.py file and import various functions into another file

```python
def calculator():
    while(1):
        a = int(input("Enter first number : "))
        b = int(input("Enter second number : "))
        print("Operators : +,-,/,*,**")
        operator = input("Enter operator : ")
        if operator == "+":
            print(f"{a} + {b} = {a+b}\n")
        elif operator == "-":
            print(f"{a} - {b} = {a-b}\n")
        elif operator == "/":
            print(f"{a} / {b} = {a/b}\n")
        elif operator == "*":
            print(f"{a} * {b} = {a*b}\n")
        elif operator == "**":
            print(f"{a} ** {b} = {a**b}\n")
        else:
            print("Invalid Operator entered")
            print("Exiting program")
            break
```

```python
import calculator
calculator.calculator()
```

```
Enter first number : 5
Enter second number : 7
Operators : +,-,/,*,**
Enter operator : -
5 - 7 = -2
```

33) Implement command line arguments (display all the arguments)

```python
import sys

# total arguments
n = len(sys.argv)
print("Total arguments passed:", n)

# Arguments passed
for i in range(0,n):
    print(f"Argumnet {i} is : {sys.argv[i]}")
```

```
C:\Users\psriz\Desktop>python CMD.py 1 2 3 4 5
Total arguments passed: 6
Argumnet 0 is : CMD.py
Argumnet 1 is : 1
Argumnet 2 is : 2
Argumnet 3 is : 3
Argumnet 4 is : 4
Argumnet 5 is : 5
```

34) Implement command line arguments (add all the numbers passed on command line)

```python
import sys

# total arguments
n = len(sys.argv)
print("Total arguments passed:", n)

# Arguments passed
print("\nName of Python script:", sys.argv[0])

print("\nArguments passed:", end = " ")
for i in range(1, n):
    print(sys.argv[i], end = " ")

# Addition of numbers
Sum = 0
# Using argparse module
for i in range(1, n):
    Sum += int(sys.argv[i])

print("\n\nResult:", Sum)
```

```
C:\Users\psriz\Desktop>python CMDArguments.py 1 2 3 4 5
Total arguments passed: 6

Name of Python script: CMDArguments.py

Arguments passed: 1 2 3 4 5

Result: 15
```

36) From the string “Bamboozled”. WAP to obtain the following output

```python
word = "Bamboozled"
# Output 1
print(word[0:2])

# Output 2
print(word[8:10])

# Output 3
a = word[8:10]
b = a.capitalize()
print(b)

# Output 4
a = word[2:10]
b = a.capitalize()
print(b)

# Output 5
a = word[0:6]
b = a.capitalize()
print(b)
```

```
Ba
ed
Ed
Mboozled
Bamboo
```

37) Find all the occurrence of 'T' int the string 'The Terrible Tiger Tore The Towel'. Count them and print also replace them with 't'

```
sentence = "The Terrible Tiger Tore The Towel"
print(sentence)
print(f"Number of times 'T' occured : {sentence.count('T')}")
s = sentence.replace("T", "t")
print(s)
```

```
The Terrible Tiger Tore The Towel
Number of times 'T' occured : 6
the terrible tiger tore the towel
```

38) WAP to reverse a string

```
sentence = "I believe what doesn't kill you simply makes you stronger"
print(sentence[::-1])
```

```
regnorts uoy sekam ylpmis uoy llik t'nseod tahw eveileb I
```

39) WAP to check if the string is palindrome

```
word = input("Enter a string: ")
if word == word[::-1]:
    print("String is a palindrome")
else:
    print("String is not a palindrome")
```

```
Enter a string: malayalam
String is a palindrome
```

40) WAP to calculate the length of the string

```
word = input("Enter a string : ")
print(f"Length of the string is : {len(word)}")
```

```
Enter a string : "You either die a hero or live long enough to see yourself become the villain."
Length of the string is : 79
```

41) WAP to count the number of characters (character frequency) in a string

```
word = "Batman"

frequency = {}

for i in word:
    if i in frequency:
        frequency[i] += 1
    else:
        frequency[i] = 1
print(f"Count of all characters in {word} is : {frequency}")
```

```
Count of all characters in Batman is : {'B': 1, 'a': 2, 't': 1, 'm': 1, 'n': 1}
```

42) WAP to get a single string from two strings separated by a space and swap the first two characters of each string

```
a = input("Enter the 1st string : ")
b = input("Enter the 2nd string : ")
print(f"Value after swapping is : {a} {b}")
temp = a[0:2]
a = b[0:2] + a[2:]
b = temp + b[2:]
print(f"Value after swapping is : {a} {b}")
```

```
Enter the 1st string : abc
Enter the 2nd string : xyz
Value after swapping is : abc xyz
Value after swapping is : xyc abz
```

43) WAP to accept a string and count the total number of vowels

```
string = input("Enter a string : ")
vowels = ['a', 'e', 'i', 'o', 'u']
count = 0
for i in string:
    if i in vowels:
        count = count + 1
print(count)
```

```
Enter a string : Muhammed Raihan
6
```

44) WAP to search a character in the string

```python
string = input("Enter a string : ")
character = input("Enter character that you want to search : ")
if character in string:
    print(f"{character} is present")
else:
    print(f"{character} is not present")
```

```
Enter a string : Muhammed Raihan
Enter character that you want to search : n
n is present
```

45) WAP to read the email id of the person in the format expected

```python
email_address = input("What is your email address? : ")
reverse = email_address[::-1]
if reverse[0:10] == "moc.liamg@":
    print(f"{email_address} is a valid email")
else:
    print(f"{email_address} is a not valid email")

# while "@" not in email_address:
#     email_address = input("Your email address must have '@' in it\nPlease write your email address again: ")
#     if len(email_address) <= 6 :
#         email_address = input("Your email address is too short\nPlease write your email address again: ")
#     if "." not in email_address:
#         email_address = input("Your email address must have '.' in it\nPlease write your email address again: ")
# while "." not in email_address:
#     email_address = input("Your email address must have '.' in it\nPlease write your email address again: ")
#     if len(email_address) <= 6 :
#         email_address = input("Your email address is too short\nPlease write your email address again: ")
#     if "@" not in email_address:
#         email_address = input("Your email address must have '@' in it\nPlease write your email address again: ")
```

```
What is your email address? : xyz@gmail.com
xyz@gmail.com is a valid email
```


```python
email_address = input("What is your email address? : ")
reverse = email_address[::-1]
if reverse[0:10] == "moc.liamg@":
    print(f"{email_address} is a valid email")
else:
    print(f"{email_address} is a not valid email")
```


```
What is your email address? : xyz@gmail.co
xyz@gmail.co is a not valid email
```

46) WAP to perform following on the list

1) Create a list of 5 names

2) Insert a name Anuj before Aditya

3) Append a name Zulu

4) Delete Avi from the list

5) Replace Anil with Anil Kumar

```python
# 1) Create a list of 5 names
name = ['Anil', 'Anmol', 'Aditya', 'Avi', 'Alka']
print(f"Output 1 : {name}")

# 2) Insert a name Anuj before Aditya
name.insert(2,'Anuj')
print(f"Output 2 : {name}")

# 3) Append a name Zulu
name.append('Zulu')
print(f"Output 3 : {name}")

# 4) Delete Avi from the list
name.remove('Avi')
print(f"Output 4 : {name}")

# 5) Replace Anil with Anil Kumar
name[0] = 'Anil Kumar'
print(f"Output 5 : {name}")
```

```
Output 1 : ['Anil', 'Anmol', 'Aditya', 'Avi', 'Alka']
Output 2 : ['Anil', 'Anmol', 'Anuj', 'Aditya', 'Avi', 'Alka']
Output 3 : ['Anil', 'Anmol', 'Anuj', 'Aditya', 'Avi', 'Alka', 'Zulu']
Output 4 : ['Anil', 'Anmol', 'Anuj', 'Aditya', 'Alka', 'Zulu']
Output 5 : ['Anil Kumar', 'Anmol', 'Anuj', 'Aditya', 'Alka', 'Zulu']
```

47) WAP to implement stack data structure which is LIFO

```python
stack = [1, 2, 3, 4, 5]
print(f"Initial stack : {stack}")
print()

def Push():
    element = int(input("Enter element that you want to push into the stack : "))
    stack.append(element)
    print(f"Pushed {element} into the stack")
    print(f"Stack after push operation : {stack}")
    print()

def Pop():
    element = stack.pop()
    print(f"Popped {element} from the stack")
    print(f"Stack after pop operation : {stack}")
    print()

Push()
Pop()
```

```
Initial stack : [1, 2, 3, 4, 5]

Enter element that you want to push into the stack : 6
Pushed 6 into the stack
Stack after push operation : [1, 2, 3, 4, 5, 6]

Popped 6 from the stack
Stack after pop operation : [1, 2, 3, 4, 5]
```

48) Suppose a list has 20 numbers. WAP that removes all duplicates from the list

```python
# 1st Method
numbers = [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 5, 8, 4, 9, 0, 10, 10, 3, 6]
unique_numbers = list(set(numbers))
print(f"Original List : {numbers}")
print(f"List after removing duplicates : {unique_numbers}")

# 2nd Method
numbers = [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 5, 8, 4, 9, 0, 10, 10, 3, 6]

# Remove duplicates using a loop
unique_numbers = []
for num in numbers:
    if num not in unique_numbers:
        unique_numbers.append(num)

print(f"Original List : {numbers}")
print(f"List after removing duplicates : {unique_numbers}")
```

```
Original List : [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 5, 8, 4, 9, 0, 10, 10, 3, 6]
List after removing duplicates : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Original List : [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 5, 8, 4, 9, 0, 10, 10, 3, 6]
List after removing duplicates : [1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 10]
```

49) WAP to obtain the median value of the list of numbers, without disturbing the order of numbers in the list

```python
def find_median():
    numbers = [7, 3, 9, 2, 5, 1, 8, 6, 4]
    print(f"Original list : {numbers}")
    sorted_list = sorted(numbers)
    n = len(sorted_list)
    if n % 2 == 0:
        # For even number of elements, median is the average of middle two elements
        mid1 = n // 2
        mid2 = mid1 - 1
        median = (sorted_list[mid1] + sorted_list[mid2]) / 2
    else:
        # For odd number of elements, median is the middle element
        median = sorted_list[n // 2]
    print(f"Median of numbers is : {median}")

find_median()
```

```
Original list : [7, 3, 9, 2, 5, 1, 8, 6, 4]
Median of numbers is : 5
```

50) WAP to count the occurrence of each character entered by the user and store them in a dictionary

```python
def count_characters(input_string):
    char_count = {}
    for char in input_string:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1
    return char_count

user_input = input("Enter a string: ")
result = count_characters(user_input)
print("Character count:")
for char, count in result.items():
    print(f"'{char}': {count}")
```

```
Enter a string: Raihan is a good boy
Character count:
'R': 1
'a': 3
'i': 2
'h': 1
'n': 1
' ': 4
's': 1
'g': 1
'o': 3
'd': 1
'b': 1
'y': 1
```

51) WAP to create two dictionaries, concatenate them and create a third dictionary

```python
def concatenate_dicts():
    # Creating two dictionaries
    dict1 = {'a': 1, 'b': 2, 'c': 3}
    print(f"First dictionary : {dict1}")
    dict2 = {'d': 4, 'e': 5, 'f': 6}
    print(f"Second dictionary : {dict2}")
    concatenated_dict = dict1
    concatenated_dict.update(dict2)
    print(f"Concatenated dictionary : {concatenated_dict}")

concatenate_dicts()
```

```
First dictionary : {'a': 1, 'b': 2, 'c': 3}
Second dictionary : {'d': 4, 'e': 5, 'f': 6}
Concatenated dictionary : {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
```

52) WAP to check weather a given key exists in the dictionary

```
name = {1:'Raihan', 2:'Rohan', 3:'Raman', 4:'Rajat', 5:'Roshan'}
index = int(input("Enter key that you want to search : "))

for key,value in name.items():
    if key == index:
        print(f"Key {key} exists and if value is : {value}")
```

```
Enter key that you want to search : 4
Key 4 exists and if value is : Rajat
```

53) WAP to remove a particular key from the dictionary

```python
def remove_key():
    name = {1:'Raihan', 2:'Rohan', 3:'Raman', 4:'Rajat', 5:'Roshan'}
    print(f"Dictionary before updation : {name}")
    key = int(input("Enter key that you want to remove : "))
    if key in name:
        name.pop(key)
        print(f"Key '{key}' removed successfully")
    else:
        print(f"Key '{key}' not found in the dictionary")
    print(f"Dictionary after updation : {name}")

remove_key()
```

```
Dictionary before updation : {1: 'Raihan', 2: 'Rohan', 3: 'Raman', 4: 'Rajat', 5: 'Roshan'}
Enter key that you want to remove : 2
Key '2' removed successfully
Dictionary after updation : {1: 'Raihan', 3: 'Raman', 4: 'Rajat', 5: 'Roshan'}
```

54) WAP to display the minimum and maximum in a dictionary

```python
def min_max_values():
    dictionary = {'a': 10, 'b': 5, 'c': 20, 'd': 15}
    if not dictionary:
        print("Dictionary is empty.")
        return

    min_value = min(dictionary.values())
    print(f"Minimum value : {min_value}")
    max_value = max(dictionary.values())
    print(f"Maximum value : {max_value}")

min_max_values()
```

```
Minimum value : 5
Maximum value : 20
```

55) WAP to reverse a tuple

```
numbers = (1,2,3,4,5)
print(type(numbers))
reverse = numbers[::-1]
print(numbers)
print(reverse)
```

```
<class 'tuple'>
(1, 2, 3, 4, 5)
(5, 4, 3, 2, 1)
```

56) WAP to calculate the product by multiplying all the numbers of a tuple

```
def product():
    numbers = (1,2,3,4,5)
    product = 1
    for i in numbers:
        product = product * i
    print(f"Product of these numbers are : {product}")

product()
```

```
Product of these numbers are : 120
```

57) WAP to check if the number exists in the tuple

```
numbers = (1,2,3,4,5,6,7,8,9,10)
search = int(input("Enter number that you want to search : "))

if search in numbers:
    print(f"{search} is present")
else:
    print(f"{search} is not present")
```

```
Enter number that you want to search : 5
5 is present
```

58) WAP to check if two sets have any elements in common. if yes display the common elements

```
number = {1,2,3,4,5}
numbers = {4,5,6,7,8}
common = number.intersection(numbers)
print(type(number))
print(f"Common elements between both the sets are : {common}")
```

```
<class 'set'>
Common elements between both the sets are : {4, 5}
```

59) Update set1 by adding items from set2, except common elements

```
number = {1,2,3,4,5}
numbers = {4,5,6,7,8}
union = number.union(numbers)
print(f"Union of both the sets are : {common}")
```

```
Union of both the sets are : {4, 5}
```

60) Remove items from set1 that are not common to both set1 and set2

```
number = {1,2,3,4,5}
numbers = {4,5,6,7,8}
set_difference = number.difference(numbers)
number = number - set_difference
print(number)
```

```
{4, 5}
```

61) WAP to print tuples which are not empty

```python
numbers = (1, 2, 3, 4, 5)
alphabets = ('a', 'b', 'c', 'd', 'e')
empty = ()

if len(empty) == 0:
    print("Tuple is empty")
else:
    print("Tuple is not empty")
```

```
Tuple is empty
```

62) WAP to open a file try.txt, add some text into it and close it

```
aboutme = open('try.txt', 'a')
```

```
aboutme.write('I am a graduate with a passion for continuous
```

```
420
```

```
aboutme.close()
```

jupyter try.txt 36 minutes ago

File Edit View Language

```
I am a graduate with a passion for continuous learning in the fields of Space, Artificial intelligence, Machine learning, Deep learning,
Computer vision and Programming. Currently, I am enrolled as a postgraduate student pursuing a Masters in Computer Application (MCA) at
Vivekananda Institute Of Professional Studies (VIPS, GGSIPU). I am committed to expanding my knowledge and skills in the field of computer
science.
```

63) WAP to read the above file and print that on the screen

```
aboutme = open('try.txt', 'r')
```

```
aboutme.read()
```

```
'I am a graduate with a passion for continuous learning in the fields of Space,
Artificial intelligence, Machine learning, Deep learning, Computer vision and P
rogramming. Currently, I am enrolled as a postgraduate student pursuing a Maste
rs in Computer Application (MCA) at Vivekananda Institute Of Professional Studi
es (VIPS, GGSIPU). I am committed to expanding my knowledge and skills in the f
ield of computer science.'
```

```
aboutme.close()
```

jupyter try.txt 36 minutes ago

File Edit View Language

```
1 I am a graduate with a passion for continuous learning in the fields of Space, Artificial intelligence, Machine learning, Deep learning,
  Computer vision and Programming. Currently, I am enrolled as a postgraduate student pursuing a Masters in Computer Application (MCA) at
  Vivekananda Institute Of Professional Studies (VIPS, GGSIPU). I am committed to expanding my knowledge and skills in the field of computer
  science.
```

64) WAP that writes four integers to a file called numbers Go to the following position and return

a) 10 positions from the beginning

b) 2 position to the right of current position

c) 10 position to the left from the end

```
file = open("numbers.txt", "w")
file.write("100,200,300,400")
file = open("numbers.txt", "rb")
file.read()
```

```
b'100,200,300,400'
```

```
# 10 positions from the beginning
file.seek(10,0)
file.read()
```

```
b'0,400'
```

```
# 2 position to the right of current position
file.seek(2,1)
file.read()
```

```
b''
```

```
# 10 position to the left from the end
file.seek(-10,2)
file.read()
```

```
b'00,300,400'
```

65) WAP to implement inheritance using shape class and also make rectangle as a derived class. declare a function area to calculate the area of rectangle and also show the colour of the rectangle. Also make required constructor

```
class shape:
    def __init__(self, color):
        self.color = color

class Rectangle(shape):
    def __init__(self, color, length, width):
        super().__init__(color)
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def display_info(self):
        print(f"I am a {self.color} rectangle with area {self.area()}. square units.")

my_rectangle = Rectangle(color="Blue", length=5.0, width=3.0)
my_rectangle.display_info()
```

```
I am a Blue rectangle with area 15.0. square units.
```

66) WAP to implement Multiple Inheritance using the example of employee class, waged employee and salaried employee. Write Appropriate constructors and functions to calculate pay.

```python
class Employee:
    def __init__(self, emp_id, name):
        self.emp_id = emp_id
        self.name = name

class WagedEmployee(Employee):
    def __init__(self, emp_id, name, hourly_rate, hours_worked):
        super().__init__(emp_id, name)
        self.hourly_rate = hourly_rate
        self.hours_worked = hours_worked

    def calculate_pay(self):
        return self.hourly_rate * self.hours_worked

class SalariedEmployee(Employee):
    def __init__(self, emp_id, name, monthly_salary):
        super().__init__(emp_id, name)
        self.monthly_salary = monthly_salary

    def calculate_pay(self):
        return self.monthly_salary

waged_emp = WagedEmployee(emp_id=101, name="John", hourly_rate=20, hours_worked=40)
print(f"Waged Employee Pay: ${waged_emp.calculate_pay()}")

salaried_emp = SalariedEmployee(emp_id=102, name="Alice", monthly_salary=5000)
print(f"Salaried Employee Pay: ${salaried_emp.calculate_pay()}")
```

```
Waged Employee Pay: $800
Salaried Employee Pay: $5000
```

67) Implement operator overloading for + operator

68) Implement operator overloading for less than equal to operator for user defined class

69) Implement operator overloading for - operator

70) Implement operator overloading for unary operator ++ and -- operator

71) Implement exception handling for divide by 0 exception

72) Implement exception handling using raise statement

73) Implement threading

74) WAP to construct a 3d array of dimension 4x2x3. Initialize the array to some value. Find the maximum along each axis (k,413)

```python
import numpy as np

# Construct a 3D array with dimensions 4x2x3
a = np.arange(0,24)
a = a.reshape(4,2,3)

# Print the initialized 3D array
print("Initialized 3D array:")
print(a)

# Find the maximum values along each axis
max_along_axis0 = np.max(a, axis=0)
max_along_axis1 = np.max(a, axis=1)
max_along_axis2 = np.max(a, axis=2)

# Print the maximum values along each axis
print("\nMaximum along axis 0:")
print(max_along_axis0)
print("\nMaximum along axis 1:")
print(max_along_axis1)
print("\nMaximum along axis 2:")
print(max_along_axis2)
```

```
Initialized 3D array:
[[[ 0  1  2]
  [ 3  4  5]]

 [[ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]]]

Maximum along axis 0:
[[18 19 20]
 [21 22 23]]

Maximum along axis 1:
[[ 3  4  5]
 [ 9 10 11]
 [15 16 17]
 [21 22 23]]

Maximum along axis 2:
[[ 2  5]
 [ 8 11]
 [14 17]
 [20 23]]
```

75) Create a program to achieve the following

- **Create an array of size 10 with each element of it set to a value 3**
- **Find the memory size of this array and its individual element**
- **Create an array b of size 10 with values ranging from 0 to 90 evenly spaced.**
- **Reverse elements of array b**
- **Add arrays a and b and store the result in c**

```python
import numpy as np

# Step 1: Create an array of size 10 with each element set to 3
a = np.full(10, 3)
print("Array a:")
print(a)

# Step 2: Find the memory size of the array and its individual element
memory_size_of_array = a.nbytes
memory_size_of_element = a.itemsize
print("\nMemory size of the entire array 'a':", memory_size_of_array, "bytes")
print("Memory size of each element in array 'a':", memory_size_of_element, "bytes")

# Step 3: Create an array b of size 10 with values ranging from 0 to 90 evenly spaced
b = np.linspace(0, 90, 10)
print("\nArray b:")
print(b)

# Step 4: Reverse the elements of array b
b_reversed = b[::-1]
print("\nReversed array b:")
print(b_reversed)

# Step 5: Add arrays a and b_reversed and store the result in c
c = a + b
print("\nResulting array c (a + b):")
print(c)
```

```
Array a:
[3 3 3 3 3 3 3 3 3 3]

Memory size of the entire array 'a': 40 bytes
Memory size of each element in array 'a': 4 bytes

Array b:
[ 0. 10. 20. 30. 40. 50. 60. 70. 80. 90.]

Reversed array b:
[90. 80. 70. 60. 50. 40. 30. 20. 10.  0.]

Resulting array c (a + b):
[ 3. 13. 23. 33. 43. 53. 63. 73. 83. 93.]
```

76) Declare a matrix o 3x3 and calculate its transpose

```
import numpy as np
a = np.arange(0,9)
a = a.reshape(3,3)
print(a)
b = a.transpose()
print()
print(b)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]

[[0 3 6]
 [1 4 7]
 [2 5 8]]
```

77) Write a menu driven program to perform all the matrix operation

```python
import numpy as np

def get_matrix(prompt):
    print(prompt)
    rows = int(input("Enter the number of rows: "))
    cols = int(input("Enter the number of columns: "))
    n = rows * cols
    elements = []
    for i in range(n):
        value = int(input(f"Enter element {i+1}: "))
        elements.append(value)
    elements = np.array(elements)
    matrix = elements.reshape(rows, cols)
    print("The matrix is : ")
    print(matrix)
    return matrix

def add_matrices():
    matrix1 = get_matrix("Enter the first matrix:")
    matrix2 = get_matrix("Enter the second matrix:")
    if matrix1.shape == matrix2.shape:
        result = np.add(matrix1, matrix2)
        print("Result of addition:")
        print(result)
    else:
        print("Matrices must have the same dimensions to be added.")
```

```python
def subtract_matrices():
    matrix1 = get_matrix("Enter the first matrix:")
    matrix2 = get_matrix("Enter the second matrix:")
    if matrix1.shape == matrix2.shape:
        result = np.subtract(matrix1, matrix2)
        print("Result of subtraction:")
        print(result)
    else:
        print("Matrices must have the same dimensions to be subtracted.")

def multiply_matrices():
    matrix1 = get_matrix("Enter the first matrix:")
    matrix2 = get_matrix("Enter the second matrix:")
    if matrix1.shape[1] == matrix2.shape[0]:
        result = np.dot(matrix1, matrix2)
        print("Result of multiplication:")
        print(result)
    else:
        print("Number of columns in the first matrix must be equal to the number of rows in the second matrix.")

def transpose_matrix():
    matrix = get_matrix("Enter the matrix:")
    result = np.transpose(matrix)
    print("Transpose of the matrix:")
    print(result)
```

```python
def determinant_matrix():
    matrix = get_matrix("Enter the matrix:")
    if matrix.shape[0] == matrix.shape[1]:
        result = np.linalg.det(matrix)
        print("Determinant of the matrix:")
        print(result)
    else:
        print("Determinant can only be calculated for square matrices.")
```

```python
def main():
    while True:
        print("\nMatrix Operations Menu:")
        print("1. Add matrices")
        print("2. Subtract matrices")
        print("3. Multiply matrices")
        print("4. Transpose a matrix")
        print("5. Find the determinant of a matrix")
        print("6. Exit")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            add_matrices()
        elif choice == 2:
            subtract_matrices()
        elif choice == 3:
            multiply_matrices()
        elif choice == 4:
            transpose_matrix()
        elif choice == 5:
            determinant_matrix()
        elif choice == 6:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
```

```
Matrix Operations Menu:
1. Add matrices
2. Subtract matrices
3. Multiply matrices
4. Transpose a matrix
5. Find the determinant of a matrix
6. Exit
Enter your choice: 1
Enter the first matrix:
Enter the number of rows: 2
Enter the number of columns: 2
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
The matrix is :
[[1 2]
 [3 4]]
Enter the second matrix:
Enter the number of rows: 2
Enter the number of columns: 2
Enter element 1: 5
Enter element 2: 6
Enter element 3: 7
Enter element 4: 8
The matrix is :
[[5 6]
 [7 8]]
Result of addition:
[[ 6  8]
 [10 12]]
```

78) WAP to count all the odd numbers in 1d array

```
import numpy as np
numbers = np.arange(0,20)
count = 0
for i in numbers:
    if i%2==0:
        continue
    else:
        print(i)
        count = count+1
print(f"Total numbers of odd numbers in the array is : {count}")
```

```
1
3
5
7
9
11
13
15
17
19
Total numbers of odd numbers in the array is : 10
```

79) WAP to check if none of the elements is zero in the array

```
import numpy as np
first = int(input("Enter the first element of the array : "))
last = int(input("Enter the last element of the array : "))
numbers= np.arange(first,last+1)
print(f"Array : {numbers}")

for i in numbers:
    if numbers[i] == 0:
        print(f"Element at index {i} is {numbers[i]}")
```

```
Enter the first element of the array : -5
Enter the last element of the array : 5
Array : [-5 -4 -3 -2 -1  0  1  2  3  4  5]
Element at index 5 is 0
```

80) WAP to extract all the prime numbers from an array

```
import numpy as np

first = int(input("Enter the first element of the array : "))
last = int(input("Enter the last element of the array : "))
array= np.arange(first, last+1)
print(f"Array : {array}")

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def print_primes(array):
    print("Prime numbers in the array:")
    for num in array:
        if is_prime(num):
            print(num, end=", ")

print_primes(array)
```

```
Enter the first element of the array : 0
Enter the last element of the array : 15
Array : [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
Prime numbers in the array:
2, 3, 5, 7, 11, 13,
```

81) Pre-process and perform different operations on data using Numpy and Pandas library

```
import pandas as pd
import numpy as np
```

```
# Read in data
df = pd.read_csv('Reviews.csv')
```

```
df.shape
```

```
(568454, 10)
```

```
df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

```
df.describe()
```

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	568454.000000	568454.000000	568454.00000	568454.000000	5.684540e+05
mean	284227.500000	1.743817	2.22881	4.183199	1.296257e+09
std	164098.679298	7.636513	8.28974	1.310436	4.804331e+07
min	1.000000	0.000000	0.00000	1.000000	9.393408e+08
25%	142114.250000	0.000000	0.00000	4.000000	1.271290e+09
50%	284227.500000	0.000000	1.00000	5.000000	1.311120e+09
75%	426340.750000	2.000000	2.00000	5.000000	1.332720e+09
max	568454.000000	866.000000	923.00000	5.000000	1.351210e+09

```
df.isnull()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
568449	False	False	False	False	False	False	False	False	False	False
568450	False	False	False	False	False	False	False	False	False	False
568451	False	False	False	False	False	False	False	False	False	False
568452	False	False	False	False	False	False	False	False	False	False
568453	False	False	False	False	False	False	False	False	False	False

568454 rows × 10 columns

```
df.isnull().sum()
```

```
Id                      0
ProductId               0
UserId                  0
ProfileName            16
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                27
Text                    0
dtype: int64
```

```
# Used to fill the missing values but in this case missiing values are catagorical data not numerical data
# df['Score'] = df['Score'].fillna(df['Score'].mean())
```

```
df = df.dropna()
```

```
df.isnull().sum()
```

```
Id                      0
ProductId               0
UserId                  0
ProfileName             0
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                 0
Text                    0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 568411 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568411 non-null  int64
 1   ProductId               568411 non-null  object
 2   UserId                  568411 non-null  object
 3   ProfileName             568411 non-null  object
 4   HelpfulnessNumerator    568411 non-null  int64
 5   HelpfulnessDenominator  568411 non-null  int64
 6   Score                   568411 non-null  int64
 7   Time                    568411 non-null  int64
 8   Summary                 568411 non-null  object
 9   Text                    568411 non-null  object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 568411 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568411 non-null  int64
 1   ProductId               568411 non-null  object
 2   UserId                  568411 non-null  object
 3   ProfileName             568411 non-null  object
 4   HelpfulnessNumerator    568411 non-null  int64
 5   HelpfulnessDenominator  568411 non-null  int64
 6   Score                   568411 non-null  int64
 7   Time                    568411 non-null  int64
 8   Summary                 568411 non-null  object
 9   Text                    568411 non-null  object
dtypes: int64(5), object(5)
memory usage: 47.7+ MB
```

```
df_cat = df.select_dtypes(include= ['object'])
df_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 568411 entries, 0 to 568453
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   ProductId    568411 non-null  object
 1   UserId       568411 non-null  object
 2   ProfileName  568411 non-null  object
 3   Summary      568411 non-null  object
 4   Text         568411 non-null  object
dtypes: object(5)
memory usage: 26.0+ MB
```

```
df['Score'].unique()
```

```
array([5, 1, 4, 2, 3], dtype=int64)
```

```
df['Score'].value_counts()
```

```
5    363111
4     80655
1     52264
3     42638
2     29743
Name: Score, dtype: int64
```

```
df.groupby('Score').mean()
```

```
C:\Users\psriz\AppData\Local\Temp\ipykernel_6844\4169637139.py:1: FutureWarning: The default value of numeric_only in DataFrame
GroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df.groupby('Score').mean()
```

Score	Id	HelpfulnessNumerator	HelpfulnessDenominator	Time
1	282277.896047	2.735133	4.869738	1.303159e+09
2	280775.355277	1.859799	3.085264	1.301239e+09
3	279643.956893	1.701018	2.466485	1.300125e+09
4	281713.258608	1.390292	1.666084	1.296722e+09
5	285887.479828	1.675273	1.874160	1.294305e+09

82) Analyze the sales data and visualize it using different plots

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# Step 1: Load the Data
df = pd.read_csv('Sales.csv')
# Step 2: Explore the Data
df.head()
```

Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4
123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	8.4
373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3

```python
df.info()
```

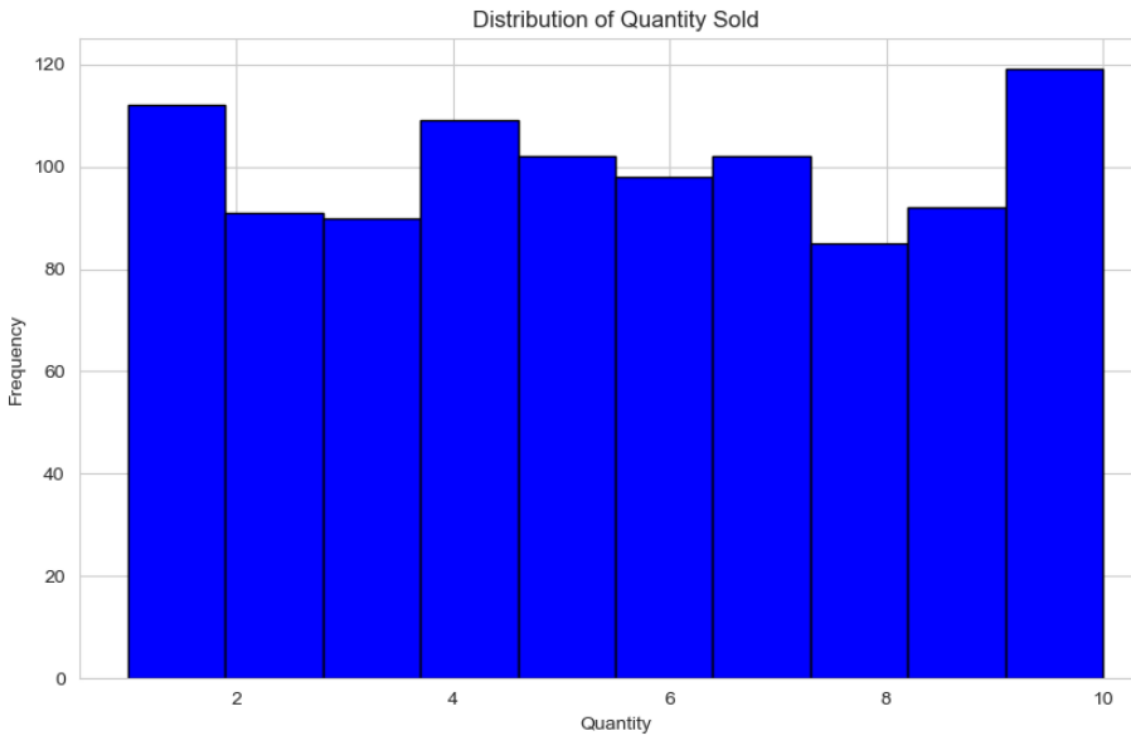
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Invoice ID               1000 non-null   object
 1   Branch                   1000 non-null   object
 2   City                     1000 non-null   object
 3   Customer type            1000 non-null   object
 4   Gender                   1000 non-null   object
 5   Product line             1000 non-null   object
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object
 11  Time                     1000 non-null   object
 12  Payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

```python
df.describe()
```

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000

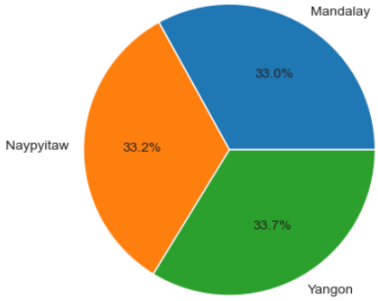
```python
# Step 3: Data Cleaning (if needed)
# For example:
# sales_data.dropna(inplace=True)
# sales_data.drop_duplicates(inplace=True)
```

```python
# Histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Quantity'], bins=10, color='blue', edgecolor='black')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.title('Distribution of Quantity Sold')
plt.show()
```



```python
# Pie Chart: Sales Distribution by Region
plt.subplot(1, 1, 1)
sales_by_region = sales_data.groupby('City')['Quantity'].sum()
plt.pie(sales_by_region, labels=sales_by_region.index, autopct='%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x1504de34190>,
  <matplotlib.patches.Wedge at 0x1504e252ce0>,
  <matplotlib.patches.Wedge at 0x1504de372e0>],
 [Text(0.5590275782441871, 0.9473585207103166, 'Mandalay'),
  Text(-1.0997280606941795, 0.0244579746058285, 'Naypyitaw'),
  Text(0.5378253125953597, -0.9595540282504701, 'Yangon')],
 [Text(0.3049241335877384, 0.5167410112965363, '33.0%'),
  Text(-0.5998516694695524, 0.013340713421360997, '33.2%'),
  Text(0.2933592614156507, -0.5233931063184382, '33.7%')])
```




```python
# Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Unit price'], df['Total'], color='orange', alpha=0.5)
plt.xlabel('Unit Price')
plt.ylabel('Total Sales')
plt.title('Scatter Plot: Unit Price vs Total Sales')
plt.show()
```


```python
# Heatmap
plt.figure(figsize=(10, 6))
heatmap_data = df.corr()
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\psriz\AppData\Local\Temp\ipykernel_6844\3551810091.py:3: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  heatmap_data = df.corr()
```