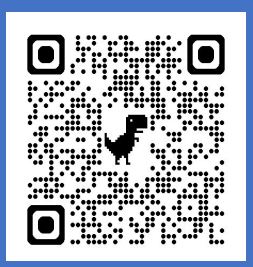
PYTHON LAB FILE



MINE GIT HUB ACCOUNT

Experiment 1: Python Installation and starting with python

2. Write Python programs to print strings in the given manner:

a) Hello Everyone !!!

b) Hello

World

c) Hello

World

d) ‘ Rohit’ s date of birth is 12\05\1999’

```
print("a) Hello Everyone !!!")
print("b) Hello\n World")
print("c) Hello\n\t World")
print("d) 'THOR's date of birth is 11\\08\\1983'")
```

OUTPUT:

```
a) Hello Everyone !!!
b) Hello
 World
c) Hello
     World
d) 'THOR's date of birth is 11\08\1983'
```

3 Declare a string variable called x and assign it the value “Hello”.

Print out the value of x

```
x = "Hello"
print("Value of x:", x)
```

OUTPUT:

```
Value of x: Hello

=== Code Execution Successful ===
```

4 Take different data types and print values using print function.

```
integer_number = 10
float_number = 3.14
boolean_value = True
string_value = "Hello, World!"
list_value = [1, 2, 3, 4, 5]
tuple_value = (1, 2, 3)
dictionary_value = {"key1": "value1", "key2": "value2"}
```

```
    # Printing values using print function
print("Integer Number:", integer_number)
print("Float Number:", float_number)
print("Boolean Value:", boolean_value)
print("String Value:", string_value)
print("List Value:", list_value)
print("Tuple Value:", tuple_value)
print("Dictionary Value:", dictionary_value)
```

OUTPUT:

```
Integer Number: 10
Float Number: 3.14
Boolean Value: True
String Value: Hello, World!
List Value: [1, 2, 3, 4, 5]
Tuple Value: (1, 2, 3)
Dictionary Value: {'key1': 'value1', 'key2': 'value2'}

=== Code Execution Successful ===
```

5 Take two variable a and b. Assign your first name and last name. Print your Name

after adding your First name and Last name together.

```
a = "John"
b = "Doe"
print("Full Name:", a + " " + b)
```

OUTPUT:

```
Full Name: John Doe

=== Code Execution Successful ===
```

6 Declare three variables, consisting of your first name, your last name and

Nickname. Write a program that prints out your first name, then your nickname in

parenthesis and then your last name.

Example output : George (woody) Washington.

```
first_name = "George"
last_name = "Washington"
nickname = "Woody"
print(first_name, "(", nickname, ")", last_name + ".")
```

OUTPUT:

```
George ( Woody ) Washington.

=== Code Execution Successful ===
```

7 Declare and assign values to suitable variables and print in the following way :

NAME : NIKUNJ BANSAL

SAP ID : 500069944

DATE OF BIRTH : 13 Oct 1999

ADDRESS : UPES

Bidholi Campus

Pincode : 248007

Programme : AI & ML

Semester : 2

```
print("NAME:", "NIKUNJ BANSAL")
print("SAP ID:", "500069944")
print("DATE OF BIRTH:", "13 Oct 1999")
print("ADDRESS: UPES")
print("        Bidholi Campus")
print("Pincode:", 248007)
print("Programme:", "AI & ML")
print("Semester:", 2)
```

OUTPUT:

```
NAME: NIKUNJ BANSAL
SAP ID: 500069944
DATE OF BIRTH: 13 Oct 1999
ADDRESS: UPES
         Bidholi Campus
Pincode: 248007
Programme: AI & ML
Semester: 2

=== Code Execution Successful ===
```

Experiment 2: Use of input statements, operators

1. Declare these variables (x, y and z) as integers. Assign a value of 9 to x, Assign

a value of 7 to y, perform addition, multiplication, division and subtraction on

these two variables and Print out the result.

```
x = 9
y = 7
z=1
print("Addition:", x + y)
print("Multiplication:", x * y)
print("Division:", x / y)
print("Subtraction:", x - y)
```

OUTPUT:

```
Addition: 16
Multiplication: 63
Division: 1.2857142857142858
Subtraction: 2

=== Code Execution Successful ===
```

2. Write a Program where the radius is taken as input to compute the area of a circle.

```
radius = float(input("Enter the radius of the circle: "))
area = 3.14159 * radius ** 2
print("Area of the circle:", area)
```

OUTPUT:

```
Enter the radius of the circle: 5
Area of the circle: 78.53975

=== Code Execution Successful ===
```

3. Write a Python program to solve (x+y)*(x+y)

Test data : x = 4 , y = 3

Expected output: 49

```
x = 4
y = 3
result = (x + y) * (x + y)
print("Result:", result)
```

OUTPUT:

```
Result: 49

=== Code Execution Successful ===
```

4. Write a program to compute the length of the hypotenuse (c) of a right triangle

using Pythagoras theorem.

```
import math
side1 = float(input("Enter length of side 1: "))
side2 = float(input("Enter length of side 2: "))
hypotenuse = math.sqrt(side1 ** 2 + side2 ** 2)
print("Length of the hypotenuse:", hypotenuse)
```

OUTPUT:

```
Enter length of side 1: 3
Enter length of side 2: 4
Length of the hypotenuse: 5.0

=== Code Execution Successful ===
```

5. Write a program to find simple interest.

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time period: "))
simple_interest = (principal * rate * time) / 100
print("Simple Interest:", simple_interest)
```

OUTPUT:

```
Enter the principal amount: 5000
Enter the rate of interest: 8
Enter the time period: 7
Simple Interest: 2800.0

=== Code Execution Successful ===
```

6. Write a program to find area of triangle when length of sides are given.

```
import math
a = float(input("Enter length of side 1: "))
b = float(input("Enter length of side 2: "))
c = float(input("Enter length of side 3: "))
s = (a + b + c) / 2
area_triangle = math.sqrt(s * (s - a) * (s - b) * (s - c))
print("Area of the triangle:", area_triangle)
```

OUTPUT:

```
Enter length of side 1: 3
Enter length of side 2: 4
Enter length of side 3: 5
Area of the triangle: 6.0

=== Code Execution Successful ===
```

7. Write a program to convert given seconds into hours, minutes and remaining seconds.

```
seconds = int(input("Enter the number of seconds: "))
hours = seconds // 3600
minutes = (seconds % 3600) // 60
remaining_seconds = (seconds % 3600) % 60
print("Hours:", hours)
print("Minutes:", minutes)
print("Remaining Seconds:", remaining_seconds)
```

OUTPUT:

```
Enter the number of seconds: 120
Hours: 0
Minutes: 2
Remaining Seconds: 0

=== Code Execution Successful ===
```

8. Write a program to swap two numbers without taking additional variable.

```
a = int(input("Enter the value of a: "))
b = int(input("Enter the value of b: "))
a = a + b
b = a - b
a = a - b
print("After swapping: a =", a, ", b =", b)
```

OUTPUT:

```
Enter the value of a: 5
Enter the value of b: 6
After swapping: a = 6 , b = 5

=== Code Execution Successful ===
```

9. Write a program to find sum of first n natural numbers.

```
n = int(input("Enter the value of n: "))
sum_natural_numbers = (n * (n + 1)) // 2
print("Sum of first", n, "natural numbers:", sum_natural_numbers)
```

OUTPUT:

```
Enter the value of n: 5
Sum of first 5 natural numbers: 15

=== Code Execution Successful ===
```

10.Write a program to print truth table for bitwise operators(& , | and ^ operators)

```
print("Truth table for AND operator (&):")
print("0 & 0 =", 0 & 0)
print("0 & 1 =", 0 & 1)
print("1 & 0 =", 1 & 0)
print("1 & 1 =", 1 & 1)

print("Truth table for OR operator (|):")
print("0 | 0 =", 0 | 0)
print("0 | 1 =", 0 | 1)
print("1 | 0 =", 1 | 0)
print("1 | 1 =", 1 | 1)

print("Truth table for XOR operator (^):")
print("0 ^ 0 =", 0 ^ 0)
print("0 ^ 1 =", 0 ^ 1)
print("1 ^ 0 =", 1 ^ 0)
print("1 ^ 1 =", 1 ^ 1)
```

OUTPUT:

```
Truth table for AND operator (&):
0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1
Truth table for OR operator (|):
0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1
Truth table for XOR operator (^):
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

=== Code Execution Successful ===
```

11.Write a program to find left shift and right shift values of a given number.

```
number = int(input("Enter a number: "))
left_shift = number << 1
right_shift = number >> 1
print("Left shift of", number, ":", left_shift)
print("Right shift of", number, ":", right_shift)
```

OUTPUT:

```
Enter a number: 5
Left shift of 5 : 10
Right shift of 5 : 2

=== Code Execution Successful ===
```

12.Using membership operator find whether a given number is in sequence (10,20,56,78,89)

```
number = int(input("Enter a number: "))
sequence = [10, 20, 56, 78, 89]
if number in sequence:
    print(number, "is in the sequence.")
else:
    print(number, "is not in the sequence.")
```

OUTPUT:

```
Enter a number: 55
55 is not in the sequence.
```

```
Enter a number: 89
89 is in the sequence.
```

13.Using membership operator find whether a given character is in a string.

```
character = input("Enter a character: ")
string = input("Enter a string: ")
if character in string:
    print(character, "is present in the string.")
else:
    print(character, "is not present in the string.")
```

OUTPUT:

```
Enter a character: T
Enter a string: TUSHAR SAINI
T is present in the string.

=== Code Execution Successful ===
```

Experiment 3 : Conditional Statements

1. Check whether given number is divisible by 3 and 5 both.

```
user_input = input("Enter a number: ")
num = int(user_input)
print("You entered:", num)

if num % 3 == 0 and num % 5 == 0:
    print(f"{num} is divisible by both 3 and 5.")
else:
    print(f"{num} is not divisible by both 3 and 5.")
```

OUTPUT:

```
Enter a number: 445
You entered: 445
445 is not divisible by both 3 and 5.
```

```
Enter a number: 15
You entered: 15
15 is divisible by both 3 and 5.

=== Code Execution Successful ===
```

2. Check whether a given number is multiple of five or not.

```python
user_input = input("Enter a number: ")
num = int(user_input)
print("You entered:", num)

if num % 5 == 0:
    print(f"{num} is a multiple of five.")
else:
    print(f"{num} is not a multiple of five.")
```

OUTPUT:

```
Enter a number: 45
You entered: 45
45 is a multiple of five.

=== Code Execution Successful ===
```

```
Enter a number: 12
You entered: 12
12 is not a multiple of five.

=== Code Execution Successful ===
```

3. Find the greatest among two numbers. If numbers are equal than print

“numbers are equal”.

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

if num1 == num2:
    print("Numbers are equal")
elif num1 > num2:
    print(f"The greatest number is {num1}")
else:
    print(f"The greatest number is {num2}")
```

output:

```
Enter the first number: 45
Enter the second number: 46
The greatest number is 46

=== Code Execution Successful ===
```

4. Find the greatest among three numbers assuming no two values are same.

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
num3 = int(input("Enter the third number: "))

if num1 > num2 and num1 > num3:
    print(f"The greatest number is {num1}")
elif num2 > num1 and num2 > num3:
    print(f"The greatest number is {num2}")
else:
    print(f"The greatest number is {num3}")
```

output:

```
Enter the first number: 33
Enter the second number: 55
Enter the third number: 0
The greatest number is 55

=== Code Execution Successful ===
```

5. Check whether the quadratic equation has real roots or imaginary roots. Display the roots.

```
a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))

discriminant = b**2 - 4*a*c
if discriminant > 0:
    root1 = (-b + discriminant**0.5) / (2*a)
    root2 = (-b - discriminant**0.5) / (2*a)
    print("Real roots:", root1, root2)
elif discriminant == 0:
    root = -b / (2*a)
    print("Real and identical roots:", root)
else:
    real_part = -b / (2*a)
    imaginary_part = (-discriminant)**0.5 / (2*a)
    print("Imaginary roots:", real_part + imaginary_part, real_part -
imaginary_part)
```

output:

```
Enter coefficient a: 2
Enter coefficient b: 3
Enter coefficient c: 4
Imaginary roots: 0.4489578808281798 -1.9489578808281798

=== Code Execution Successful ===
```

6. Find whether a given year is a leap year or not.

```
year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

output:

```
Enter a year: 2012
2012 is a leap year.

=== Code Execution Successful ===
```

```
Enter a year: 2001
2001 is not a leap year.

=== Code Execution Successful ===
```

7. Write a program which takes any date as input and display next date of the calendar

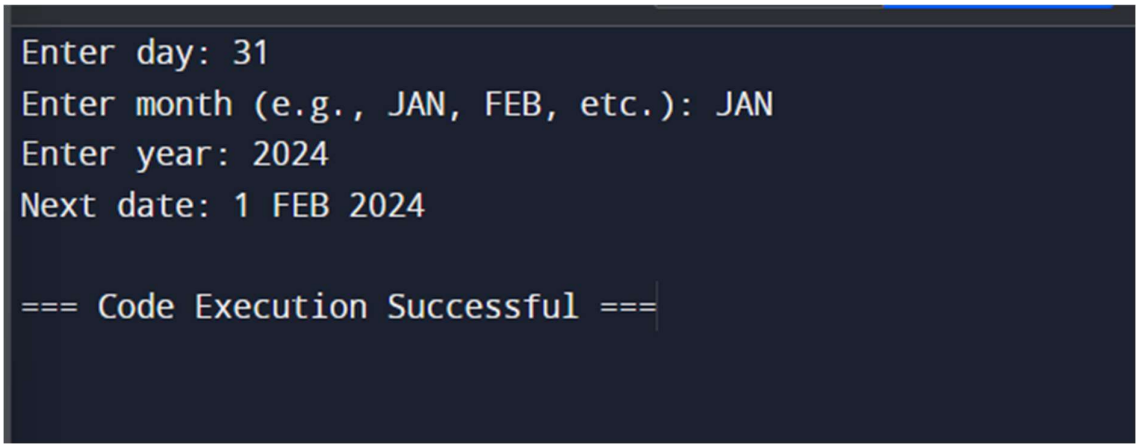
e.g.

I/P: day=20 month=9 year=2005

O/P: day=21 month=9 year 2005

```
day = int(input("Enter day: "))
month = input("Enter month (e.g., JAN, FEB, etc.): ").upper()  # Convert input
to uppercase
year = int(input("Enter year: "))

month_names = ['', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG',
'SEP', 'OCT', 'NOV', 'DEC']
days_in_month = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

# Simplified leap year check
if year % 4 == 0 and (year % 100!= 0 or year % 400 == 0):
    days_in_month[2] = 29

# Calculate the next date
next_day = day + 1
next_month = month_names[month_names.index(month)]

if next_day > days_in_month[month_names.index(next_month)]:
    next_day = 1
    next_month = month_names[(month_names.index(next_month) + 1) % 12]

if next_month == month_names[1]:  # If next month is February, check for leap
year
    if year % 4 == 0 and (year % 100!= 0 or year % 400 == 0):
        next_day = 1
        next_month = month_names[month_names.index(next_month) + 1]

print("Next date:", next_day, next_month, year)
```

OUTPUT:



8. Print the grade sheet of a student for the given range of cgpa. Scan marks of five subjects and calculate the percentage.

CGPA=percentage/10

CGPA range:

0 to 3.4 -> F

3.5 to 5.0->C+

5.1 to 6->B

6.1 to 7-> B+

7.1 to 8-> A

8.1 to 9->A+

9.1 to 10-> O (Outstanding)

Sample Gradesheet

Name: Rohit Sharma

Roll Number: R17234512 SAPID: 50005673

Sem: 1 Course: B.Tech. CSE AI&ML

Subject name: Marks

PDS: 70

Python: 80

Chemistry: 90

English: 60

Physics: 50

Percentage: 70%

CGPA:7.0

Grade:

```
name = "Rohit Sharma"
roll_number = "R17234512"
sap_id = "50005673"
sem = 1
course = "B.Tech. CSE AI&ML"
marks = {"PDS": 70, "Python": 80, "Chemistry": 90, "English": 60, "Physics": 50}

total_marks = sum(marks.values())
percentage = total_marks / (len(marks) * 100) * 100
cgpa = percentage / 10

if cgpa >= 0 and cgpa <= 3.4:
    grade = "F"
elif cgpa <= 5.0:
    grade = "C+"
elif cgpa <= 6:
    grade = "B"
elif cgpa <= 7:
    grade = "B+"
elif cgpa <= 8:
    grade = "A"
elif cgpa <= 9:
    grade = "A+"
else:
    grade = "O (Outstanding)"

print("Sample Gradesheet")
print("Name:", name)
print("Roll Number:", roll_number)
print("SAPID:", sap_id)
print("Sem:", sem)
```

```
print("Course:", course)
print("Subject name: Marks")
for subject, mark in marks.items():
    print(subject + ":", mark)
print("Percentage:", f"{percentage:.2f}%")
print("CGPA:", f"{cgpa:.1f}")
print("Grade:", grade)
```

OUTPUT:

```
Sample Gradesheet
Name: Rohit Sharma
Roll Number: R17234512
SAPID: 50005673
Sem: 1
Course: B.Tech. CSE AI&ML
Subject name: Marks
PDS: 70
Python: 80
Chemistry: 90
English: 60
Physics: 50
Percentage: 70.00%
CGPA: 7.0
Grade: B+

=== Code Execution Successful ===
```

Experiment 4: Loops

1. Find a factorial of given number.

```python
num = int(input("Namaskaram\nEnter a number whose factorial you wanna find: "))
factorial = 1

if num < 0:
    print("Sorry, factorial does not exist for negative numbers\n Ab zaa jaake maths padh")
elif num == 0:
    print("The factorial of 0 is 1 \n just like without GF you are single ")
else:
    for i in range(1, num + 1):
        factorial *= i
    print(f"The factorial of {num} is {factorial}")
```

OUTPUT:

```
Namaskaram
Enter a number whose factorial you wanna find: 5
The factorial of 5 is 120

=== Code Execution Successful ===
```

2. Find whether the given number is Armstrong number.

```python
num = int(input("Namaskaram,\nEnter a number you wanna check if a armsstrong number or not: "))
original_num = num
sum_of_cubes = 0

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

if original_num == sum_of_cubes:
    print(f"{original_num} is an Armstrong number")
```

```
else:
    print(f"{original_num} is not an Armstrong number")
```

OUTPUT:

```
Namaskaram,
Enter a number you wanna check if a armsstrong number or not: 5
5 is not an Armstrong number

=== Code Execution Successful ===
```

```
Namaskaram,
Enter a number you wanna check if a armsstrong number or not: 153
153 is an Armstrong number

=== Code Execution Successful ===
```

3. Print Fibonacci series up to given term.

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1

if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print(f"Fibonacci sequence up to {nterms}:")
    print(n1)
else:
    print("Fibonacci sequence:")
    while nterms > 0:
        print(n1, end=" ")
        nth = n1 + n2
        n1 = n2
        n2 = nth
        nterms -= 1
```

OUTPUT:

```
How many terms? 5
Fibonacci sequence:
0 1 1 2 3
=== Code Execution Successful ===
```

4. Write a program to find if given number is prime number or not.

```python
# Check if a given number is prime
num = int(input("Enter a number: "))

if num <= 1:
   print(f"{num} is not a prime number")
else:
   for i in range(2, num):
      if num % i == 0:
         print(f"{num} is not a prime number")
         break
   else:
      print(f"{num} is a prime number")
```

OUTPUT:

```
Enter a number: 45
45 is not a prime number

=== Code Execution Successful ===
```

```
Enter a number: 5
5 is a prime number

=== Code Execution Successful ===
```

5. Check whether given number is palindrome or not.

```python
# Check if a given number is palindrome
num = input("Enter a number: ")

if num == num[::-1]:
   print(f"{num} is a Palindrome")
else:
   print(f"{num} is not a Palindrome")
```

OUTPUT:

```
Enter a number: 101
101 is a Palindrome

=== Code Execution Successful ===
```

```
Enter a number: 95
95 is not a Palindrome

=== Code Execution Successful ===
```

6. Write a program to print sum of digits.

```python
# Print the sum of digits in a given number
original_num = int(input("Enter a number: "))  # Store the original input number
a = original_num  # Corrected variable assignment
```

```
sum_of_digits = 0

while original_num > 0:
    digit = original_num % 10
    sum_of_digits += digit
    original_num //= 10

print(f"Sum of digits in {a} is {sum_of_digits}")
```

Output:

```
Enter a number: 44
Sum of digits in 44 is 8

=== Code Execution Successful ===
```

7. Count and print all numbers divisible by 5 or 7 between 1 to 100.

```
# Count and print numbers divisible by 5 or 7 between 1 and 100
for i in range(1, 101):
    if i % 5 == 0 or i % 7 == 0:
        print(i, end=" ")
```

OUTPUT:

```
5 7 10 14 15 20 21 25 28 30 35 40 42 45 49 50 55 56 60 63 65 70 75 77 80 84 85 90 91 95
    98 100
=== Code Execution Successful ===
```

8. Convert all lower cases to upper case in a string.

```
# Convert all lowercase letters in a string to uppercase
input_string = input("Enter a string: ")
uppercase_string = input_string.upper()
print(f"Uppercase string: {uppercase_string}")
```

OUTPUT:

```
Enter a string: tushar
Uppercase string: TUSHAR

=== Code Execution Successful ===
```

9. Print all prime numbers between 1 and 100.

```python
# Print all prime numbers between 1 and 100
for num in range(1, 101):
    if num > 1:
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                break
        else:
            print(num, end=" ")
```

OUTPUT:

```
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
=== Code Execution Successful ===
```

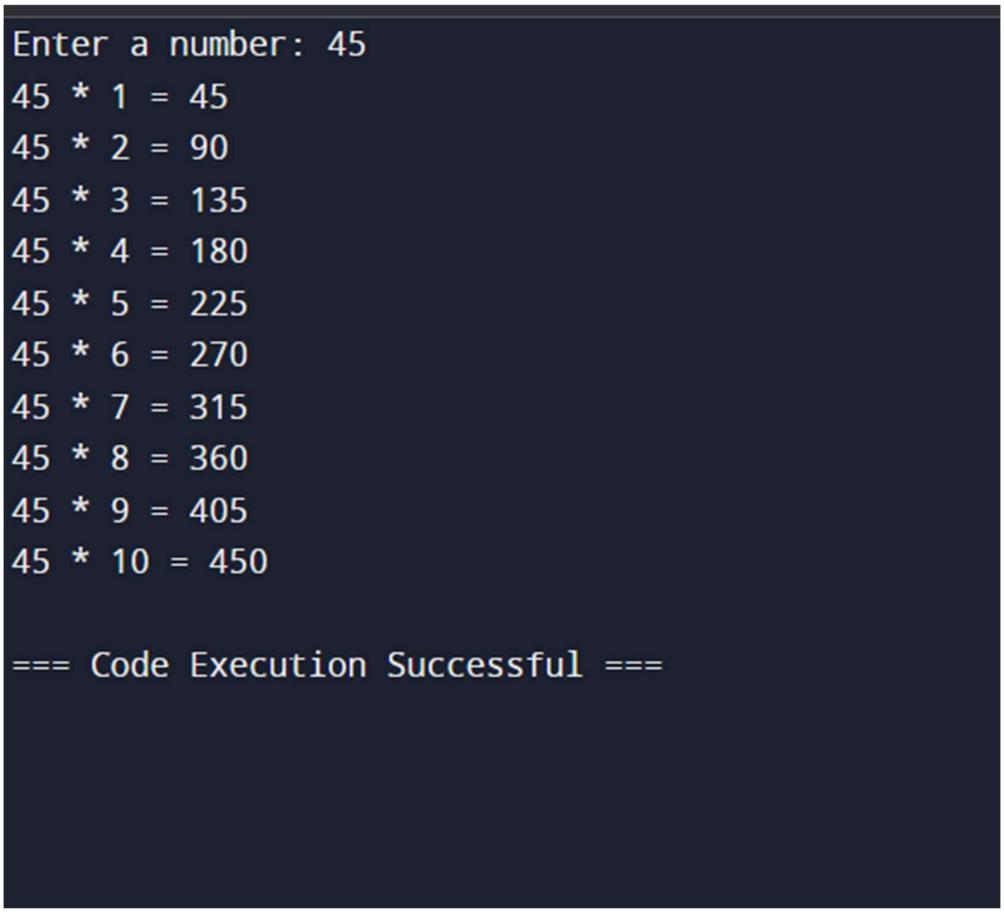
10.Print the table for a given number:

5 * 1 = 5

5 * 2 = 10……..

```python
# Print the table for a given number
given_number = int(input("Enter a number: "))
for i in range(1, 11):
    product = given_number * i
    print(f"{given_number} * {i} = {product}")
```

OUTPUT:



Experiment 5: String and Sets

1. Write a program to count and display the number of capital letters in a given string.

```
def count_capitals(s):
    return sum(1 for c in s if c.isupper())

s = input("Write a sentence of your choice ")
print("Number of capital letters:", count_capitals(s))
```

OUTPUT:

```
Write a sentence of your choice MY NAME IS TUSHAR SAINI
Number of capital letters: 19

=== Code Execution Successful ===
```

2. Count total number of vowels in a given string.

```
def count_vowels(s):
    return sum(s.count(vowel) for vowel in 'aeiouAEIOU')

s = input("Write a sentence of your choice ")
print("Number of vowels:", count_vowels(s))
```

OUTPUT:

```
Write a sentence of your choice HELLO world
Number of vowels: 3

=== Code Execution Successful ===
```

3. Input a sentence and print words in separate lines.

```
s = input("Write a sentence of your choice ")
for word in s.split():
    print(word)
```

OUTPUT:

```
Write a sentence of your choice HELLO WORLD
HELLO
WORLD

=== Code Execution Successful ===
```

4. WAP to enter a string and a substring. You have to print the number of times

that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

Sample Input

ABCDCDC

CDC

Sample Output

2

```
def count_substring_occurrences(main_string, sub_string):
    # Initialize the counter
    count = 0

    # Get the length of the main string and the substring
    main_length = len(main_string)
    sub_length = len(sub_string)

    # Traverse the main string
    for i in range(main_length - sub_length + 1):  # +1 because we want to check up to the last possible position
        # Check if the substring starts at the current position
        if main_string[i:i + sub_length] == sub_string:
            count += 1  # Increment the counter

    return count


# Corrected input handling
main_string = input("Enter a string: \n")
sub_string = input("Enter the substring: ")

# Call the function and print the result
print(count_substring_occurrences(main_string, sub_string))
```

OUTPUT:

```
Enter a string:
HELLO WORLD THIS IS HP
Enter the substring: HP
1

=== Code Execution Successful ===
```

5. Given a string containing both upper and lower case alphabets. Write a Python

program to count the number of occurrences of each alphabet (case

insensitive) and display the same.

Sample Input

ABaBCbGc

Sample Output

2A

3B

2C

1G

```
def count_alphabets(string):
    # Convert the string to lowercase to ignore case
    string = string.lower()

    # Initialize an empty dictionary to store the counts
    alphabet_counts = {}

    # Iterate through each character in the string
    for char in string:
        # If the character is an alphabet (a-z), update its count in the dictionary
        if char.isalpha():
            if char in alphabet_counts:
```

```
            alphabet_counts[char] += 1
        else:
            alphabet_counts[char] = 1

    # Return the dictionary of alphabet counts
    return alphabet_counts


# Take input from the user
user_input = input("Enter a string: ")

# Call the function and print the result
alphabet_counts = count_alphabets(user_input)

# Print each alphabet and its count
for alphabet, count in alphabet_counts.items():
    print(f"{count}{alphabet}")
```

OUTPUT:

```
Enter a string: HELLO WORLD
1h
1e
3l
2o
1w
1r
1d

=== Code Execution Successful ===
```

6. Program to count number of unique words in a given sentence using sets.

```
def count_unique_words(sentence):
    # Split the sentence into words
```

```
    words = sentence.split()

    # Convert the list of words into a set to remove duplicates
    unique_words_set = set(words)

    # Return the number of unique words
    return len(unique_words_set)


# Take input from the user
sentence = input("Enter a sentence: ")

# Call the function and print the result
num_unique_words = count_unique_words(sentence)

# Print the number of unique words
print(f"The sentence contains {num_unique_words} unique words.")
```

OUTPUT:

```
Enter a sentence: HELLO HP
The sentence contains 2 unique words.

=== Code Execution Successful ===
```

7. Create 2 sets s1 and s2 of n fruits each by taking input from user and find:

a) Fruits which are in both sets s1 and s2

b) Fruits only in s1 but not in s2

c) Count of all fruits from s1 and s2

```
def process_fruits(s1, s2):
    # a) Fruits which are in both sets s1 and s2
    common_fruits = s1.intersection(s2)
    print("Fruits in both sets:", common_fruits)

    # b) Fruits only in s1 but not in s2
    exclusive_s1_fruits = s1.difference(s2)
    print("Fruits only in s1:", exclusive_s1_fruits)
```

```
    # c) Count of all fruits from s1 and s2
    total_count = len(s1) + len(s2)
    print("Total count of fruits:", total_count)

# Take input from the user for s1
try:
    s1_input = input("Enter fruits for set s1, separated by space:
").strip().upper().split()
    s1 = set(s1_input)
except ValueError:
    print("Error: Invalid input for set s1. Please enter fruits separated by spaces.")

# Take input from the user for s2
try:
    s2_input = input("Enter fruits for set s2, separated by space:
").strip().upper().split()
    s2 = set(s2_input)
except ValueError:
    print("Error: Invalid input for set s2. Please enter fruits separated by spaces.")

# Process the fruits
process_fruits(s1, s2)
```

OUTPUT:

```
Enter fruits for set s1, separated by space: APPLE BANANA
Enter fruits for set s2, separated by space: ORANGE BANANA
Fruits in both sets: {'BANANA'}
Fruits only in s1: {'APPLE'}
Total count of fruits: 4

=== Code Execution Successful ===
```

8. Take two sets and apply various set operations on them :

S1 = {Red ,yellow, orange , blue }

S2 = {violet, blue , purple}

```
def set_operations(S1, S2):
    # Union of S1 and S2
    union = S1.union(S2)
```

```
    print("Union of S1 and S2:", union)

    # Intersection of S1 and S2
    intersection = S1.intersection(S2)
    print("Intersection of S1 and S2:", intersection)

    # Difference of S1 and S2
    diff_S1_S2 = S1.difference(S2)
    print("Difference of S1 and S2:", diff_S1_S2)

    # Difference of S2 and S1
    diff_S2_S1 = S2.difference(S1)
    print("Difference of S2 and S1:", diff_S2_S1)

    # Symmetric difference of S1 and S2
    symmetric_diff = S1.symmetric_difference(S2)
    print("Symmetric difference of S1 and S2:", symmetric_diff)

    # Count of elements in S1 and S2
    print("Count of elements in S1:", len(S1))
    print("Count of elements in S2:", len(S2))

    # Size of the union of S1 and S2
    union_size = len(union)
    print("Size of the union of S1 and S2:", union_size)


# Take input from the user for S1
S1 = set(input("Enter elements for set S1, separated by space: ").split())

# Take input from the user for S2
S2 = set(input("Enter elements for set S2, separated by space: ").split())

# Perform set operations
set_operations(S1, S2)
```

OUTPUT:

```
Enter elements for set S1, separated by space: APPLE BANANA
Enter elements for set S2, separated by space: ORANGE BANANA
Union of S1 and S2: {'APPLE', 'ORANGE', 'BANANA'}
Intersection of S1 and S2: {'BANANA'}
Difference of S1 and S2: {'APPLE'}
Difference of S2 and S1: {'ORANGE'}
Symmetric difference of S1 and S2: {'APPLE', 'ORANGE'}
Count of elements in S1: 2
Count of elements in S2: 2
Size of the union of S1 and S2: 3

=== Code Execution Successful ===
```

Experiment 6: Lists, tuples, dictionary

1. Scan n values in range 0-3 and print the number of times each value has occurred.

```python
n = int(input("Enter the number of values: "))
values = []
for _ in range(n):
    value = int(input("Enter a value between 0 and 3: "))
    if 0 <= value <= 3:
        values.append(value)
    else:
        print("Error: Value must be between 0 and 3. Please try again.")

# Counting occurrences of each number from 0 to 3
counts = [values.count(i) for i in range(4)]
for i in range(4):
    print(f"{i} has occurred {counts[i]} times")
```

OUTPUT:

```
Enter the number of values: 5
Enter a value between 0 and 3: 0
Enter a value between 0 and 3: 1
Enter a value between 0 and 3: 2
Enter a value between 0 and 3: 4
ERROR!
Error: Value must be between 0 and 3. Please try again.
Enter a value between 0 and 3: 3
0 has occurred 1 times
1 has occurred 1 times
2 has occurred 1 times
3 has occurred 1 times

=== Code Execution Successful ===
```

2. Create a tuple to store n numeric values and find average of all values.

```
n = int(input("Enter the number of numeric values: "))
if n == 0:
    print("Error: Cannot calculate average with zero values.")
else:
    values = tuple(float(input("Enter a numeric value: ")) for _ in range(n))
    average = sum(values) / n
    print(f"Average is {average:.2f}")  # Use formatted string to display the
average with two decimal places
```

OUTPUT:

```
Enter the number of numeric values: 5
Enter a numeric value: 1
Enter a numeric value: 2
Enter a numeric value: 3
Enter a numeric value: 4
Enter a numeric value: 5
Average is 3.00

=== Code Execution Successful ===
```

3. WAP to input a list of scores for N students in a list data type. Find the score of

the runner-up and print the output.

Sample Input

N = 5

Scores= 2 3 6 6 5

Sample output

5

Note: Given list is [2, 3, 6, 6, 5]. The maximum score is 6, second maximum is

5. Hence, we print 5 as the runner-up score.

```
n = int(input("Enter the number of students: "))
scores = [int(score) for score in input("Enter the scores: ").split()]
scores.sort()
runner_up_score = scores[-2]
print(f"Runner-up score is {runner_up_score}")
```

OUTPUT:

```
Enter the number of students: 5
Enter the scores: 12 13 14 11 0
Runner-up score is 13

=== Code Execution Successful ===
```

4. Create a dictionary of n persons where key is name and value is city.

a) Display all names

b) Display all city names

c) Display student name and city of all students.

d) Count number of students in each city.

```
n = int(input("Enter the number of persons: "))
persons = {}
for _ in range(n):
    name = input("Enter name: ")
    city = input("Enter city: ")
    persons[name] = city

print("Names: ", list(persons.keys()))
print("Cities: ", list(persons.values()))
print("Persons: ", persons)

# Correctly count the number of persons in each city
city_counts = {city: sum(1 for person in persons.values() if person == city) for
city in set(persons.values())}
print("Number of persons in each city: ", city_counts)
```

OUTPUT:

```
Enter the number of persons: 3
Enter name: ANDY
Enter city: DDN
Enter name: SAINI
Enter city: RTK
Enter name: PRANKUR
Enter city: ALWAR
Names:  ['ANDY', 'SAINI', 'PRANKUR']
Cities:  ['DDN', 'RTK', 'ALWAR']
Persons:  {'ANDY': 'DDN', 'SAINI': 'RTK', 'PRANKUR': 'ALWAR'}
Number of persons in each city:  {'RTK': 1, 'DDN': 1, 'ALWAR': 1}

=== Code Execution Successful ===
```

5. Store details of n movies in a dictionary by taking input from the user. Each movie must store details like name, year, director name, production cost, collection made (earning) & perform the following :-

a) print all movie details

b) display name of movies released before 2015

c) print movies that made a profit.

d) print movies directed by a particular director.

```
n = int(input("Enter the number of movies: "))
movies = {}
for _ in range(n):
    name = input("Enter movie name: ")
    year = int(input("Enter release year: "))
    director = input("Enter director name: ")
    cost = float(input("Enter production cost: "))
    earning = float(input("Enter collection made: "))
    movies[name] = {"year": year, "director": director, "cost": cost, "earning":
earning}

print("All movie details: ", movies)
print("Movies released before 2015: ", [name for name, details in
movies.items() if details['year'] < 2015])
print("Movies that made a profit: ", [name for name, details in movies.items() if
```

```
details['earning'] > details['cost']])
director_name = input("Enter director name: ")
print("Movies directed by ", director_name, ": ", [name for name, details in
movies.items() if details['director'] == director_name])
```

OUTPUT:

```
Enter the number of movies: 3
Enter movie name: R ONE
Enter release year: 2000
Enter director name: ABC
Enter production cost: 232323
Enter collection made: 444444
Enter movie name: DDLJ
Enter release year: 2005
Enter director name: XYZ
Enter production cost: 200
Enter collection made: 800
Enter movie name: NAAGIN
Enter release year: 1995
Enter director name: QWER
Enter production cost: 50
Enter collection made: 150
All movie details:  {'R ONE': {'year': 2000, 'director': 'ABC', 'cost': 232323.0,
    'earning': 444444.0}, 'DDLJ': {'year': 2005, 'director': 'XYZ', 'cost': 200.0,
    'earning': 800.0}, 'NAAGIN': {'year': 1995, 'director': 'QWER', 'cost': 50.0,
    'earning': 150.0}}
Movies released before 2015:  ['R ONE', 'DDLJ', 'NAAGIN']
Movies that made a profit:  ['R ONE', 'DDLJ', 'NAAGIN']
Enter director name: ABC
Movies directed by  ABC :  ['R ONE']

=== Code Execution Successful ===
```

Experiment 7: Functions

1. Write a Python function to find the maximum and minimum numbers from a sequence of numbers. (Note: Do not use built-in functions.)

```
def find_max_min(numbers):
    max_num = numbers[0]
    min_num = numbers[0]
```

```
    for num in numbers:
        if num > max_num:
            max_num = num
        elif num < min_num:
            min_num = num

    return max_num, min_num


numbers = list(map(int, input("Enter numbers separated by space: ").split()))
max_num, min_num = find_max_min(numbers)
print(f"Maximum Number: {max_num}, Minimum Number: {min_num}")
```

OUTPUT:

```
Enter numbers separated by space: 1 2 3 4
Maximum Number: 4, Minimum Number: 1

=== Code Execution Successful ===
```

2. Write a Python function that takes a positive integer and returns the sum of the

cube of all the positive integers smaller than the specified number.

```
def sum_of_cubes(n):
    if n <= 0:
        return 0
    else:
        return n**3 + sum_of_cubes(n-1)

n = int(input("Enter a positive integer: "))
result = sum_of_cubes(n)
print(f"Sum of cubes: {result}")
```

OUTPUT:

```
Enter a positive integer: 3
Sum of cubes: 36

=== Code Execution Successful ===
```

3. Write a Python function to print 1 to n using recursion. (Note: Do not use loop)

```python
def print_numbers(n):
    if n > 0:
        print(n)
        print_numbers(n-1)

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

OUTPUT:

```
Enter a number: 5
5
4
3
2
1

=== Code Execution Successful ===
```

4. Write a recursive function to print Fibonacci series upto n terms.

```python
def fibonacci(n):
    if n <= 0:
```

```
    return []
  elif n == 1:
    return [0]
  elif n == 2:
    return [0, 1]
  else:
    fib_series = fibonacci(n-1)
    fib_series.append(fib_series[-1] + fib_series[-2])
    return fib_series

n = int(input("Enter the number of terms: "))
fibonacci_series = fibonacci(n)
print(f"Fibonacci Series: {fibonacci_series}")
```

OUTPUT:

```
Enter the number of terms: 3
Fibonacci Series: [0, 1, 1]

=== Code Execution Successful ===
```

5. Write a lambda function to find volume of cone.

```
volume_cone = lambda r, h: (1/3) * 3.14159 * r**2 * h
r = float(input("Enter radius: "))
h = float(input("Enter height: "))
print(f"Volume of Cone: {volume_cone(r, h)}")
```

OUTPUT:

```
Enter radius: 5
Enter height: 4
Volume of Cone: 104.71966666666665

=== Code Execution Successful ===
```

6. Write a lambda function which gives tuple of max and min from a list.

Sample input: [10, 6, 8, 90, 12, 56]

Sample output: (90,6)

```
max_min_tuple = lambda lst: (max(lst), min(lst))
lst = list(map(int, input("Enter numbers separated by space: ").split()))
print(max_min_tuple(lst))
```

OUTPUT:

```
Enter numbers separated by space: 1 2 3 4
(4, 1)

=== Code Execution Successful ===
```

7. Write functions to explain mentioned concepts:

a. Keyword argument

b. Default argument

c. Variable length argument

```
def greet(name="World"):
    print(f"Hello, {name}!")

greet(name="Alice")  # Positional argument
greet(name="Bob")  # Keyword argument
```

```python
#B)
def greet(name="World"):
    print(f"Hello, {name}!")

greet()  # Uses default argument
greet("Alice")  # Overrides default argument

#C)
def sum_numbers(*args):
    return sum(args)

numbers = list(map(int, input("Enter numbers separated by space: ").split()))
total = sum_numbers(*numbers)
print(f"Sum: {total}")
```

OUTPUT:

```
Hello, Alice!
Hello, Bob!
Hello, World!
Hello, Alice!
Enter numbers separated by space: 1 2 3 4
Sum: 10

=== Code Execution Successful ===
```

Experiment 8: File Handling and Exception Handling

1. Add few names, one name in each row, in “name.txt file”.

a. Count no of names

b. Count all names starting with vowel

c. Find longest name

```python
#a)
with open('name.txt', 'w') as f:
    f.write('Alice\n')
    f.write('Bob\n')
    f.write('Charlie\n')

with open('name.txt', 'r') as f:
    lines = f.readlines()
    print(f"Number of names: {len(lines)}")
#B)
with open('name.txt', 'r') as f:
    lines = f.readlines()
    vowel_names = [line.strip() for line in lines if line.strip()[0].lower() in 'aeiou']
    print(f"Number of names starting with a vowel: {len(vowel_names)}")
#c)
with open('name.txt', 'r') as f:
    lines = f.readlines()
    longest_name = max(lines, key=len).strip()
    print(f"Longest name: {longest_name}")
```

OUTPUT:

```
C:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Users\hp\Desktop\Python\Experiments\Experiment_3\NEW.py
Number of names: 3
Number of names starting with a vowel: 1
Longest name: Charlie

Process finished with exit code 0
```

2. Store integers in a file.

a. Find the max number

b. Find average of all numbers

c. Count number of numbers greater than 100

```python
#a)
with open('integers.txt', 'w') as f:
    f.write('100\n')
    f.write('200\n')
    f.write('300\n')

with open('integers.txt', 'r') as f:
    numbers = [int(line.strip()) for line in f]
    max_number = max(numbers)
```

```
    print(f"Max number: {max_number}")

#b)
with open('integers.txt', 'r') as f:
    numbers = [int(line.strip()) for line in f]
    avg = sum(numbers) / len(numbers)
    print(f"Average of all numbers: {avg}")

#c)
with open('integers.txt', 'r') as f:
    numbers = [int(line.strip()) for line in f]
    count = sum(1 for num in numbers if num > 100)
    print(f"Number of numbers greater than 100: {count}")
```

OUTPUT:

```
C:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Users\hp\Desktop\Python\Experiments\Experiment_3\NEW.py
Max number: 300
Average of all numbers: 200.0
Number of numbers greater than 100: 2

Process finished with exit code 0
```

3. Assume a file city.txt with details of 5 cities in given format (cityname population(in lakhs) area(in sq KM)):

Example:

Dehradun 5.78 308.20

Delhi 190 1484

……

Open file city.txt and read to:

a. Display details of all cities

b. Display city names with population more than 10Lakhs

c. Display sum of areas of all cities

```
import os


# Function to create city.txt file with sample data
def create_city_file():
    if not os.path.exists('city.txt'):
```

```
        with open('city.txt', 'w') as file:
            file.write("Dehradun 5.78 308.20\n")
            file.write("Delhi 190 1484\n")
            file.write("Mumbai 12.52 603\n")
            file.write("Kolkata 14.17 206\n")
            file.write("Chennai 8.87 426\n")


# Function to perform operations on city.txt with error handling
def process_city_file():
    total_area = 0
    with open('city.txt', 'r') as file:
        for line in file:
            try:
                city_name, population, area = line.strip().split()
                population_float = float(population)

                print(f"City Name: {city_name}, Population: {population_float}
Lakhs, Area: {area} sq KM")

                if population_float > 10:
                    print(f"{city_name} has a population more than 10 Lakhs.")

                total_area += float(area)
            except ValueError as e:
                print(f"Error processing line: {line.strip()} - {e}")

    print("\nTotal Area of All Cities:", total_area)


# Main execution
if __name__ == "__main__":
    # Create city.txt file if it doesn't exist
    create_city_file()

    # Process city.txt file
    process_city_file()
```

4. Input two values from user where the first line contains N, the number of test cases. The next N lines contain the space separated values of a and b. Perform

integer division and print a/b. Handle exception in case of ZeroDivisionError or ValueError.

Sample input

1 0

2 $

3 1

Sample Output :

Error Code: integer division or modulo by zero

Error Code: invalid literal for int() with base 10: '$' 3

```
try:
    N = int(input("Enter the number of test cases: "))
    for _ in range(N):
        a, b = map(str, input("Enter a and b separated by space: ").split())
        try:
            result = int(a) // int(b)
            print(result)
        except ZeroDivisionError:
            print("Error Code: integer division or modulo by zero")
        except ValueError:
            print("Error Code: invalid literal for int() with base 10: '" + b + "'")
except ValueError:
    print("Error Code: invalid literal for int() with base 10: '" + input() + "'")
```

output:

```
Enter the number of test cases: 5
Enter a and b separated by space: 2 4
0
Enter a and b separated by space: 4 2
2
Enter a and b separated by space: 3 1
3
Enter a and b separated by space: 3 0
Error Code: integer division or modulo by zero
Enter a and b separated by space: 0 9
0

=== Code Execution Successful ===
```

5. Create multiple suitable exceptions for a file handling program.

```
try:
    with open('non_existent_file.txt', 'r') as f:
        pass  # Attempt to read a non-existent file
except FileNotFoundError:
    print("FileNotFoundError: The file does not exist.")
except PermissionError:
    print("PermissionError: You do not have permission to access the file.")
except IsADirectoryError:
    print("IsADirectoryError: The path is a directory, not a file.")
except NotADirectoryError:
    print("NotADirectoryError: The path is not a directory.")
except UnicodeDecodeError:
    print("UnicodeDecodeError: The file contains non-UTF-8 characters.")
```

output:

```
ERROR!
FileNotFoundError: The file does not exist.

=== Code Execution Successful ===
```

Experiment 9: Classes and objects

1.Create a class of student (name, sap id, marks[phy,chem,maths]). Create 3 objects by taking inputs from the user and display details of all students.

```python
class Student:
    def __init__(self, name, sap_id, marks):
        self.name = name
        self.sap_id = sap_id
        self.marks = marks  # Assuming marks is a dictionary with subjects as keys

    def display(self):
        print(f"Name: {self.name}, SAP ID: {self.sap_id}")
        print("Marks: ", self.marks)

# Taking inputs from the user
n = int(input("Enter the number of students: "))
students = []

for i in range(n):
    name = input("Enter student name: ")
    sap_id = input("Enter SAP ID: ")
    marks = {'phy': float(input("Enter physics mark: ")),
             'chem': float(input("Enter chemistry mark: ")),
             'maths': float(input("Enter maths mark: "))}
    students.append(Student(name, sap_id, marks))

# Displaying details of all students
for student in students:
    student.display()
```

output:

```
Enter the number of students: 3
Enter student name: tushar
Enter SAP ID: 500122590
Enter physics mark: 90
Enter chemistry mark: 90
Enter maths mark: 90
Enter student name: prankur
Enter SAP ID: 500124514
Enter physics mark: 88
Enter chemistry mark: 88
Enter maths mark: 88
Enter student name: aman
Enter SAP ID: 500122000
Enter physics mark: 777
Enter chemistry mark: 7
Enter maths mark: 77
Name: tushar, SAP ID: 500122590
Marks:  {'phy': 90.0, 'chem': 90.0, 'maths': 90.0}
Name: prankur, SAP ID: 500124514
Marks:  {'phy': 88.0, 'chem': 88.0, 'maths': 88.0}
Name: aman, SAP ID: 500122000
Marks:  {'phy': 77.0, 'chem': 7.0, 'maths': 77.0}

=== Code Execution Successful ===
```

2. Add constructor in the above class to initialize student details of n students and

implement following methods:

a) Display() student details

b) Find Marks_percentage() of each student

c) Display result() [Note: if marks in each subject >40% than Pass else Fail]

Write a Function to find average of the class.

```python
def find_marks_percentage(student):
    total = sum(student.marks.values())
    percentage = (total / len(student.marks)) * 100
    return percentage

def display_result(student):
    percentage = find_marks_percentage(student)
    if percentage > 40:
        print(f"{student.name} passed with {percentage}%")
    else:
        print(f"{student.name} failed with {percentage}%")

def find_average(students):
    total = sum(find_marks_percentage(student) for student in students)
    return total / len(students)

# Displaying results and finding average
for student in students:
    display_result(student)

average = find_average(students)
print(f"Class average: {average}%")
```

output:

```
Enter the number of students: 2
Enter student name: tushar
Enter SAP ID: 123
Enter phy mark: 90
Enter chem mark: 90
Enter maths mark: 90
Enter student name: prankur
Enter SAP ID: 456
Enter phy mark: 88
Enter chem mark: 88
Enter maths mark: 88
Name: tushar, SAP ID: 123
Marks:  {'phy': 90.0, 'chem': 90.0, 'maths': 90.0}
Name: prankur, SAP ID: 456
Marks:  {'phy': 88.0, 'chem': 88.0, 'maths': 88.0}
tushar passed with 90.0%
prankur passed with 88.0%
Class average: 89.0%

=== Code Execution Successful ===
```

3. Create programs to implement different types of inheritances.

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

    def display(self):
        print(f"Employee Name: {self.name}, Salary: {self.salary}")

class Manager(Employee):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department
```

```
    def display(self):
        super().display()
        print(f"Department: {self.department}")

# Creating objects and displaying details
emp = Employee("John Doe", 50000)
mgr = Manager("Jane Smith", 60000, "HR")

emp.display()
mgr.display()
```

output:

```
C:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Users\hp\Des
Employee Name: John Doe, Salary: 50000
Employee Name: Jane Smith, Salary: 60000
Department: HR

Process finished with exit code 0
```

4. Create a class to implement method Overriding.

```
##Method overriding is demonstrated in the Manager class above, where the display method is overridden to include additional information.
```

5. Create a class for operator overloading which adds two Point Objects where Point has x & y values

e.g. if

P1(x=10,y=20)

P2(x=12,y=15)

P3=P1+P2 => P3(x=22,y=35)

```
class Point:
    def __init__(self, x, y):
        self.x = x
```

```
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

# Creating Point objects and adding them
P1 = Point(10, 20)
P2 = Point(12, 15)
P3 = P1 + P2

print(f"P3: ({P3.x}, {P3.y})")
```

output:

```
P3: (22, 35)

=== Code Execution Successful ===
```

Experiment 10: Data Analysis and Visualization

1. Create numpy array to find sum of all elements in an array.

```
import numpy as np

# Taking input from the user
array_elements = list(map(int, input("Enter elements of the array separated by space: ").split()))

# Creating a numpy array and finding the sum
array = np.array(array_elements)
sum_of_elements = np.sum(array)
print(f"Sum of all elements: {sum_of_elements}")
```

output:

```
Enter elements of the array separated by space: 1 2 3 4 5
Sum of all elements: 15

=== Code Execution Successful ===
```

2. Create numpy array of (3,3) dimension. Now find sum of all rows & columns individually. Also find 2nd maximum element in the array.

```
import numpy as np


# Creating a 3x3 numpy array
array_3x3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Finding sum of all rows and columns
row_sums = np.sum(array_3x3, axis=1)
col_sums = np.sum(array_3x3, axis=0)

# Finding 2nd maximum element
sorted_array = np.sort(array_3x3.flatten())
second_max = sorted_array[-2]

print(f"Row sums: {row_sums}")
print(f"Column sums: {col_sums}")
print(f"Second maximum element: {second_max}")
```

Output:

```
Row sums: [ 6 15 24]
Column sums: [12 15 18]
Second maximum element: 8

=== Code Execution Successful ===
```

3. Perform Matrix multiplication of any 2 n*n matrices.

```
import numpy as np

# Function to take input for a single row of a matrix
def input_row():
```

```
    row_elements = input("Enter elements of a row separated by space:
").strip().split()
    if len(row_elements)!= 3:
        raise ValueError("Each row must contain exactly 3 elements.")
    return list(map(int, row_elements))

# Taking input from the user for two 3x3 matrices
matrix1 = np.array([input_row() for _ in range(3)])
matrix2 = np.array([input_row() for _ in range(3)])

# Check if the matrices are 3x3
if matrix1.shape!= (3, 3) or matrix2.shape!= (3, 3):
    raise ValueError("Both matrices must be 3x3.")

# Performing matrix multiplication
product = np.dot(matrix1, matrix2)

print(f"Matrix product:\n{product}")
```

Output:

```
Enter elements of a row separated by space: 1 2 3
Enter elements of a row separated by space: 5 4 3
Enter elements of a row separated by space: 6 5 4
Enter elements of a row separated by space: 8 7 6
Enter elements of a row separated by space: 0 9 8
Enter elements of a row separated by space: 0 0 0
Matrix product:
[[ 8 25 22]
 [40 71 62]
 [48 87 76]]

=== Code Execution Successful ===
```

4. Write a Pandas program to get the powers of an array values element-wise.

Note: First array elements raised to powers from second array

Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]}

Expected Output:

X Y Z

0 78 84 86

1 85 94 97

2 96 89 96

3 80 83 72

4 86 86 83

```
import pandas as pd

# Sample data
data = {'X': [78, 85, 96, 80, 86],
        'Y': [84, 94, 89, 83, 86],
        'Z': [86, 97, 96, 72, 83]}

df = pd.DataFrame(data)

# Getting powers of array values element-wise
df['X'] = df['X'].apply(lambda x: x**df['Y'])
df['Y'] = df['Y'].apply(lambda x: x**df['Z'])

print(df)
```

5. Write a Pandas program to get the first 3 rows of a given DataFrame.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',

'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

First three rows of the data frame:

attempts name qualify score

a 1 Anastasia yes 12.5

b 3 Dima no 9.0

c 2 Katherine yes 16.5

```
# Sample Python dictionary data and list labels
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

# Creating a DataFrame
df = pd.DataFrame(exam_data, index=labels)

# Getting the first 3 rows
first_three_rows = df.head(3)

print("\nFirst three rows of the data frame:")
print(first_three_rows)
```

Output:

```
First three rows of the data frame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes

=== Code Execution Successful ===
```

6. Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

```
import pandas as pd
import numpy as np

# Step 2: Load your data
# For demonstration, let's create a sample DataFrame
data = {
    'A': [1, 2, np.nan, 4, 5],
    'B': [np.nan, 2, 3, 4, 5],
    'C': [1, 2, 3, np.nan, 5]
}
df = pd.DataFrame(data)

# Step 3: Identify missing values
print("Missing values before replacement:")
print(df.isnull())

# Step 4: Replace missing values
# Option 1: Replace with a constant value (e.g., 0)
df_filled_constant = df.fillna(0)
print("\nMissing values after replacing with 0:")
print(df_filled_constant.isnull())

# Option 2: Replace with the mean of the column
df_filled_mean = df.fillna(df.mean())
print("\nMissing values after replacing with the mean:")
print(df_filled_mean.isnull())

# Option 3: Replace with the median of the column
df_filled_median = df.fillna(df.median())
print("\nMissing values after replacing with the median:")
print(df_filled_median.isnull())

# Option 4: Interpolate the values
df_filled_interpolate = df.interpolate()
print("\nMissing values after interpolating:")
print(df_filled_interpolate.isnull())
```

```
C:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Users\
Missing values before replacement:
       A      B      C
0  False   True  False
1  False  False  False
2   True  False  False
3  False  False   True
4  False  False  False

Missing values after replacing with 0:
       A      B      C
0  False  False  False
1  False  False  False
2  False  False  False
3  False  False  False
4  False  False  False

Missing values after replacing with the mean:
```

7. Create a program to demonstrate different visual forms using Matplotlib.

```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Line plot
plt.plot(x, y)
plt.title('Line Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()

# Bar plot
plt.bar(x, y)
plt.title('Bar Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

```
# Scatter plot
plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

output: