PYTHON LAB FILE

2024

Name: Vaibhav Jain

Roll no.: R2142231457

Sap_ID: 500123594

Program, Semester, Batch: B.TECH CSE (Batch – 43)

School of Computer Science,
University of Petroleum and Energy Studies,
Dehradun

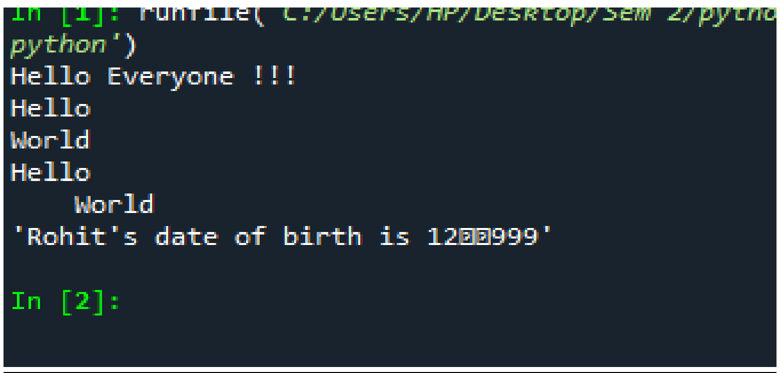
LAB 1

1. 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’

<u>Code-:</u>

```
a = "Hello Everyone !!!"

b = """Hello
World"""

c = """Hello
   World"""

d = "'Rohit's date of birth is 12\05\1999'"

print(a)

print(b)

print(c)

print(d)
```

<u>Output-:</u>



```
In [1]: runfile('C:/Users/HP/Desktop/Sem 2/pytho
python')
Hello Everyone !!!
Hello
World
Hello
    World
'Rohit's date of birth is 12999'

In [2]:
```

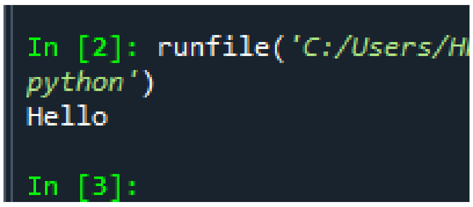
2. Declare a string variable called x and assign it the value “Hello”.
   Print out the value of x

<u>Code-:</u>

x = "Hello"

print(x)

Output-:



```
In [2]: runfile('C:/Users/H
python')
Hello

In [3]:
```

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

Code-:

x = 12

y = "Hello"

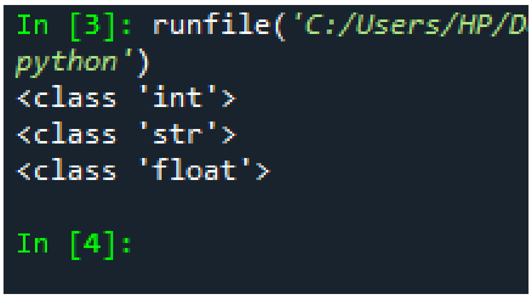
z = 12.2

print(type(x))

print(type(y))

print(type(z))

Output-:



```
In [3]: runfile('C:/Users/HP/D
python')
<class 'int'>
<class 'str'>
<class 'float'>

In [4]:
```

4. 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.

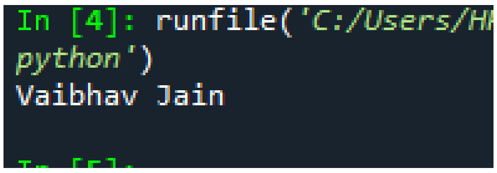
Code-:

a = "Vaibhav "

b = "Jain"

print(a+b)

Output-:



```
In [4]: runfile('C:/Users/HI
python')
Vaibhav Jain
```

5. 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.

   Code-:

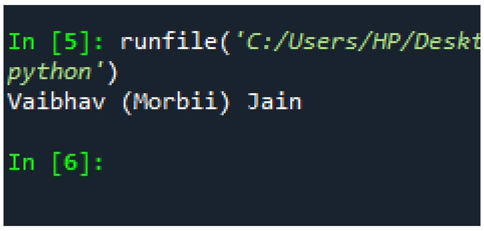
   a = "Vaibhav "

   b = "(Morbii) "

   c = "Jain"

   print(a+b+c)

   Output-:



```
In [5]: runfile('C:/Users/HP/Deskt
python')
Vaibhav (Morbii) Jain

In [6]:
```

6. 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

   Code-:

   Name = "NIKUNJ BANSAL"

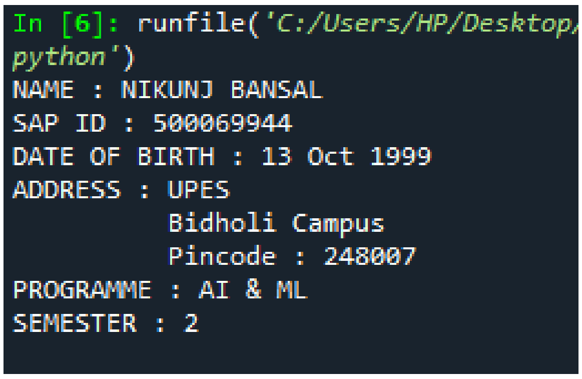
   Sap = "500069944"

   DOB = "13 Oct 1999"

   Address = """UPES
   Bidholi Campus

```
        Pincode : 248007"""

Programme = "AI & ML"

Sem = "2"

print("NAME :",Name)

print("SAP ID :",Sap)

print("DATE OF BIRTH :",DOB)

print("ADDRESS :",Address)

print("PROGRAMME :",Programme)

print("SEMESTER :",Sem)
```

Output-:



```
In [6]: runfile('C:/Users/HP/Desktop/
python')
NAME : NIKUNJ BANSAL
SAP ID : 500069944
DATE OF BIRTH : 13 Oct 1999
ADDRESS : UPES
          Bidholi Campus
          Pincode : 248007
PROGRAMME : AI & ML
SEMESTER : 2
```

LAB 2

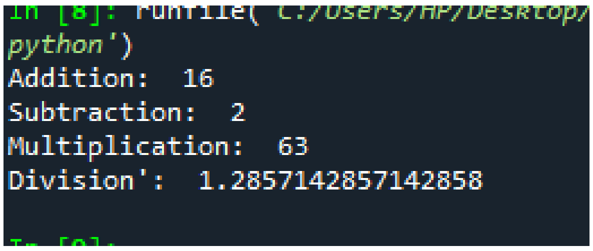
1. Declare these variables (x, y and z) as integers. Assign a value of 9 to x, Assign a value of 7 to y, perform addition, multiplication, division and subtraction on these two variables and Print out the result.

Code-:

```
#Declaring Variables
x = 9
y = 7

add = x + y
subtract = x - y
divide = x / y
multiply = x * y
```

```
#Results
print("Addition: ",add)
print("Subtraction: ",subtract)
print("Multiplication: ",multiply)
print("Division': ",divide)
```

Output-:



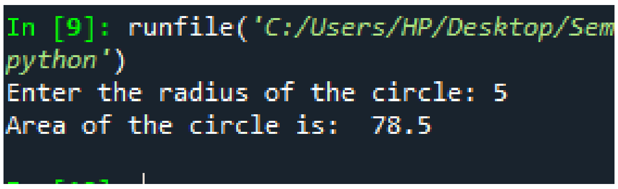
```
In [8]: runfile('C:/Users/HP/Desktop/
python')
Addition:  16
Subtraction:  2
Multiplication:  63
Division':  1.2857142857142858

In [9]:
```

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

Code-:

```
#Taking input for the radius
radius = float(input("Enter the radius of the circle: "))

#Area
area = 3.14*radius*radius

print("Area of the circle is: ", area)
```

Output-:



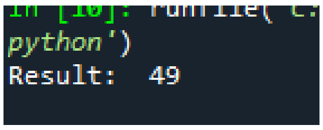
```
In [9]: runfile('C:/Users/HP/Desktop/Sem
python')
Enter the radius of the circle: 5
Area of the circle is:  78.5
```

3. Write a Python program to solve (x+y)*(x+y)
   Test data : x = 4 , y = 3

Code-:

```
x = 4
y = 3

result = (x+y)*(x+y)

print("Result: ", result)
```

Output-:



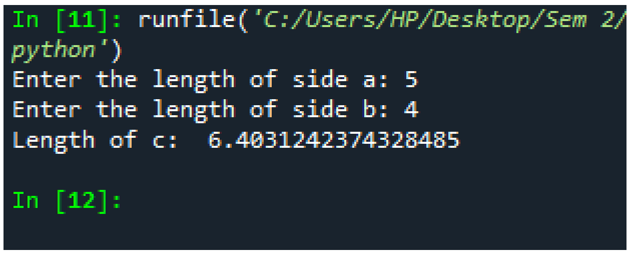
```
python')
Result:  49
```

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

<u>Code-:</u>

```
import math

a = float(input("Enter the length of side a: "))
b = float(input("Enter the length of side b: "))

c = math.sqrt(a*a + b*b)

print("Length of c: ", c)
```

<u>Output-:</u>



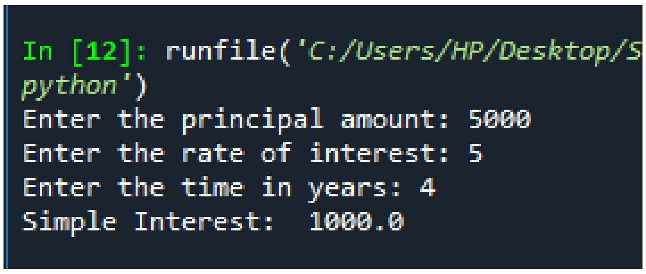
```
In [11]: runfile('C:/Users/HP/Desktop/Sem 2/
python')
Enter the length of side a: 5
Enter the length of side b: 4
Length of c:  6.4031242374328485

In [12]:
```

5. Write a program to find simple interest.

<u>Code-:</u>

```
#Taking input from the user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time in years: "))

#Calculating simple interest
SI = (principal*rate*time)/100

print("Simple Interest: ", SI)
```

<u>Output-:</u>



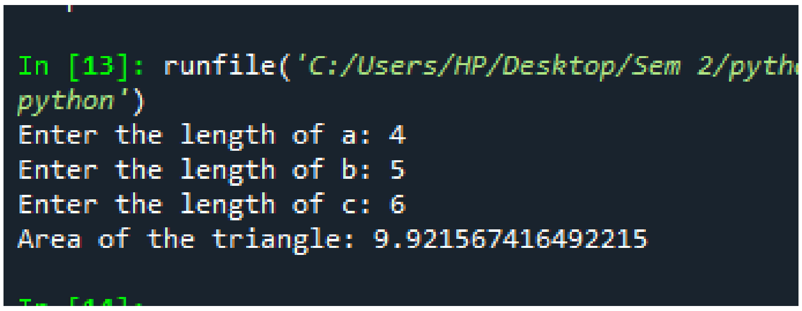
```
In [12]: runfile('C:/Users/HP/Desktop/S
python')
Enter the principal amount: 5000
Enter the rate of interest: 5
Enter the time in years: 4
Simple Interest:  1000.0
```

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

Code-:

```
import math

# Input sides of the triangle
a = float(input("Enter the length of a: "))
b = float(input("Enter the length of b: "))
c = float(input("Enter the length of c: "))

# Calculate the semi-perimeter
s = (a + b + c) / 2

# Calculate the area
area = math.sqrt(s * (s - a) * (s - b) * (s - c))

print("Area of the triangle:", area)
```

Output-:



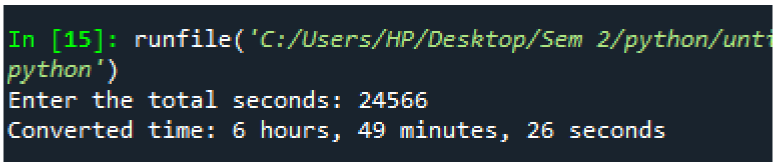
```
In [13]: runfile('C:/Users/HP/Desktop/Sem 2/pyth
python')
Enter the length of a: 4
Enter the length of b: 5
Enter the length of c: 6
Area of the triangle: 9.921567416492215
```

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

Code-:

```
# Input the seconds
sec = int(input("Enter the total seconds: "))

# Calculating hours, minutes, and remaining seconds
hours = sec // 3600
minutes = (sec % 3600) // 60
rem = sec % 60

# Print out the result
print(f"Converted time: {hours} hours, {minutes} minutes, {rem} seconds")
```

Output-:

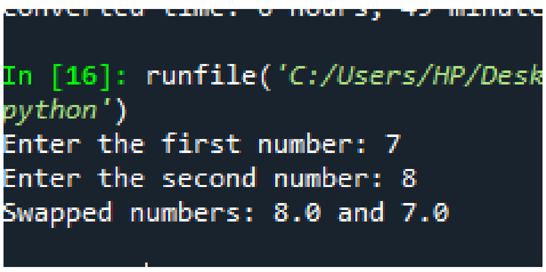

```
In [15]: runfile('C:/Users/HP/Desktop/Sem 2/python/unti
python')
Enter the total seconds: 24566
Converted time: 6 hours, 49 minutes, 26 seconds
```

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

Code-:

```
#Swap
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))

#Swapping
a = a+b
b = a-b
a = a-b

print(f"Swapped numbers: {a} and {b}")
```

Output-:

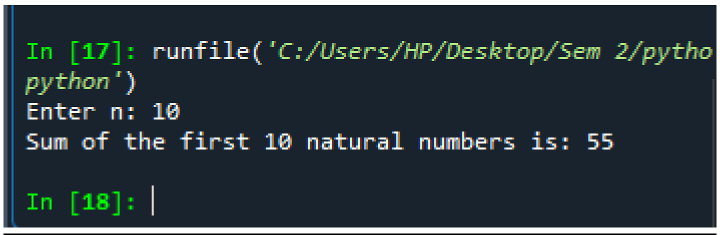

```
In [16]: runfile('C:/Users/HP/Desk
python')
Enter the first number: 7
Enter the second number: 8
Swapped numbers: 8.0 and 7.0
```

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

Code-:

```
# Input the value of n
n = int(input("Enter n: "))

sum = n * (n + 1) // 2

print(f"Sum of the first {n} natural numbers is: {sum}")
```

Output-:

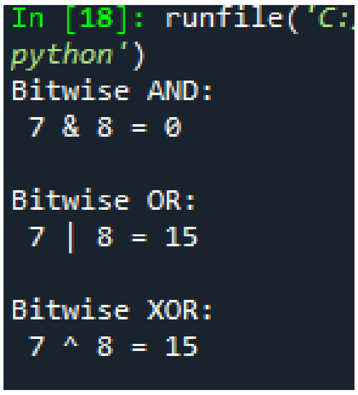

```
In [17]: runfile('C:/Users/HP/Desktop/Sem 2/pytho
python')
Enter n: 10
Sum of the first 10 natural numbers is: 55

In [18]:
```

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

Code-:

```
a = 7
b = 8

#The truth table for bitwise AND (&) operator
print("Bitwise AND:")
print(f" {a} & {b} = {a & b}")

#The truth table for bitwise OR (|) operator
print("\nBitwise OR:")
print(f" {a} | {b} = {a|b}")

#The truth table for bitwise XOR (^) operator
print("\nBitwise XOR:")
print(f" {a} ^ {b} = {a^b}")
```

Output-:



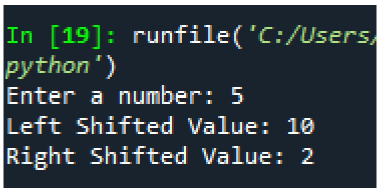
```
In [18]: runfile('C:
python')
Bitwise AND:
 7 & 8 = 0

Bitwise OR:
 7 | 8 = 15

Bitwise XOR:
 7 ^ 8 = 15
```

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

Code-:

```
num = int(input("Enter a number: "))

# Perform left shift and right shift operations
left = num << 1
right = num >> 1

print("Left Shifted Value:", left)
print("Right Shifted Value:", right)
```

Output-:

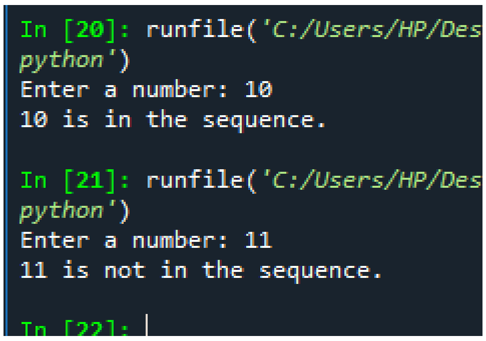

```
In [19]: runfile('C:/Users,
python')
Enter a number: 5
Left Shifted Value: 10
Right Shifted Value: 2
```

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

Code-:

```
num = int(input("Enter a number: "))

# Check if the number is in the sequence
sequence = [10, 20, 56, 78, 89]
if num in sequence:
    print(f"{num} is in the sequence.")
else:
    print(f"{num} is not in the sequence.")
```

Output-:

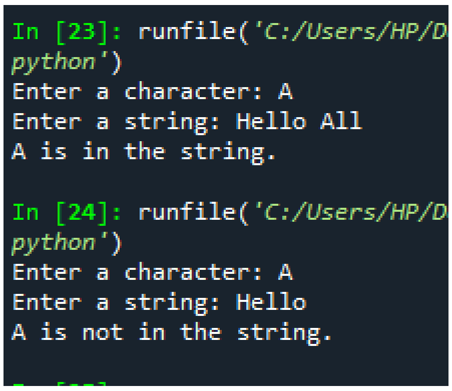

```
In [20]: runfile('C:/Users/HP/Des
python')
Enter a number: 10
10 is in the sequence.

In [21]: runfile('C:/Users/HP/Des
python')
Enter a number: 11
11 is not in the sequence.

In [22]:
```

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

Code-:

```
char = input("Enter a character: ")

# Input a string
string = input("Enter a string: ")

# Check if the character is in the string
if char in string:
    print(f"{char} is in the string.")
else:
    print(f"{char} is not in the string.")
```

Output-:



```
In [23]: runfile('C:/Users/HP/D
python')
Enter a character: A
Enter a string: Hello All
A is in the string.

In [24]: runfile('C:/Users/HP/D
python')
Enter a character: A
Enter a string: Hello
A is not in the string.
```

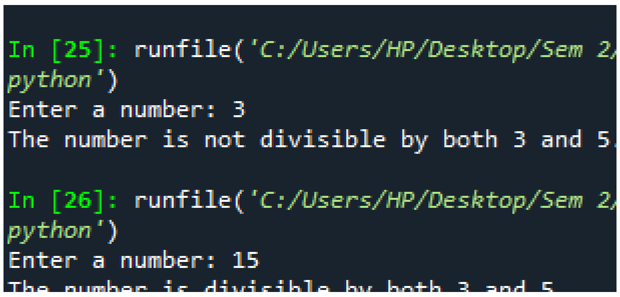
LAB 3

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

Code-:

```
num = int(input("Enter a number: "))

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

Output-:



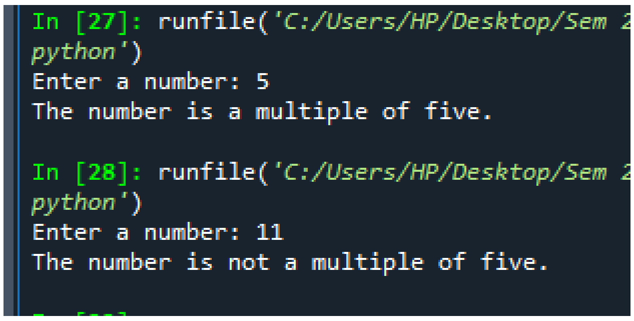
```
In [25]: runfile('C:/Users/HP/Desktop/Sem 2,
python')
Enter a number: 3
The number is not divisible by both 3 and 5.

In [26]: runfile('C:/Users/HP/Desktop/Sem 2,
python')
Enter a number: 15
The number is divisible by both 3 and 5
```

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

Code-:

```
number = int(input("Enter a number: "))
if number % 5 == 0:
   print("The number is a multiple of five.")
else:
   print("The number is not a multiple of five.")
```

Output-:



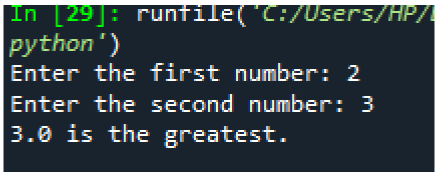
```
In [27]: runfile('C:/Users/HP/Desktop/Sem 2
python')
Enter a number: 5
The number is a multiple of five.

In [28]: runfile('C:/Users/HP/Desktop/Sem 2
python')
Enter a number: 11
The number is not a multiple of five.
```

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

Code-:

```
num 1 = float(input("Enter the first number: "))
num 2 = float(input("Enter the second number: "))

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

Output-:



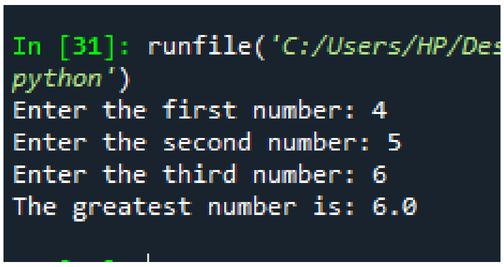
```
In [29]: runfile('C:/Users/HP/
python')
Enter the first number: 2
Enter the second number: 3
3.0 is the greatest.
```

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

Code-:

```
a= float(input("Enter the first number: "))
b= float(input("Enter the second number: "))
c = float(input("Enter the third number: "))

maxn= max(a, b, c)
print(f"The greatest number is: {maxn")
```

Output-:

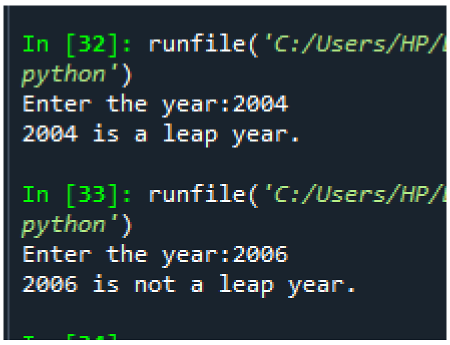

```
In [31]: runfile('C:/Users/HP/De
python')
Enter the first number: 4
Enter the second number: 5
Enter the third number: 6
The greatest number is: 6.0
```

5. Check whether the quadratic equation has real roots or imaginary roots. Display the roots.
6. Find whether a given year is a leap year or not.

Code-:

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

if year % 4 == 0:

   if year % 100 == 0:

      if year % 400 == 0:

         print(year,"is a leap year.")

      else:
         print(year,"is not a leap year.")

   else:
      print(year,"is a leap year.")

else:
   print(year,"is not a leap year.")
```

Output-:


```
In [32]: runfile('C:/Users/HP/
python')
Enter the year:2004
2004 is a leap year.

In [33]: runfile('C:/Users/HP/
python')
Enter the year:2006
2006 is not a leap year.
```

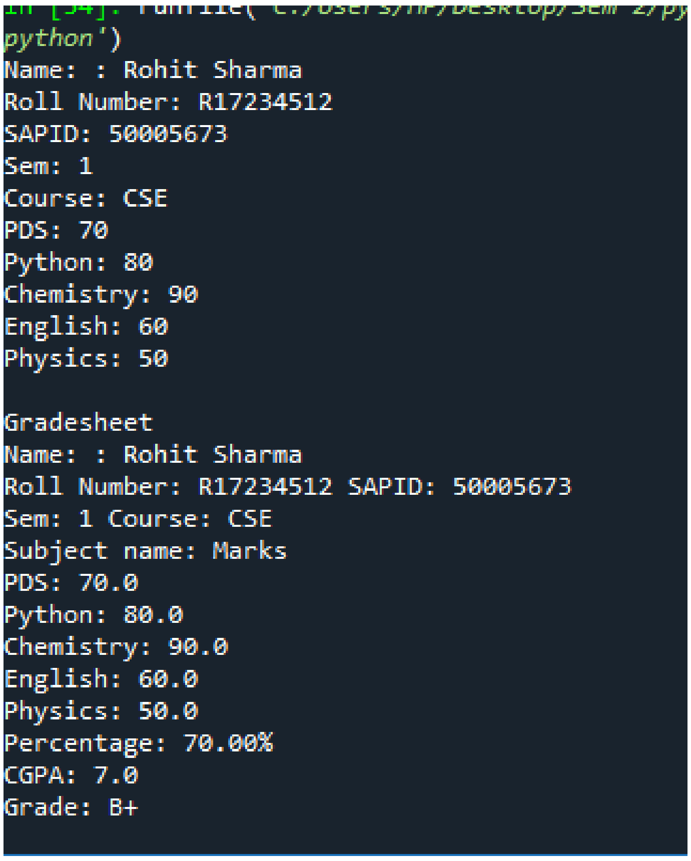
7. Print the grade sheet of a student for the given range of cgpa.
Scan marks of five subjects and calculate the percentage.
CGPA=percentage/10
CGPA range:
0 to 3.4 -> F
3.5 to 5.0->C+
5.1 to 6->B

6.1 to 7-> B+
7.1 to 8-> A
8.1 to 9->A+
9.1 to 10-> O (Outstanding)

Code-:

```
# Input student details
name = input("Name: ")
roll = input("Roll Number: ")
SAP = input("SAPID: ")
sem = input("Sem: ")
course = input("Course: ")

# Input marks for five subjects
marks = {
   "PDS": float(input("PDS: ")),
   "Python": float(input("Python: ")),
   "Chemistry": float(input("Chemistry: ")),
   "English": float(input("English: ")),
   "Physics": float(input("Physics: "))
}

#percentage
total = sum(marks.values())
percentage = (total / (len(marks) * 100)) * 100

# Calculate CGPA
cgpa = percentage / 10

if cgpa >= 9.1:
   grade = "O (Outstanding)"
elif 8.1 <= cgpa < 9.1:
   grade = "A+"
elif 7.1 <= cgpa < 8.1:
   grade = "A"
elif 6.1 <= cgpa < 7.1:
   grade = "B+"
elif 5.1 <= cgpa < 6.1:
   grade = "B"
elif 3.5 <= cgpa < 5.1:
   grade = "C+"
else:
   grade = "F"

print("\nGradesheet")
print(f"Name: {name}")
print(f"Roll Number: {roll} SAPID: {SAP}")
print(f"Sem: {sem} Course: {course}")

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

Output-:



```
In [34]: runfile('C:/Users/HP/Desktop/Sem 2/py
python')
Name: : Rohit Sharma
Roll Number: R17234512
SAPID: 50005673
Sem: 1
Course: CSE
PDS: 70
Python: 80
Chemistry: 90
English: 60
Physics: 50

Gradesheet
Name: : Rohit Sharma
Roll Number: R17234512 SAPID: 50005673
Sem: 1 Course: CSE
Subject name: Marks
PDS: 70.0
Python: 80.0
Chemistry: 90.0
English: 60.0
Physics: 50.0
Percentage: 70.00%
CGPA: 7.0
Grade: B+
```

LAB 4

1. Find a factorial of given number.

Code-:

```
 num = int(input("Enter a number:"))
factorial = 1

if num < 0:
   print("Factorial is not defined for negative numbers.")
elif num == 0 or num == 1:
   print("The factorial of", num, "is 1")
else:
   for i in range(1, num + 1):
        factorial *= i
    print("The factorial of", num, "is:", factorial)
```

2. Find whether the given number is Armstrong number.

Code-:

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

while temp > 0:
    digits = temp%10
    sum += digits**order
    temp //= 10

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

3. Print Fibonacci series up to given term.

Code-:

```
num = int(input("Enter the number of terms: "))
a, b = 0, 1

print("Fibonnaci Series:")

for _ in range(num):
    print(a, end=" ")

    a, b = b, a+b
```

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

Code-:

```
num = int(input("Enter the number to check: "))

if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            print(f"{num} is not a prime number")
            break
    else:
        print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

5. Check whether given number is palindrome or not.

Code-:

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

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

if temp == reverse:
    print(f"{temp} is a palindrome.")
else:
    print(f"{temp} is not a palindrome.")
```

6. Write a program to print sum of digits.

Code-:

```
num = int(input("Enter a number:"))
sum = 0

while num > 0:
    digit = num % 10
    sum += digit
    num = num // 10

print(f"The sum of digits is: {sum}")
```

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

Code-:

```
count = 0

print("Numbers divisible by 5 or 7 between 1 to 100:")
for num in range(1, 101):
    if num % 5 == 0 or num % 7 == 0:
        count += 1
        print(num)

    print("Total count:", count)
```

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

Code-:

```
 string = input("Enter a string:")
uppercase = string.upper()

print("Uppercase string:", uppercase)
```

9. Print all prime numbers between 1 and 100.

Code-:

```
 print("Prime numbers between 1 and 100 are:")

for num in range(1,101):

  if num > 1:
    for i in range(2, num):
      if (num % i) == 0:
        break
    else:
      print(num)
```

10. Print the table for a given number:
   5 * 1 = 5
   5 * 2 = 10………..

Code-:

```
 num = int(input("Enter a number: "))

print(f"Multiplication table for {num}:")
for i in range(1, 11):
   result = num * i
   print(f"{num} * {i} = {result}")
```

LAB 5

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

Code-:

```
def count_capital_letters(input_string):
    count = 0
    for char in input_string:
        if char.isupper():
            count += 1
    return count

def main():
    input_string = input("Enter a string: ")
    capital_count = count_capital_letters(input_string)
    print("Number of capital letters:", capital_count)
```

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

Code-:

```
def count_vowels(input_string):
    vowels = 'aeiouAEIOU'
    vowel_count = 0
    for char in input_string:
        if char in vowels:
            vowel_count += 1
    return vowel_count

def main():
    input_string = input("Enter a string: ")
    vowel_count = count_vowels(input_string)
    print("Total number of vowels:", vowel_count)
```

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

Code-:

```
def print_words_separate_lines(sentence):
    words = sentence.split()
    for word in words:
        print(word)

def main():
    input_sentence = input("Enter a sentence: ")
    print("Words in separate lines:")
```

```
print_words_separate_lines(input_sentence)
```

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

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

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

Code-:

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

  # Create a set to store unique words
  unique_words = set(words)

  # Return the count of unique words
  return len(unique_words)

def main():
  input_sentence = input("Enter a sentence: ")
  unique_word_count = count_unique_words(input_sentence)
  print("Number of unique words:", unique_word_count)
```

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

Code-:

```
def main():
    n = int(input("Enter the number of fruits for each set: "))

    # Input for set s1
    print("\nEnter fruits for set s1:")
    s1 = set()
    for i in range(n):
        fruit = input(f"Enter fruit {i+1}: ")
        s1.add(fruit.lower())  # Convert to lowercase to make it case-insensitive

    # Input for set s2
    print("\nEnter fruits for set s2:")
    s2 = set()
    for i in range(n):
        fruit = input(f"Enter fruit {i+1}: ")
        s2.add(fruit.lower())  # Convert to lowercase to make it case-insensitive

    # Fruits which are in both sets s1 and s2
    common_fruits = s1.intersection(s2)
    print("\nFruits which are in both sets s1 and s2:")
    print(common_fruits)

    # Fruits only in s1 but not in s2
    unique_to_s1 = s1.difference(s2)
    print("\nFruits only in set s1 but not in set s2:")
    print(unique_to_s1)

    # Count of all fruits from s1 and s2
    total_fruits = len(s1.union(s2))
    print("\nCount of all fruits from s1 and s2:", total_fruits)
```

8. Take two sets and apply various set operations on them :
   S1 = {Red ,yellow, orange , blue }
   S2 = {violet, blue , purple}

<u>Code-:</u>

```
def main():
    S1 = {"Red", "Yellow", "Orange", "Blue"}
    S2 = {"Violet", "Blue", "Purple"}


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


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


    # Difference (elements in S1 but not in S2)
    print("Difference of S1 and S2:", S1.difference(S2))


    # Subset
    print("Is S1 a subset of S2?", S1.issubset(S2))


    # Superset
    print("Is S1 a superset of S2?", S1.issuperset(S2))


    # Disjoint
    print("Are S1 and S2 disjoint?", S1.isdisjoint(S2))
```

LAB 6

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

<u>Code-:</u>

```
        def calculate_average(numeric_values):
```

```
    # Calculate the sum of all values in the tuple
    total_sum = sum(numeric_values)

    # Calculate the average
    average = total_sum / len(numeric_values)

    return average

def main():
    n = int(input("Enter the number of values: "))

    # Input the numeric values into a tuple
    numeric_values = tuple(float(input(f"Enter value {i+1}: ")) for i in range(n))

    # Calculate the average of all values
    average = calculate_average(numeric_values)

    print("Average of all values:", average)

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

2. 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.

Code-:

```
def find_runner_up_score(scores):

        # Sort the list of scores in descending order
        sorted_scores = sorted(scores, reverse=True)

        # Find the second highest score
        runner_up_score = None
        for score in sorted_scores:
            if score < sorted_scores[0]:
                runner_up_score = score
                break
```

```
    return runner_up_score

def main():
    N = int(input("Enter the number of students: "))
    scores = list(map(int, input("Enter the scores separated by space: ").split()))

    if len(scores) != N:
        print("Number of scores entered does not match the specified number of students.")
        return

    runner_up_score = find_runner_up_score(scores)
    if runner_up_score is not None:
        print("Runner-up score:", runner_up_score)
    else:
        print("There is no runner-up score.")

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

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

Code-:

```
def count_occurrences(n):

    occurrences = {0: 0, 1: 0, 2: 0, 3: 0}  # Initialize dictionary to count occurrences

    # Input values and count occurrences
    for _ in range(n):
        value = int(input("Enter a value between 0 and 3: "))
        if value in occurrences:
            occurrences[value] += 1
        else:
            print("Invalid value entered. Please enter a value between 0 and 3.")

    return occurrences

def main():
    n = int(input("Enter the number of values: "))
    if n <= 0:
        print("Number of values must be greater than zero.")
        return

    occurrences = count_occurrences(n)

    # Print the number of times each value has occurred
    print("\nOccurrences of each value:")
    for value, count in occurrences.items():
        print(f"{value}: {count} times")

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

LAB 7

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

Code-:

```
#Find maximum and minimum numbers from the sequence of number
def max_min(data):
   l_num = data[0]
   s_num = data[0]

   for num in data:
      if num > l_num:
         l_num = num
      elif num < s_num:
         s_num = num
   return l_num, s_num

print(max_min([0, 10, 15, 40, -5, 42, 17, 28, 75]))
```

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.

Code-:

```
#sum of the cube of the all positive integers smaller by the specified number
def sum_of_cubes(n):
   n = n-1
   total = 0

   while n > 0:
      total = total + (n * n * n)
      n = n- 1
   return total

print("Sum of cubes:", sum_of_cubes(4))
```

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

Code-:

```
# To print 1 to n using recursion
def printNums(n):
   if n > 0:
        printNums(n - 1)
        print(n, end=' ')

n = 50
printNums(n)
```

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

Code-:

# To print Fibonacci series upto n terms.

```
def fibonacci(n):
  if n <= 1:
     return n
  else:
     return(fibonacci(n-1) + fibonacci(n-2))

n = 10

print("Fibonacci sequence:")
for i in range(n):
   print(fibonacci(i))
```

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

Code-:

```
# To find volume of cone.
import math

cone_volume = lambda radius, height: (1/3) * math.pi * radius**2 * height

print(cone_volume(2,3))

# lambda function which gives tuple of max and min from a list
my_list = [10, 5, 20, 8, 15]
a=[]

min_value = lambda lst: min(lst)
max_value = lambda lst: max(lst)

max_result = max_value(my_list)
min_result = min_value(my_list)

a.append(min_result)
a.append(max_result)

b=tuple(a)
print(b)
```

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)

Code-:

```
# lambda function which gives tuple of max and min from a list
my_list = [10, 5, 20, 8, 15]
a=[]

min_value = lambda lst: min(lst)
max_value = lambda lst: max(lst)

max_result = max_value(my_list)
min_result = min_value(my_list)

a.append(min_result)
a.append(max_result)

b=tuple(a)
print(b)
```

7. Write functions to explain mentioned concepts:
   a. Keyword argument
   b. Default argument
   c. Variable length argument

Code-:

```
#Write functions to explain mentioned concepts:
#a.  Keyword argument
def greet(name, message):
   print(f"Hello, {name}! {message}")

greet(name="Alice", message="How are you?")

#b.  Default argument
def greet(name, message="How are you?"):
   print(f"Hello, {name}! {message}")

greet("Bob")

greet("Alice", "Nice to see you!")

#c.  Variable length argument
def calculate_sum(*args):
   total = 0
   for num in args:
      total += num
   return total
```

```
print(calculate_sum(1, 2, 3))
print(calculate_sum(5, 10, 15, 20))
```

LAB 8

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

Code-:

```
#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
 def count_names(filename):
    with open(filename, 'r') as file:
       names = file.readlines()
    return len(names)
 def count_names_starting_with_vowel(filename):
    vowels = "aeiouAEIOU"
    with open(filename, 'r') as file:
       names = file.readlines()
    count = sum(1 for name in names if name.strip()[0] in vowels)
    return count
 def find_longest_name(filename):
    with open(filename, 'r') as file:
       names = file.readlines()
    longest_name = max(names, key=lambda x: len(x.strip()))
    return longest_name.strip()
 def main():
    filename = "myfile.txt"
    total_names = count_names(filename)
    print("Total number of names:", total_names)
    names_starting_with_vowel = count_names_starting_with_vowel(filename)
    print("Number of names starting with a vowel:", names_starting_with_vowel)

    longest_name = find_longest_name(filename)
    print("Longest name:", longest_name)


 if name == "main":
    main()
```

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

Code-:

```
#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
 def read_integers(filename):
    with open(filename, 'r') as file:
       integers = [int(line.strip()) for line in file]
    return integers

 def find_max_number(integers):
    return max(integers)

 def calculate_average(integers):
    return sum(integers) / len(integers)

 def count_numbers_greater_than_100(integers):
    return sum(1 for num in integers if num > 100)

 def main():
    filename = 'myfile.txt'
    integers = read_integers(filename)

    if integers:
       print("Max number:", find_max_number(integers))
       print("Average of all numbers:", calculate_average(integers))
       print("Count of numbers greater than 100:", count_numbers_greater_than_100(integers))
    else:
       print("No integers found in the file.")

 if name == "main":
    main()
```

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:
    d. Display details of all cities
    e. Display city names with population more than 10Lakhs
    f. Display sum of areas of all cities

Code-:

```
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
 def details(filename):
    cities = []
    with open(filename, 'r') as file:
       for line in file:
          city_details = line.strip().split()
          city_name = city_details[0]
          population = float(city_details[1])
          area = float(city_details[2])
          cities.append((city_name, population, area))
    return cities

 def display_all_cities(cities):
    print("City Details:")
    for city in cities:
       print("City:", city[0])
       print("Population (in lakhs):", city[1])
       print("Area (in sq KM):", city[2])
       print()

 def display_cities_population_more_than_10_lakhs(cities):
    print("Cities with population more than 10 Lakhs:")
    for city in cities:
       if city[1] > 10:
          print(city[0])

 def calculate_sum_of_areas(cities):
    total_area = sum(city[2] for city in cities)
    print("Sum of areas of all cities:", total_area)

 def main():
    filename = 'city.txt'  # Change this to the name of your file
    cities = details(filename)
    if cities:
       display_all_cities(cities)
       display_cities_population_more_than_10_lakhs(cities)
       calculate_sum_of_areas(cities)
    else:
       print("No city details found in the file.")
 if name == "main":
    main()
```

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

Code-:

```
#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.
 def perform_division(a, b):
    try:
       result = int(a) // int(b)
       print(result)
    except ValueError as ve:
       print("Error Code:", ve)
    except ZeroDivisionError as zde:
       print("Error Code:", zde)

 if name == "main":
    N = int(input("Enter the number of test cases: "))
    for _ in range(N):
       a, b = input().split()
       perform_division(a, b)


#5.  Create multiple suitable exceptions for a file handling program.
 def read(filename):
    try:
       with open(filename, 'r') as file:
          content = file.read()
          print(content)
    except FileNotFoundError:
       print(f"Error: File '{filename}' not found.")
       except PermissionError:
       print(f"Error: Permission denied to open '{filename}'.")
    except IsADirectoryError:
       print(f"Error: '{filename}' is a directory, not a file.")
    except UnicodeDecodeError:
       print(f"Error: Unable to decode file '{filename}'. It may not be a text file.")
    except Exception as e:
       print(f"An unexpected error occurred: {e}")
    except ValueError as ve:
       print(f"Error code:",ve)
 if name == "main":
    filename = input("Enter the name of the file to read: ")
    read(filename)
```

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

Code-:

```
#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.
 def perform_division(a, b):
    try:
       result = int(a) // int(b)
       print(result)
    except ValueError as ve:
       print("Error Code:", ve)
    except ZeroDivisionError as zde:
       print("Error Code:", zde)

 if name == "main":
    N = int(input("Enter the number of test cases: "))
    for _ in range(N):
       a, b = input().split()
       perform_division(a, b)
```

```
#5.  Create multiple suitable exceptions for a file handling program.
 def read(filename):
    try:
       with open(filename, 'r') as file:
          content = file.read()
          print(content)
    except FileNotFoundError:
       print(f"Error: File '{filename}' not found.")
       except PermissionError:
       print(f"Error: Permission denied to open '{filename}'.")
    except IsADirectoryError:
       print(f"Error: '{filename}' is a directory, not a file.")
    except UnicodeDecodeError:
       print(f"Error: Unable to decode file '{filename}'. It may not be a text file.")
    except Exception as e:
       print(f"An unexpected error occurred: {e}")
    except ValueError as ve:
       print(f"Error code:",ve)
 if name == "main":
    filename = input("Enter the name of the file to read: ")
    read(filename)
```

LAB 9

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.
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*]*

Code-:

```
class Student:
  def init(self, name, sap_id, marks):
    self.name = name
    self.sap_id = sap_id
    self.marks = marks

  def display_details(self):
    print("Name:", self.name)
    print("SAP ID:", self.sap_id)
    print("Physics Marks:", self.marks['physics'])
    print("Chemistry Marks:", self.marks['chemistry'])
    print("Mathematics Marks:", self.marks['mathematics'])
    print()

if name == "main":
  students = []

  for i in range(3):
    name = input("Enter student name: ")
    sap_id = input("Enter SAP ID: ")
    physics_marks = float(input("Enter Physics marks: "))
    chemistry_marks = float(input("Enter Chemistry marks: "))
    mathematics_marks = float(input("Enter Mathematics marks: "))
    marks = {'physics': physics_marks,'chemistry': chemistry_marks,'mathematics': mathematics_marks }
    student = Student(name, sap_id, marks)
    students.append(student)
  print("\nDetails of all students:")
  for student in students:
    student.display_details()
```

3. Create programs to implement different types of inheritances.

Single Inheritance

Code-:

```
class Parent:
  def parent_method(self):
        print("Parent's method")

class Child(Parent):
    def child_method(self):
        print("Child's method")
```

Multiple Inheritance

```
class Parent1:
    def method1(self):
        print("Parent 1's method")

class Parent2:
    def method2(self):
        print("Parent 2's method")


class ChildMultiple(Parent1, Parent2):
    def child_method(self):
        print("Child's method")
```

Multilevel Inheritance

```
class Grandparent:
    def grandparent_method(self):
        print("Grandparent's method")


class ParentMultilevel(Grandparent):
    def parent_method(self):
        print("Parent's method")


class ChildMultilevel(ParentMultilevel):
    def child_method(self):
        print("Child's method")
```

Hierarchical Inheritance

```
class ParentHierarchical:
    def parent_method(self):
        print("Parent's method")


class Child1Hierarchical(ParentHierarchical):
    def child1_method(self):
        print("Child1's method")

class Child2Hierarchical(ParentHierarchical):
    def child2_method(self):
        print("Child2's method")
```

```
# Single Inheritance
print("Single Inheritance:")
```

```
child_obj = Child()
child_obj.parent_method()
child_obj.child_method()

    # Multiple Inheritance
print("\nMultiple Inheritance:")
child_multiple_obj = ChildMultiple()
child_multiple_obj.method1()
child_multiple_obj.method2()
child_multiple_obj.child_method()

    # Multilevel Inheritance
print("\nMultilevel Inheritance:")
child_multilevel_obj = ChildMultilevel()
child_multilevel_obj.grandparent_method()
child_multilevel_obj.parent_method()
child_multilevel_obj.child_method()

    # Hierarchical Inheritance
print("\nHierarchical Inheritance:")
child1_hierarchical_obj = Child1Hierarchical()
child2_hierarchical_obj = Child2Hierarchical()
child1_hierarchical_obj.parent_method()
child1_hierarchical_obj.child1_method()
child2_hierarchical_obj.parent_method()
child2_hierarchical_obj.child2_method()
```

4. Create a class to implement method Overriding.

Code-:

```
class Parent:
    def method(self):
        print("Parent's method")

class Child(Parent):
    def method(self):
        print("Child's overridden method")
par_obj = Parent()
child_obj = Child()
par_obj.method()
child_obj.method()
```

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)

Code-:

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

   def add(self, other):
      if isinstance(other, Point): #subclass
         new_x = self.x + other.x
         new_y = self.y + other.y
         return Point(new_x, new_y)
      else:
         raise TypeError("Unsupported operand type(s) for +: 'Point' and '{}'".format(type(other)))

   def str(self):
      return "Point(x={}, y={})".format(self.x, self.y)
P1 = Point(25, 60)
P2 = Point(92, 10)
P3 = P1 + P2
print("P1:", P1)
print("P2:", P2)
print("P3:",P3)
```

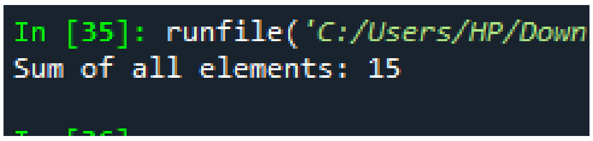
LAB 10

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

Code-:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

sum_of_elements = np.sum(arr)
print(f"Sum of all elements: {sum_of_elements}")
```

Output-:



```
In [35]: runfile('C:/Users/HP/Down
Sum of all elements: 15
```

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

Code-:

import numpy as np

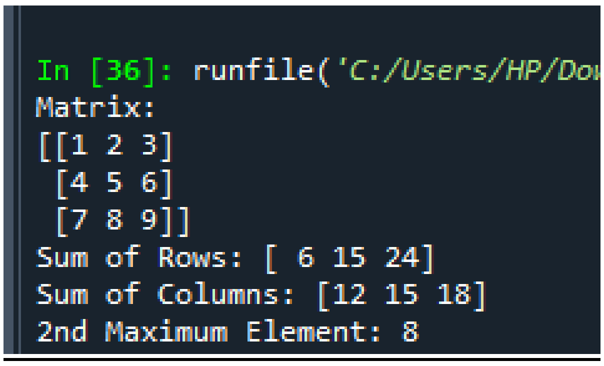
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

sum_of_rows = np.sum(matrix, axis=1)
sum_of_columns = np.sum(matrix, axis=0)

second_max_element = np.partition(matrix.flatten(), -2)[-2]

print(f"Matrix:\n{matrix}")
print(f"Sum of Rows: {sum_of_rows}")
print(f"Sum of Columns: {sum_of_columns}")
print(f"2nd Maximum Element: {second_max_element}")

Output-:



```
In [36]: runfile('C:/Users/HP/Do
Matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Sum of Rows: [ 6 15 24]
Sum of Columns: [12 15 18]
2nd Maximum Element: 8
```

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

Code-:

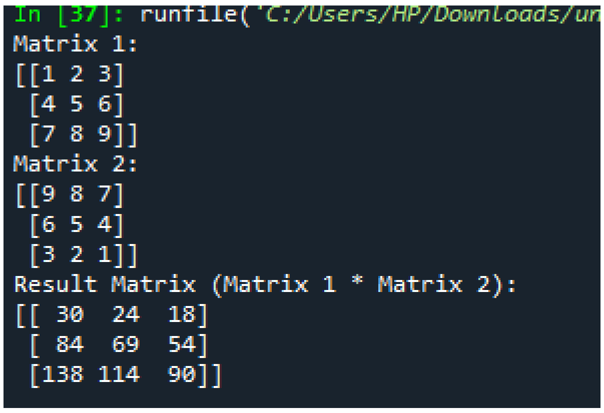
import numpy as np

matrix1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix2 = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

result_matrix = np.dot(matrix1, matrix2)

print(f"Matrix 1:\n{matrix1}")
print(f"Matrix 2:\n{matrix2}")
*print(f"Result Matrix (Matrix 1 * Matrix 2):\n{result_matrix}")*

Output-:



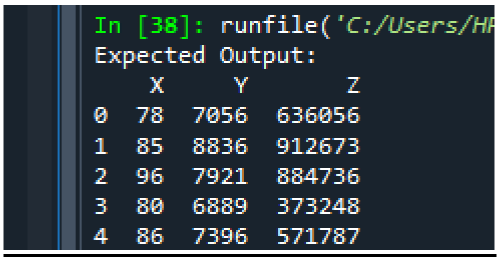
```
In [37]: runfile('C:/Users/HP/Downloads/un
Matrix 1:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Matrix 2:
[[9 8 7]
 [6 5 4]
 [3 2 1]]
Result Matrix (Matrix 1 * Matrix 2):
[[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
```

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

Code-:

```
import pandas as pd
data = {'X': [78, 85, 96, 80, 86], 'Y': [84, 94, 89, 83, 86], 'Z': [86, 97, 96, 72, 83]}
df = pd.DataFrame(data)
powers_df = df.pow([1, 2, 3])
print("Expected Output:")
print(powers_df)
```

Output-:



```
In [38]: runfile('C:/Users/HP
Expected Output:
    X     Y       Z
0  78  7056  636056
1  85  8836  912673
2  96  7921  884736
3  80  6889  373248
4  86  7396  571787
```

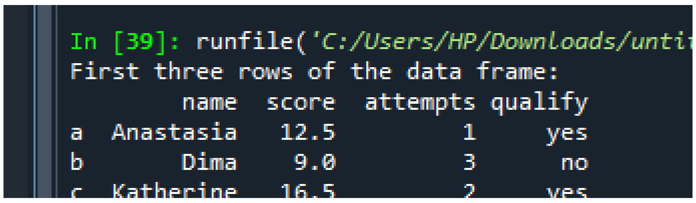
5. Write a Pandas program to get the first 3 rows of a given DataFrame.

Code-:

import pandas as pd

```
import numpy as np

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']

df = pd.DataFrame(exam_data, index=labels)

first_three_rows = df.head(3)

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

Output-:



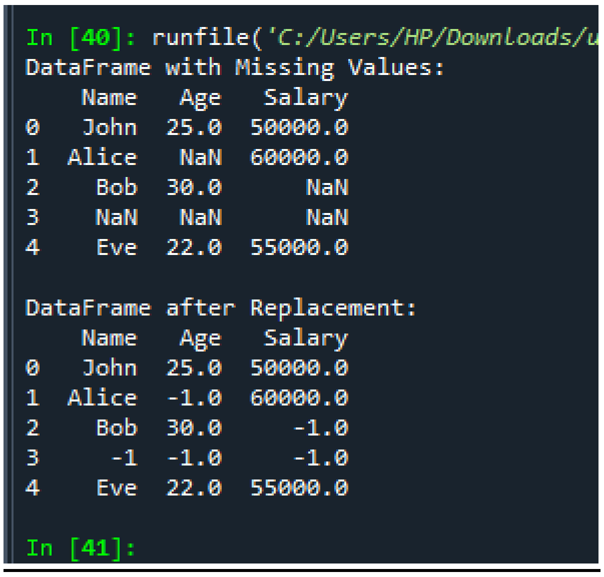
```
In [39]: runfile('C:/Users/HP/Downloads/unti
First three rows of the data frame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

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

Code-:

```
import pandas as pd
import numpy as np

data = {'Name': ['John', 'Alice', 'Bob', np.nan, 'Eve'],
      'Age': [25, np.nan, 30, np.nan, 22],
      'Salary': [50000, 60000, np.nan, np.nan, 55000]}

df = pd.DataFrame(data)

df_filled = df.fillna(-1)

print("DataFrame with Missing Values:")
print(df)
print("\nDataFrame after Replacement:")
print(df_filled)
```

Output-:



```
In [40]: runfile('C:/Users/HP/Downloads/u
DataFrame with Missing Values:
    Name   Age   Salary
0   John  25.0  50000.0
1  Alice   NaN  60000.0
2    Bob  30.0      NaN
3    NaN   NaN      NaN
4    Eve  22.0  55000.0

DataFrame after Replacement:
    Name   Age   Salary
0   John  25.0  50000.0
1  Alice  -1.0  60000.0
2    Bob  30.0     -1.0
3     -1  -1.0     -1.0
4    Eve  22.0  55000.0

In [41]:
```

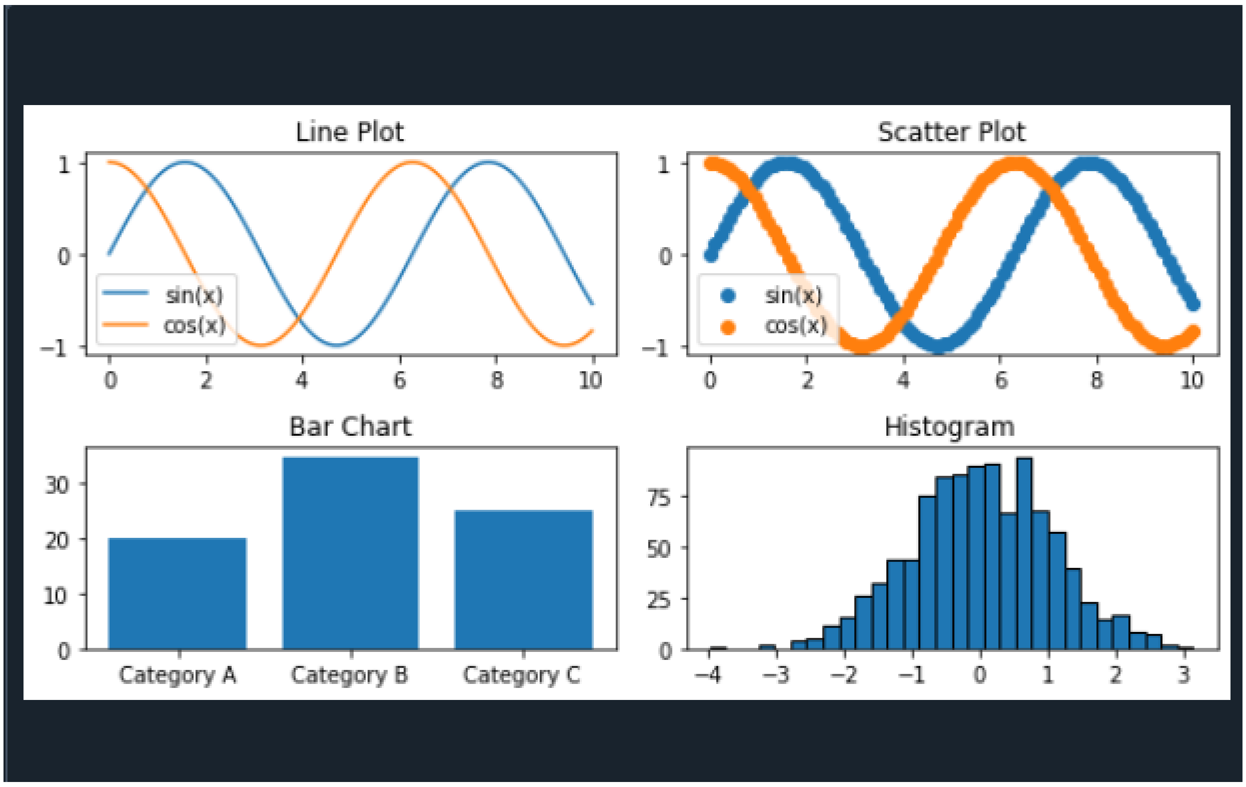
7. Create a program to demonstrate different visual forms using Matplotlib.

Code-:

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

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

plt.figure(figsize=(8, 4))
plt.subplot(2, 2, 1)
plt.plot(x, y1, label='sin(x)')
plt.plot(x, y2, label='cos(x)')
plt.title('Line Plot')
plt.legend()

plt.subplot(2, 2, 2)
plt.scatter(x, y1, label='sin(x)')
plt.scatter(x, y2, label='cos(x)')
plt.title('Scatter Plot')
plt.legend()

categories = ['Category A', 'Category B', 'Category C']
values = [20, 35, 25]
plt.subplot(2, 2, 3)
plt.bar(categories, values)
plt.title('Bar Chart')

data = np.random.randn(1000)
plt.subplot(2, 2, 4)
plt.hist(data, bins=30, edgecolor='black')
plt.title('Histogram')
```

plt.tight_layout()
plt.show()

Output-:


Line Plot
Scatter Plot
Bar Chart
Histogram
sin(x)
cos(x)
sin(x)
cos(x)
Category A
Category B
Category C