-:Python Assignments:-

Name: Mohit Limbachiya

Roll No.: 26

Div: D

DAY-1

1. Write a python program to print “Hello World”.

```
print("Hello Word")
```

Output:

```
Hello Word
```

2. Write a python program to get user input using input( ) function.

```
name=input("enter Name...")
print(name)
age=input("enter your Age..")
print(age)
```

Output:

```
enter Name...31/01/2002
31/01/2002
enter your Age..21
21
```

3. WAP to take only numerical input.

```
no1=eval(input("enter your number 1.."))
print(no1)
no2=eval(input("enter your number 2.."))
print(no2)
```

Output:

```
enter your number 1..10
10
enter your number 2..20
20
```

4. Convert the number to floating point number.

```
no1=eval(input("enter your number 1.."))
print(no1)

no2=float(no1)
print(no2)
```

Output:

```
enter your number 1..10
10
10.0
```

5. WAP to Add, Subtract, Multiply and Divide 2 numbers.

```
no1=eval(input("enter your number 1.."))
print(no1)
no2=eval(input("enter your number 2.."))
print(no2)
```

```
print("\n")
add=no1+no2
sub=no1-no2
mul=no1*no2
div=no1/no2

print(add)
print(sub)
print(mul)
print(div)
```

Output:

```
enter your number 1..10
10
enter your number 2..20
20



30
-10
200
0.5
```

6. Print the quotient of remainder separately for division operation.

```
no1=eval(input("enter your number 1.."))
print(no1)
no2=eval(input("enter your number 2.."))
print(no2)
```

```
print("Quotiont",(no1/no2))
print("Reminder",(no1%no2))
```

Output:

```
enter your number 1..10
10
enter your number 2..20
20
Quotiont 0.5
Reminder 10
```

7. Write a Python Program to Print Bio Data.

```
print("Biodata")

print("Name : M0hit")
print("Birth Date : 31/01/2002")
print("Age: 22")
print("Mb no: 9484452440")
```

Output:

```
Biodata
Name : M0hit
Birth Date : 31/01/2002
Age: 22
Mb no: 9484452440
```

8. Print your name with Hello…..using input() function.

```
name="M0hit"

print("Hello",name)
```

Output:

Hello M0hit

9. Take an input string and print it three times using one single print Statement.

```
name=input("enter Name...")
print(name*3,"\n")
print(name,name,name)
```

Output:

enter Name...M0hit

M0hitM0hitM0hit

M0hit M0hit M0hit

DAY-2

1. Write a program to find meter to kilometer.

```
m=eval(input("enter meter..."))
```

```
k=m/1000
print(k)
```

Output:

```
enter meter...1000
1.0
```

2. Write a program to find area of a rectangle. (Area=l*b). Take input parameters from user.

```
l=eval(input("enter length..."))
b=eval(input("enter breath..."))

print("Area of ragtangle is : ",(l*b))
```

Output:

```
enter length...10
enter breath...20
Area of ragtangle is :  200
```

3. Write a program to find volume of cube. (Area=l*b*h). Take input parameters from user.

```
l=eval(input("enter length..."))
b=eval(input("enter breath..."))
h=eval(input("enter height..."))

Area=(l*b*h)
```

```
print(Area)
```

Output:

```
enter length...10
enter breath...20
enter height...20
4000
```

4. Write a program to find area of triangle. (Area=(l*b)/2). Take input parameters from user.

```
l=eval(input("enter length..."))
b=eval(input("enter breath..."))

Area=(l*b)/2

print(Area)
```

Output:

```
enter length...10
enter breath...20
100.0
```

5. WAP to convert the given temperature from Fahrenheit to Celsius using the formula

C = (F – 32) / 1.8.

```
f=eval(input("enter Farenhit..."))
C=(f-32)/1.8
Print(C)
```

Output:

```
enter Farenhit...10
-12.222222222222221
```

6.WAP to convert temperature from Celsius to Fahrenheit where temperature in Celsius is entered by user. (F= C(9/5) + 32).

```
c=eval(input("enter celsius..."))
f=(c*(9/5)+32)
print(f)
```

Output:

```
enter celsius...10
50.0
```

7. Take a number as user input and convert it into Binary, Octal and Hexadecimal numbers.

```
n=eval(input("enter any number"))

print("decimal to binary",bin(n))
print("decimal to octal",oct(n))
print("decimal to hexadecimal",hex(n))
```

Output:

```
enter any number10
decimal to binary 0b1010
decimal to octal 0o12
decimal to hexadecimal 0xa
```

8. Take binary, octal and hexadecimal numbers as an input and convert them to Decimal number.

```
n1=eval(input("enter any binary "))
n2=eval(input("enter any octal "))
n3=eval(input("enter any hexadecimal "))

print("Binary to decimal",int(n1))
print("octal to decimal",int(n2))
print("hexadecimal to decimal",int(n3))
```

Output:

```
enter any binary 10
enter any octal 20
enter any hexadecimal 5
Binary to decimal 10
```

octal to decimal 20

hexadecimal to decimal 5

9. Without applying condition statement display output as “true” if the 1st number greater than the 2 nd number and “false” if 2 nd number is larger than the 1 st one.

```
a = eval(input("Enter value of a :-"))
b = eval(input("Enter value of b :-"))

c= (a>b)

print("Your ans is :- ",c)
```

Output:

```
Enter value of a :-10
Enter value of b :-20
Your ans is :-  False
```

10. Take 3 different inputs from user and display their data type.

```
a= (20,)
b= 1.5
c="M0hit"

print("data type is ",type(a))
print("data type is ",type(b))
print("data type is ",type(c))
```

Output:

```
data type is  <class 'tuple'>
data type is  <class 'float'>
data type is  <class 'str'>
```

11. Find the minimum and maximum of 2 numbers.

```
a= eval(input("Enter value of a :-"))
b= eval(input("Enter value of b :-"))

print("maximum...",max(a,b))

print("minimum...",min(a,b))
```

Output:

```
Enter value of a :-10
Enter value of b :-20
maximum... 20
minimum... 10
```

12. Perform binary “AND” and “OR” operation for given 2 integer numbers from user input.

```
a= eval(input("Enter value of a :-"))
b= eval(input("Enter value of b :-"))

print("And operation of A and b :- ",a&b)
print("or operation of A or b :- ",a|b)
```

Output:

Enter value of a :-10

Enter value of b :-20

And operation of A and b :- 0

or operation of A or b :- 30

13. Write a program to calculate area of circle. (pi*r*r).

```
PI = 3.14
r = eval(input("Enter radius :- "))

area = PI*r*r

print("area of circle is :- ", area)
```

Output:

Enter radius :- 4

area of circle is :- 50.24

14. Write a program in C to calculate simple interest using formula SI = (P*R*N) / 100. Take all parameters as input from user.

```
p = eval(input("Enter Principle amount :- "))
r = eval(input("Enter rate of intrest :- "))
n = eval(input("Enter Number of year :- "))
```

```
SI = (p*r*n)/100

print("Simple intrest of :- ", SI)
```

Output:

```
Enter Principle amount :- 10000
Enter rate of intrest :- 100
Enter Number of year :- 10
Simple intrest of :-  100000.0
```

15. Without applying condition statement display output as “true” if a number is an even number and “false” if the number is an odd number.

```
a = eval(input("Enter Number :- "))
print(a%2==0)
```

Output:

```
Enter Number :- 100
True
```

16. Find the minimum and maximum of 2 numbers.

```
a = eval(input("Entar value of a :- "))
b = eval(input("Entar value of b :- "))

print("maximum:-",max(a,b))
```

```
print("minimum:-",min(a,b))
```

Output:

```
Entar value of a :- 10
Entar value of b :- 20
maximum:- 20
minimum:- 10
```

DAY-3

1. Print multiple lines using single print statement. as –

I like “Python Programming” very much

It is my favorite subject.

```
print('I like “Python Programming” very much \n It is my favorite subject')
```

Output:

```
I like “Python Programming” very much
 It is my favorite subject
```

2. Print a part of the above string “very much” using the slice operator.

```
str1='I like “Python Programming” very much \n It is my favorite subject'

print(str1[28:37])
```

Output:

very much

3. Print the last 5 characters from the above given string.

str1='I like “Python Programming” very much \n It is my favorite subject'

print(str1[28:37])

Output:

bject

4. Print only the second line of the given string.

str1='I like “Python Programming” very much \n It is my favorite subject'

print(str1[38:])

Output:

It is my favorite subject

5. Take two strings as input from the user and concatenate them.

```
str1=input("enter string 1...")
str2=input("enter string 2...")

print(str1+str2)
```

Output:

```
enter string 1...hello
enter string 2...welcome
hellowelcome
```

6. Take a number and a string from the user and repeat the string for that many times.

```
str1=input("enter string 1...")
no=eval(input("enter number..."))

print(str1*no)
```

Output:

```
enter string 1...hello
enter number...3
hellohellohello
```

7. **Take an input character from the user and check whether that character is present in the above given string or not. – Using 'in' operator and using 'not in' operator.**

```
str1='I like “Python Programming” very much \n It is my favorite subject'

ch=input("enter character or word..")

print("In operator",ch in str1)

print("Not In opeator",ch not in str1)
```

Output:

```
enter character or word..xyz
In operator False
Not In opeator True
```

DAY-4

1. Create a menu driven program for string manipulation

a. Find the length of a string

b. Print the string in upper case

c. Print the string in lower case

d. Print the string with initial capital

e. Split the string

```
txt="Hello,Welcome to python class."
```

```
print("title:",txt.title())
print("lower:",txt.lower())
print("Upper:",txt.upper())
print("length:",len(txt))
print("spilt:",txt.split(","))
```

Output:

```
title: Hello,Welcome To Python Class.
lower: hello,welcome to python class.
Upper: HELLO,WELCOME TO PYTHON CLASS.
length: 30
spilt: ['Hello', 'Welcome to python class.']
```

2. Take two strings as input s1 and s2 and check whether s2 is present in s1 or not.

```
s1 = input("Enter the first string: ")
s2 = input("Enter the second string: ")

if s2 in s1:
   print("found")
else:
   print("not found")
```

Output:

```
Enter the first string: xyz
Enter the second string: abc
```

not found

3. If s2 is a part of s1 then print the 1st and last occurrences of it.

```
# Taking two strings as input
s1 = input("Enter the first string: ")
s2 = input("Enter the second string: ")

# Checking if s2 is present in s1
if s2 in s1:
    first_index = s1.index(s2)
    last_index = s1.rindex(s2)
    print("present")
    print("first: ",first_index)
    print("last: ",last_index)
else:
    print(f"{s2} is not present in {s1}.")
```

Output:

```
Enter the first string: xyz
Enter the second string: a
a is not present in xyz.
```

4. If s2 is present in s1 then also count number of times it occurs in s1.

```
s1 = input("Enter the first string: ")
s2 = input("Enter the second string: ")

if s2 in s1:
   print(s1.count(s2))
```

Output:

```
Enter the first string: welcome to python class
Enter the second string: welcome
1
```

5. Count total number of words in the string input by user.

```
s = input("Enter a string: ")
w = len(s.split())

print(w)
```

Output:

```
Enter a string: M0hit
1
```

6. Take two string and apply all string function.

```
str1="Hello,Good morning "
str2="Welcome to python lab "
txt=str1+str2

print("endswith:",txt.endswith("ss."))
print("Count",txt.count("o"))
print("Find:",txt.find("W"))
print("Rfind:",txt.rfind("l"))
print("title:",txt.title())
print("lower:",txt.lower())
print("Upper:",txt.upper())
print("length:",len(txt))
print("cpitalized:",txt.capitalize())
print("index:",txt.index("Welcome"))
print("rindex:",txt.rindex("python"))
print("spilt:",txt.split(","))
print("replace:",txt.replace("class","lab"))
print("isdigit",txt.isdigit())
print("isalpha",txt.isalpha())
print("Count start end",txt.count("G",4,10))
```

Output:

```
endswith: False
Count 7
Find: 19
Rfind: 37
title: Hello,Good Morning Welcome To Python Lab
```

lower: hello,good morning welcome to python lab

Upper: HELLO,GOOD MORNING WELCOME TO PYTHON LAB

length: 41

cpitalized: Hello,good morning welcome to python lab

index: 19

rindex: 30

spilt: ['Hello', 'Good morning Welcome to python lab ']

replace: Hello,Good morning Welcome to python lab

isdigit False

isalpha False

Count start end 1

DAY-5

1. Create a tuple for name say t1 (FirstName, MiddleName, LastName).

```
t1=("M0hit")

print(t1)
```

Output:

```
('M0hit')
```

2. Create a tuple say t2 for marks of 5 subjects.

```
marks=(55,60,70,80,90)

print(marks)
```

Output:

(55, 60, 70, 80, 90)

3. Make a total of all the marks and print it. (with and without using sum() method).

```
marks=(55,60,70,80,90)
print(marks)
print(sum(marks))
```

Output:

(55, 60, 70, 80, 90)

355

4. Make a tuple t3 having 2 elements as t1 and t2 (tuples created above) – It is called a nested tuple.

```
t1=(10,15,20,25,30,35)
t2=("apple","banana","mango","grapes")
t3=(t1,t2)
for i in t3:
    print(i)
```

Output:

(10, 15, 20, 25, 30, 35)

('apple', 'banana', 'mango', 'grapes')

5. Take an input number and find whether that is present as an element in the tuple t3 or not.

```
t3 = (1, 3, 5, 7, 9)
print(t3)
s = int(input("Enter search number "))

for i in t3:
   if  i == s:
      print("found")
      break
else:
   print("not found")
```

Output:

```
(1, 3, 5, 7, 9)
Enter search number 1
Found
```

6. Create a tuple of 5 fruits. Ask the user to input a fruit name and search that name in the given fruit tuple. Display suitable messages.

```
fruits =()
print("enter five fruits:")

for _ in range(5):
   f=input().strip().capitalize()
   fruits += (f,)
print("fruits",fruits)
```

```
s = input("Search fruit ")

for i in fruits:
   if  i == s:
      print("found")
      break
else:
   print("not found")
```

Output:

```
enter five fruits:
banana
orange
mango
kiwi
chiku
fruits ('Banana', 'Orange', 'Mango', 'Kiwi', 'Chiku')
Search fruit banana
not found
```

7. Create a tuple of cities of Gujarat by taking user input.

```
city =()
print("enter five cities of gujarat:")

for _ in range(5):
```

```
    ucity=input().strip().capitalize()
    city += (ucity,)
print("Gujarat cities",city)
```

Output:

```
enter five cities of gujarat:
rajkot
ahmedabad
surat
baroda
gandhinagar
Gujarat cities ('Rajkot', 'Ahmedabad', 'Surat', 'Baroda', 'Gandhinagar')
```

8. Find the length of name of each city in the above tuple. With and without len() method.

```
city =()
print("enter five cities of gujarat:")

for _ in range(5):
    ucity=input().strip().capitalize()
    city += (ucity,)
print("Gujarat cities",city)

#with length function

for i in city:
    print("city",i,"",len(i))
```

```
#without length function
for i in city:
    count = 0
    for j in i:
        count += 1
    print(i," city ",count)
```

Output:

```
enter five cities of gujarat:
rajkot
jamnagar
surat
gir somnath
ahmedabad
Gujarat cities ('Rajkot', 'Jamnagar', 'Surat', 'Baroda, 'Ahmedabad')
city Rajkot   6
city Jamnagar   8
city Surat   5
city Baroda   6
city Ahmedabad   9
Rajkot  city  6
Jamnagar  city  8
Surat  city  5
Baroda  city  6
Ahmedabad  city  9
```

9. Create a nested tuple t4 of your (name, (hobbies), (friends), degree) by taking user inputs.

```
name = input("Enter your name: ")
hobbies = tuple(input("Enter your hobbies: ").split(','))
friends = tuple(input("Enter your friends:  ").split(','))
degree = input("Enter your degree: ")

t4 = (name, hobbies, friends, degree)

print("Nested Tuple t4:", t4)
```

Output:

```
Enter your name: M0hit
Enter your hobbies: Gaming
Enter your friends:  Hit, Ketan, Vivek
Enter your degree: mca
Nested Tuple t4: ('M0hit', (' Gaming ',), (‘Hit, Ketan, Vivek’,), 'mca')
```

10. Find an element in the nested tuple (t4) and print its position if found, otherwise print “Not found”.

```
name = input("Enter your name: ")
hobbies = tuple(input("Enter your hobbies: ").split(','))
friends = tuple(input("Enter your friends:  ").split(','))
degree = input("Enter your degree: ")

t4 = (name, hobbies, friends, degree)

print("Nested Tuple t4:", t4)

s = input("enter search element..")
```

```
for i in t4:
    if i==s:
        print("found")
        break
else:
    print("not found")
```

Output:

```
Enter your name: M0hit
Enter your hobbies: Gaming
Enter your friends:  Hit, Ketan, Vivek
Enter your degree: mca
Nested Tuple t4: ('M0hit', ('Gaming',), (‘Hit, Ketan, Vivek’,), 'mca')
enter search element..M0hit
found
```

DAY-6

1. Find the minimum and maximum of 3 numbers.

```
n1=int(input("enter number 1......"))
n2=int(input("enter number 2......"))
n3=int(input("enter number 3......"))

if n1>n2 and n1>n2:
    print("n1 is maximum")
elif n2>n1 and n2>n3:
    print("n2 is maximum")
else:
    print("n3 is maximum")

if n1<n2 and n1<n3:
    print("n1 is minimum")
elif n2<n1 and n2<n3:
    print("n2 is mainmum")
else:
    print("n3 is minmum")
```

Output:

enter number 1.....40

enter number 2.....50

enter number 3.....10

n2 is maximum

n3 is minimum

2. WAP to print if a number is even or odd.

```
no=int(input("enter number ......"))

if no%2==0:
    print("number is even")
else:
    print("number is odd")
```

Output:

enter number 10

number is even

3. WAP to check if a number is prime number or not.

```
no=int(input("enter number ......"))

if no<=2:
    print("number is not prime")
else:
    for i in range(2,no):
        if no%i==0:
            print("number is not prime")
            break
    else:
        print("number is prime")
```

Output:

```
enter number ......10
number is not prime
```

4. WAP to know if a number is zero, positive or negative.

```
no=int(input("enter number ......"))

        if no>0:
           print("number is positive")
        elif no==0:
           print("number is zero")
        else:
           print("number is negative")
```

Output:

```
enter number ......10
number is positive
```

5. Check for a palindrome number.

```
no=input("Enter number :- ")

if no == no[::-1]:
    print(no," is palindrome")
```

```
else:
    print(no," is not palindrome")
```

Output:

```
Enter number :- 10
10  is not palindrome
```

6. Print the number in reverse order (eg. 1234 o/p 4321).

```
no=input("Enter number :- ")
print("Before Reverse Number....",no)
no=no[::-1]
print("After Reverse Number....",no)
```

Output:

```
Enter number :- 1234
Before Reverse Number.... 1234
After Reverse Number.... 4321
```

7. Print numbers from 1 to 50.

```
for i in range(1,51):
    print(i)
```

Output:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

8. Print the ‘*’ patterns using range().

```
for i in range(5):
    for j in range(5):
        print("* " ,  end="")
```

```
    print()
```

Output:

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

9. Print the number pyramid using range().

```
for i in range(0,5):
    print("" * (5-i), end="")
    for j in range(0,i+1):
        print("*",end="")
    print()
```

Output:

```
    *
   * *
  * * *
 * * * *
* * * * *
```

10. Print all prime numbers between 1 to 50.

```
for num in range(2, 51):
   prime = True
   for i in range(2, int(num**0.5) + 1):
      if num % i == 0:
         prime = False
         break
   if prime:
      print(num, end="")
```

Output:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

11. Print all numbers divisible by 7 between 1 to 100.

```
for i in range(1,100):
   if i%7==0:
      print(i)
```

Output:

7
14
21
28
35
42
49

56

63

70

77

84

91

98

DAY-7

1. Create a list of students say L1.

```
l1=["M0hit","Hit","Vivek","Ketan"]
print(l1)
```

Output:

```
['M0hit', 'Hit', 'Vivek', 'Ketan']
```

2. Count total number of students from the above list.

```
l1=["M0hit","Hit","Vivek","Ketan"]
print(len(l1))
```

Output:

4

3. Add one more student in the list L1.

```
l1=["M0hit","Hit","Vivek","Ketan"]

l1.append("Kunj")
print(l1)
```

Output:

['M0hit', 'Hit', 'Vivek', 'Ketan', ‘Kunj’]

4. Display all the students in the sorted order.

```
l1=["M0hit","Hit","Vivek","Ketan"]

sorted_students = sorted(l1)

print("student in sorted order: ")

for i in sorted_students:
   print(i)
```

Output:

student in sorted order:
Ketan

M0hit

Hit

Vivek

5. Check a particular student's name is present in the list or not.

```
l1=["M0hit","Hit","Vivek","Ketan"]

student_name=input("enter student name...")

if student_name in l1:
   print(student_name,"is present in the list")
else:
   print(student_name,"is not present in the list")
```

Output:

```
enter student name...M0hit
M0hit is present in the list
```

6. If the student's name is present in the list, print total number of same name students in the list L1 and display the position of 1 st occurrence of that name.

```
l1=["M0hit","Hit","Vivek","Ketan"]

student_name=input("enter student name...")

if student_name in l1:
   print(student_name,"is present in the list")
   print(l1.count(student_name))
```

```
    print(l1.index(student_name,3,5))
else:
    print(student_name,"is not present in the list")
```

Output:

```
enter student name...M0hit
M0hit is present in the list
3
4
```

7. Remove the last student from the list L1.

```
l1=["M0hit","Hit","Vivek","Ketan","Maulik"]

r=l1.pop()
print(l1)
```

Output:

```
['M0hit', 'Hit', 'Vivek', 'Ketan']
```

8. Remove a particular student from the list. (Take a name of student from the user.).

```
l1=["M0hit","Hit","Ketan","M0hit"]

r=input("enter student name...")

if r in l1:
```

```
    l1.remove(r)
    print(l1)
else:
    print("element not found ")
```

Output:

```
enter student name...M0hit
['Hit', 'Ketan', 'Vivek', 'M0hit']
```

9. While removing the student from the list, if multiple students have same name then remove all of them from the list.

```
l1=["M0hit","Hit","Vivek","Ketan"]


r=input("enter student name...")

if r in l1:
    while r in l1:
        l1.remove(r)
    print(l1)
else:
    print("element not found ")
```

Output:

```
enter student name...M0hit
```

['Hit', 'Ketan', 'Vivek']

10. Create a list of 10 numbers and find the maximum and minimum numbers from it.

```
ln=[5,10,15,20,25,30,35,40,45,50]

print("minimum number: ",min(ln))
print("maximum number: ",max(ln))
```

Output:

```
minimum number:  5
maximum number:  50
```

11. Create a list of alphabets and count total number of vowels in it.

```
alphabets = ['a','b','c','i','o','m','e','f','h','e','a']

vowel_count = 0

for i in alphabets:
    if i in ['a', 'e', 'i', 'o', 'u']:
        vowel_count += 1

print("Total number of vowels:", vowel_count)
```

Output:

```
Total number of vowels: 6
```

12. Create a list of even numbers between 1 to 21 using range().

```
even_numbers = list(range(2, 21, 2))

print("List of even numbers between 1 and 21:", even_numbers)
```

Output:

List of even numbers between 1 and 21: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

13. Create a list of employees (nested list) with their personal details like [name, age, salary,expertise] in a list. Ask the user to enter name and display the details of that employee. If the employee is not in the list, print error message.

```
emp1 = ["M0hit", 30, 50000, "Software Engineer"]
emp2 = ["Vivek", 35, 60000, "Data Scientist"]
emp3 = ["Hit", 28, 45000, "Web Developer"]
emp4 = ["Ketan", 32, 55000, "UX Designer"]

emp=[emp1,emp2,emp3,emp4]

search_name = input("Enter the name of the employee: ")

for i in emp:
    if i[0] == search_name:
        print("found")
        print("Name: ",i[0])
        print("Age: ",i[1])
```

```
        print("Salary: ",i[2])
        print("Expertise: ",i[3])
        break
else:
   print("not found")
```

Output:

```
Enter the name of the employee: M0hit
found
Name:  M0hit
Age:  30
Salary:  50000
Expertise:  Software Engineer
```

14. Create a list by taking any 5 inputs from the user.

```
stud=[]
n=5
for i in range(n):
   list=input("enter list  ")
   stud.append(list)
print(stud)
```

Output:

```
enter list  1
enter list  11
enter list  21
```

enter list 31

enter list 41

['1', '11', '21', '31', '41']

15. Display the students from L1 list, whose name contains the character ‘a’.

```
L1 = [
   {"name": "M0hit"},
   {"name": "Hit"},
   {"name": "Bhautik"},
   {"name": "Sarah"}
]
```

```
students_with_a = [student for student in L1 if 'a' in student['name'] or 'A' in student['name']]
```

```
for student in students_with_a:
   print(student['name'])
```

Output:

Bhautik

Sarah

16. Create a list of 10 numbers and find the total of odd numbers and even numbers.

```
l1=[1,2,3,4,5,6,7,8,9,10]

odd=[num for num in l1 if num%2!=0]
even=[num for num in l1 if num%2==0]

print(len(odd))
print(len(even))
```

```
5
5
```

17. Put all the odd numbers in 1 list and even numbers in another list using list comprehension.

```
l1=[1,2,3,4,5,6,7,8,9,10]

odd=[num for num in l1 if num%2!=0]
even=[num for num in l1 if num%2==0]

print(odd)
print(even)
```

Output:

```
[1, 3, 5, 7, 9]
[2, 4, 6, 8, 10]
```

18.Create a list having mixed elements like string and integers. Print only integer elements from the list with and without using list comprehension.

```
mixed_list = ['apple', 5, 'banana', 10, 'cherry', 15]

# Using list comprehension to extract integer elements
integer_elements = [x for x in mixed_list if isinstance(x, int)]

print("Integer elements using list comprehension:", integer_elements)

mixed_list = ['apple', 5, 'banana', 10, 'cherry', 15]

# Without list comprehension
integer_elements = []
for x in mixed_list:
    if isinstance(x, int):
        integer_elements.append(x)

print("Integer elements without list comprehension:", integer_elements)
```

Output:

```
Integer elements using list comprehension: [5, 10, 15]
Integer elements without list comprehension: [5, 10, 15]
```

DAY-8

1. Create a dictionary of employees where empId will be the key and value will be the name of an employee.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Kunj"}

print(employees)
```

Output:

```
{'E01': 'M0hit', 'E02': 'Vivek', 'E03': 'Ketan', 'E04': 'Hit', 'E05': ‘Kunj’}
```

2. Display how many employees are there in the dictionary.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Maulik",
"E06":"Nilesh"}

print(len(employees))
```

Output:

```
6
```

3. Display all empID and make a separate list of just IDs.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Maulik"}

print(employees.keys())
```

Output:

dict_keys(['E01', 'E02', 'E03', 'E04', 'E05'])

4. Display all employee names and take them to a separate list.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Kunj"}

print(employees.values())
```

Output:

dict_values(['M0hit', 'Vivek', 'Ketan', 'Hit', Kunj])

5. Take an empId from the user and check if that employee is there in the dictionary or not.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Kunj"}

empid=input("enter employee id: ")
if empid in employees:
   print("id found")
else:
   print("not found")
```

Output:

enter employee id: E01
id found

6. If an empID is there in the dictionary then display the name of that employee or if not available then add an ID and Name of the employee in the dictionary.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Sahdev"}

empid=input("enter employee id: ")
if empid in employees:
   print("id found")
else:
   print("not found")
   name=input("enter employee name..")
   employees[empid]=name
   print("Employee added successfully!")

print(employees)
```

Output:

```
enter employee id: E06
not found
enter employee name.. Khuman
Employee added successfully!
{'E01': 'M0hit', 'E02': 'Vivek', 'E03': 'Ketan', 'E04': 'Hit', 'E05': Sahdev, 'E06': 'Khuman'}
```

7. Change the name of the employee of empID taken by the user.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Maulik"}

empid=input("enter employee id: ")
```

```
if empid in employees:
    print("found")
    newname=input("enter name you want to change..")
    employees[empid]=newname
    print("updated successfully")
    print(employees)
else:
    print("not found")
```

Output:

```
enter employee id: E05
found
enter name you want to change..Dhrumil
updated succussfullyy
{'E01': 'M0hit', 'E02': 'Vivek', 'E03': 'Ketan', 'E04': 'Hit', 'E05': ' Dhrumil '}
```

8. Remove an employee whose ID is provided by the user.

```
employees={"E01":"M0hit","E02":"Vivek","E03":"Ketan","E04":"Hit","E05":"Maulik"}

empid=input("enter employee id: ")

if empid in employees:
    print("found")
    del employees[empid]
    print("remove succussfullyy")
```

```
    print(employees)
else:
    print("not found")
```

Output:

```
enter employee id: E01
found
remove succussfullyy
{'E02': 'Vivek', 'E03': 'Ketan', 'E04': 'Hit', 'E05': ‘Maulik’}
```

9. Take 5 names of students as an input from the user and create a dictionary with keys as their initials and value is a list as [age, degree, favorite subject].

```
students = {}


for i in range(1):
    name = input("Enter student name: ")
    age = int(input("Enter student age: "))
    degree = input("Enter student degree: ")
    favorite_subject = input("Enter student's favorite subject: ")

    initials = ''.join([name.split()[0][0]])
    students[initials] = [age, degree, favorite_subject]

print("\nDictionary of students with initials as keys:")
print(students)
```

Output:

Enter student name: M0hit

Enter student age: 22

Enter student degree: bca

Enter student's favorite subject: python

Dictionary of students with initials as keys:

{'r': [20, 'bca', 'python']}

10. Display the youngest student from the above dictionary.

```
students = {}

def student_info():
    stud_info = {}
    for i in range(2):
        name = input("Enter student name: ")
        age = int(input("Enter student age: "))
        degree = input("Enter student degree: ")
        favorite_subject = input("Enter student favorite subject: ")
        stud_info[name] = {"age":age, "degree":degree, "subject":favorite_subject}
    return stud_info
students = student_info()
print(students)

youngest_student = min(students.keys())
print("youngest student name is:",youngest_student)
```

Output:

Enter student name: M0hit

Enter student age: 20

Enter student degree: bca

Enter student favorite subject: java

Enter student name: Nilesh

Enter student age: 21

Enter student degree: bba

Enter student favorite subject: account

{'M0hit': {'age': 20, 'degree': 'bca', 'subject': 'java'}, ‘Nilesh ': {'age': 21, 'degree': 'bba', 'subject': 'account'}}

youngest student name is: Nilesh

11. Create a dictionary of students having rollno of the student is as key and value is a list of marks obtained by that student in 5 subjects.

```
student_marks = {}


for i in range(2):
    roll_number = input("Enter student's roll number: ")
    marks = []
    for j in range(2):
        subject_marks = float(input("Enter marks: "))
        marks.append(subject_marks)
    student_marks[roll_number] = marks



print("students dictionary:")
print(student_marks)
```

Output:

Enter student's roll number: 28

Enter marks: 70

Enter marks: 80

Enter student's roll number: 17

Enter marks: 77

Enter marks: 88

students dictionary:

{'28': [70.0, 80.0], '17': [77.0, 88.0]}

12. Create a dictionary from the above one, where key is rollno and value is (total of all subjects, percentage and grade) a tuple of his result.

```
student_marks = {}

for i in range(2):
    roll_number = input("Enter student's roll number: ")
    marks = []
    for j in range(2):
        subject_marks = float(input("Enter marks: "))
        marks.append(subject_marks)
    student_marks[roll_number] = marks



print("students dictionary:")
print(student_marks)

student_results = {}
```

```
for roll_number, marks in student_marks.items():
    total_marks = sum(marks)



    total_subjects = len(marks)
    percentage = (total_marks / (total_subjects * 100)) * 100

    if percentage >= 90:
        grade = 'A'
    elif percentage >= 80:
        grade = 'B'
    elif percentage >= 70:
        grade = 'C'
    elif percentage >= 60:
        grade = 'D'
    else:
        grade = 'F'
    student_results[roll_number] = (total_marks, percentage, grade)

print("student results ")
print(student_results)
```

Output:

```
Enter student's roll number: 28
Enter marks: 77
Enter marks: 88
Enter student's roll number: 17
```

```
Enter marks: 88
Enter marks: 77
students dictionary:
{'28': [77.0, 88.0], '17': [88.0, 77.0]}
student results
{'28': (165.0, 82.5, 'B'), '17': (165.0, 82.5, 'B')}
```

13. Display the rollno who has scored highest marks (total).

```
student_marks = {}

for i in range(2):
    roll_number = input("Enter student's roll number: ")
    marks = []
    for j in range(2):
        subject_marks = float(input("Enter marks: "))
        marks.append(subject_marks)
    student_marks[roll_number] = marks


print("students dictionary:")
print(student_marks)

student_results = {}

for roll_number, marks in student_marks.items():
    total_marks = sum(marks)
```

```
    total_subjects = len(marks)
    percentage = (total_marks / (total_subjects * 100)) * 100

    if percentage >= 90:
        grade = 'A'
    elif percentage >= 80:
        grade = 'B'
    elif percentage >= 70:
        grade = 'C'
    elif percentage >= 60:
        grade = 'D'
    else:
        grade = 'F'
    student_results[roll_number]={"totalmarks":total_marks,
"percentage":percentage, "grade":grade}

print("student results ")
print(student_results)

maximummarks = max(student_results.keys())
print("maximum mark has this roll number: ",maximummarks)
```

Output:

```
Enter student's roll number: 06
Enter marks: 77
Enter marks: 88
Enter student's roll number: 28
Enter marks: 77
```

Enter marks: 86

students dictionary:

{'06': [77.0, 88.0], '28': [77.0, 86.0]}

student results

{'06': {'totalmarks': 165.0, 'percentage': 82.5, 'grade': 'B'}, '28': {'totalmarks': 163.0, 'percentage': 81.5, 'grade': 'B'}}

maximum mark has this roll number: 28

14. Take user input for roll no of 5 students and their coma separated marks of 6 subjects (out of 50). Display the minimum and maximum marks of each subject in a separate dictionary object.

```
minmarks = {}
maxmarks = {}

for i in range(2):
    rollno = input("Enter roll number {}: ".format(i+1))
    marks = input("Enter marks of 6 subjects  {} : ".format(i+1))
    markslist = list(map(int, marks.split(',')))

    for j in range(6):
        sub = "Subject {}".format(j+1)
        if sub not in minmarks:
            minmarks[sub] = markslist[j]
            maxmarks[sub] = markslist[j]
        else:
            if markslist[j] < minmarks[sub]:
                minmarks[sub] = markslist[j]
            if markslist[j] > maxmarks[sub]:
```

```
            maxmarks[sub] = markslist[j]

print("\nMinimum Marks:")
for sub, minmark in minmarks.items():
   print("{}: {}".format(sub, minmark))

print("\nMaximum Marks:")
for sub, maxmark in maxmarks.items():
   print("{}: {}".format(sub, maxmark))



print(minmarks)
print(maxmarks)
```

Output:

```
Enter roll number 1: 28
Enter marks of 6 subjects  1 : 77,66,88,78,78,75
Enter roll number 2: 17
Enter marks of 6 subjects  2 : 77,88,98,97,87,78

Minimum Marks:
Subject 1: 77
Subject 2: 66
Subject 3: 88
Subject 4: 78
Subject 5: 78
Subject 6: 75
```

Maximum Marks:

Subject 1: 77

Subject 2: 88

Subject 3: 98

Subject 4: 97

Subject 5: 87

Subject 6: 78

{'Subject 1': 77, 'Subject 2': 66, 'Subject 3': 88, 'Subject 4': 78, 'Subject 5': 78, 'Subject 6': 75}

{'Subject 1': 77, 'Subject 2': 88, 'Subject 3': 98, 'Subject 4': 97, 'Subject 5': 87, 'Subject 6': 78}

15. Create a dictionary of 6 fruits (by taking user input) and store their buying and selling price as a tuple value. Display the fruit name whose buying price is less than its selling price and also the fruit name whose buying price is more than its selling price. Display the fruit that has earned no profit no loss.

```
fruits={}

for _ in range(3):
    fruitnm=input("enter fruit name ...")
    bp=int(input("enter buying price..."))
    sp=int(input("enter selling price.."))
    fruits[fruitnm]=(bp,sp)

print("\n")

print(fruits)

print("\n")
```

```
print("buying price less then selling price..")

for i,price in fruits.items():
    if price[0]<price[1]:
        print(i)

print("buying price more then selling price..")

for i,price in fruits.items():
    if price[0]>price[1]:
        print(i)

print("no profit no loss")
for i,price in fruits.items():
    if price[0]==price[1]:
        print(i)
```

Output:

```
enter fruit name ...bannana
enter buying price...40
enter selling price..50
enter fruit name ...chiku
enter buying price...40
enter selling price..60
enter fruit name ...kiwi
enter buying price...50
enter selling price..70
```

{'bannana': (40, 50), 'chiku': (40, 60), 'kiwi': (50, 70)}

buying price less then selling price..

bannana

chiku

kiwi

buying price more then selling price..

no profit no loss

Day-9

Essential Assignment:

1. **Take user input and create a menu driven program to perform mathematical operations like addition, subtraction, multiplication, division, integer division, power. Return values from the functions**

```
# Main function to print menu....
def menu():
    while True:
  print("\n Menu...")
  print("1... Addition of two number")
  print("2... Substraction of two number")
  print("3... Multiplication of two number")
  print("4... Division of two number")
  print("5... Integer Division of two number")
  print("6... Power of two number")
  print("7... Exit")

  choice = int(input("Enter the your choice (1 to 7):"))

  if choice == 7:
     print("Existing...")
     break
```

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

        if choice == 1:
            print(f"{num1} & {num2}'s Addition is :", Addition(num1, num2))

        elif choice == 2:
            print(f"{num1} & {num2}'s Substraction is :", Substraction(num1, num2))

        elif choice == 3:
            print(f"{num1} & {num2}'s Multiplication is :", Multiplication(num1, num2))

        elif choice == 4:
            print(f"{num1} & {num2}'s Division is :", Division(num1, num2))

        elif choice == 5:
            print(f"{num1} & {num2}'s Integer_Division is :", Integer_Division(num1, num2))

        elif choice == 6:
            print(f"{num1} & {num2}'s Power is :", Power(num1, num2))

        else:
            print("Invalid choice... Plase enter a number 1 to 7!")

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

*out put

```
            Menu...
1... Addition of two number
2... Substraction of two number
3... Multiplication of two number
4... Division of two number
5... Integer Division of two number
6... Power of two number
7... Exit
Enter the your choice (1 to 7):1
Enter first number :1
Enter second number :3
     1    & 3's Addition is : 4
```

2**. Create functions to calculate**

a.Area of a rectangle = width * length

b.Area of a triangle = ½ * Height * Base

c.Area of a circle = pi*r*r

```
def Rectangle(width, length):
    return width * length

def Triangle(heigth, base):
    return (heigth * base)/2

def Circle(PI, radius):
    return PI * radius *radius

while True:
    print("\n Menu")
    print("1... Area of Rectanle")
    print("2... Area of Triangle")
    print("3... Area of Circle")

    choice = int(input("Enter the your choice (1 to 3) :"))

    if choice == 1:
        width = eval(input("Enter the value of width :"))
        length = eval(input("Enter the value of length :"))
        print("Area of Rectangle is :", Rectangle(width, length))

    elif choice == 2:
        heigth = eval(input("Enter the value of heigth :"))
        base = eval(input("Enter the value of base :"))
        print("Area of Triangle is :", Triangle(heigth, base))

    elif choice == 3:
        PI = 3.13
        radius = eval(input("Enter the value of radius :"))
        print("Area of Circle is :", Circle(PI, radius))
```

*out put

```
Menu
1... Area of Rectanle
2... Area of Triangle
3... Area of Circle
Enter the your choice (1 to 3) :1
Enter the value of width :12
Enter the value of length :16
Area of Rectangle is : 192
```

3. Create functions to convert decimal numbers to binary, octal and hexadecimal numbers. Always return values from the functions

```
def binary(a):
   return bin(a)

def octal(b):
   return oct(b)

def hexadecimal(c):
   return hex(c)

while True:
   print("\n Convert")
   print("1.. Convert to binary")
   print("2.. Convert to octal")
   print("3.. Convert to hexadecimal")

   choice = int(input("Enter the choice (1 to 3) :"))

   number = eval(input("Enter the number :"))

   if choice == 1:
      print(f"{number} is convert into binary :", binary(number))

   elif choice == 2:
      print(f"{number} is convert into octal :", octal(number))

   elif choice == 3:
      print(f"{number} is convert into HexaDecimal is :", hexadecimal(number))
```

*out put

```
          Convert
1.. Convert to binary
2.. Convert to octal
3.. Convert to hexadecimal
Enter the choice (1 to 3) :2
Enter the number :11
     11  s convert into octal : 0o13
```

4. **Write an UDF to return a list having only unique values by removing duplicate values from the provided input list.**

Eg. I/P: Sample List : [1,2,3,3,3,3,4,5] O/P: Unique List : [1, 2, 3, 4, 5]

```
def duplicate(list):
   unique_list = []
```

```
    for i in list:
        if i not in unique_list:
            unique_list.append(i)
    return unique_list


list = [1,2,3,3,3,3,4,5]
unique_list = duplicate(list)


print("Unique List is :", unique_list)
```

*out put

Unique List is : [1, 2, 3, 4, 5]

5. **Write a Python function to multiply all the numbers in a list.**

```
def multiply(list):
    result = 1
    for i in list:
        result  *= i
    return result



list = [1,2,3,4,5]
print(list)
multiply_list = multiply(list)

print("Multiply all element :", multiply_list)
```

*out put

[1, 2, 3, 4, 5]

Multiply all element : 120

6. Write a UDF to check the inputted number is between specified range or not.

```
def check(n, start, end):
    return start <= n <= end

s_range = int(input("Enter the start of the range :"))
e_range = int(input("Enter the end of the range :"))
number = int(input("Enter the number to check :"))

if check(number,s_range, e_range):
    print(f"number {number} is within the range {s_range} to {e_range}")
else:
    print(f"number {number} is not within the range {s_range} to {e_range}")
```

*out put

```
Enter the start of the range :1
Enter the end of the range :90
Enter the number to check :45
number 45 is within the range 1 to 90
```

7. Write a function to calculate total number of Uppercase and lowercase characters in the string.

```
def check(str):
    lower = 0
    upper = 0

    for i in str:
        if i.islower():
            lower += 1
        elif i.isupper():
            upper += 1

    return lower,upper
```

```
str = input("Enter the string :")

lower,upper = check(str)


print("Number of lower characters :", lower)
print("Number of upper characters :", upper)
```

*out put

```
Enter the string :milan
Number of lower characters : 5
Number of upper characters : 0
```

8. Write an UDF to check if the user given number is a prime number or not. And generate prime numbers from the given range.

```
def isprime(num):
    for i in range(2,num):
        if num % 2 == 0:
            return  False
        else:
            for i in range(3, int(num**0.5) + 1, 2):
                if num % i == 0:
                    return False
        return True

def range_prime(start,end):
    prime_number = []
    for i in range(start, end + 1):
        if isprime(i):
            prime_number.append(i)
    return prime_number

num = int(input("Enter the number :"))
is_prime = isprime(num)

if is_prime:
    print(f"{num} is prime...")
else:
```

```
    print(f"{num} is not prime...")

s_range = int(input("Enter start of the range :"))
e_range = int(input("Enter end of the range :"))
prime_number = range_prime(s_range,e_range)

print("Prime number are :", prime_number)
```

*out put

```
Enter the number :10
10 is not prime...
Enter start of the range :12
Enter end of the range :45
Prime number are : [13, 17, 19, 23, 29, 31, 37, 41, 43]
```

9. Write a python program to calculate factorial values of a number.

```
def factorial(num):
    if num == 0:
        return True
    else:
        return num * factorial(num-1)

num = int(input("Enter the number :"))

if num < 0:
    print(f"{num} is negative number! not consider in factorial...")
elif num == 0:
    print("number is 0! not consider in facotrial...")
elif num > 0:
    print(f"{num} number's factorial is :", factorial(num))
```

*out put

```
Enter the number :2
2    number's factorial is : 2
```

10. **Write a program to check the data type of a given input and accordingly**

perform ‘+’ and ‘*’ operations on the input values.

```
def perform_operations(input_value):
    if isinstance(input_value, (int)):
        return input_value + input_value, input_value * input_value
    elif isinstance(input_value, str):
        return input_value + input_value, None
    else:
        return "Unsupported data type (Please enter int or str!)", None

input_value = eval(input("Enter a value: "))
addition_result, multiplication_result = perform_operations(input_value)

print("Result of addition:", addition_result)
if multiplication_result is not None:
    print("Result of multiplication:", multiplication_result)
```

*out put

```
Enter a value: 14
Result of addition: 28
Result of multiplication: 196
```

Day-10

Essential Assignment:

1..Write a findString() function to find all the positions of occurrences of string2 in string1 and return that value. If string2 is not present in string1 then display suitable message.

Eg. Str1 = Hello all, Good Morning to all. (pass it as a parameter in the function)

Str2 = ‘all’ (pass it as a parameter, but if not passed take a default argument)

o/p: String 2 found at positions: [6, 27]

```
def findString(string1, string2='all'):
    positions = []
    index = 0
    while index < len(string1):
        index = string1.find(string2, index)
        if index == -1:
            break
        positions.append(index)
        index += 1
    if positions:
        print(f"String '{string2}' found at positions: {positions}")
    else:
        print(f"String '{string2}' not found in '{string1}'")


str1 = "Hello all, Good Morning to all."
str2 = "all"
findString(str1, str2)
```

*out put

```
String 'all' found at positions: [6, 27]
```

2. **Create a list of fruits and using different functions perform the following operations: Show the use of global() , global keyword and don’t return from the functions**
    a. **Add a fruit at the last in the above list**
    b. **Insert a fruit at a particular position (pass it as an argument. If the position is not passed then take default argument as 1)**
    c. **Update the fruit (use keyword arguments)**
    d. **Remove a fruit from the list (pass an index position/ pass a name of the fruit as an argument)**

e. Arrange the fruits in an order

```
fruits = ['apple', 'banana', 'orange']

def add_fruit(fruit):
    global fruits
    fruits.append(fruit)

def insert_fruit(fruit, position=1):
    global fruits
    fruits.insert(position - 1, fruit)

def update_fruit(index, new_fruit):
    global fruits
    fruits[index] = new_fruit

def remove_fruit(identifier):
    global fruits
    if isinstance(identifier, int):
        del fruits[identifier]
    else:
        fruits.remove(identifier)

def arrange_fruits():
    global fruits
    fruits.sort()

print("Initial list of fruits:", fruits)
```

```
add_fruit('grapes')
print("After adding grapes:", fruits)

insert_fruit('kiwi', 2)
print("After inserting kiwi at position 2:", fruits)

update_fruit(1, 'pear')
print("After updating fruit at index 1 to pear:", fruits)

remove_fruit(0)
print("After removing fruit at index 0:", fruits)

remove_fruit('banana')
print("After removing banana:", fruits)

arrange_fruits()
print("After arranging fruits:", fruits)
```

*out put

Initial list of fruits: ['apple', 'banana', 'orange']

After adding grapes: ['apple', 'banana', 'orange', 'grapes']After inserting kiwi at position 2: ['apple', 'kiwi', 'banana', 'orange', 'grapes']

After updating fruit at index 1 to pear: ['apple', 'pear', 'banana', 'orange', 'grapes']

After removing fruit at index 0: ['pear', 'banana', 'orange', 'grapes']

After removing banana: ['pear', 'orange', 'grapes']

After arranging fruits: ['grapes', 'orange', 'pear']

3. **Create a function to generate prime numbers. Ask total numbers form the user and pass in the function which will return a list of prime numbers.**
    a. **Eg. GeneratePrime(10) function will return 1st 10 prime numbers starting from 2 like 2,3,5,7,11,13,15,1719,23**

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

def generate_primes(total):
    primes = []
    num = 2
    while len(primes) < total:
        if is_prime(num):
            primes.append(num)
        num += 1
    return primes

total_numbers = int(input("Enter the total number of prime numbers you want to generate: "))
prime_list = generate_primes(total_numbers)
print("Prime numbers:", prime_list)
```

*out put

```
Enter the total number of prime numbers you want to generate: 25
Prime numbers: [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]
```

4. **Create a dictionary object (for storing price and features of mobile phones) using create() function and define that dictionary as a global variable. Use that dictionary object to add element, update element, and delete an element using separate functions.**

```
def create():
    global mobiles_dict
```

```
    mobiles_dict = {}

def add_mobile(name, price, features):
    global mobiles_dict
    mobiles_dict[name] = {'price': price, 'features': features}

def update_mobile(name, price=None, features=None):
    global mobiles_dict
    if name in mobiles_dict:
        if price:
            mobiles_dict[name]['price'] = price
        if features:
            mobiles_dict[name]['features'] = features
    else:
        print(f"Mobile '{name}' not found in the dictionary.")

def delete_mobile(name):
    global mobiles_dict
    if name in mobiles_dict:
        del mobiles_dict[name]
        print(f"Mobile '{name}' deleted successfully.")
    else:
        print(f"Mobile '{name}' not found in the dictionary.")

create()

add_mobile('iPhone', 999, ['A14 Bionic chip', '5G support'])
add_mobile('Samsung Galaxy', 899, ['Quad-camera setup', '120Hz display'])
```

```
print("Initial dictionary:", mobiles_dict)

update_mobile('iPhone', features=['A15 Bionic chip', 'Improved camera'])
print("After updating iPhone:", mobiles_dict)

delete_mobile('Samsung Galaxy')
print("After deleting Samsung Galaxy:", mobiles_dict)
```

*out put

```
Initial dictionary: {'iPhone': {'price': 999, 'features': ['A14 Bionic chip', '5G support']}, 'Samsung Galaxy': {'price': 899, 'features': ['Quad-camera setup', '120Hz display']}}
After updating iPhone: {'iPhone': {'price': 999, 'features': ['A15 Bionic chip', 'Improved camera']}, 'Samsung Galaxy': {'price': 899, 'features': ['Quad-camera setup', '120Hz display']}}
Mobile 'Samsung Galaxy' deleted successfully.
After deleting Samsung Galaxy: {'iPhone': {'price': 999, 'features': ['A15 Bionic chip', 'Improved camera']}}
```

Day-11

Essential Assignment:

2. Create a lambda function that will return maximum of two numbers

```
maximum = lambda num1,num2: num1 if num1>num2 else num2
num1 = eval(input("Enter the value of number 1 :"))
num2 = eval(input("Enter the value of number 2 :"))
```

```
result = maximum(num1, num2)

print("Maximum number is :",result)
```

*out put

```
Enter the value of number 1 :1

Enter the value of number 2 :2

Maximum number is : 2
```

3. **Create a lambda function that will return maximum of three numbers**

```
maximum = lambda num1,num2,num3 : max(num1,num2,num3)

num1 = eval(input("Enter the number 1 :"))
num2 = eval(input("Enter the number 2 :"))
num3 = eval(input("Enter the number 3 :"))

result = maximum(num1,num2,num3)
print("Maximum number is :",result)
```

*out put

```
Enter the number 1 :2

Enter the number 2 :3

Enter the number 3 :4

Maximum number is : 4
```

4. **Write a lambda function that takes one number and if the number is even, returns that number multiplied by 5 else if the number is odd, returns that number multiplied by 10**

```
multi = lambda num1 : num1 * 5 if num1 % 2 == 0  else num1 * 10


num1 = eval(input("Enter the number :"))
```

```
result = multi(num1)
print(result)
```

*out put

```
Enter the number :12
60
```

5. **Take a list of mixed elements and**
   a. **Write a lambda function to separate integer elements as an output list.**
   b. **Write another lambda function to separate string elements as an output list.**

```
list = [1,2,3,4,"Milan",'Bhadani']

integer = lambda list: [element for element in list if isinstance(element, int)]
string = lambda list: [element for element in list if isinstance(element, str)]

integer_list = integer(list)
string_list = string(list)

print("Integer list is :", integer_list)
print("String list is :", string_list)
```

*out put

```
Integer list is : [1, 2, 3, 4]
String list is : ['Milan', 'Bhadani']
```

6. **Modify the above program using filter ()**

```
list1 = [1,2,3,4,"Milan",'Bhadani']

integer_list = list(filter(lambda l: isinstance(l, int), list1))
string_list = list(filter(lambda l: isinstance(l, str), list1))

print("Integer list :", integer_list)
print("String list :", string_list)
```

*out put

```
Integer list : [1, 2, 3, 4]
String list : ['Milan', 'Bhadani']
```

7. Filter all vowels from the given string.

```
string = input("Enter the string :")

vowel = lambda char : char.lower() in 'aeiou'

is_vowels = list(filter(vowel, string))

print("Vowels are :", is_vowels)
```

*out put

```
Enter the string :Milan
Vowels are : ['i', 'a']
```

8. From the provided list filter, the even numbers and odd numbers as a separate output list

```
list1 = [1,2,3,4,5,6,7,8,9,10,11,12]

integer_even_num = lambda numbers: numbers % 2 == 0
integer_odd_num = lambda numbers: numbers % 2 != 0

even_list = list(filter(integer_even_num, list1))
odd_list = list(filter(integer_odd_num, list1))

print("Even number are :", even_list)
print("Odd number are :", odd_list)
```

*out put

```
Even number are : [2, 4, 6, 8, 10, 12]
Odd number are : [1, 3, 5, 7, 9, 11]
```

9. Write a lambda function that will 2 inputs. If inputs are integers, it will return the product of 2 numbers. Else perform concatenation.

```
result = lambda input1, input2 : input1 * input2 if isinstance(input1, int) and
isinstance(input2, int) else str(input1)+ str(input2)

result1 = result(10,20)
result2 = result("Milan", 5)

print("Result 1 is :", result1)
print("Result 2 is :", result2)
```

*out put

```
Result 1 is : 200
Result 2 is : Milan5
```

Day-12

Essential Assignment:

10. Take a list of words and print all palindrome words using filter() [Hint: string slicing str1[::-1]]

```
def palindrome(words):
   return words == words[::-1]

# user input
# word = input("Enter the word :").split(",")
# print("Actually word :", word)

word = ['radar', 'level', 'python', 'madam']
```

```
palindrome_word = list(filter(palindrome,word))

print("Palindrome word is :", palindrome_word)
```

*out put

```
Palindrome word is : ['radar', 'level', 'madam']
```

11. Sort the list elements using lambda, where elements are in form of tuple. Eg. Fruits = [(120,'Apple'), (20,'Banana'),(50,'Chikoo'),(100,'Pinaple')]

```
Fruits = (120,'Apple'), (20,'Banana'),(50,'Chikoo'),(100,'Pinaple')

sort = sorted(Fruits, key = lambda f:f[0])

print(sort)
```

*out put

```
[(20, 'Banana'), (50, 'Chikoo'), (100, 'Pinaple'), (120, 'Apple')]
```

12. Take a list of students and filter the students whose name is less than 6 characters.

```
student = ['neeraj', 'raj', 'milan', 'jeel', 'monil']

filter_s = list(filter(lambda name : len(name) < 6, student ))

print(filter_s)
```

*out put

```
['raj', 'milan', 'jeel', 'monil']
```

13. Write filter() to remove empty strings from the list

```
string = ['neeraj', 'raj', 'milan','', 'jeel','', 'monil']

empty_string = list(filter(lambda empty : empty != '', string))
```

```
print(empty_string)
```

*out put

```
['neeraj', 'raj', 'milan', 'jeel', 'monil']
```

14. Write filter() to remove negative numbers from the list

```
number = [1,2,3,4,-4,-5,-25,0,45]
negative_number = list(filter(lambda n : n > 0, number))
print(negative_number)
```

*out put

```
[1, 2, 3, 4, 45]
```

15. Write filter() to remove duplicate numbers from the list (hint: list.count())

```
number = [1,2,3,4,-4,-5,-25,0,45, -4, 1,2]
filter_list = list(filter(lambda numbers: number.count(numbers) == 1, number))
print(filter_list)
```

*out put

```
[3, 4, -5, -25, 0, 45]
```

16. Sorting the dictionary elements using lambda (by using sorted () method) according to age and if age is same then sort my name
Eg. stud= [{'name': 'Amit', 'age': 25}, {'name': 'Bina', 'age': 22}, {'name': 'Dax', 'age': 25}]

```
student = [{'name': 'Naimish', 'age': 25}, {'name': 'Rohit', 'age': 22}, {'name': 'Dax', 'age':
```

```
25}]

sort_student = sorted(student, key=lambda s: (s['age'], s['name']))

for s in sort_student:
   print("Name:", s['name'], "-> Age:", s['age'])
```

*out put

```
Name: Rohit -> Age: 22
Name: Naimish -> Age: 25
Name: Dax -> Age: 25
```

Day-13

Essential Assignment:

1. **Find the average of all the elements passed as an argument in lambda (using variable length arguments)**

```
average = lambda *args: sum(args) / len(args) if len(args) > 0 else None


# Example usage:
result = average(10, 20, 30, 40, 50)
print("Average:", result)
```

*out put

Average: 30.0

2. **Take 2 lists and add the elements of it if the 1st number is greater than the other else find the difference between them**

   Eg. nums1 = [6, 5, 3, 9] nums2 = [0, 1, 7, 7]

   O/P [6, 4, 10, 2]

```
nums1 = [6, 5, 3, 9]

nums2 = [0, 1, 7, 7]


result = []


for num1, num2 in zip(nums1, nums2):
        if num1 > num2:
        result.append(num1 + num2)
 else:
  result.append(abs(num1 - num2))


print(result)
```

*out put

```
[6, 6, 4, 16]
```

3. Take a list of person names and display them all in upper case using map()

```
person_names = ['milan', 'jay', 'ashish', 'jeel']


upper_case_names = list(map(str.upper, person_names))


print(upper_case_names)
```

*out put

['MILAN', 'JAY', 'ASHISH', 'JEEL']

4. **Take a list of floating-point numbers and display list of all round numbers. Also round them with just 2 decimal points. Using map()**
   Eg. [6.56773, 9.57668, 4.00914, 56.24241, 9.01344]
   o/p [7, 10, 4, 56, 9] and [6.57, 9.58, 4.01, 56.24, 9.01]

```
numbers = [6.56773, 9.57668, 4.00914, 56.24241, 9.01344]

rounded_integers = list(map(round, numbers))
rounded_floats_2dp = list(map(lambda x: round(x, 2), numbers))

print("Rounded integers:", rounded_integers)
print("Rounded floats (2 decimal points):", rounded_floats_2dp)
```

*out put

Rounded integers: [7, 10, 4, 56, 9]

Rounded floats (2 decimal points): [6.57, 9.58, 4.01, 56.24, 9.01]

5. **Take a string as an input and display the output to analysis the string based on separate words. Using map()**
   a. **Display the words in upper case along with the length of each word**
   b. **Display total number of each vowel in each word**

   Eg. Str1 = ‘Hello how are you?’
   o/p: [{'HELLO': [{'e': 1}, {'o': 1}], 'length': 5}, {'HOW': [{'o': 1}], 'length': 3}, {'ARE': [{'a': 1}, {'e': 1}], 'length': 3}, {'YOU?': [{'o': 1}, {'u': 1}], 'length': 4}]

```
def analyze_word(word):
upper_word = word.upper()
length = len(word)
```

```
    vowel_count = {vowel: word.count(vowel) for vowel in 'aeiou'}
    return {upper_word: [{'length': length}, vowel_count]}

def analyze_string(string):
    words = string.split()
    return list(map(analyze_word, words))

input_string = 'Hello how are you?'
output = analyze_string(input_string)
print(output)
```

*out put

```
[{'HELLO': [{'length': 5}, {'a': 0, 'e': 1, 'i': 0, 'o': 1, 'u': 0}]}, {'HOW': [{'length': 3}, {'a': 0, 'e': 0, 'i': 0, 'o': 1, 'u': 0}]}, {'ARE': [{'length': 3}, {'a': 1, 'e': 1, 'i': 0, 'o': 0, 'u': 0}]}, {'YOU?': [{'length': 4}, {'a': 0, 'e': 0, 'i': 0, 'o': 1, 'u': 1}]}]
```

6. **Find the square of each element of a list (using map())**

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)
```

*out put

```
[1, 4, 9, 16, 25]
```

7. **Use a lambda function to calculate grades for a list of scores (using map())**
   Eg scores = [88, 92, 78, 95, 86] o/p: ['B', 'A', 'C', 'A', 'B']

```
scores = [88, 92, 78, 95, 86]

grades = list(map(lambda score: 'A' if score >= 90 else 'B' if score >= 80 else 'C' if score >= 70 else 'D', scores))

print(grades)
```

*out put

```
'B', 'A', 'C', 'A', 'B']
```

8. Add all the elements of the list (using reduce())

```
from functools import reduce

numbers = [1, 2, 3, 4, 5]

total = reduce(lambda x, y: x + y, numbers)

print(total)
```

*out put

```
15
```

9. Multiply all the elements of the list (using reduce())

```
from functools import reduce

numbers = [1, 2, 3, 4, 5]

product = reduce(lambda x, y: x * y, numbers)

print(product)
```

*out put

120

10. Find the maximum element from the list using reduce()

```
from functools import reduce

numbers = [12, 45, 23, 67, 89, 34]

max_number = reduce(lambda x, y: x if x > y else y, numbers)

print(max_number)
```

*out put

89

11. **Take a matrix as input and transpose its elements using lambda**

Eg. matrix = [[1, 2],[3,4],[5,6],[7,8]]
o/p: [[1, 3, 5, 7], [2, 4, 6, 8]]

```
matrix = [[1, 2], [3, 4], [5, 6], [7, 8]]

transposed_matrix = list(map(lambda x: list(x), zip(*matrix)))

print(transposed_matrix)
```

*out put

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

12. Find the factorial of a number using lambda (recursive)

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

number = 5
print("Factorial of", number, "is", factorial(number))
```

*out put

```
Factorial of 5 is 120
```

Day-14

Essential Assignment:

1. **Define a function 'sort_matrix' that takes a matrix 'M' (list of lists) as input Eg. M = [[1, 2, 3], [2, 4, 5], [1, 1, 1]] o/p [[1, 1, 1], [1, 2, 3], [2, 4, 5]]**

```
def sort_matrix(M):
    sorted_matrix = sorted(M)
    return sorted_matrix

M = [[1, 2, 3], [2, 4, 5], [1, 1, 1]]
sorted_M = sort_matrix(M)
print(sorted_M)
```

*out put

```
[[1, 1, 1], [1, 2, 3], [2, 4, 5]]
```

2. **Define a function 'extract_string' that takes a list of strings 'str_list1' and an integer 'l' as input Eg. str_list1 = ['Python', 'list', 'exercises', 'practice', 'solution'] l = 8 o/p ['practice', 'solution']**

```
def extract_string(str_list1, l):
    return [string for string in str_list1 if len(string) >= l]

str_list1 = ['Python', 'list', 'exercises', 'practice', 'solution']
l = 8
result = extract_string(str_list1, l)
print(result)
```

*out put

```
['exercises', 'practice', 'solution']
```

3. **Create a list of tuples named 'subject_marks', each tuple containing a subject and its corresponding marks Eg. subject_marks = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)] o/p [('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]**

```
subject_marks = [('English', 88), ('Science', 90), ('Maths', 97),
                 ('Social sciences', 82)]

print(subject_marks)
```

*out put

```
[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
```

4. **Create a list of dictionaries named 'models', each dictionary representing a mobile phone model with 'make', 'model', and 'color' keys Eg. models = [{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': ', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}] o/p : [{'make': 'Samsung', 'model': 17, 'color':**

'Blue'}, {'make': 'Mi Max', 'model': 20, 'color': 'Gold'}, {'make': 'Nokia', 'model': 21, 'color': 'Black'}]

```
models = [
   {'make': 'Nokia', 'model': 216, 'color': 'Black'},
   {'make': 'Mi Max', 'model': 20, 'color': 'Gold'},
   {'make': 'Samsung', 'model': 17, 'color': 'Blue'}
]

print(models)
```

*out put

```
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': 20, 'color': 'Gold'}, {'make': 'Samsung', 'model': 17, 'color': 'Blue'}]
```

5. **Create a list of mixed inputs by taking user input and write a lambda function to check the element in the list is a digit or not. [Hint: str.isdigit()] Eg. l = ['1','2','a','b','3','c','d'] o/p [True, True, False, False, True, False, False]**

```
user_input = input("Enter elements of the list separated by space: ")
mixed_list = user_input.split()

check_digit = lambda x: x.isdigit()

result = list(map(check_digit, mixed_list))

print(result)
```

*out put

```
Enter elements of the list separated by space: a
[False]
```

Day:- 15

1. Take 2 lists and add the elements of it if the 1st number is greater than the other else find the difference between them

Eg. nums1 = [6, 5, 3, 9] nums2 = [0, 1, 7, 7]
O/P [6, 4, 10, 2]

Syntax:-

```
l1 = [6, 5, 3, 9]

l2 = [0, 1, 7, 7]

l3 = []


for i in range(len(l1)):
    if l1[i] > l2[i]:
        l3.append(l1[i] - l2[i])
    else:
        l3.append(l2[i] + l1[i])


print(l3)
```

Output:-

```
[6, 4, 10, 2]
```

2. Take a list of person names and display them all in upper case using map()

Syntax:-

```
persons = ['m0hit', 'hit', 'ketan', 'maulik', 'vivek',  'khuman']

print("Before:",persons)


uppercase_persons = list(map(str.upper, persons))

print("After:",uppercase_persons)
```

Output:-

```
Before: ['m0hit', 'hit', 'ketan', 'maulik', 'vivek', 'khuman']

After: ['M0HIT', 'HIT', 'KETAN', 'MAULIK', 'VIVEK', 'KHUMAN']
```

3. Take a list of floating-point numbers and display list of all round numbers. Also round them with just 2 decimal points. Using map()

Eg. [6.56773, 9.57668, 4.00914, 56.24241, 9.01344]

o/p [7, 10, 4, 56, 9] and [6.57, 9.58, 4.01, 56.24, 9.01]

Syntax:-

```
numbers = [6.56773, 9.57668, 4.00914, 56.24241, 9.01344]


rounded_integers = list(map(lambda x: round(x), numbers))

rounded_2_decimal = list(map(lambda x: round(x, 2), numbers))


print("Rounded integer:", rounded_integers)

print("With decimal points:", rounded_2_decimal)
```

Output:-

Rounded integer: [7, 10, 4, 56, 9]

With decimal points: [6.57, 9.58, 4.01, 56.24, 9.01]

4. Take a list of words and print all palindrome numbers using filter() [Hint: string slicing str1[::-1]]

Syntax:-

```
def is_palindrome(word):
    return word == word[::-1]


words = ['m0hit', 'tenet', 'python', 'noon', 'radar', 'hello']


palindrome = list(filter(is_palindrome, words))
print("Palindrome words:", palindrome)
```

Output:-

Palindrome words: ['tenet', 'noon', 'radar']

Day:- 16

1. Take a list of students and filter the students whose name is less than 6 characters.

Syntax:-

```
students = ['M0hit', 'Dhrumil', 'Nilesh', 'Bhautik', 'Khuman', 'Hit', 'Vivek']
stud_with_shortname = list(filter(lambda x: len(x) < 6, students))
```

```
print(stud_with_shortname)
```

Output:-

```
['M0hit', 'Vivek', 'Hit']
```

2. Take a string as an input and display the output to analysis the string based on separate words. Using map()

a. Display the words in upper case along with the length of each word

b. Display total number of each vowel in each word

Eg. Str1 = 'Hello how are you?'

o/p: [{'a': 0, 'e': 1, 'length': 5}, {'a': 0, 'e': 0, 'length': 3}, {'a': 1, 'e': 1, 'length': 3}, {'a': 0, 'e': 0, 'length': 4}]

Syntax:-

```
str1 = 'Hello, Good Morning!'
def count_vowels(word):
    vowels = 'aeiou'
    return {vowel: word.lower().count(vowel) for vowel in vowels}

processed_words = map(lambda word: {'length': len(word),
**count_vowels(word)}, str1.split())

for word_info in processed_words:
    print(word_info)
```

Output:-

{'length': 6, 'a': 0, 'e': 1, 'i': 0, 'o': 1, 'u': 0}

{'length': 4, 'a': 0, 'e': 0, 'i': 0, 'o': 2, 'u': 0}

{'length': 8, 'a': 0, 'e': 0, 'i': 1, 'o': 1, 'u': 0}

3. Take a matrix as input and transpose its elements using lambda

Eg. matrix = [[1, 2],[3,4],[5,6],[7,8]]

o/p: [[1, 3, 5, 7], [2, 4, 6, 8]]

Syntax:-

```
matrix = [[1, 2], [3, 4], [5, 6], [7, 8]]
transposed_matrix = [[row[i] for row in matrix] for i in range(len(matrix[0]))]
print("Transposed matrix:", transposed_matrix)
```

Output:-

Transposed matrix: [[1, 3, 5, 7], [2, 4, 6, 8]]

Day:- 17

1. Find the factorial of a number using lambda (recursive)

Syntax:-

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

```
number = 5

print("Factorial of", number, "is:", factorial(number))
```

Output:-

```
Factorial of 5 is: 120
```

2. Create a menu driven program with user defined functions to insert update delete elements in the dictionary object of employees

Emp = {empCode:[name, age, salary, (expert areas)],.....}

Syntax:-

```
insert_employee = lambda emp_dict: emp_dict.update({input("Enter employee code: "): [input("Enter employee name: "), int(input("Enter employee age: ")), float(input("Enter employee salary: ")), input("Enter employee expert areas: ").split(',')]})


update_employee = lambda emp_dict: emp_dict.update({input("Enter employee code to update: "): [input("Enter new name: "), int(input("Enter new age: ")), float(input("Enter new salary: ")), input("Enter new expert areas: ").split(',')]})


delete_employee = lambda emp_dict: emp_dict.pop(input("Enter employee code to delete: "), None)


def main():
    emp_dict = {}
```

```
while True:
    print("\nEmployee Management System")
    print("1. Insert Employee")
    print("2. Update Employee")
    print("3. Delete Employee")
    print("4. Display All Employees")
    print("5. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        insert_employee(emp_dict)
    elif choice == 2:
        update_employee(emp_dict)
    elif choice == 3:
        delete_employee(emp_dict)
    elif choice == 4:
        print("Employee Details:")
        for emp_code, details in emp_dict.items():
            print(f"Employee Code: {emp_code}, Details: {details}")
    elif choice == 5:
        print("Exiting the program.")
        break
    else:
```

```
            print("Invalid choice!")


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

Output:-

```
Employee Management System
1. Insert Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee code: 1
Enter employee name: M0hit
Enter employee age: 22
Enter employee salary: 50000
Enter employee expert areas: Python

Employee Management System
1. Insert Employee
2. Update Employee
3. Delete Employee
```

4. Display All Employees

5. Exit

Enter your choice: 2

Enter employee code to update: 1

Enter new name: Hit

Enter new age: 23

Enter new salary: 30000

Enter new expert areas: C

3. Write a python script to generate result for a particular student.

1. Create a student data in a dictionary object as shown below:

stud = {1: {"name":'Amit', "age":23,"marks": [(10,15,12), (11,12,13)]},

2: {"name":'Bhumi',"age":22,"marks": [(13,15,11), (10,10,13)]},

3: {"name":'Bharat',"age":23,"marks": [(12,12,14), (13,14,15)]} }

NOTE: Here students are getting marks of 3 subjects in 2 attempts of test in a form of tuple

2. Create separate user defined functions to (Create a menu for options)

a. Take user input for creating entry of any new student (addStud())

b. Print all marks of a specific student. (Result(name))

c. Display overall result of all students in a given format

Syntax:-

```
stud = {
   1: {"name": 'M0hit', "age": 22, "marks": [(78, 89, 69), (80, 70, 70)]},
```

```
    2: {"name": 'Hit', "age": 25, "marks": [(67, 92, 76), (62, 76, 67)]},
    3: {"name": 'Ketan', "age": 24, "marks": [(80, 70, 56), (82, 68, 71)]}
}

addStud = lambda: stud.update({int(input("Enter student ID: ")): {"name": input("Enter student name: "), "age": int(input("Enter student age: ")), "marks": eval(input("Enter marks for two attempts (in the format [(s1, s2, s3), (s1, s2, s3)]): "))}})

Result = lambda name: print(f"Marks of {name}: {[details['marks'] for stud_id, details in stud.items() if details['name'] == name]}")

displayOverallResult = lambda: print("Overall Result:\n" + '\n'.join([f"Student ID: {stud_id}, Name: {details['name']}, Marks: {details['marks']}" for stud_id, details in stud.items()]))

def main():
    while True:
        print("\nMenu:")
        print("1. Add New Student Entry")
        print("2. Print Marks of Specific Student")
        print("3. Display Overall Result of All Students")
        print("4. Exit")
        choice = int(input("Enter your choice: "))
```

```
        if choice == 1:
            addStud()
            print("Student entry added successfully!")
        elif choice == 2:
            name = input("Enter student name: ")
            Result(name)
        elif choice == 3:
            displayOverallResult()
        elif choice == 4:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice!")

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

Output:-

```
Menu:
1. Add New Student Entry
2. Print Marks of Specific Student
3. Display Overall Result of All Students
4. Exit
```

Enter your choice: 3

Overall Result:

Student ID: 1, Name: M0hit, Marks: [(78, 89, 69), (80, 70, 70)]

Student ID: 2, Name: Hit, Marks: [(67, 92, 76), (62, 76, 67)]

Student ID: 3, Name: Ketan, Marks: [(80, 70, 56), (82, 68, 71)]

Menu:

1. Add New Student Entry

2. Print Marks of Specific Student

3. Display Overall Result of All Students

4. Exit

Enter your choice: 1

Enter student ID: 4

Enter student name: Maulik

Enter student age: 20

Enter marks for two attempts (in the format [(s1, s2, s3), (s1, s2, s3)]): (56,78,64), (79,58,65)

Student entry added successfully!

Menu:

1. Add New Student Entry

2. Print Marks of Specific Student

3. Display Overall Result of All Students

4. Exit

Enter your choice: 4

Exiting the program.

Day:- 18

1. Take a string as an input and display the output to analysis the string based on separate words. Using map()

a. Display the words in upper case along with the length of each word

b. Display total number of each vowel in each word

Eg. Str1 = 'Hello how are you?'

o/p: [{'HELLO': [{'e': 1}, {'o': 1}], 'length': 5}, {'HOW': [{'o': 1}], 'length': 3}, {'ARE': [{'a': 1}, {'e': 1}], 'length': 3}, {'YOU?': [{'o': 1}, {'u': 1}], 'length': 4}]

Syntax:-

```
Str1 = "Hello, Good Morning"

output = list(map(lambda word: {'length': len(word), **{word.upper(): [{vowel: word.lower().count(vowel)} for vowel in 'aeiou']}}, Str1.split()))

print(output)
```

Output:-

```
[{'length': 6, 'HELLO,': [{'a': 0}, {'e': 1}, {'i': 0}, {'o': 1}, {'u': 0}]}, {'length': 4, 'GOOD': [{'a': 0}, {'e': 0}, {'i': 0}, {'o': 2}, {'u': 0}]}, {'length': 7, 'MORNING': [{'a': 0}, {'e': 0}, {'i': 1}, {'o': 1}, {'u': 0}]}]
```

2. Create a list of mixed inputs by taking user input and write a lambda function to check the element in the list is a digit or not. [Hint: str.isdigit()]

Eg. l = ['1','2','a','b','3','c','d']

o/p [True, True, False, False, True, False, False]

Syntax:-

```
user_input = input("Enter elements: ")

input_list = user_input.split()

is_digit = lambda x: x.isdigit()

output = list(map(is_digit, input_list))

print(output)
```

Output:-

```
Enter elements: 1 M 0 2 H 4 I 6 3 T

[True, False, True, True, False, True, False, True, True, False]
```

3. Create a list of tuples named 'subject_marks', each tuple containing a subject and its corresponding marks

Eg. subject_marks = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
o/p [('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]

Syntax:-

```
subject_marks = [('Python', 88), ('Java', 90), ('C', 97), ('BDT', 82)]

output = sorted(subject_marks, key=lambda x: x[1])

print(output)
```

Output:-

```
[('BDT', 82), ('Python', 88), ('Java', 90), ('C', 97)]
```

4. Sorting the dictionary elements using lambda (by using sorted () method) according to age and if age is same then sort my name

Eg. stud= [{'name': 'M0hit', 'age': 22}, {'name': 'Hit', 'age': 21}, {'name': 'Sahdev', 'age': 25},{'name': 'Nilesh', 'age': 30}]

Syntax:-

```
stud = [
   {'name': 'M0hit', 'age': 22},
   {'name': 'Hit', 'age': 21},
   {'name': 'Sahdev', 'age': 25},
   {'name': 'Nilesh', 'age': 30}
]
output = sorted(stud, key=lambda x: (x['age'], x['name']))
print(output)
```

Output:-

```
[{'name': 'Hit', 'age': 21}, {'name': 'M0hit', 'age': 22}, {'name': 'Sahdev', 'age': 25}, {'name': 'Nilesh', 'age': 30}]
```

Day:- 19

1. Create a text file with different modes like w, w+, a, a+ and write few lines in it

Syntax:-

```
with open("file_w.txt", "w") as f_w:
   f_w.write("This is a text file created using write mode.\n")
   f_w.write("New content will overwrite existing content.\n")


with open("file_w_plus.txt", "w+") as f_w_plus:
```

```
    f_w_plus.write("This is a text file created using write mode with reading.\n")
    f_w_plus.write("New content will overwrite existing content.\n")
    f_w_plus.seek(0)
    print("Content of file_w_plus.txt after writing:")
    print(f_w_plus.read())

with open("file_a.txt", "a") as f_a:
    f_a.write("This is a text file created using append mode.\n")
    f_a.write("New content will be appended to existing content.\n")

with open("file_a_plus.txt", "a+") as f_a_plus:
    f_a_plus.write("This is a text file created using append mode with reading.\n")
    f_a_plus.write("New content will be appended to existing content.\n")
    f_a_plus.seek(0)
    print("Content of file_a_plus.txt after writing:")
    print(f_a_plus.read())
```

Output:-

```
Content of file_w_plus.txt after writing:
This is a text file created using write mode with reading.
New content will overwrite existing content.
```

Content of file_a_plus.txt after writing:

This is a text file created using append mode with reading.

New content will be appended to existing content.

2. Read the content of the whole file together

Syntax:-

```
with open("file_w.txt", "r") as f:
    content = f.read()
    print("Content of file_w.txt:")
    print(content)

with open("file_w_plus.txt", "r") as f:
    content = f.read()
    print("\nContent of file_w_plus.txt:")
    print(content)

with open("file_a.txt", "r") as f:
    content = f.read()
    print("\nContent of file_a.txt:")
    print(content)

with open("file_a_plus.txt", "r") as f:
```

```
    content = f.read()
    print("\nContent of file_a_plus.txt:")
    print(content)
```

Output:-

```
Content of file_a.txt:
This is a text file created using append mode.
New content will be appended to existing content.
This is a text file created using append mode.
New content will be appended to existing content.
```

3. Print the length of the file data

Syntax:-

```
with open("file_w.txt", "r") as f:
    content = f.read()
    file_length = len(content)
    print("Length of the data in file_w.txt:", file_length)

with open("file_w_plus.txt", "r") as f:
    content = f.read()
    file_length = len(content)
    print("\nLength of the data in file_w_plus.txt:", file_length)
```

```
with open("file_a.txt", "r") as f:
    content = f.read()
    file_length = len(content)
    print("\nLength of the data in file_a.txt:", file_length)

with open("file_a_plus.txt", "r") as f:
    content = f.read()
    file_length = len(content)
    print("\nLength of the data in file_a_plus.txt:", file_length)
```

Output:-

```
Length of the data in file_w.txt: 91
Length of the data in file_w_plus.txt: 104
Length of the data in file_a.txt: 194
Length of the data in file_a_plus.txt: 220
```

4. Read the file content line by line

Syntax:-

```
with open("file_w.txt", "r") as f:
    print("Content of file_w.txt:")
    for line in f:
        print(line.strip())
```

```
with open("file_w_plus.txt", "r") as f:
    print("\nContent of file_w_plus.txt:")
    for line in f:
        print(line.strip())

with open("file_a.txt", "r") as f:
    print("\nContent of file_a.txt:")
    for line in f:
        print(line.strip())

with open("file_a_plus.txt", "r") as f:
    print("\nContent of file_a_plus.txt:")
    for line in f:
        print(line.strip())
```

Output:-

```
Content of file_w.txt:
This is a text file created using write mode.
New content will overwrite existing content.

Content of file_w_plus.txt:
This is a text file created using write mode with reading.
New content will overwrite existing content.
```

Content of file_a.txt:

This is a text file created using append mode.

New content will be appended to existing content.

This is a text file created using append mode.

New content will be appended to existing content.

Content of file_a_plus.txt:

This is a text file created using append mode with reading.

New content will be appended to existing content.

This is a text file created using append mode with reading.

New content will be appended to existing content.

5. Print total number of words in each line in the file

Syntax:-

```
with open("file_w.txt", "r") as f:
    print("Total number of words in each line of file_w.txt:")
    for line in f:
        words = line.split()
        num_words = len(words)
        print(f"Line: {line.strip()}, Number of words: {num_words}")
```

```
with open("file_w_plus.txt", "r") as f:
    print("\nTotal number of words in each line of file_w_plus.txt:")
    for line in f:
        words = line.split()
        num_words = len(words)
        print(f"Line: {line.strip()}, Number of words: {num_words}")

with open("file_a.txt", "r") as f:
    print("\nTotal number of words in each line of file_a.txt:")
    for line in f:
        words = line.split()
        num_words = len(words)
        print(f"Line: {line.strip()}, Number of words: {num_words}")

with open("file_a_plus.txt", "r") as f:
    print("\nTotal number of words in each line of file_a_plus.txt:")
    for line in f:
        words = line.split()
        num_words = len(words)
        print(f"Line: {line.strip()}, Number of words: {num_words}")
```

Output:-

```
Total number of words in each line of file_w.txt:
```

Line: This is a text file created using write mode., Number of words: 9

Line: New content will overwrite existing content., Number of words: 6

Total number of words in each line of file_w_plus.txt:

Line: This is a text file created using write mode with reading., Number of words: 11

Line: New content will overwrite existing content., Number of words: 6

Total number of words in each line of file_a.txt:

Line: This is a text file created using append mode., Number of words: 9

Line: New content will be appended to existing content., Number of words: 8

Line: This is a text file created using append mode., Number of words: 9

Line: New content will be appended to existing content., Number of words: 8

Total number of words in each line of file_a_plus.txt:

Line: This is a text file created using append mode with reading., Number of words: 11

Line: New content will be appended to existing content., Number of words: 8

Line: This is a text file created using append mode with reading., Number of words: 11

Line: New content will be appended to existing content., Number of words: 8

6. Print all the words in reverse.

Syntax:-

```
with open("file_w.txt", "r") as f:
    print("All the words in reverse in each line of file_w.txt:")
    for line in f:
        words = line.split()
        reversed_words = [word[::-1] for word in words]
        reversed_line = " ".join(reversed_words)
        print(reversed_line)

with open("file_w_plus.txt", "r") as f:
    print("\nAll the words in reverse in each line of file_w_plus.txt:")
    for line in f:
        words = line.split()
        reversed_words = [word[::-1] for word in words]
        reversed_line = " ".join(reversed_words)
        print(reversed_line)

with open("file_a.txt", "r") as f:
    print("\nAll the words in reverse in each line of file_a.txt:")
    for line in f:
        words = line.split()
        reversed_words = [word[::-1] for word in words]
```

```
        reversed_line = " ".join(reversed_words)
        print(reversed_line)

with open("file_a_plus.txt", "r") as f:
    print("\nAll the words in reverse in each line of file_a_plus.txt:")
    for line in f:
        words = line.split()
        reversed_words = [word[::-1] for word in words]
        reversed_line = " ".join(reversed_words)
        print(reversed_line)
```

Output:-

```
Total number of words in each line of file_w.txt:
Line: This is a text file created using write mode., Number of words: 9
Line: New content will overwrite existing content., Number of words: 6

Total number of words in each line of file_w_plus.txt:
Line: This is a text file created using write mode with reading., Number of words: 11
Line: New content will overwrite existing content., Number of words: 6

Total number of words in each line of file_a.txt:
Line: This is a text file created using append mode., Number of words: 9
```

Line: New content will be appended to existing content., Number of words: 8

Line: This is a text file created using append mode., Number of words: 9

Line: New content will be appended to existing content., Number of words: 8

Total number of words in each line of file_a_plus.txt:

Line: This is a text file created using append mode with reading., Number of words: 11

Line: New content will be appended to existing content., Number of words: 8

Line: This is a text file created using append mode with reading., Number of words: 11

Line: New content will be appended to existing content., Number of words: 8

Day:- 20

1. Write multiple lines in a text file. Using list object

Syntax:-

```
lines = [
    "This is the first line.",
    "This is the second line.",
    "This is the third line.",
    "And this is the fourth line."
]
with open("output.txt", "w") as file:
```

```
    for line in lines:
        file.write(line + "\n")
```

Output:- (in File)

```
This is the first line.
This is the second line.
This is the third line.
And this is the fourth line.
```

2. Take a filename from the user to read that file

Syntax:-

```
filename = input("Enter the filename to read: ")
with open(filename, "r") as file:
    content = file.read()
    print("Content of the file:")
    print(content)
```

Output:-

```
Enter the filename to read: output.txt
Content of the file:
This is the first line.
This is the second line.
This is the third line.
And this is the fourth line.
```

3. If the file to be read is not available then print suitable message

Syntax:-

```
filename = input("Enter the filename to read: ")


try:
        with open(filename, "r") as file:
            content = file.read()
            print("Content of the file:")
            print(content)
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
    Enter the filename to read: output
    The file 'output' is not available. Please enter a valid filename.
```

4. After reading the file content, append the text at the end of the file.

Syntax:-

```
    filename = input("Enter the filename to read: ")
    try:
        with open(filename, "r+") as file:
            content = file.read()
```

```
        print("Content of the file:")
        print(content)
        file.seek(0, 2)
        file.write("\nThis text is appended to the end of the file.")
        print("Text appended successfully.")
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Enter the filename to read: output.txt
Content of the file:
This is the first line.
This is the second line.
This is the third line.
And this is the fourth line.


Text appended successfully.
```

5. Open a file and append a line at the beginning of the file content

Syntax:-

```
filename = input("Enter the filename to open: ")


try:
```

```
        with open(filename, "r+") as file:
            content = file.read()
            print("Content of the file before appending:")
            print(content)

            file.seek(0, 0)
            file.write("This line is appended at the beginning of the file.\n" + content)
            print("Line appended at the beginning successfully.")
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Enter the filename to open: output.txt
Content of the file before appending:
This is the first line.
This is the second line.
This is the third line.
And this is the fourth line.

This text is appended to the end of the file.
Line appended at the beginning successfully.
```

6. Copy the content of one file to another

Syntax:-

```
source_filename = input("Enter the source filename: ")
destination_filename = input("Enter the destination filename: ")

try:
    with open(source_filename, "r") as source_file:
        content = source_file.read()

    with open(destination_filename, "w") as destination_file:
        destination_file.write(content)

    print("Content copied from", source_filename, "to", destination_filename, "successfully.")
except FileNotFoundError:
    print("One or both of the specified files are not available. Please enter valid filenames.")
```

Output:-

```
Enter the source filename: output.txt
Enter the destination filename: output2.txt
Content copied from output.txt to output2.txt successfully.
```

7. Read 10th to 15th byte from the file and print.

Syntax:-

```
filename = input("Enter the filename to read: ")


try:
    with open(filename, "rb") as file:
        file.seek(9)
        content = file.read(6)
        print("Content from 10th to 15th byte:", content)
except FileNotFoundError:
    print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Enter the filename to read: output.txt
Content from 10th to 15th byte: b' is ap'
```

8. Read an existing file and take a user input string to be appended in that file. Also ask the position where new line need to be appended. Update the file content and print the updated file. [Hint: Make a file with new line character after each line]

Syntax:-

```
filename = input("Enter the filename to update: ")
try:
    with open(filename, "r+") as file:
        content = file.read()
        print("Current content of the file:")
```

```
        print(content)


        new_content = input("Enter the string to append: ")
        position = int(input("Enter the position where new content should be
appended (0-indexed): "))
        file.seek(position)
        file.write("\n" + new_content)
        file.seek(0, 2)


        updated_content = file.read()
        print("Updated content of the file:")
        print(updated_content)
    except FileNotFoundError:
       print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Enter the filename to update: output.txt
Current content of the file:
This line is appended at the beginning of the file.
This is the first line.
This is the second line.
This is the third line.
And this is the fourth line.
```

This text is appended to the end of the file.

Enter the string to append: This is the fifth line

Enter the position where new content should be appended (0-indexed): 4

Updated content of the file:

9. Read an alternate bytes/ character from the file.

Syntax:-

```
filename = input("Enter the filename to read: ")
try:
    with open(filename, "rb") as file:
        content = file.read()
        alternate_content = content[::2]

        print("Alternate bytes/characters from the file:")
        print(alternate_content.decode())
except FileNotFoundError:
    print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

Enter the filename to read: output.txt

Alternate bytes/characters from the file:

Ti

Ti steffhln einn ftefl.

hsi h is ie

Ti stescn ie

Ti stetidln.

n hsi h orhln.

hstx sapne oteedo h ie

10. Read alternate lines from the file.

Syntax:-

```
filename = input("Enter the filename to read: ")


try:
    with open(filename, "r") as file:
        lines = file.readlines()


        alternate_lines = lines[::2]


        print("Alternate lines from the file:")
        for line in alternate_lines:
            print(line.strip())
except FileNotFoundError:
```

```
        print(f"The file '{filename}' is not available. Please enter a valid filename.");
```

Output:-

```
Enter the filename to read: output.txt

Alternate lines from the file:

This

This is the first line.

This is the third line.
```

Day:- 21

1. Write a python script to read the text file content and print the output in form of line wise total words in the file.

File Content as below:

Hello, How are you?

Very good morning

Have a nice day to all

Good Bye...

Output: [(1,4), (2,3), (3,6), (4,2)]

Syntax:-

```
filename = "output.txt"

try:

    with open(filename, "r") as file:

        lines = file.readlines()
```

```
        line_word_counts = []

        for i, line in enumerate(lines, start=1):
            words = line.split()
            word_count = len(words)
            line_word_counts.append((i, word_count))

        print("Output:")
        print(line_word_counts)
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Output:
[(1, 4), (2, 3), (3, 6), (4, 2)]
```

2. Open a text file using with statement and write and read the content from that file.

Syntax:-

```
filename = "output.txt"
content_to_write = "This is some text that we want to write to the file."

try:
```

```
        with open(filename, "w") as file:
            file.write(content_to_write)
            print("Content written to the file successfully.")

        with open(filename, "r") as file:
            content_read = file.read()
            print("Content read from the file:")
            print(content_read)
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Content written to the file successfully.
Content read from the file:
This is some text that we want to write to the file.
```

3. Take the user input for data to be written in the text file. Enter the data line by line, till '@' character is entered by the user at the end.

Syntax:-

```
filename = "output.txt"

try:
    with open(filename, "w") as file:
```

```
        print("Enter the data to be written in the text file (enter '@' to end):")
        while True:
            line = input()
            if line.strip() == "@":
                break
            file.write(line + "\n")

        print("Data has been written to the file successfully.")
    except FileNotFoundError:
        print(f"The file '{filename}' is not available. Please enter a valid filename.")
```

Output:-

```
Enter the data to be written in the text file (enter '@' to end):
output.txt
@
Data has been written to the file successfully.
```

4. Create a text file having string and numeric data. Write a script to separate the string and numbers in two different files [Hint: ch.isdigit() method will return true , if character is a number]

Syntax:-

```
filename = "mixed_data.txt"

with open(filename, "w") as file:
```

```
    file.write("M0hit1 123\n")
    file.write("456 Hit2\n")
    file.write("Maulik3 789\n")
    file.write("1234 Ketan4\n")

string_filename = "string_data.txt"
numeric_filename = "numeric_data.txt"

try:
    with open(filename, "r") as file:
        with open(string_filename, "w") as string_file, open(numeric_filename, "w") as numeric_file:
            for line in file:
                words = line.split()
                for word in words:
                    if word.isdigit():
                        numeric_file.write(word + "\n")
                    else:
                        string_file.write(word + "\n")
    print("Separation of string and numeric data is complete.")
except FileNotFoundError:
    print(f"The file '{filename}' is not available. Please create the file first.")
```

Output:-

Separation of string and numeric data is complete.

M0hit1 123

456 Hit2

Maulik3 789

1234 Ketan4

5. Create a menu driven program to perform various file operations through python functions as:

a) Create a file – (define the filename as a default argument)

b) Read the content of a specified file – Return the content in a string

c) Append the content in the specified file

d) Rename a file – (Take filename as keyword arguments)

e) Delete a file - (define the filename as a default argument)

f) Create a directory / folder

g) Display all the files present in the specified folder

h) Display only .txt file names from the specified folder

i) Display the files, starting with letter 't' in their filename.

j) Display all python files from a specified folder.(Either .py extension or filename/folder name contains 'py' in between)

k) Display the file names having .txt extension

Syntax:-

```
import os


def create_file(filename="example.txt"):
    try:
        with open(filename, "w") as file:
            print(f"File '{filename}' created successfully.")
    except IOError:
        print(f"Error: Unable to create file '{filename}'.")


def read_file(filename):
    try:
        with open(filename, "r") as file:
            content = file.read()
            return content
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return None


def append_file(filename, content):
    try:
        with open(filename, "a") as file:
            file.write(content)
```

```
            print(f"Content appended to file '{filename}' successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def rename_file(old_filename, new_filename):
    try:
        os.rename(old_filename, new_filename)
        print(f"File '{old_filename}' renamed to '{new_filename}' successfully.")
    except FileNotFoundError:
        print(f"Error: File '{old_filename}' not found.")

def delete_file(filename="example.txt"):
    try:
        os.remove(filename)
        print(f"File '{filename}' deleted successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def create_folder(folder_name):
    try:
        os.mkdir(folder_name)
        print(f"Folder '{folder_name}' created successfully.")
```

```
    except FileExistsError:
        print(f"Error: Folder '{folder_name}' already exists.")

def list_files(folder="."):
    files = os.listdir(folder)
    print("Files in the specified folder:")
    for file in files:
        print(file)

def list_txt_files(folder="."):
    files = os.listdir(folder)
    txt_files = [file for file in files if file.endswith(".txt")]
    print("Text files in the specified folder:")
    for txt_file in txt_files:
        print(txt_file)

def list_files_starting_with_t(folder="."):
    files = os.listdir(folder)
    t_files = [file for file in files if file.startswith("t")]
    print("Files starting with 't' in the specified folder:")
    for t_file in t_files:
        print(t_file)
```

```
def list_python_files(folder="."):
    files = os.listdir(folder)
    python_files = [file for file in files if file.endswith(".py") or "py" in file]
    print("Python files in the specified folder:")
    for python_file in python_files:
        print(python_file)


def list_txt_files_extension(folder="."):
    files = os.listdir(folder)
    txt_files = [file for file in files if file.endswith(".txt")]
    print("Text files with .txt extension in the specified folder:")
    for txt_file in txt_files:
        print(txt_file)


def main():
    while True:
        print("\nFile Operations Menu:")
        print("a) Create a file")
        print("b) Read the content of a specified file")
        print("c) Append content in a specified file")
        print("d) Rename a file")
```

```
        print("e) Delete a file")
        print("f) Create a directory / folder")
        print("g) Display all files present in the specified folder")
        print("h) Display only .txt file names from the specified folder")
        print("i) Display files starting with letter 't' in their filename")
        print("j) Display all python files from a specified folder")
        print("k) Display file names having .txt extension")
        print("q) Quit")

        choice = input("Enter your choice: ").lower()

        if choice == "a":
            filename = input("Enter the filename to create (default is example.txt):
")
            create_file(filename)
        elif choice == "b":
            filename = input("Enter the filename to read: ")
            content = read_file(filename)
            if content:
                print("Content of the file:")
                print(content)
        elif choice == "c":
```

```
            filename = input("Enter the filename to append content to: ")
            content = input("Enter the content to append: ")
            append_file(filename, content)
        elif choice == "d":
            old_filename = input("Enter the old filename: ")
            new_filename = input("Enter the new filename: ")
            rename_file(old_filename, new_filename)
        elif choice == "e":
            filename = input("Enter the filename to delete: ")
            delete_file(filename)
        elif choice == "f":
            folder_name = input("Enter the folder name to create: ")
            create_folder(folder_name)
        elif choice == "g":
            folder = input("Enter the folder to list files from (default is current
directory): ")
            list_files(folder)
        elif choice == "h":
            folder = input("Enter the folder to list .txt files from (default is current
directory): ")
            list_txt_files(folder)
        elif choice == "i":
```

```
            folder = input("Enter the folder to list files starting with 't' from (default is current directory): ")
            list_files_starting_with_t(folder)
        elif choice == "j":
            folder = input("Enter the folder to list python files from (default is current directory): ")
            list_python_files(folder)
        elif choice == "k":
            folder = input("Enter the folder to list .txt files with .txt extension from (default is current directory): ")
            list_txt_files_extension(folder)
        elif choice == "q":
            print("Exiting the program...")
            break
        else:
            print("Invalid choice. Please try again.")

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

Output:-

```
File Operations Menu:
a) Create a file
b) Read the content of a specified file
```

c) Append content in a specified file

d) Rename a file

e) Delete a file

f) Create a directory / folder

g) Display all files present in the specified folder

h) Display only .txt file names from the specified folder

i) Display files starting with letter 't' in their filename

j) Display all python files from a specified folder

k) Display file names having .txt extension

q) Quit

Enter your choice:

Day:- 22

1. Write a Python script to take a user input to enter a list of elements for employee data

and write it into a file. Handle the following exceptions for it.

a. If an age entered is not a number

b. If salary is not defined and trying to append in the list

c. Entered age must be between 18 and 25 only

d. Salary must be greater than or equal to 10000

e. Calculate HRA and check for the ZeroDivisionError

f. Open a file in read mode and try to write into it

g. Try to write the whole list object into a file

Syntax:-

```
def calculate_hra(salary):
    try:
        hra = salary * 0.1
        return hra
    except ZeroDivisionError:
        print("Error: Salary cannot be zero.")


def write_employee_data(filename, employee_data):
    try:
        with open(filename, "w") as file:
            for employee in employee_data:
                file.write(str(employee) + "\n")
        print(f"Employee data written to '{filename}' successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except IOError:
        print(f"Error: Unable to write to file '{filename}'.")


employee_data = []
```

```
while True:
    try:
        name = input("Enter employee name (press 'q' to quit): ")
        if name.lower() == 'q':
            break
        age = int(input("Enter employee age: "))
        if not 18 <= age <= 25:
            raise ValueError("Age must be between 18 and 25.")
        salary = float(input("Enter employee salary: "))
        if salary < 10000:
            raise ValueError("Salary must be greater than or equal to 10000.")
        hra = calculate_hra(salary)
        employee = {"Name": name, "Age": age, "Salary": salary, "HRA": hra}
        employee_data.append(employee)
    except ValueError as ve:
        print(f"ValueError: {ve}")
    except ZeroDivisionError:
        print("ZeroDivisionError: Salary cannot be zero.")
    except KeyboardInterrupt:
        print("\nOperation aborted by the user.")
        break
```

```
filename = input("Enter filename to save employee data: ")
write_employee_data(filename, employee_data)
```

Output:-

```
File Operations Menu:
a) Create a file
b) Read the content of a specified file
c) Append content in a specified file
d) Rename a file
e) Delete a file
f) Create a directory / folder
g) Display all files present in the specified folder
h) Display only .txt file names from the specified folder
i) Display files starting with letter 't' in their filename
j) Display all python files from a specified folder
k) Display file names having .txt extension
q) Quit
Enter your choice:
```

Day:- 23

1. Write a Python script to take a user input to enter a list of elements for student result data

{1: (23,34,43,42,26), 2: (25,35,45,43,30), ….}And write it into a file. Handle the exceptions if any.

a. Create a user defined exception if entered marks are > 50

b. Take marks (coma separated) of 5 subjects from user. If marks are for more or less than 5 then raise a user defined exception.

c. Make a total of all subjects and calculate %

d. Write a UDF to search data (based on rollno) from the binary file. And print the student’ s result in proper format

Syntax:-

```
import pickle


class MarksError(Exception):
    pass


class SubjectCountError(Exception):
    pass


def calculate_result(marks):
    total_marks = sum(marks)
    percentage = (total_marks / (len(marks) * 50)) * 100
    return total_marks, percentage


def enter_marks(roll_no):
```

```
    try:
        marks_str = input(f"Enter marks for student {roll_no} (comma-separated):
")
        marks = list(map(int, marks_str.split(',')))
        if len(marks) != 5:
            raise SubjectCountError("Exactly 5 subjects must be entered.")
        for mark in marks:
            if mark > 50:
                raise MarksError("Marks cannot be greater than 50.")
        return marks
    except ValueError:
        print("Error: Please enter numeric marks.")
    except MarksError as me:
        print(f"Error: {me}")
    except SubjectCountError as sce:
        print(f"Error: {sce}")
    return None


def write_student_result(filename, result_data):
    try:
        with open(filename, "wb") as file:
            pickle.dump(result_data, file)
```

```
        print(f"Student result data written to '{filename}' successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except IOError:
        print(f"Error: Unable to write to file '{filename}'.")


def search_student_result(filename, roll_no):
    try:
        with open(filename, "rb") as file:
            result_data = pickle.load(file)
            if roll_no in result_data:
                marks = result_data[roll_no]
                total_marks, percentage = calculate_result(marks)
                print(f"Student Result (Roll No. {roll_no}):")
                print(f"Marks: {marks}")
                print(f"Total Marks: {total_marks}")
                print(f"Percentage: {percentage:.2f}%")
            else:
                print(f"Student with Roll No. {roll_no} not found in the records.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except IOError:
```

```
        print(f"Error: Unable to read file '{filename}'.")
    except pickle.UnpicklingError:
        print(f"Error: Unable to load data from file '{filename}'.")


def main():
    try:
        result_data = {}
        while True:
            roll_no = int(input("Enter student roll number (press 0 to stop): "))
            if roll_no == 0:
                break
            marks = enter_marks(roll_no)
            if marks:
                result_data[roll_no] = marks

        filename = input("Enter filename to save student result data: ")
        write_student_result(filename, result_data)

        search_roll_no = int(input("Enter roll number to search for student result:
"))
        search_student_result(filename, search_roll_no)
```

```
    except KeyboardInterrupt:
        print("\nOperation aborted by the user.")
    except ValueError:
        print("Error: Invalid input for roll number.")


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

Output:-

```
Enter student roll number (press 0 to stop): 1
Enter marks for student 1 (comma-separated): 40, 30, 36, 48, 42
Enter student roll number (press 0 to stop): 2
Enter marks for student 2 (comma-separated): 40, 30, 36, 48, 52
Error: Marks cannot be greater than 50.
Enter student roll number (press 0 to stop): 0
Enter filename to save student result data: Results.txt
Student result data written to 'Results.txt' successfully.
```

2. Create a dictionary object for student details (Rollno, name, age, hobby, marks…)

Syntax:-

```
student_details = {
    1: {"Name": "M0hit", "Age": 20, "Hobby": "Reading", "Marks": [85, 90, 75, 80]},
```

```
    2: {"Name": "Kunj", "Age": 21, "Hobby": "Music", "Marks": [70, 75, 80, 85]},
    3: {"Name": "Sahdev", "Age": 19, "Hobby": "Sports", "Marks": [65, 70, 75, 80]},
}
```

3. Create a user defined exception if entered marks are > 50

Syntax:-

```
class MarksError(Exception):
    pass


def enter_marks():
    try:
        marks_str = input("Enter marks (comma-separated): ")
        marks = list(map(int, marks_str.split(',')))
        for mark in marks:
            if mark > 50:
                raise MarksError("Marks cannot be greater than 50.")
        return marks
    except ValueError:
        print("Error: Please enter numeric marks.")
    except MarksError as me:
        print(f"Error: {me}")
```

```
    return None


marks = enter_marks()
print("Entered marks:", marks)
```

Output:-

```
Enter marks (comma-separated): 48, 40, 38, 34
Entered marks: [48, 40, 38, 34]
```

4. Store all those dictionary data in a binary file.

Syntax:-

```
import pickle


student_details = {
    1: {"Rollno": 1, "Name": "Mohit", "Age": 20, "Hobby": "Reading", "Marks": [85, 90, 75, 80]},
    2: {"Rollno": 2, "Name": "Hit", "Age": 21, "Hobby": "Music", "Marks": [70, 75, 80, 85]},
    3: {"Rollno": 3, "Name": "Maulik", "Age": 26, "Hobby": "Sports", "Marks": [70, 75, 80, 85]},
    4: {"Rollno": 4, "Name": "Nilesh", "Age": 31, "Hobby": "Dancing", "Marks": [70, 75, 80, 85]},
    5: {"Rollno": 5, "Name": "Sahdev", "Age": 25, "Hobby": "Gaming", "Marks": [70, 75, 80, 85]},
```

```
    6: {"Rollno": 6, "Name": "Kunj", "Age": 35, "Hobby": "Dancing", "Marks": [70, 75, 80, 85]},
}



filename = "student_data.bin"

with open(filename, "wb") as file:
    pickle.dump(student_details, file)

print("Student details stored in binary file successfully.")
```

Output:-

```
Student details stored in binary file successfully.
```

5. Create separate functions for addData, updateData, deleteData from the binary file

Syntax:-

```
import pickle

def add_data(filename, student_data):
    try:
        with open(filename, "ab") as file:
            pickle.dump(student_data, file)
```

```
        print("Student data added successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except IOError:
        print(f"Error: Unable to write to file '{filename}'.")


def update_data(filename, roll_no, new_data):
    try:
        temp_filename = "temp_file.bin"
        with open(filename, "rb") as file, open(temp_filename, "wb") as temp_file:
            while True:
                try:
                    student_data = pickle.load(file)
                    if "Rollno" in student_data and student_data["Rollno"] == roll_no:
                        student_data.update(new_data)
                    pickle.dump(student_data, temp_file)
                except EOFError:
                    break
        import os
        os.remove(filename)
        os.rename(temp_filename, filename)
        print("Student data updated successfully.")
```

```
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except IOError:
        print(f"Error: Unable to write to file '{filename}'.")


def delete_data(filename, roll_no):
    try:
        temp_filename = "temp_file.bin"
        with open(filename, "rb") as file, open(temp_filename, "wb") as temp_file:
            while True:
                try:
                    student_data = pickle.load(file)
                    if "Rollno" in student_data and student_data["Rollno"] != roll_no:
                        pickle.dump(student_data, temp_file)
                except EOFError:
                    break
        import os
        os.remove(filename)
        os.rename(temp_filename, filename)
        print("Student data deleted successfully.")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
```

```
    except IOError:
        print(f"Error: Unable to write to file '{filename}'.")


student_data = {
    "Rollno": 1,
    "Name": "M0hit",
    "Age": 22,
    "Hobby": "Reading",
    "Marks": [85, 90, 75, 80]
}


filename = "student_data.bin"
add_data(filename, student_data)


update_data(filename, 1, {"Age": 21})


delete_data(filename, 1)
```

Output:-

```
Student data added successfully.
Student data updated successfully.
Student data deleted successfully.
```

6. Write a program for Banking Application having following features (User defined functions)

a. Add Customers (While opening an account, balance must be minimum 5000)

b. Print Statement (Show the Account#, Name and Current Balance)

c. Withdraw Money (It must be +ve integer value only)

d. Deposit Money (It must be +ve integer value only)

Syntax:-

```
class BankAccount:
    def __init__(self):
        self.customers = {}


    def add_customer(self, account_no, name, initial_balance):
        if initial_balance < 5000:
            print("Initial balance must be at least 5000.")
            return False
        if account_no in self.customers:
            print("Account number already exists.")
            return False
        self.customers[account_no] = {"Name": name, "Balance": initial_balance}
        print("Customer added successfully.")
        return True
```

```
    def print_statement(self, account_no):
        if account_no in self.customers:
            print(f"Account#: {account_no}, Name:
{self.customers[account_no]['Name']}, Balance:
{self.customers[account_no]['Balance']}")
        else:
            print("Account not found.")

    def withdraw_money(self, account_no, amount):
        if account_no in self.customers:
            if amount > 0 and amount <= self.customers[account_no]["Balance"]:
                self.customers[account_no]["Balance"] -= amount
                print("Amount withdrawn successfully.")
            else:
                print("Invalid amount or insufficient balance.")
        else:
            print("Account not found.")

    def deposit_money(self, account_no, amount):
        if account_no in self.customers:
            if amount > 0:
                self.customers[account_no]["Balance"] += amount
                print("Amount deposited successfully.")
```

```
            else:
                print("Invalid amount.")
        else:
            print("Account not found.")

bank = BankAccount()

bank.add_customer(101, "M0hit", 6000)
bank.add_customer(102, "Hit", 5020)
bank.add_customer(103, "Nilesh", 1220)
bank.add_customer(104, "Dhrumil", 500)
bank.add_customer(105, "Vivek", 5300)

bank.print_statement(101)
bank.print_statement(102)
bank.print_statement(103)
bank.print_statement(104)
bank.print_statement(105)

bank.withdraw_money(101, 2000)
bank.withdraw_money(102, 5000)
bank.withdraw_money(103, 3000)
```

```
bank.withdraw_money(104, 4620)

bank.withdraw_money(105, 500)


bank.deposit_money(101, 3000)

bank.deposit_money(102, -1000)

bank.deposit_money(103, -2000)

bank.deposit_money(104, 2000)

bank.deposit_money(105, 5000)
```

Output:-

```
Customer added successfully.

Customer added successfully.

Initial balance must be at least 5000.

Initial balance must be at least 5000.

Customer added successfully.

Account#: 101, Name: M0hit, Balance: 6000

Account#: 102, Name: Hit, Balance: 5020

Account not found.

Account not found.

Account#: 105, Name: Vivek, Balance: 5300

Amount withdrawn successfully.

Amount withdrawn successfully.

Account not found.
```

Account not found.

Amount withdrawn successfully.

Amount deposited successfully.

Invalid amount.

Account not found.

Account not found.

Amount deposited successfully.

Day:- 24

1. Create a file for logging/ storing messages, when exception occurs

Syntax:-

```
import logging


logging.basicConfig(filename='error.log', level=logging.ERROR,
format='%(asctime)s - %(levelname)s - %(message)s')


try:
    x = 1 / 0
except Exception as e:
    logging.error(f'An exception occurred: {e}')
    print(f'An exception occurred: {e}')
```

Output:-

An exception occurred: division by zero

2. Print today’ s date and time using datetime module, also print year,month and day.

Syntax:-

```
import datetime


current_datetime = datetime.datetime.now()


print("Current date and time:", current_datetime)


print("Year:", current_datetime.year)
print("Month:", current_datetime.month)
print("Day:", current_datetime.day)
```

Output:-

Current date and time: 2024-04-25 19:11:56.679890

Year: 2024

Month: 4

Day: 25

3. Use different formatting options to print weekdays

Syntax:-

```
import datetime


current_date = datetime.datetime.now()


print("Weekday abbreviated:", current_date.strftime("%a"))
print("Weekday full name:", current_date.strftime("%A"))
print("Weekday as a number (0-6, 0 is Monday):", current_date.strftime("%w"))
```

Output:-

```
Weekday abbreviated: Thu
Weekday full name: Thursday
Weekday as a number (0-6, 0 is Monday): 4
```

4. Create a Database in MySQL – name MyDB

Syntax:-

```
import mysql.connector


conn = mysql.connector.connect(
  host="localhost",
  user="your_username",
  password="your_password"
)
```

```
cursor = conn.cursor()


cursor.execute("CREATE DATABASE MyDB")


cursor.close()
conn.close()


print("Database MyDB created successfully.")
```

Output:-

```
Database MyDB created successfully.
```

5. Create a table – employee having fields – eno, ename, age, salary, doj

Syntax:-

```
import mysql.connector


conn = mysql.connector.connect(
    host="localhost",
    user="your_username",
    password="your_password",
    database="MyDB"
)
```

```
cursor = conn.cursor()


cursor.execute("""
    CREATE TABLE employee (
        eno INT PRIMARY KEY,
        ename VARCHAR(255),
        age INT,
        salary DECIMAL(10, 2),
        doj DATE
    )
""")


cursor.close()
conn.close()


print("Table employee created successfully.")
```

Output:-

```
Table employee created successfully.
```

6. Insert, update and delete specific rows in that table using python script

Syntax:-

```
import mysql.connector
```

```
conn = mysql.connector.connect(
    host="localhost",
    user="your_username",
    password="your_password",
    database="MyDB"
)

cursor = conn.cursor()

def insert_employee(eno, ename, age, salary, doj):
    sql = "INSERT INTO employee (eno, ename, age, salary, doj) VALUES (%s, %s, %s, %s, %s)"
    val = (eno, ename, age, salary, doj)
    cursor.execute(sql, val)
    conn.commit()
    print("Employee inserted successfully.")

def update_employee(eno, new_salary):
    sql = "UPDATE employee SET salary = %s WHERE eno = %s"
    val = (new_salary, eno)
    cursor.execute(sql, val)
```

```
    conn.commit()
    print("Employee updated successfully.")

def delete_employee(eno):
    sql = "DELETE FROM employee WHERE eno = %s"
    val = (eno,)
    cursor.execute(sql, val)
    conn.commit()
    print("Employee deleted successfully.")

insert_employee(101, 'John Doe', 30, 50000.00, '2022-05-01')

update_employee(101, 55000.00)

delete_employee(101)

cursor.close()
conn.close()
```

Output:-

```
Employee inserted successfully.
Employee updated successfully.
Employee deleted successfully.
```

7. Create a menu driven CRUD operation program to perform above tasks.

Syntax:-

```
import mysql.connector


conn = mysql.connector.connect(
    host="localhost",
    user="your_username",
    password="your_password",
    database="MyDB"
)


cursor = conn.cursor()


def insert_employee():
    eno = int(input("Enter employee number: "))
    ename = input("Enter employee name: ")
    age = int(input("Enter employee age: "))
    salary = float(input("Enter employee salary: "))
    doj = input("Enter employee date of joining (YYYY-MM-DD): ")
    sql = "INSERT INTO employee (eno, ename, age, salary, doj) VALUES (%s,
%s, %s, %s, %s)"
```

```
    val = (eno, ename, age, salary, doj)
    cursor.execute(sql, val)
    conn.commit()
    print("Employee inserted successfully.")

def read_employees():
    cursor.execute("SELECT * FROM employee")
    employees = cursor.fetchall()
    for employee in employees:
        print(employee)

def update_employee():
    eno = int(input("Enter employee number whose salary to be updated: "))
    new_salary = float(input("Enter new salary: "))
    sql = "UPDATE employee SET salary = %s WHERE eno = %s"
    val = (new_salary, eno)
    cursor.execute(sql, val)
    conn.commit()
    print("Employee updated successfully.")

def delete_employee():
    eno = int(input("Enter employee number to be deleted: "))
```

```
    sql = "DELETE FROM employee WHERE eno = %s"
    val = (eno,)
    cursor.execute(sql, val)
    conn.commit()
    print("Employee deleted successfully.")

while True:
    print("\nMenu:")
    print("1. Insert employee")
    print("2. Read all employees")
    print("3. Update employee salary")
    print("4. Delete employee")
    print("5. Exit")
    choice = input("Enter your choice: ")

    if choice == '1':
        insert_employee()
    elif choice == '2':
        read_employees()
    elif choice == '3':
        update_employee()
    elif choice == '4':
```

```
            delete_employee()
        elif choice == '5':
            break
        else:
            print("Invalid choice. Please enter a valid option.")

    cursor.close()
    conn.close()
```

Output:-

```
Menu:
1. Insert employee
2. Read all employees
3. Update employee salary
4. Delete employee
5. Exit
Enter your choice:
```

Day:- 25

1. Create a MySQL database table called Tournament. Use exception handling while creating a database table.

Syntax:-

```
import mysql.connector
```

```
try:
    conn = mysql.connector.connect(
        host="localhost",
        user="your_username",
        password="your_password",
        database="MyDB"
    )

    cursor = conn.cursor()

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS Tournament (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        location VARCHAR(255),
        start_date DATE,
        end_date DATE
    )
    """)

    print("Tournament table created successfully.")
```

```
except mysql.connector.Error as err:
    print(f"Error: {err}")

finally:
    cursor.close()
    conn.close()
```

Output:-

Tournament table created successfully.

2. Having fields like Name, Age, Sport_Play, NoOfTournaments

Syntax:-

```
import mysql.connector

try:
    conn = mysql.connector.connect(
        host="localhost",
        user="your_username",
        password="your_password",
        database="MyDB"
    )
```

```
        cursor = conn.cursor()

        cursor.execute("""
        CREATE TABLE IF NOT EXISTS Tournament (
            id INT AUTO_INCREMENT PRIMARY KEY,
            Name VARCHAR(255) NOT NULL,
            Age INT,
            Sport_Play VARCHAR(255),
            NoOfTournaments INT
        )
        """)

        print("Tournament table created successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        cursor.close()
        conn.close()
```

Output:-

Tournament table created successfully.

3. After inserting data in the table, write a menu driven program to

a. Display sport wise no. of players (sports name - count of players)

b. Display player' s details for a specific sport.

c. Update the tournament field (increment by 1) for a user specified player (when he has played match)

d. Search players' detail by their name.

e. Delete a record of specific player

Syntax:-

```
import mysql.connector


def display_sportwise_players(cursor):
    try:
        cursor.execute("""
            SELECT Sport_Play, COUNT(*) AS Player_Count
            FROM Tournament
            GROUP BY Sport_Play
        """)
        sportwise_players = cursor.fetchall()
        print("Sport-wise number of players:")
        for sport, count in sportwise_players:
            print(f"{sport} - {count}")
```

```
    except mysql.connector.Error as err:
        print(f"Error: {err}")

def display_players_details(cursor, sport_name):
    try:
        cursor.execute(f"""
            SELECT *
            FROM Tournament
            WHERE Sport_Play = '{sport_name}'
        """)
        players_details = cursor.fetchall()
        print(f"Players' details for {sport_name}:")
        for player in players_details:
            print(player)
    except mysql.connector.Error as err:
        print(f"Error: {err}")

def update_tournament_field(cursor, player_name):
    try:
        cursor.execute(f"""
            UPDATE Tournament
            SET NoOfTournaments = NoOfTournaments + 1
```

```
            WHERE Name = '{player_name}'
        """)
        print(f"Tournament field updated for {player_name}.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

def search_players_details(cursor, player_name):
    try:
        cursor.execute(f"""
            SELECT *
            FROM Tournament
            WHERE Name = '{player_name}'
        """)
        player_details = cursor.fetchone()
        if player_details:
            print(f"Player's details for {player_name}: {player_details}")
        else:
            print(f"No player found with the name {player_name}.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

def delete_player_record(cursor, player_name):
```

```
    try:
        cursor.execute(f"""
            DELETE FROM Tournament
            WHERE Name = '{player_name}'
        """)
        print(f"Record of {player_name} deleted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

def main():
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="your_username",
            password="your_password",
            database="MyDB"
        )

        cursor = conn.cursor()

        while True:
            print("\nMenu:")
```

```
print("a. Display sport-wise number of players")
print("b. Display player's details for a specific sport")
print("c. Update the tournament field for a specific player")
print("d. Search players' detail by their name")
print("e. Delete a record of specific player")
print("f. Exit")
choice = input("Enter your choice: ")

if choice == 'a':
    display_sportwise_players(cursor)
elif choice == 'b':
    sport_name = input("Enter sport name: ")
    display_players_details(cursor, sport_name)
elif choice == 'c':
    player_name = input("Enter player name: ")
    update_tournament_field(cursor, player_name)
elif choice == 'd':
    player_name = input("Enter player name: ")
    search_players_details(cursor, player_name)
elif choice == 'e':
    player_name = input("Enter player name: ")
    delete_player_record(cursor, player_name)
```

```
            elif choice == 'f':
                break
            else:
                print("Invalid choice. Please try again.")

        except mysql.connector.Error as err:
            print(f"Error: {err}")

        finally:
            cursor.close()
            conn.close()

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

Output:-

Menu:

a. Display sport-wise number of players

b. Display player's details for a specific sport

c. Update the tournament field for a specific player

d. Search players' detail by their name

e. Delete a record of specific player

f. Exit

Enter your choice:

4. Take the DB table backup in the binary file. (Use Pickle.dump())

Syntax:-

```
import sqlite3
import pickle


def backup_table_to_binary(db_file, table_name, backup_file):
    try:
        conn = sqlite3.connect(db_file)
        cursor = conn.cursor()

        cursor.execute(f"SELECT * FROM {table_name}")
        table_data = cursor.fetchall()

        conn.close()

        data_to_backup = {
            'table_name': table_name,
            'data': table_data
        }
```

```
        with open(backup_file, 'wb') as file:
            pickle.dump(data_to_backup, file)

        print(f"Backup of table '{table_name}' created successfully in '{backup_file}'.")
    except sqlite3.Error as e:
        print("SQLite error:", e)
    except Exception as e:
        print("Error:", e)

backup_table_to_binary("my_database.db", "my_table", "table_backup.bin")
```

Output:-

```
Backup of table 'my_table' created successfully in 'table_backup.bin'.
```

5. Delete all the records from the above table

Syntax:-

```
import sqlite3

def delete_all_records(db_file, table_name):
    try:
        conn = sqlite3.connect(db_file)
        cursor = conn.cursor()
```

```
            cursor.execute(f"DELETE FROM {table_name}")

            conn.commit()

            conn.close()

            print("All records deleted successfully from the table.")
        except sqlite3.Error as e:
            print("SQLite error:", e)
        except Exception as e:
            print("Error:", e)

    delete_all_records("my_database.db", "my_table")
```

Output:-

```
All records deleted successfully from the table.
```

6. Reload the data from text file. (Use Pickle.load())

Syntax:-

```
import pickle

def reload_data_from_text_file(file_path):
```

```
    try:
        with open(file_path, 'rb') as file:
            data = pickle.load(file)

        print("Data reloaded from the text file successfully.")
        return data
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        return None
    except pickle.PickleError as e:
        print("Error:", e)
        return None

data = reload_data_from_text_file("data_backup.txt")
if data:
    print("Reloaded data:", data)
```

Output:-

```
Data reloaded from the text file successfully.
Reloaded data: {'key1': 'value1', 'key2': 'value2', ...}
```

Day-28

1. **Create a dataframe from an excel sheet**.

```
import pandas as pd
excel_df = pd.read_excel('std.xlsx')
print(excel_df)
```

Output:

Roll Number	Name	Age	Marks
C01	hit	20	99
C02	m0hit	21	59
C03	khuman	21	78
C04	maulik	21	48
C05	bhautik	21	68
C06	vivek	21	77
C07	sahdev	1	88
C08	meet	21	99
C09	jeet	22	65
C10	Rinkal	20	35
C11	Tanvi	20	65

2. Create a dataframe from a .csv file

```
import pandas as pd
csv_df = pd.read_csv('std1.csv')
print(csv_df)
```

Output:

Roll Number\tName\tAge\tMarks

C01\tJignesh\t20\t99

C02\tVasu\t21\t59

C03\tRohit\t21\t78

C04\tVistrut\t21\t48

C05\tZeel\t21\t68

C06\tDhavel\t21\t77

C07\tYash\t1\t88

C08\tVivek\t21\t99

C09\tVandan\t22\t65

C10\tRinkal\t20\t35

C11\tTanvi\t20\t65

3. Create a dataframe from the dictionary object

```
data = {'RollNo': ['C12','C13','C14'],
        'Name': ['M0hit', 'Abhishek', 'deep'],
        'Age': [20, 21, 22],
        'Marks': [85, 90, 88]}
dict_df = pd.DataFrame(data)
```

Output:

RollNo	Name	Age	Marks
C12	M0hit	20	85
C13	Abhishek	21	90
C14	deep	22	88

4. Create a dataframe from a list object

```
data_list = [(12, 'M0hit', 20, 85),
             (13, 'Abhishek', 21, 90),
             (14, 'Deep', 22, 88)]
list_df = pd.DataFrame(data_list, columns=['RollNo', 'Name', 'Age', 'Marks'])
```

Output:

RollNo	Name	Age	Marks
12	M0hit	20	85
13	Abhishek	21	90
14	Deep	22	88

5. Display total number of rows and columns in the dataframe

```
print("Rows and Columns in Excel dataframe:", excel_df.shape)
print("Rows and Columns in CSV dataframe:", csv_df.shape)
print("Rows and Columns in Dictionary dataframe:", dict_df.shape)
print("Rows and Columns in List dataframe:", list_df.shape)
```

Output:

```
Rows and Columns in Excel dataframe: (11, 4)
Rows and Columns in CSV dataframe: (11, 1)
Rows and Columns in Dictionary dataframe: (3, 4)
Rows and Columns in List dataframe: (3, 4)
```

6. Display only 1st 3 rows from dataframe

```
print("First 3 rows from Excel dataframe:")
print(excel_df.head(3))
```

Output:

First 3 rows from Excel dataframe:

Roll Number	Name	Age	Marks
C01	hit	20	99
C02	m0hit	21	59
C03	khuman	21	78

7. Display only last two rows from dataframe

```
print("Last 2 rows from CSV dataframe:")
print(csv_df.tail(2))
```

Output:

Last 2 rows from CSV dataframe:

Roll Number\tName\tAge\tMarks

C10\tRinkal\t20\t35

C11\tTanvi\t20\t65

8. Display 3rd to 7th row of the dataframe

```
print("3rd to 7th row from Dictionary dataframe:")
print(dict_df.iloc[2:7])
```

Output:

3rd to 7th row from Dictionary dataframe:

	Roll Number	Name	Age	Marks
2	C03	khuman	21	78
3	C04	maulik	21	48
4	C05	bhautik	21	68
5	C06	vivek	21	77
6	C07	sahdev	1	88

9. Display all the rows in reverse order.

```
print("All rows in reverse order from List dataframe:")
print(list_df.iloc[::-1])
```

Output:

All rows in reverse order from List dataframe:

	Roll Number	Name	Age	Marks
C11	Tanvi	20	65	
C10	Rinkal	20	35	

C09 jeet 22 65

C08 meet 21 99

C07 sahdev 1 88

C06 vivek 21 77

C05 bhautik 21 68

C04 maulik 21 48

C03 khuman 21 78

C02 m0hit 21 59

C01 hit 20 99

10. Display all column names of the dataframe.

```
print("Column names of Excel dataframe:")
print(excel_df.columns.tolist())
```

Output:

Column names of Excel dataframe:

['Roll Number', 'Name', 'Age', 'Marks']

Day-29

1. Display only name and age of all students from the dataframe

```
print("Name and Age of all students:")
print(excel_df[['Name', 'Age']])
```

Output:

Name and Age of all students:

	Name	Age
	hit	20
	m0hit	21
	khuman	21
	maulik	21
	bhautik	21
	vivek	21
	sahdev	1
	meet	21
	jeet	22
	Rinkal	20
	Tanvi	20

2. Display maximum and minimum marks from the dataframe.

```
max_marks = excel_df['Marks'].max()
min_marks = excel_df['Marks'].min()
print("Maximum Marks:", max_marks)
print("Minimum Marks:", min_marks)
```

Output:

Maximum Marks: 99

Minimum Marks: 35

3. Display the statistical analysis of marks from the student dataframe

```
print("Statistical analysis of marks:")
print(excel_df['Marks'].describe())
```

Output:

```
Statistical analysis of marks:
count    11.00000
mean     71.00000
std      19.97999
min      35.00000
25%      62.00000
50%      68.00000
75%      83.00000
max      99.00000
```

4. Display the name of the student having marks > 50

```
print("Students having marks > 50:")
print(excel_df[excel_df['Marks'] > 50]['Name'])
```

Output:

```
Students having marks > 50:
0        hit
1      m0hit
2      khuman
4     bhautik
5       vivek
6      sahdev
7        meet
8        jeet
10      Tanvi
```

Name: Name, dtype: object

5. Display the rollno and name of the student whose age is > 20

```
print("RollNo and Name of students whose age is > 20:")
print(excel_df.loc[excel_df['Age'] > 20, ['RollNo', 'Name']])
```

Output:

```
RollNo and Name of students whose age is > 20:
  RollNo     Name
   C02    m0hit
   C03    khuman
   C04   maulik
   C05  bhautik
   C06    vivek
   C08     meet
   C09     jeet
```

6. Display the students having age between 20 and 25

```
print("Students having age between 20 and 25:")
print(excel_df[(excel_df['Age'] >= 20) & (excel_df['Age'] <= 25)])
```

Output:

```
Students having age between 20 and 25:
   RollNo     Name  Age  Marks
0    C01      hit   20     99
1    C02    m0hit   21     59
2    C03    khuman   21     78
3    C04   maulik   21     48
4    C05  bhautik   21     68
```

5 C06 vivek 21 77

7 C08 meet 21 99

8 C09 jeet 22 65

9 C10 Rinkal 20 35

10 C11 Tanvi 20 65

7. Display the name of the student who has scored maximum marks

```
max_marks_student = excel_df[excel_df['Marks'] == max_marks]['Name'].iloc[0]
print("Student with maximum marks:", max_marks_student)
```

Output:

Student with maximum marks: hit

8. Display the students who have scored more than average marks (use mean)

```
avg_marks = excel_df['Marks'].mean()
print("Students who scored more than average marks:")
print(excel_df[excel_df['Marks'] > avg_marks])
```

Output:

Students who scored more than average marks:

RollNo Name Age Marks

0 C01 hit 20 99

2 C03 khuman 21 78

5 C06 vivek 21 77

6 C07 sahdev 1 88

7 C08 meet 21 99

Day-30

```
import pandas as pd
```

1. Create a scalar series (dictionary object with single value) and convert it into a dataframe

```
scalar_series = pd.Series({'Value': 10})

scalar_df = scalar_series.to_frame()

print("Scalar DataFrame:")

print(scalar_df)
```

Output:

```
Scalar DataFrame:

      0

Value  10
```

2. Create MultiIndex.from_arrays like Students [ ], Score [ ], Age [ ]

```
arrays = [['Student1', 'Student2', 'Student3'], ['Score1', 'Score2', 'Score3'], ['Age1', 'Age2', 'Age3']]

multi_index = pd.MultiIndex.from_arrays(arrays, names=('Students', 'Score', 'Age'))

print("\nMultiIndex from arrays:")

print(multi_index)
```

Output:

```
MultiIndex from arrays:

MultiIndex([('Student1', 'Score1', 'Age1'),

            ('Student2', 'Score2', 'Age2'),

            ('Student3', 'Score3', 'Age3')],

           names=['Students', 'Score', 'Age'])
```

3. Create MultiIndex.from_frame

a. Use dictionary object for employee data for empId, Name, and Salary

```
employee_data = {'empId': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie'], 'Salary': [50000, 60000, 70000]}

employee_df = pd.DataFrame(employee_data)
```

Output:

```
DataFrame with MultiIndex from dictionary:
  empId     Name  Salary
0     1    M0hit   50000
1     2     Hit   60000
2     3   Ketan   70000
```

b. Create DataFrames by read_excel(), read_csv() and set multiindex

```
employee_excel = pd.read_excel('emp.xlsx')

employee_csv = pd.read_csv('emp1.csv')

employee_excel.set_index(['empId', 'Name'], inplace=True)

employee_csv.set_index(['empId', 'Name'], inplace=True)

print("\nDataFrame with MultiIndex from dictionary:")

print(employee_df)

print("\nDataFrame with MultiIndex from excel:")

print(employee_excel)

print("\nDataFrame with MultiIndex from csv:")

print(employee_csv)
```

Output:

```
DataFrame with MultiIndex from excel:

Empty DataFrame

Columns: [Salary]

Index: []


DataFrame with MultiIndex from csv:

        Salary
```

empId Name

1 hit 100000

4. Create DataFrame from List object with Index values

```
student_data = [('Student1', 'Pass', 80, 85, 90),
        ('Student2', 'Fail', 60, 55, 70),
        ('Student3', 'Pass', 75, 85, 80),
        ('Student4', 'Pass', 90, 85, 95),
        ('Student5', 'Fail', 40, 45, 50)]
columns = ['Student', 'Result', 'Subject1', 'Subject2', 'Subject3']
student_df = pd.DataFrame(student_data, columns=columns)
student_df.set_index('Student', inplace=True)
print("\nDataFrame from List object with Index values:")
print(student_df)
```

Output:

DataFrame from List object with Index values:

Student	Result	Subject1	Subject2	Subject3
Student1	Pass	80	85	90
Student2	Fail	60	55	70
Student3	Pass	75	85	80
Student4	Pass	90	85	95
Student5	Fail	40	45	50

5. Find total number of 'pass' students and 'fail' students in each subject from above list – use of groupby

```
pass_fail_counts = student_df.groupby('Result').apply(lambda x: (x == 'Pass').sum())
print("\nPass/Fail counts in each subject:")
```

```
print(pass_fail_counts)
```

Output:

```
Pass/Fail counts in each subject:
        Result  Subject1  Subject2  Subject3
Result
Fail         0         0         0         0
Pass         3         0         0         0
```

6. Find minimum and maximum marks of each subject

```
min_marks = student_df.min()
max_marks = student_df.max()
print("\nMinimum marks of each subject:")
print(min_marks)
print("\nMaximum marks of each subject:")
print(max_marks)
```

Output:

```
Minimum marks of each subject:
Result      Fail
Subject1      40
Subject2      45
Subject3      50
Maximum marks of each subject:
Result      Pass
Subject1      90
Subject2      85
Subject3      95
```

Day-31

import pandas as pd

1. **Create separate data frames with some common index and some common columns**

df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}, index=['X', 'Y', 'Z'])

df2 = pd.DataFrame({'A': [7, 8, 9], 'C': [10, 11, 12]}, index=['Y', 'Z', 'W'])

Output:

	A	B
X	1	4
Y	2	5
Z	3	6

	A	C
Y	7	10
Z	8	11
W	9	12

2. **Concat dataframes with outer join (axis=1)**

outer_concat = pd.concat([df1, df2], axis=1, sort=False)

print("Outer join:")

print(outer_concat)

Output:

Outer join:

	A	B	A	C
X	1.0	4.0	NaN	NaN
Y	2.0	5.0	7.0	10.0
Z	3.0	6.0	8.0	11.0
W	NaN	NaN	9.0	12.0

3. **Concat dataframes with inner join (join="inner")**

```
inner_concat = pd.concat([df1, df2], axis=1, join="inner")
print("\nInner join:")
print(inner_concat)
```

Output:

```
Inner join:
  A  B  A   C
Y  2  5  7  10
Z  3  6  8  11
```

4. Concat dataframes with "left" join (.reindex())

```
left_concat = pd.concat([df1, df2.reindex(df1.index)], axis=1)
print("\nLeft join:")
print(left_concat)
```

Output:

```
Left join:
  A  B    A    C
X  1  4  NaN   NaN
Y  2  5  7.0  10.0
Z  3  6  8.0  11.0
```

5. Concat dataframes by Ignoring indexes on the concatenation axis

```
ignore_index_concat = pd.concat([df1, df2], ignore_index=True, sort=False)
print("\nConcatenate ignoring indexes:")
print(ignore_index_concat)
```

Output:

```
Concatenate ignoring indexes:
   A    B    C
0  1  4.0   NaN
1  2  5.0   NaN
2  3  6.0   NaN
```

3 7 NaN 10.0

4 8 NaN 11.0

5 9 NaN 12.0

6. Concat named series with a dataframe

```
s = pd.Series([13, 14, 15], name='D', index=['X', 'Y', 'Z'])
concat_with_series = pd.concat([df1, s], axis=1)
print("\nConcatenate with named series:")
print(concat_with_series)
```

Output:

Concatenate with named series:

```
  A  B   D
X 1  4  13
Y 2  5  14
Z 3  6  15
```

7. Save all the resultant dataframes in the separate sheets of an excel file

```
with pd.ExcelWriter('output.xlsx') as writer:
    outer_concat.to_excel(writer, sheet_name='Outer Join')
    inner_concat.to_excel(writer, sheet_name='Inner Join')
    left_concat.to_excel(writer, sheet_name='Left Join')
    ignore_index_concat.to_excel(writer, sheet_name='Ignore Index Concat')
    concat_with_series.to_excel(writer, sheet_name='Concat with Series')
```

Day-32

1. Create 2 data frames from SQL tables and merge them based on common column (keys)

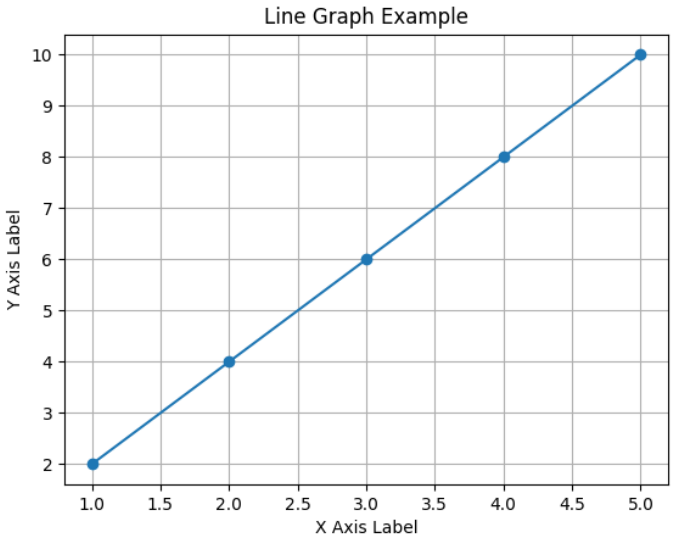
```
import pandas as pd
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',
    database='school'
)

df1 = pd.read_sql_query('SELECT * FROM table1', conn)
df2 = pd.read_sql_query('SELECT * FROM table2', conn)

merged_df = pd.merge(df1, df2, on='common_column')

print(merged_df)

conn.close()
```

2. Plot a line graph,

```
import matplotlib.pyplot as plt

x_values = [1, 2, 3, 4, 5]
```

```
y_values = [2, 4, 6, 8, 10]

plt.plot(x_values, y_values, marker='o', linestyle='-')

plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')
plt.title('Line Graph Example')

plt.grid(True)
plt.show()
```

Output:-

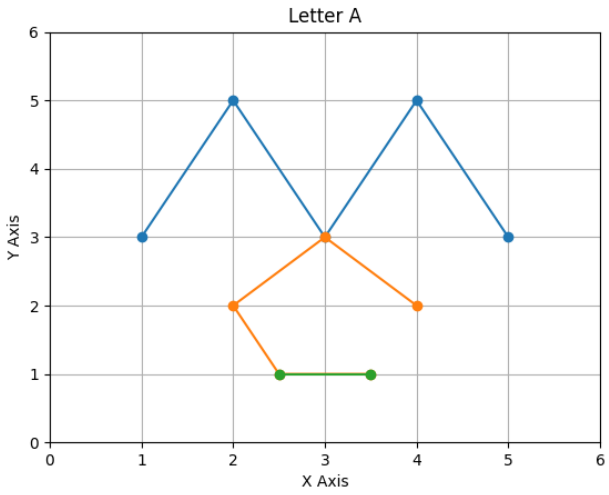


3. Make a letter 'A' using plot method with x and y points and then after passing the array of points (multiple lines)

```
import matplotlib.pyplot as plt

x_points = [1, 2, 3, 4, 5, 4, 3, 2, 2.5, 3.5]
```

```
y_points = [3, 5, 3, 5, 3, 2, 3, 2, 1, 1]

plt.plot(x_points[:5], y_points[:5], marker='o', linestyle='-')
plt.plot(x_points[5:], y_points[5:], marker='o', linestyle='-')
plt.plot([2.5, 3.5], [1, 1], marker='o', linestyle='-')

plt.xlim(0, 6)
plt.ylim(0, 6)

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Letter A')

plt.grid(True)
plt.show()
```

Output:-



4. Make letters ‘E’, ‘F’, ‘H’, ‘I’, ‘K’, ‘L’, ‘M’, ‘W’, ‘X’ and ‘Z’ using plot()

```
import matplotlib.pyplot as plt

letters = {
    'E': [[1, 5], [1, 1], [1, 5], [3, 3], [1, 5], [5, 5]],
    'F': [[1, 5], [1, 1], [1, 5], [3, 3]],
    'H': [[1, 1], [1, 5], [3, 3], [1, 5], [5, 5]],
    'I': [[2, 4], [1, 5], [2, 4]],
    'K': [[1, 1], [1, 5], [3, 1], [1, 3], [3, 5]],
    'L': [[1, 1], [1, 5]],
    'M': [[1, 1], [1, 3], [2, 2], [3, 3], [3, 5], [5, 5]],
    'W': [[1, 1], [1, 3], [2, 2], [3, 3], [3, 5], [5, 5], [5, 5]],
    'X': [[1, 5], [5, 1]],
    'Z': [[1, 5], [5, 1]]
}

for letter, points in letters.items():
    for line in points:
        plt.plot(line, [letter]*2, marker='', linestyle='-')

plt.xlim(0, 6)
plt.ylim(0, len(letters) + 1)

plt.xlabel('X Axis')
plt.ylabel('Letter')
plt.title('Letters')
```

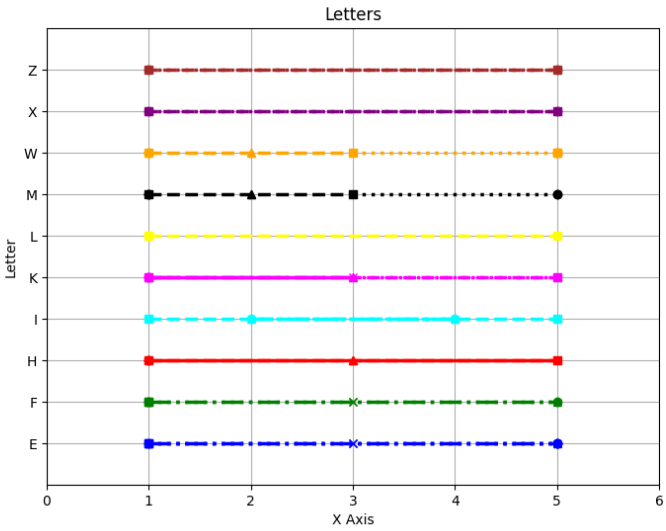
```
plt.grid(True)
plt.yticks(range(1, len(letters) + 1), letters.keys())
plt.show()
```

5. Use of linestyle as 'dotted' , ‘dashed’, ‘dashdot’ for plotting above characters, Make all letters colourful, set different line width, use different markers

```
import matplotlib.pyplot as plt

letters = {
    'E': [[1, 5], [1, 1], [1, 5], [3, 3], [1, 5], [5, 5]],
    'F': [[1, 5], [1, 1], [1, 5], [3, 3]],
    'H': [[1, 1], [1, 5], [3, 3], [1, 5], [5, 5]],
    'I': [[2, 4], [1, 5], [2, 4]],
    'K': [[1, 1], [1, 5], [3, 1], [1, 3], [3, 5]],
    'L': [[1, 1], [1, 5]],
    'M': [[1, 1], [1, 3], [2, 2], [3, 3], [3, 5], [5, 5]],
    'W': [[1, 1], [1, 3], [2, 2], [3, 3], [3, 5], [5, 5], [5, 5]],
    'X': [[1, 5], [5, 1]],
    'Z': [[1, 5], [5, 1]]
}

colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black', 'orange', 'purple', 'brown']
line_styles = ['dotted', 'dashed', 'dashdot', 'solid']
markers = ['o', 's', '^', 'x', '+']
```

```
fig, ax = plt.subplots(figsize=(8, 6))


for i, (letter, points) in enumerate(letters.items()):
    for j, line in enumerate(points):
        plt.plot(line, [i+1]*2, color=colors[i], linestyle=line_styles[j%len(line_styles)], linewidth=2.5, marker=markers[j%len(markers)])


plt.xlim(0, 6)
plt.ylim(0, len(letters) + 1)


plt.xlabel('X Axis')
plt.ylabel('Letter')
plt.title('Letters')


plt.grid(True)
plt.yticks(range(1, len(letters) + 1), letters.keys())
plt.show()
```

Output:-

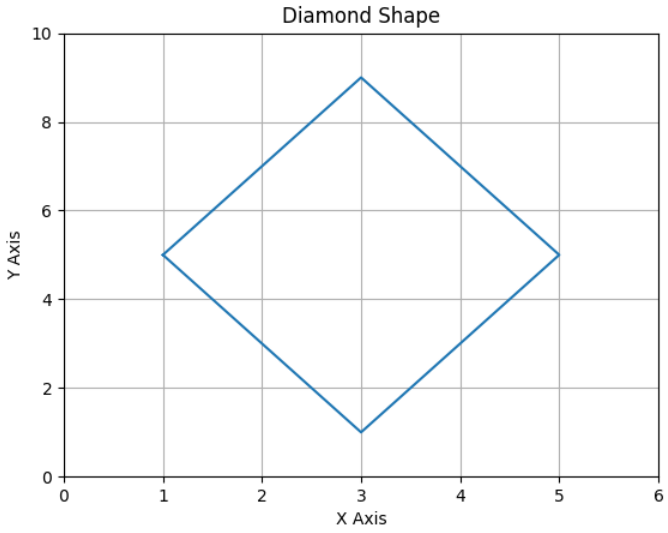


6. Make a diamond shape using multiple lines

```
import matplotlib.pyplot as plt

diamond_points = [
    [1, 5],
    [3, 1],
    [5, 5],
    [3, 9],
    [1, 5]
]

x_values = [point[0] for point in diamond_points]
y_values = [point[1] for point in diamond_points]

plt.plot(x_values, y_values, marker='', linestyle='-')
```

```
plt.xlim(0, 6)
plt.ylim(0, 10)

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Diamond Shape')

plt.grid(True)
plt.show()
```

Output:-



7. Set the graph title, xlabel and ylabel

```
import matplotlib.pyplot as plt

diamond_points = [
    [1, 5],
```

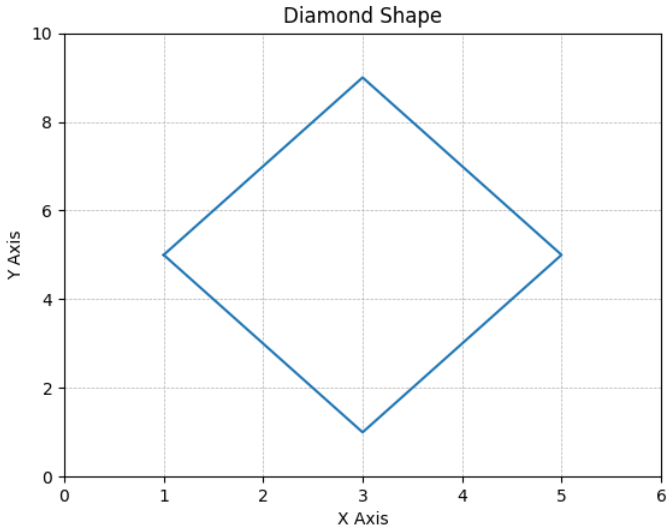
```
    [3, 1],
    [5, 5],
    [3, 9],
    [1, 5]
]

x_values = [point[0] for point in diamond_points]
y_values = [point[1] for point in diamond_points]

plt.plot(x_values, y_values, marker='', linestyle='-')

plt.xlim(0, 6)
plt.ylim(0, 10)

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Diamond Shape')

plt.grid(True)
plt.show()
```

8. Draw grids with (x and y axis , linestyle, linewidth)

```
import matplotlib.pyplot as plt

diamond_points = [
    [1, 5],
```

```
    [3, 1],
    [5, 5],
    [3, 9],
    [1, 5]
]

x_values = [point[0] for point in diamond_points]
y_values = [point[1] for point in diamond_points]

plt.plot(x_values, y_values, marker='', linestyle='-')

plt.xlim(0, 6)
plt.ylim(0, 10)

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Diamond Shape')

plt.grid(True, linestyle='--', linewidth=0.5)

plt.show()
```

Output:-



9. Save all line graphs in separate .png files

```
import matplotlib.pyplot as plt

data = {
    'graph1': {'x': [1, 2, 3, 4, 5], 'y': [2, 4, 6, 8, 10]},
    'graph2': {'x': [1, 2, 3, 4, 5], 'y': [5, 4, 3, 2, 1]},
    'graph3': {'x': [1, 2, 3, 4, 5], 'y': [3, 6, 9, 12, 15]}
}

for name, values in data.items():
    plt.plot(values['x'], values['y'])
    plt.xlabel('X Axis')
    plt.ylabel('Y Axis')
    plt.title(name)
    plt.grid(True)
    plt.savefig(f'{name}.png')
    plt.close()
```

```
print("All graphs have been saved as separate .png files.")
```

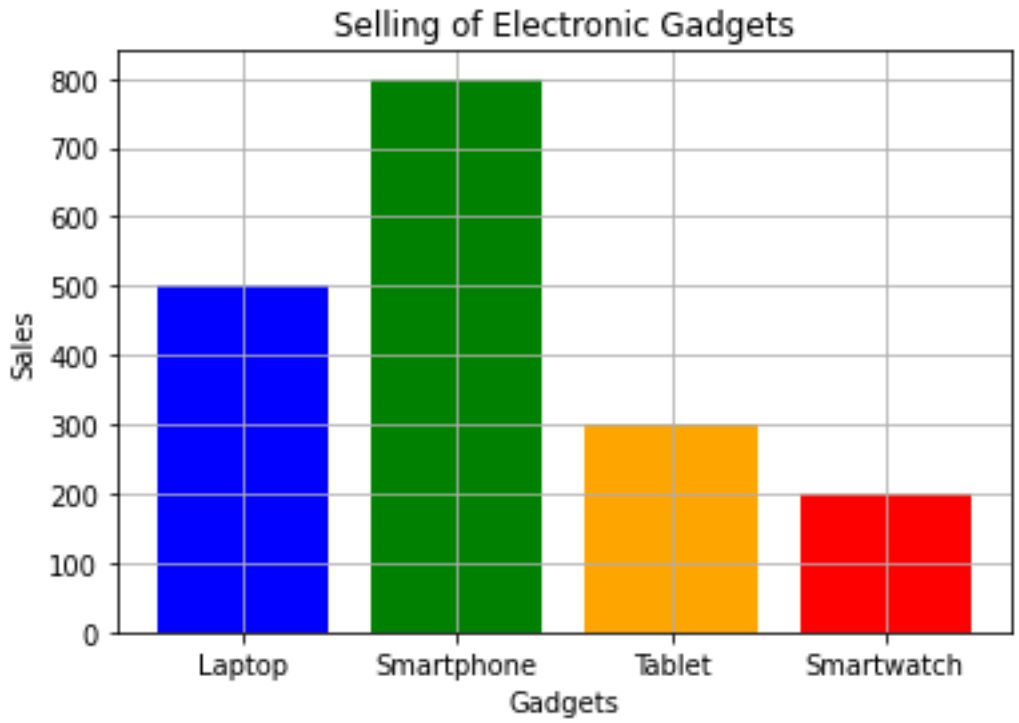
Day-33

```
import matplotlib.pyplot as plt
```

1. Draw bargraph for selling of different electronic gadgets

```
gadgets = ['Laptop', 'Smartphone', 'Tablet', 'Smartwatch']
sales = [500, 800, 300, 200]

plt.bar(gadgets, sales, color=['blue', 'green', 'orange', 'red'])
plt.title('Selling of Electronic Gadgets')
plt.xlabel('Gadgets')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```

Output:



2. Draw a histogram for the yearly performance of a student

yearly_performance = [80, 85, 90, 92, 88, 85, 90, 95, 85, 90]

plt.hist(yearly_performance, bins=10, color='skyblue', edgecolor='black')

plt.title('Yearly Performance of a Student')

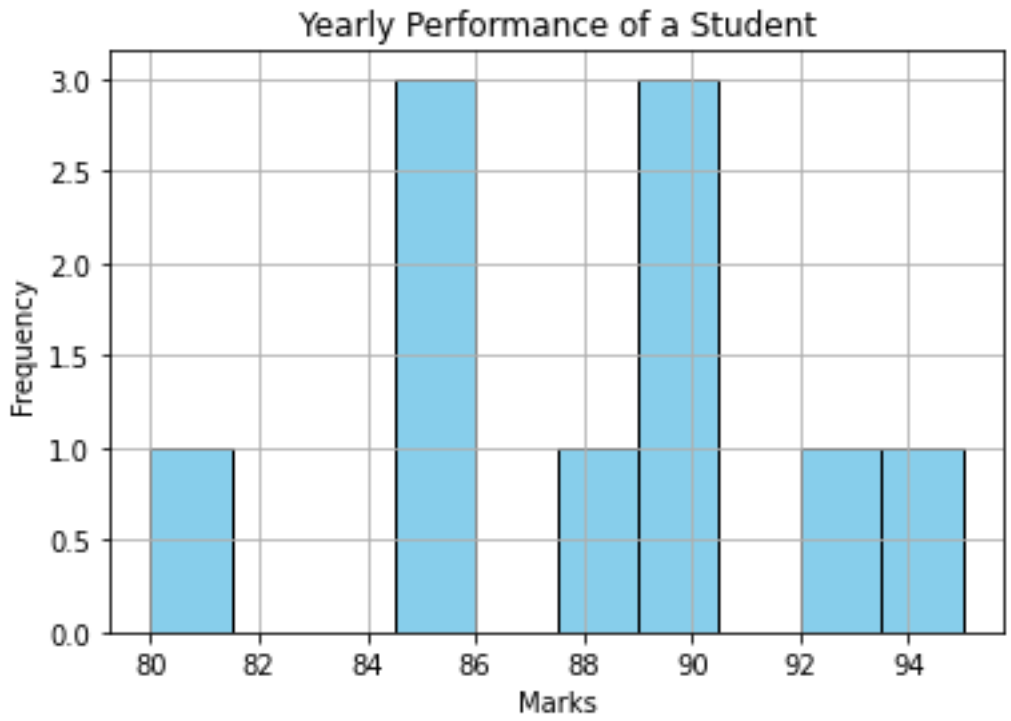
plt.xlabel('Marks')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

Output:



3. Draw a pie chart for the student's participation in different games

games = ['Football', 'Basketball', 'Cricket', 'Tennis']

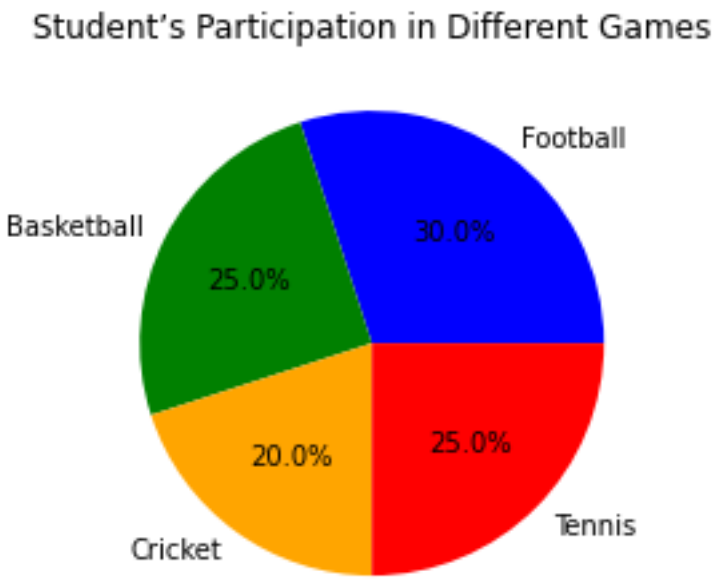
participation = [30, 25, 20, 25]

plt.pie(participation, labels=games, autopct='%1.1f%%', colors=['blue', 'green', 'orange', 'red'])

plt.title("Student's Participation in Different Games")

plt.show()

Output:



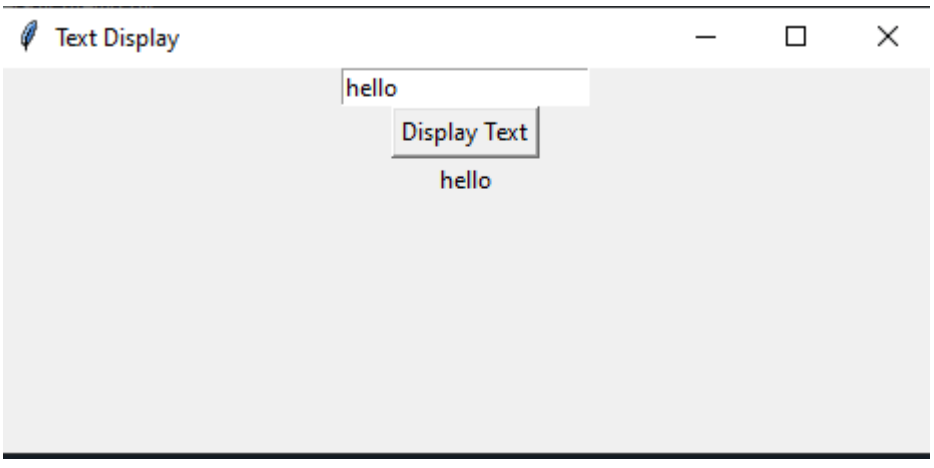
Day-34

1. **Create a GUI program that takes user input in the Entry widget and on button click that text must be displayed on the Label widget**

```
import tkinter as tk

def display_text():
```

```
    text = entry.get()
    label.config(text=text)

root = tk.Tk()
root.title("Text Display")

entry = tk.Entry(root)
entry.pack()

button = tk.Button(root, text="Display Text", command=display_text)
button.pack()

label = tk.Label(root, text="")
label.pack()

root.mainloop()
```

Output:

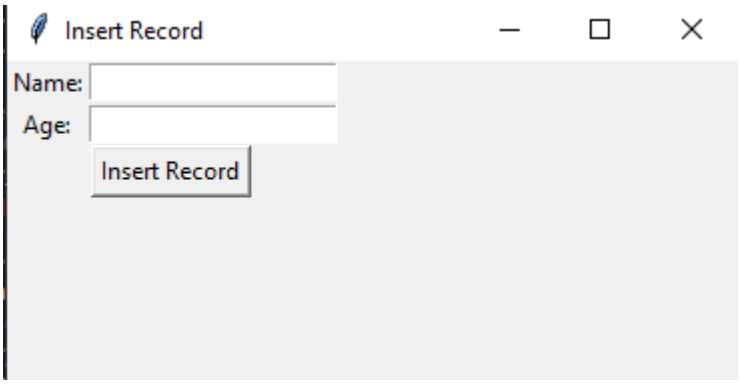


1. Insert a record in the DB table by taking user input from the GUI

2. Update and delete the record in the DB table.

```
import tkinter as tk
import sqlite3

def insert_record():
    name = name_entry.get()
    age = age_entry.get()
    conn = sqlite3.connect('test.db')
    c = conn.cursor()
    c.execute("INSERT INTO records (name, age) VALUES (?, ?)", (name, age))
    conn.commit()
    conn.close()
    name_entry.delete(0, tk.END)
    age_entry.delete(0, tk.END)

root = tk.Tk()
root.title("Insert Record")

name_label = tk.Label(root, text="Name:")
name_label.grid(row=0, column=0)
name_entry = tk.Entry(root)
name_entry.grid(row=0, column=1)

age_label = tk.Label(root, text="Age:")
age_label.grid(row=1, column=0)
age_entry = tk.Entry(root)
age_entry.grid(row=1, column=1)
```

```
insert_button = tk.Button(root, text="Insert Record", command=insert_record)
insert_button.grid(row=2, columnspan=2)

root.mainloop()
```



1. Create a Calculator application using tkinter module in Python
Perform different mathematical operations using GUI – button widget

```
import tkinter as tk


def button_click(event):
    text = event.widget.cget("text")
    if text == "=":
        try:
            result = eval(entry.get())
            entry.delete(0, tk.END)
            entry.insert(tk.END, result)
        except Exception as e:
            entry.delete(0, tk.END)
```

```
            entry.insert(tk.END, "Error")
    elif text == "C":
        entry.delete(0, tk.END)
    else:
        entry.insert(tk.END, text)

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root, font=("Arial", 20))
entry.grid(row=0, column=0, columnspan=4)

buttons = [
    ("7", 1, 0), ("8", 1, 1), ("9", 1, 2), ("/", 1, 3),
    ("4", 2, 0), ("5", 2, 1), ("6", 2, 2), ("*", 2, 3),
    ("1", 3, 0), ("2", 3, 1), ("3", 3, 2), ("-", 3, 3),
    ("0", 4, 0), (".", 4, 1), ("=", 4, 2), ("+", 4, 3),
    ("C", 5, 0)
]

for (text, row, column) in buttons:
    button = tk.Button(root, text=text, font=("Arial", 15), padx=20, pady=10)
    button.grid(row=row, column=column, padx=5, pady=5)
    button.bind("<Button-1>", button_click)
```

root.mainloop()