Lab1:

Title:

practise using data types like lists, dictonories, and sets.
write a Python program to create a list of integers, append values, sort the list, and remove duplicates.

code:

```
name = 'binod'
 age = 21
 names= ['dipak', 'ram', 'hari', 'pari', 'alina', 'molina']   # list
 tupleho = ('sekenu', 'teku', 23, 'ram')   # tuple
 setho = ('remae', 'ram', 'ram', 'zyan'}
 dictho = {'name': 'Binod Raj Pant', 'age': 21, 'study': 'BE comp', 'clz': 'NAST'}

 print(f"{name} is of type : {type(name)}")
 print(f"{names} is of type : {type (names)}")
 print(f"{tupleho} is of type : {type(tupleho)}")
 print(f"{setho} is of type : {type(setho)}")
 print(f"{dictho} is of type : {type(dictho)}")
```

~~Output~~: # creating list of integer, append values, sort & remove duplicate.

```
 listint = [12, 11, 7, 8, 5, 4, 3, 2]
 print(f"{listint} is of type : {type(listint)}")
 listint.append (999)    # appending to list
 print(f" updated {listint} is of type : {type(listint)}")
listint.append (999)  # multiple similar value
 print (f" updated {listint} is of type : {type(listint)}")

listint. sort()   # To sort list
print(f" ~~stored~~ sorted list : {listint is of type : {type(listint)}")

uniqueval = set(listint)   # convert into set for uniqueness
print (f" sorted and unique : {uniqueval} is of type: {type(uniqueval)}
 # again convert to list
lis = list(uniqueval)
print (f" sorted and unique value as list : {lis} is of
              type : {type(lis)}")
```

Title:

Create a simple script that performs basic arithematic operations :

Code:

```
num1 = input("enter number 1 : ")
num2 = input(" Enter number 2: ")

sum = num1 + num2
sub = num1 - num2
mult = num1 * num2
div  = num1 / num2
mod  = num1 % num2

print(f" The sum of {num1} and {num2} is : {sum} ")
print(f" The subsraction of {num1} and {num2} is :{sub}")
print(f" The multiplication of {num1} and {num2} is : {mult}")
print(f" The division of {num1} and {num2} is : {div}")
print (f" The modulus div of {num1} and {num2} is : {mod}")
```

output:

```
Enter number 1 : 10
enter number 2 : 2
 The sum of 10 and 2 is : 12
 The substraction of 10 and 2 is : 8
 The multiplication of 10 and 2 is : 20
 The division of 10 and 2 is : 5.0
 The modulus div of 10 and 2 is : 0
```

title :-

WAP that stores student name as keys and their marks as values in a dictonary, print avg marks and name of the topscorer.

code :-

```
dictname = {'Binod':55, 'shekhar':69, 'monit':66, 'sandy':77,
            'Bikram':99}

average = sum(dictname.values()) / len(dictname)

print(f" The average marks is : {average}")

highest_value = max( dictname, key = dictname.get)

print(f" the highest marks is of : {highest_value}")
```

output :-

```
 The average marks is : 73.2
 The highest marks is of : Bikram
```

Lab 2:

Date: 2082/03/17

Title: WAP to check eligibility for voting, if age >18 eligible else not.

code:

```
age = int(input('enter your age:'))
if age >=18:
    print(f" The age {age} is eligible for voting")
else:
    print(f" The age {age} is not eligible for voting")
```

output:

```
Enter your age : 21
The age 21 is eligible for voting.
```

Title: WAP to check whether student is pass or fail

code:

```
marks = int(input('enter your marks:'))
if marks >=45:
    print(f" {marks}, you are passed! ")
else:
    print(f" sorry, you failed.")
```

output:

```
enter your marks: 33
sorry, you failed.
```

Title: WAP to find grade of students based on pou.

code:

```
marks = int(input("enter your marks"))
if marks >= 90:
    print(" A ")
elif marks >= 80:
    print(" A-")
elif marks >= 70:
    print(" B ")
elif marks >= 60:
    print("B -")
```

```
elif  marks >= 50:
    print("c")
elif  marks >= 45:
    print("c-")
else:
    print(f"{marks}, you failed")
```

output:

```
    Enter your marks 98
    A
```

Title: wap to print 110 to 1010 using for and while loop.

code: [for loop]

```
for i in range(110, 1011):
    print(i)
```

code: [while loop]

```
int = 1010
while int != 1010
    print(int)
    ~~int = +1~~
    int += 1
```

Title:- wap to demonstrate function, takes two numbers as input, and return their sum, difference, product & quotient.

Code:-

```
def calc(a, b):
    sum = a+b
    sub = a-b
    mul = a*b
    div = a/b
    mod = a%b
    return  sum, sub, mul, div, mod

a = int(input("Enter first number: "))
b = int(input(" enter second number: "))
s, ssu, m, d, mod = calc(a, b)
```

```
print('sum:', s)
print(' sub:', ssu)
print('mul:', m)
print('div:', d)
print(' mod:', mod)
```

output:

```
    Enter first number : 10
    Enter second number : 2

     sum : 12
     sub : 8
     mul : 20
     div : 5
     mod : 0
```

Title: create a fnc^n that return exponent of given input

code:

```
 a = int(input('enter num1:'))
 b = int(input('enter exp value :'))

 def expon(a, b):
     exp = a**b
     return exp

 e = expon(a, b)
 print(f" exponent of {a} and {b} is :", e)
```

output:

```
    enter num1: 2
    enter exp value : 3
    exponent of 2 and 3 is : 8
```

title: wtp using recursive func^n to calc' factorial of given input.

```
~~a = int(input('enter num:'))~~
~~def fact(a):~~
    ~~fact = fact * fact(n-1)~~
    ~~return fact~~

~~value = fact(a)~~
~~print(f" The factorial of {a} is : ", value)~~
~~n = in~~
```

code:

```
n = int(input("Enter num:"))
def fact(n):
    if n==1:
        return n
    else:
        return n * fact(n-1)

value = fact(n)
print(f" The factorial of {n} is : ", value)
```

output:

```
Enter num: 4
The factorial of 4 is : 24.
```

Lab 3:

2082 / 03/08

Title: WAP to read / write a file

code: # write to file binod.txt

```
info = input("Enter the text here:")
f = open("binod", "w")
f.write(info)
f.close
print("successfully done!")
```

code:- # Read from file binod.txt

```
f = open("binod")
print(f.read())
f.close()
```

Title: wap using with statement to read / write to / from a file.

code:- # write in file using with statement

```
info = input("Enter the text here:")
with open("binod", "w") as f:
    f.write(info)
    print("success!")
```

code: # ~~write~~ Read from a file using with statement

```
with open("binod", "r") as f:
    print(f.read())
```

...ing recursive fincn to calc'

Title: wap to read and write a csv file using csv module.

code: # write to csv file named as binod.csv

```
import csv
with open ('binod.csv', 'w', newline ='') as file:
    csv_writer = csv.writer (file)
    csv_writer, writerow (['name', 'email', 'mobile']) # write header
    csv_writer. writerow (['Binod', 'binod@gmail.com', '999999']) # data
```

code: # Read the csv file

```
import csv
with open ('binod, csv', 'r', newline ='') as file:
    csv_reader = csv. reader (file)
    for item in csv_reader:
        print (item)
```

Title: wap to read and write json file using json module.

code: # write

```
import json
data = {'name': 'Binod', 'age': 21, 'city': 'dnn' }
with open ('binod.json', 'w') as file:
    json. dump( data, file, indent = 4 ) # indent for pretty
```

code: # read

```
import json
with open ('binod.json', 'r') as file:
    data = json. load (file)
    print (date)
```

Title: wap to create a method which shows student details name roll and mark as attribute and a method to calc' avg marks and call necessary fncn.

code:

```
class Student:
    def __init__ (self, name, roll_number, marks):
        self.name = name
        self. roll_number = roll_number
        self. marks = marks

    def display_info (self):
        print (f" student name : {self.name}, roll number:
        {self.roll_number}, average function : {self.average()}")

    def average (self):
        return sum (self. marks) / len ( self. marks)

s = Student ('Binod', 07, [34, 34, 54, 61, 78, 95])
s. display_info()
```

Title: create a base class vehicle with attribute like make an model, define a class car from vehicle with additional attribute like milage and capacity., create an object of child class and display their details. (inheritence).

code: vehicle.py

```
class Vehicle ():
    def __init__ (self, make, model):
        self. make = make
        self. model = model

    def show_info (self)
        print ( f" made on { self. make} and the model is
                { self.model}" }
```

car.py

```
from vehicle import Vehicle
class Car (Vehicle):
    def __init__ (self, make, model, mileage, capacity):
        super(). __init__ (make, model)
        self. mileage = mileage
~~def~~ self. capacity = capacity
    def showModC (self):
        print (f" the mileage is { self. mileage } & capacity is {self.capacity
c = Car ("2024", "tesla-7", 150, 7)
c. show_info()
c. showModC()
```