Darshan
UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Submission

24010101675 Dhruv R. Pithwa

Lab1

March 17, 2025

Python Programming - 2301CS404

Lab - 1

24010101675 Dhruv R. Pithwa

0.0.1 01) WAP to print "Hello World"

[ ]:
```
print("Hello World")
```

```
Hello World
```

0.0.2 02) WAP to print addition of two numbers with and without using input().

[2]:
```
a = int(input("Enter a number: "))
b = int(input("Enter a number: "))

print("sum:",a + b)
```

```
sum: 11
```

0.0.3 03) WAP to check the type of the variable.

[ ]:
```
l = [1,"hellow",3.14,3+4j]

for i in l:
    print(type(i))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'complex'>
```

0.0.4 04) WAP to calculate simple interest.

[4]:
```
p = int(input("Enter Principal: "))
r = float(input("Enter Interest Rate: "))
t = int(input("Enter Time(Years): "))

print("Simple Interest:",(p*r*t))
```

```
Simple Interest: 2100000.0
```

0.0.5 05) WAP to calculate area and perimeter of a circle.

[ ]:
```
r = float(input("Enter Radius: "))
print("Area of Circle:",3.14*r*r)
print("perimeter of Circle:",2*3.14*r)
```

```
Area of Circle: 94.985
perimeter of Circle: 34.54
```

0.0.6 06) WAP to calculate area of a triangle.

[6]:
```
b = int(input("Enter Base: "))
h = int(input("Enter Height: "))
print("Area of Triangle:",0.5*b*h)
```

```
Area of Triangle: 4.0
```

0.0.7 07) WAP to compute quotient and remainder.

[ ]:
```
a = int(input("Enter a number: "))
b = int(input("Enter a divisor: "))
print("Quotient:",a//b)
print("Remainder:",a%b)
```

```
Quotient: 2
Remainder: 1
```

0.0.8 08) WAP to convert degree into Fahrenheit and vice versa.

[7]:
```
c = int(input("Enter Celsius: "))
print("Fahrenheit:",(c*9/5)+32)
```

```
Fahrenheit: 86.0
```

[8]:
```
f = int(input("Enter Fahrenheit: "))
print("Celsius:",(f-32)*5/9)
```

```
Celsius: 30.0
```

0.0.9 09) WAP to find the distance between two points in 2-D space.

[ ]:
```
import math

a = int(input("Enter x1: "))
b = int(input("Enter y1: "))
x = int(input("Enter x2: "))
y = int(input("Enter y2: "))
```

```
print("Distance:",round((math.sqrt((x-a)**2+(y-b)**2)),2))
```

```
Distance: 4.24
```

0.0.10 10) WAP to print sum of n natural numbers.

[ ]:
```
sum = 0
for i in range(1,int(input("Enter a number: "))+1):
    sum += i
print("Sum :",sum)
```

```
Sum : 15
```

0.0.11 11) WAP to print sum of square of n natural numbers.

[1]:
```
sum = 0
for i in range(1,int(input("Enter a number: "))+1):
    sum += i**2
print("Sum :",sum)
```

```
Sum : 55
```

0.0.12 12) WAP to concate the first and last name of the student.

[ ]:
```
fn = input("First name: ")
ln = input("Last name: ")

print(fn+ln)
```

```
DhruvPihwa
```

0.0.13 13) WAP to swap two numbers.

[2]:
```
a = int(input("Enter a number: "))
b = int(input("Enter a number: "))

print("Before Swapping:",a,b)
a,b = b,a
print("After Swapping:",a,b)
```

```
Before Swapping: 3 8
After Swapping: 8 3
```

0.0.14 14) WAP to get the distance from user into kilometer, and convert it into meter, feet, inches and centimeter.

[3]:
```
a = float(input("Enter a num in kilometer: "))

print("In meters:",a*1000)
print("In centimeters:",a*100000)
print("In feet",a*3280.84)
print("In inches",a*39370.1)
```

```
In meters: 12000.0
In centimeters: 1200000.0
In feet 39370.08
In inches 472441.19999999995
```

0.0.15 15) WAP to get day, month and year from the user and print the date in the given format: 23-11-2024.

[9]:
```
day = int(input("Enter day: "))
month = int(input("Enter month: "))
year = int(input("Enter year: "))

print(day,month,year,sep="-")
```

```
12-11-2020
```

Lab2

March 17, 2025

0.0.1 02) WAP to check whether the given number is odd or even.

[5]:
```
print('Even' if int(input('Enter a num: ')) % 2 == 0 else 'Odd' )
```

```
Odd
```

0.0.2 03) WAP to find out largest number from given two numbers using simple if and ternary operator.

[ ]:
```
a = int(input('Enter a num: '))
b = int(input('Enter a num: '))

# if (a > b):
#     print(a)
# else:
#     print(b)
print(a if a > b else b)
```

```
6
```

0.0.3 04) WAP to find out largest number from given three numbers.

[ ]:
```
a = int(input("Enter a num"))
b = int(input("Enter a num"))
c = int(input("Enter a num"))

print(a if a > b and a > c else b if b > c else c)
```

```
11
```

0.0.4 05) WAP to check whether the given year is leap year or not.

[If a year can be divisible by 4 but not divisible by 100 then it is leap year but if it is divisible by 400 then it is leap year]

[ ]:
```
year = int(input('Enter a year'))

print(year, 'is' if year % 4 == 0 and year % 100 != 0 or year % 400 == 0 else␣
 ↪'is not','leap year' )
```

```
2000  is leap year
```

0.0.5 06) WAP in python to display the name of the day according to the number given by the user.

[ ]:
```
day = int(input('Enter a day'))

match(day):
    case 0: val = 'Sun'
    case 1: val = 'Mon'
    case 2: val = 'Tue'
    case 3: val = 'Wen'
    case 4: val = 'Thu'
    case 5: val = 'Fri'
    case 6: val = 'Sat'
    case _: val = 'Invalid Input'

print(val)
```

```
Invalid
```

0.0.6 07) WAP to implement simple calculator which performs (add,sub,mul,div) of two no. based on user input.

[1]:
```
a = float(input('Enter a num'))
b = float(input('Enter a num'))

match(input('Enter opration')):
    case '+': ans = a + b
    case '-': ans = a - b
    case '*': ans = a * b
    case '/': ans = f"{(a/b):10.2f}"
    case _  : ans = 'Invalid opration'

print(ans)
```

```
30.0
```

0.0.7 08) WAP to read marks of five subjects. Calculate percentage and print class accordingly.

Fail below 35 Pass Class between 35 to 45 Second Class between 45 to 60 First Class between 60 to 70 Distinction if more than 70

[29]:
```
def getClass(n):
    if n < 35:
        return 'Fail'

    elif n < 45:
```

```
        return 'Pass'
    elif n < 60:
        return 'Second Class'
    elif n < 70:
        return 'First Class'
    else:
        return 'Distinction'

t = int(input('Enter Total Marks per subject: '))

sum = 0
while True:
    n = int(input('Enter marks (0 to exit): '))

    if n == 0:
        print('Total Marks:', getClass(sum/t*100))
        break

    sum += n
```

```
Total Marks: Distinction
```

0.0.8 09) Three sides of a triangle are entered through the keyboard, WAP to check whether the triangle is isosceles, equilateral, scalene or right-angled triangle.

[ ]:
```
l = [int(input('Enter a num: ')) for i in range(3)]
l.sort()
print('Isosceles' if l[0] == l[1] or l[1] == l[2] or l[0] == l[2] else␣
 ↪'equilateral' if l[0] == l[1] == l[2] else 'Scalene')
print('Right Angled' if l[0]**2 + l[1]**2 == l[2]**2 else exit())
```

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

0.0.9 10) WAP to find the second largest number among three user input numbers.

[4]:
```
a = int(input('Enter a num: '))
b = int(input('Enter a num: '))
c = int(input('Enter a num: '))

print('Largest:', a if a > b and a > c else b if b > c else c)
```

```
Largest: 20
```

0.0.10 11) WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

a. First 1 to 50 units – Rs. 2.60/unit
b. Next 50 to 100 units – Rs. 3.25/unit
c. Next 100 to 200 units – Rs. 5.26/unit

d. above 200 units – Rs. 8.45/unit

[5]:
```
units = int(input('Enter Units: '))

if units <= 50:
    bill = units * 2.60
elif units <= 100:
    bill = (50 * 2.60) + ((units - 50) * 3.25)
elif units <= 200:
    bill = (50 * 2.60) + (50 * 3.25) + ((units - 100) * 5.26)
else:
    bill = (50 * 2.60) + (50 * 3.25) + (100 * 5.26) + ((units - 200) * 8.45)

print('Bill:', bill)
```

```
Bill: 130.0
```

Lab3

March 17, 2025

Python Programming - 2301CS404

Lab - 3

24010101675 Dhruv R. Pithwa

1 for and while loop

1.0.1 01) WAP to print 1 to 10.

[11]:
```
for i in range(1,11): print(i,end=' ')
```

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

1.0.2 02) WAP to print 1 to n.

[10]:
```
for i in range(1,int(input('Enter end: '))+1): print(i,end=' ')
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

1.0.3 03) WAP to print odd numbers between 1 to n.

[9]:
```
for i in range(1,int(input('Enter end: '))+1): print(i if i % 2 != 0 else␣
↪'',end=' ')
```

```
1  3  5  7  9
```

1.0.4 04) WAP to print numbers between two given numbers which is divisible by 2 but not divisible by 3.

[16]:
```
for i in range(int(input('Enter Start: ')),int(input('Enter end: '))+1):
    if (i % 2 == 0 and i % 3 != 0):
        print(i,end=' ')
```

```
2 4 8 10
```

1.0.5 05) WAP to print sum of 1 to n numbers.

[25]:
```
sum = 0
for i in range(1,int(input('Enter end: '))+1): sum += i
print(sum,end=' ')
```

```
6
```

1.0.6 06) WAP to print sum of series 1 + 4 + 9 + 16 + 25 + 36 + ...n.

[1]:
```
sum = 0
for i in range(1,int(input('Enter end: '))+1): sum += i**2
print(sum,end=' ')
```

```
385
```

1.0.7 07) WAP to print sum of series 1 − 2 + 3 − 4 + 5 − 6 + 7 ... n.

[2]:
```
sum = 0
for i in range(1,int(input('Enter end: '))+1): sum += i if i % 2 != 0 else i*-1
print(sum,end=' ')
```

```
-5
```

1.0.8 08) WAP to print multiplication table of given number.

[28]:
```
n = int(input('Enter a num: '))
for i in range(1,11): print(n,'x',i,'=',n*i)
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

1.0.9 09) WAP to find factorial of the given number.

[30]:
```
fac = 1
for i in range(1,int(input('Ender a num'))+1): fac *= i
print(fac)
```

```
120
```

1.0.10 10) WAP to find factors of the given number.

[34]:
```
n = int(input('Enter a num: '))
for i in range(1,n+1):
    if n % i == 0: print(i,end=' ')
```

```
1 2 3 6
```

1.0.11 11) WAP to find whether the given number is prime or not.

[3]:
```
n = int(input('Enter a num: '))
f = True
for i in range(2,n):
    if n % i == 0:
        f = False
        break
print('Prime' if f else 'Not Prime')
```

```
Prime
```

1.0.12 12) WAP to print sum of digits of given number.

[37]:
```
n = int(input('Enter a num: '))
sum = 0
while n > 0:
    sum += n % 10
    n //= 10
print(sum)
```

```
9
```

1.0.13 13) WAP to check whether the given number is palindrome or not

[46]:
```
n = int(input('Enter a num: '))
rev = 0
tmp = n
while n > 0:
    rev = rev*10 + n % 10
    n //= 10
print('' if rev == tmp else 'Not', 'Palindrome')
```

```
Not Palindrome
```

1.0.14 14) WAP to print GCD of given two numbers.

[51]:
```
def gcd(a,b):
    #base case
    if a == 0: return b
    if b == 0: return a
```

```
    if a == b: return a
    #recursive case
    if a > b: return gcd(a-b,b)
    return gcd(a,b-a)


a = int(input('Enter a num: '))
b = int(input('Enter a num: '))
print(gcd(a,b))
```

```
11
```

Lab4

March 17, 2025

Python Programming - 2301CS404

Lab - 4

24010101675 Dhruv R. Pithwa

1 String

1.0.1 01) WAP to check whether the given string is palindrome or not.

[4]:
```
str1 = input('Enter a string: ')
print(str1,' is ', '' if str1 == str1[::-1] else 'not ', 'a palindrome', sep='')
```

```
abab is not a palindrome
```

1.0.2 02) WAP to reverse the words in the given string.

[1]:
```
str1 = input('Enter a string: ')
print(' '.join(str1.split(' ')[::-1]))
for i in  str1.split(' '):
    print(i[::-1], end=' ')
```

```
world hellow
wolleh dlrow
```

1.0.3 03) WAP to remove ith character from given string.

[3]:
```
str1 = input('Enter a string: ')
i = int(input('Enter a index: '))
print( str1.replace(str1[i+1], '', 1) )
```

```
wadwdaw
```

1.0.4 04) WAP to find length of string without using len function.

[5]:
```
str1 = input('Enter a string: ')
c = 0
for i in str1: c += 1
print(c)
```

6

1.0.5 05) WAP to print even length word in string.

[6]:
```
str1 = input('Enter a string: ')
for i in str1.split(' '):
    if (len(i)%2 == 0):
        print(i, end=' ')
```

```
wdadwa dwad
```

1.0.6 06) WAP to count numbers of vowels in given string.

[7]:
```
str1 = input('Enter a string: ')
vowel = ['a', 'e', 'i', 'o', 'u']
c = 0
for i in str1:
    if i in vowel:
        c += 1
print(c, 'vowels in', str1)
```

```
6 vowels in oiwabcycaopqwdo
```

1.0.7 07) WAP to capitalize the first and last character of each word in a string.

[ ]:
```
str1 = input('Enter a string: ')
for i in str1.split(' '):
    i = i[0].upper() + i[1:-1] +i[-1].upper()
    print(i,end=' ')
```

```
HhhH WwwW
```

1.0.8 08) WAP to convert given array to string.

[ ]:
```
arr = [1, 2, 3, 4, 5]

print(''.join(map(str, arr)))
```

```
12345
```

1.0.9 09) Check if the password and confirm password is same or not.

1.0.10 In case of only case's mistake, show the error message.

[ ]:
```
pas = input('Enter a password: ').strip()
cpas = input('Enter confirm password: ').strip()
flag = True
for i in pas:
    if i not in cpas:
        flag = False
```

```
        print('Password not match at', pas.index(i))
        break
if flag: print('Password match')
```

```
Password match
```

1.0.11 10) : Display credit card number.

1.0.12 card no. : 1234 5678 9012 3456

1.0.13 display as : **** **** **** 3456

[ ]:
```
cardno = input('Enter a card number (space seprated)')

cardno = cardno.split(' ')
result = ''

for i in range(len(cardno)-1):
    i = cardno[i]
    result += i.replace(i[::], '*'*len(i)+' ')

result += cardno[-1]
print(result)
```

```
**** **** **** 2313
```

1.0.14 11) : Checking if the two strings are Anagram or not.

1.0.15 s1 = decimal and s2 = medical are Anagram

[ ]:
```
str1 = input('Enter a string: ')
str2 = input('Enter a string: ')

print(str1,'and',str2,'are','Anagram' if sorted(str1) == sorted(str2) else 'Not␣
 ↪Anagram')
```

```
aab and abb are Not Anagram
```

1.0.16 12) : Rearrange the given string. First lowercase then uppercase alphabets.

1.0.17 input : EHlsarwiwhtwMV

1.0.18 output : lsarwiwhtwEHMV

[ ]:
```
str1 = input('Enter a string: ')
up = ''
low = ''

for i in str1:
    if i.isupper(): up += i
    else: low += i
```

```
print(low + up)
```

```
asdasdaHHASASA
```

Lab5

March 17, 2025

Python Programming - 2301CS404

Lab - 5

24010101675 Dhruv R. Pithwa

1 List

1.0.1 01) WAP to find sum of all the elements in a List.

[21]:
```
ls = [1,2,3,4,5,6,7,8,9,10]

def udf_sum(ls):
    sum = 0
    for i in ls: sum += i
    return sum

# print(sum(ls))
print(udf_sum(ls))
```

```
55
```

1.0.2 02) WAP to find largest element in a List.

[7]:
```
# print(max(ls))

def udf_max(ls):
    max = 0
    for i in ls:
        if max < i: max = i
    return max

print(udf_max(ls))
```

```
10
```

1.0.3 03) WAP to find the length of a List.

[8]:
```
print(len(ls))
```

```
10
```

1.0.4 04) WAP to interchange first and last elements in a list.

[9]:
```
tmp = ls[0]
ls[0] = ls[len(ls)-1]
ls[len(ls)-1] = tmp

print(ls)
```

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

1.0.5 05) WAP to split the List into two parts and append the first part to the end.

[14]:
```
p1 = ls[:len(ls)//2]
p2 = ls[len(ls)//2:]
ls = p2 + p1
print(ls)
```

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

1.0.6 06) WAP to interchange the elements on two positions entered by a user.

[15]:
```
def udf_changeElementPos(m,n,ls):
    tmp = ls[m]
    ls[m] = ls[n]
    ls[n] = tmp
    return ls

print(udf_changeElementPos(int(input('Enter from(index)')),int(input('Enter␣
 ↪to(index)')),ls))
```

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

1.0.7 07) WAP to reverse the list entered by user.

[20]:
```
ls = []
while(True):
    val = input('Enter nums (q to quite)')
    if val == 'q':
        break
    else:
        ls.append(int(val))

ls.reverse()
```

```
print(ls)
```

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

1.0.8 08) WAP to print even numbers in a list.

[22]:
```
for i in ls:
    if i%2 == 0:
        print(i,end=' ')
```

```
2 4 6 8 10
```

1.0.9 09) WAP to count unique items in a list.

[2]:
```
ls = [11,22,11,22,33,44,44,55]

def findUnique(ls):
    unique = []
    for i in ls:
        if i in ls and i not in unique:
            unique.append(i)
    return len(unique)

print(findUnique(ls))
```

```
5
```

1.0.10 10) WAP to copy a list.

[3]:
```
ls2 = ls.copy()

print(ls2)
```

```
[11, 22, 11, 22, 33, 44, 44, 55]
```

1.0.11 11) WAP to print all odd numbers in a given range.

[40]:
```
ls = [i for i in range(int(input('Enter from(idex): ')),int(input('Enter␣
 ↪to(index): '))) if i%2 != 0]
print(ls)
```

```
[1, 3, 5, 7, 9]
```

1.0.12 12) WAP to count occurrences of an element in a list.

[67]:
```
ls = [11,22,11,22,33,44,44,55,11]

def udf_findOccurrence(ls):
    result = {}
    for i in ls:
        if i not in result:
            result[i] = 1
        else:
            result[i] += 1

    return result

print(udf_findOccurrence(ls))
```

```
{11: 3, 22: 2, 33: 1, 44: 2, 55: 1}
```

1.0.13 13) WAP to find second largest number in a list.

[57]:
```
ls = [1,2,3,4,5,6,7,8,9]

def udf_findSecondMax(ls):
    ls.remove(max(ls))
    return max(ls)

print(udf_findSecondMax(ls))
```

```
8
```

1.0.14 14) WAP to extract elements with frequency greater than K.

[85]:
```
ls = [11,22,11,22,33,44,44,55,11,22]

def udf_findGreaterthanK(ls,k):
    tmp = udf_findOccurrence(ls)
    result = {}
    for i in tmp:
        if tmp[i] > k:
            result[i] = tmp[i]
    return result

print(udf_findGreaterthanK(ls,2))

def udf_findGreaterthanKCount(ls,k):
    result = set()
    for i in ls:
        if ls.count(i) > k:
```

```
            result.add(i)
    return list(result)

print(udf_findGreaterthanKCount(ls,2))
```

```
{11: 3, 22: 3}
[11, 22]
```

1.0.15 15) WAP to create a list of squared numbers from 0 to 9 with and without using List Comprehension.

[73]:
```
ls = []
for i in range(10):
    ls.append(i**2)

print(ls)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

1.0.16 16) WAP to create a new list (fruit whose name starts with 'b') from the list of fruits given by user.

[87]:
```
ls = []
while(True):
    val = input('Enter nums (q to quite)')
    if val == 'q':
        break
    else:
        if val.startswith('b'):
            ls.append(val)

print(ls)
```

```
['banana', 'brownie']
```

1.0.17 17) WAP to create a list of common elements from given two lists.

[2]:
```
ls = [1,2,3,4,5]
ls2 = [4,5,6,7,8]

def udf_findCommon(ls,ls2):
    # result = []
    # for i in ls:
    #     if i in ls2:
    #         result.append(i)
    return [i for i in ls if i in ls2]

print(udf_findCommon(ls,ls2))
```

```
[4, 5]
```

Lab6

March 17, 2025

Python Programming - 2301CS404

Lab - 6

24010101675 Dhruv R. Pithwa

1 Tuple

1.0.1 01) WAP to find sum of tuple elements.

[2]:
```
tp = (1,2,3,4)

def sum_tuple(tp):
    return sum(tp)

print(sum_tuple(tp))
```

```
10
```

1.0.2 02) WAP to find Maximum and Minimum K elements in a given tuple.

[28]:
```
tp = (2,1,4,3,5,6,7,8,9,10)

def k_min_max(tp, k):
    tp = list(tp)
    tp.sort()
    tp = tuple(tp)
    return tp[:k] + tp[-k:]

print(k_min_max(tp, 3))
```

```
(1, 2, 3, 8, 9, 10)
(2, 1, 4, 3, 5, 6, 7, 8, 9, 10)
```

1.0.3 03) WAP to find tuples which have all elements divisible by K from a list of tuples.

[32]:
```
def divisible_byK(tp, k):
    return tuple(filter(lambda x: x % k == 0, tp))

print(divisible_byK(tp, 2))
```

```
(2, 4, 6, 8, 10)
```

1.0.4 04) WAP to create a list of tuples from given list having number and its cube in each tuple.

[54]:
```
ls = [1,2,3,4,5]

def cube(ls):
    return list((i,i ** 3) for i in ls)

print(cube(ls))
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
```

1.0.5 05) WAP to find tuples with all positive elements from the given list of tuples.

[41]:
```
ls = [(1,-2,3), (4,5,6), (-7,8,9), (10,11,12)]

def find_positive_tuple(ls):
    return list(filter(lambda x: all(map(lambda y: y > 0, x)), ls))

print(find_positive_tuple(ls))
```

```
[(4, 5, 6), (10, 11, 12)]
```

1.0.6 06) WAP to add tuple to list and vice – versa.

[59]:
```
tp = (1,2)
ls = [3,4]

def add_elements(tp, ls):
    a = tp + tuple(ls)
    b = ls + list(tp)
    return a, b

print(add_elements(tp, ls))
```

```
((1, 2, 3, 4), [3, 4, 1, 2])
```

1.0.7 07) WAP to remove tuples of length K.

[64]:
```
ls = [(1,2),(3,4,5),(6,7,8,9),(10,11,12,13,14)]

def remove_Klength(ls, k):
    return list(filter(lambda x: len(x) != k, ls))

print(remove_Klength(ls, 2))
```

```
[(3, 4, 5), (6, 7, 8, 9), (10, 11, 12, 13, 14)]
```

1.0.8 08) WAP to remove duplicates from tuple.

[65]:
```
tp = (1,1,2,2,3,3,4,4,5,5)

def remove_duplicates(tp):
    return tuple(set(tp))

print(remove_duplicates(tp))
```

```
(1, 2, 3, 4, 5)
```

1.0.9 09) WAP to multiply adjacent elements of a tuple and print that resultant tuple.

[67]:
```
tp = (1,2,3,4,5,6)

def multiply_adjacent(tp):
    return tuple(tp[i] * tp[i+1] for i in range(len(tp)-1))

print(multiply_adjacent(tp))
```

```
(2, 6, 12, 20, 30)
```

1.0.10 10) WAP to test if the given tuple is distinct or not.

[72]:
```
tp = (1,1,2,2,3,3,4,4,5,5)
tp2 = (1,2,3,4,5,6)

def distiinct_tuple(tup):
    tmp = tuple(set(tup))
    return True if len(tmp) == len(tup) else False

print(distiinct_tuple(tp2))
```

```
True
```

Lab7

March 17, 2025

Python Programming - 2301CS404

Lab - 7

24010101675 Dhruv R. Pithwa

1 Set & Dictionary

1.0.1 01) WAP to iterate over a set.

[26]:
```
s = {1,2,3,4,5}

for i in s:
    print(i, end=' ')
```

```
1 2 3 4 5
```

1.0.2 02) WAP to convert set into list, string and tuple.

[29]:
```
ls = list(s)
tp = tuple(s)
string = ','.join(str(i) for i in s)

print(ls)
print(tp)
print(string)
```

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

1.0.3 03) WAP to find Maximum and Minimum from a set.

[30]:
```
max = max(s)
min = min(s)

print(max)
print(min)
```

```
5
1
```

1.0.4 04) WAP to perform union of two sets.

[31]:
```
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

print(s1.union(s2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

1.0.5 05) WAP to check if two lists have at-least one element common.

[37]:
```
ls1 = [1,2,3,4,5]
ls2 = [4,5,6,7,8]

def filter_list(ls1, ls2):
    ans = set(ls1).intersection(set(ls2))
    if len(ans) > 0:
        print("Common elements are:", ans)
        return
    print("No common elements")

filter_list(ls1, ls2)
```

```
Common elements are: {4, 5}
```

1.0.6 06) WAP to remove duplicates from list.

[42]:
```
ls = [1,2,2,3,4,5]

def remove_duplicates(ls):
    return list(set(ls))
print(remove_duplicates(ls))
```

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

1.0.7 07) WAP to find unique words in the given string.

[54]:
```
String = "Hello World Hello Python"
unique_words = set(String.split())
print(unique_words)
```

```
{'Python', 'Hello', 'World'}
```

1.0.8 08) WAP to remove common elements of set A & B from set A.

[55]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}

print(A.difference(B))
```

```
{1, 2, 3}
```

1.0.9 09) WAP to check whether two given strings are anagram or not using set.

[74]:
```
strA = 'aab'
strB = 'abb'

def udf_findOccurrence(el):
    result = {}
    for i in el:
        if i not in result:
            result[i] = 1
        else:
            result[i] += 1

    return result

def isAnagram(strA, strB):
    return udf_findOccurrence(strA) == udf_findOccurrence(strB)

print(isAnagram(strA, strB))
```

```
False
```

1.0.10 10) WAP to find common elements in three lists using set.

[76]:
```
ls1 = [1,2,3,4,5]
ls2 = [4,5,6,7,8,1]
ls3 = [1,2,9,4,5]

def find_common(ls1, ls2, ls3):
    return set(ls1).intersection(set(ls2), set(ls3))

print(find_common(ls1, ls2, ls3))
```

```
{1, 4, 5}
```

1.0.11 11) WAP to count number of vowels in given string using set.

[80]:
```
string = "Hello World"

def find_num_of_vowels(string):
    vowels = {'a','e','i','o','u'}
    count = 0
    for i in string:
        if i in vowels:
            count += 1
    return count

print(find_num_of_vowels(string))
```

```
3
```

1.0.12 12) WAP to check if a given string is binary string or not.

[2]:
```
string = "010101010101"

def is_binary(string):
    return set(string) == {'0','1'}

print(is_binary(string))
```

```
True
```

1.0.13 13) WAP to sort dictionary by key or value.

[6]:
```
dict1 = {2:40, 3:20, 1:30, 4:10}

def sort_dict(d, by_key=True):
    if by_key:
        return dict(sorted(d.items()))
    else:
        return dict(sorted(d.items(), key=lambda item: item[1]))

print("Sorted by key:", sort_dict(dict1, by_key=True))
print("Sorted by value:", sort_dict(dict1, by_key=False))
```

```
Sorted by key: {1: 30, 2: 40, 3: 20, 4: 10}
Sorted by value: {4: 10, 3: 20, 1: 30, 2: 40}
```

1.0.14 14) WAP to find the sum of all items (values) in a dictionary given by user. (Assume: values are numeric)

[7]:
```
def create_dict():
    key = 1
    dict1 = {}
    while True:
        val = input("Enter a number (q to quit): ")
        if val == 'q':
            break
        dict1[key] = int(val)
        key += 1
    return dict1

dict1 = create_dict()

print(dict1)
print('Sum:', sum(dict1.values()))
```

```
{1: 5, 2: 6, 3: 7, 4: 8}
Sum: 26
```

1.0.15 15) WAP to handle missing keys in dictionaries.

Example : Given, dict1 = {'a': 5, 'c': 8, 'e': 2}

if you look for key = 'd', the message given should be 'Key Not Found', otherwise print the value of 'd' in dict1.

[10]:
```
dect1 = {'a': 100, 'b': 200, 'c': 300}

def find_key(d, key):
    if key in d:
        return {key: d[key]}
    return 'Key not found'

print(find_key(dect1, 'd'))
```

```
Key not found
```

Lab8

March 17, 2025

Python Programming - 2301CS404

Lab - 8

24010101675 Dhruv R. Pithwa

1 User Defined Function

1.0.1 01) Write a function to calculate BMI given mass and height. (BMI = mass/h**2)

[2]:
```
def bmi(wight, height):
    return wight // (height ** 2)

print("Your BMI is: ", bmi(52, 1.65))
```

```
Your BMI is:  19.0
```

1.0.2 02) Write a function that add first n numbers.

[5]:
```
def sum_n(n):
    return sum(range(1, n+1))

print("The sum of the first 10 numbers is: ", sum_n(5))
```

```
The sum of the first 10 numbers is:  15
```

1.0.3 03) Write a function that returns 1 if the given number is Prime or 0 otherwise.

[10]:
```
def is_prime(n):
    for i in range(2, n//2+1):
        if n % i == 0:
            return 0
    return 1

print(is_prime(83))
```

```
1
```

1.0.4 04) Write a function that returns the list of Prime numbers between given two numbers.

[12]:
```
def list_primes(n,m):
    return [x for x in range(n, m+1) if is_prime(x)]

print(list_primes(1, 100))
```

```
[1, 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]
```

1.0.5 05) Write a function that returns True if the given string is Palindrome or False otherwise.

[13]:
```
def is_palindrome(s):
    return s == s[::-1]

print(is_palindrome("racecar"))
print(is_palindrome("hello"))
```

```
True
False
```

1.0.6 06) Write a function that returns the sum of all the elements of the list.

[15]:
```
def sum_of_list(lst):
    sum = 0
    for i in lst: sum += i
    return sum

print(sum_of_list([1, 2, 3, 4, 5]))
```

```
15
```

1.0.7 07) Write a function to calculate the sum of the first element of each tuples inside the list.

[16]:
```
lst_tup = [(1, 2), (3, 4), (5, 6)]

def sum_of_tuple(lst):
    sum = 0
    for i in lst:
        sum += i[0]
    return sum

print(sum_of_tuple(lst_tup))
```

```
9
```

1.0.8 08) Write a recursive function to find nth term of Fibonacci Series.

[20]:
```
def fibo_recursion(n):
    if n <= 1:
        return n
    else:
        return fibo_recursion(n-1) + fibo_recursion(n-2)

print(fibo_recursion(5))
```

```
0
```

1.0.9 09) Write a function to get the name of the student based on the given rollno.

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

[21]:
```
stutents = {101: "Ajay", 102: "Rahul", 103: "Sonam", 104: "Raj"}

def find_student(id):
    if id in stutents:
        return stutents[id]
    else:
        return "Student not found"

print(find_student(101))
```

```
Ajay
```

1.0.10 10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

[22]:
```
score = [100,200,434,654,234,400]

def sum_of_tens_zeros(lst):
    sum = 0
    for i in lst:
        if i % 10 == 0:
            sum += i
    return sum

print(sum_of_tens_zeros(score))
```

```
700
```

1.0.11 11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

Before : {'a': 10, 'b':20, 'c':30, 'd':40}

After : {10:'a', 20:'b', 30:'c', 40:'d'}

[23]:
```
dict1 = {'a': 100, 'b': 200, 'c': 300}

def invert_dict(d):
    return {v: k for k, v in d.items()}

print(invert_dict(dict1))
```

```
{100: 'a', 200: 'b', 300: 'c'}
```

1.0.12 12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

[27]:
```
def is_pangram(s):
    return len(set(s)) == 26

print(is_pangram("The quick brown fox jumps over the lazy dog"))
```

```
False
```

1.0.13 13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

[29]:
```
sentence = "The Quick Brown Fox Jumps Over The Lazy Dog"

def count_upper_lower(s):
    upper = 0
    lower = 0
    for i in s:
        if i.isupper():
            upper += 1
        elif i.islower():
            lower += 1
    return upper, lower

print(count_upper_lower(sentence))
```

```
(9, 26)
```

1.0.14 14) Write a lambda function to get smallest number from the given two numbers.

[30]:
```
minNum = lambda x, y: x if x < y else y

print(minNum(10, 20))
```

```
10
```

1.0.15 15) For the given list of names of students, extract the names having more that 7 characters. Use filter().

[32]:
```
stutents = ['Ajay', 'Rahul', 'Raghunath', 'Chandrika','Suryakant']
list(filter(lambda name: len(name) > 7, stutents))
```

[32]:
```
['Raghunath', 'Chandrika', 'Suryakant']
```

1.0.16 16) For the given list of names of students, convert the first letter of all the names into uppercase. use map().

[34]:
```
list(map(lambda name: name.title(), ['ajay', 'rahul', 'raghunath',␣
 ↪'chandrika','suryakant']))
```

[34]:
```
['Ajay', 'Rahul', 'Raghunath', 'Chandrika', 'Suryakant']
```

1.0.17 17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)
5. Keyword-Only & Positional Only Arguments

[7]:
```
# 1. Positional Arguments
def positional_args(a, b):
    return a + b

print('positional_args 10, 20: (sum)',positional_args(10, 20))

# 2. Keyword Arguments
def keyword_args(a, b):
    return a * b

print('\nkeyword_args a=10,b=20: (mul)',keyword_args(a=10, b=20))

# 3. Default Arguments
def default_args(a, b=5):
    return a - b
```

```
print('\ndefault_args 10 default 5 : (sub)',default_args(10))
print('default_args 10, 20 : (sub)',default_args(10, 20))

# 4. Variable Length Positional Arguments
def var_args(*args):
    return sum(args)

print('\nvar_args 10, 20, 30, 40, 50: (sum)',var_args(10, 20, 30, 40, 50))


# 4. Variable Length Keyword Arguments
def var_kwargs(**kwargs):
    return kwargs

print('\nvar_kwargs a=10, b=20, c=30, d=40, e=50: (dict)',var_kwargs(a=10,␣
 ↪b=20, c=30, d=40, e=50))

# 5. Keyword Arguments and Positional Arguments
def pow_positonal_kwargs(a, **kwargs):
    return {k: v**a for k, v in kwargs.items()}

print('\npow_positonal_kwargs 5, b=2, c=3, d=4, e=5:␣
 ↪(dict)',pow_positonal_kwargs(5, b=2, c=3, d=4, e=5))
```

```
positional_args 10, 20: (sum) 30

keyword_args a=10,b=20: (mul) 200

default_args 10 default 5 : (sub) 5
default_args 10, 20 : (sub) -10

var_args 10, 20, 30, 40, 50: (sum) 150

var_kwargs a=10, b=20, c=30, d=40, e=50: (dict) {'a': 10, 'b': 20, 'c': 30, 'd':
40, 'e': 50}

pow_positonal_kwargs 5, b=2, c=3, d=4, e=5: (dict) {'b': 32, 'c': 243, 'd':
1024, 'e': 3125}
```

Lab9

March 17, 2025

Python Programming - 2301CS404

Lab - 9

24010101675 Dhruv R. Pithwa

1 File I/O

1.0.1 01) WAP to read and display the contents of a text file. (also try to open the file in some other directory)

- in the form of a string

- line by line

- in the form of a list

[53]:
```
fp = open('pub/file.txt', 'r')
print('String:')
print(fp.read())
fp.seek(0)
print('\nline by line:')
print(fp.readline())
print(fp.readline())
print(fp.readline())
fp.seek(0)
print('\nin list:')
print(fp.readlines())
fp.close()
```

```
String:
This is a sample text file.
You can write any text you want here.
Feel free to add more lines as needed.
This is just an example.
Another Content Writing

line by line:
This is a sample text file.
```

```
You can write any text you want here.

Feel free to add more lines as needed.


in list:
['This is a sample text file.\n', 'You can write any text you want here.\n',
'Feel free to add more lines as needed.\n', 'This is just an example.\n',
'Another Content Writing ']
```

1.0.2 02) WAP to create file named "new.txt" only if it doesn't exist.

[1]:
```
fp = open('pub/new.txt', 'w')
fp.write('Hello, world!')
fp.close()
```

1.0.3 03) WAP to read first 5 lines from the text file.

[2]:
```
fp = open('pub/file.txt', 'r')
for i in range(5):
    print(fp.readline(), end='')
fp.close()
```

```
This is a sample text file.
You can write any text you want here.
Feel free to add more lines as needed.
This is just an example.
Another Content Writing
```

1.0.4 04) WAP to find the longest word(s) in a file

[3]:
```
fp = open('pub/file.txt', 'r')
words = fp.read().replace('.','').split()
max_len = len(max(words, key=len))
print(max_len)
print([word for word in words if len(word) == max_len])
fp.close()
```

```
7
['example', 'Another', 'Content', 'Writing']
```

1.0.5 05) WAP to count the no. of lines, words and characters in a given text file.

[49]:
```
fp = open('pub/file.txt', 'r')
lines = len(fp.readlines())
fp.seek(0)
words = len(fp.read().split())
fp.seek(0)
```

```
chars = len(''.join(fp.read().replace('.','').split()))
print('lines:', lines, 'words:', words, 'chars:', chars)
fp.close()
```

```
lines: 5 words: 30 chars: 120
```

1.0.6 06) WAP to copy the content of a file to the another file.

[63]:
```
with open('pub/file.txt', 'r') as fp:
    content = fp.read()
    cp = open(fp.name.replace('.txt',' copy.txt'), 'w+')
    cp.write(content)
    cp.seek(0)
    print(cp.read())
    cp.close()
```

```
This is a sample text file.
You can write any text you want here.
Feel free to add more lines as needed.
This is just an example.
Another Content Writing
```

1.0.7 07) WAP to find the size of the text file.

[68]:
```
import os

fileSize = os.stat('pub/file.txt').st_size
print('file size in kb: {:.3f}'.format(fileSize/1024))
```

```
file size in kb: 0.154
```

1.0.8 08) WAP to create an UDF named frequency to count occurances of the specific word in a given text file.

[70]:
```
def frequency(file,word):
    with open(file, 'r') as fp:
        content = fp.read().replace('.','').split()
        return content.count(word)

print(frequency('pub/file.txt', 'Python'))
```

```
5
```

1.0.9 09) WAP to get the score of five subjects from the user, store them in a file. Fetch those marks and find the highest score.

[ ]:
```
def set_scores():
    while True:
        score = input('Enter score (q to quit): ')
        if score == 'q':
            break
        with open('pub/scores.txt', 'a') as fp:
            fp.write(score+', ')

def max_score(file):
    with open(file, 'r') as fp:
        scores = list(map(int, fp.read().replace(',',' ').split()))
        return max(scores)

# set_scores()
print(max_score('pub/scores.txt'))
```

```
120
```

1.0.10 10) WAP to write first 100 prime numbers to a file named primenumbers.txt

(Note: each number should be in new line)

[ ]:
```
def is_prime(n):
    for i in range(2, n//2+1):
        if n % i == 0:
            return 0
    return 1

def first_n_primes(n):
    primes = []
    i = 2
    while len(primes) < n:
        if is_prime(i):
            primes.append(i)
        i += 1
    return primes

def write_primes(file, n):
    with open(file, 'w') as fp:
        for prime in first_n_primes(n):
            fp.write(str(prime)+'\n')

def read_primes(file):
    with open(file, 'r') as fp:
        return list(map(int, fp.readlines()))
```

```
write_primes('pub/primenumbers.txt', 5)
print(read_primes('pub/primenumbers.txt'))
```

```
[2, 3, 5, 7, 11]
```

1.0.11 11) WAP to merge two files and write it in a new file.

[84]:
```
def merge_files(file1, file2, file3):
    with open(file1, 'r') as fp1, open(file2, 'r') as fp2, open(file3, 'w') as␣
 ↪fp3:
        fp3.write(fp1.read() + '\n' +fp2.read())

def read_file(file):
    with open(file, 'r') as fp:
        print(fp.read())

merge_files('pub/file.txt', 'pub/new.txt', 'pub/merged.txt')
read_file('pub/merged.txt')
```

```
This is a sample text file.
You can write any text you want here.
Feel free to add more lines as needed.
This is just an example.
Another Content Writing
Python Python Python Python Python
Hello, world!
```

1.0.12 12) WAP to replace word1 by word2 of a text file. Write the updated data to new file.

[85]:
```
def update_file(file, old, new):
    with open(file, 'r') as fp:
        content = fp.read()
    with open(fp.name.replace('.txt',' updated.txt'), 'w') as fp:
        fp.write(content.replace(old, new))

update_file('pub/file.txt', 'Python', 'Java')
read_file('pub/file updated.txt')
```

```
This is a sample text file.
You can write any text you want here.
Feel free to add more lines as needed.
This is just an example.
Another Content Writing
Java Java Java Java Java
```

1.0.13 13) Demonstrate tell() and seek() for all the cases(seek from beginning-end-current position) taking a suitable example of your choice.

[93]:
```
with open('pub/file.txt', 'rb') as fp:
    print('cursor position (tell):', fp.tell())
    print('cursor position (seek from start):', fp.seek(5,0))
    print('cursor position (seek from current):', fp.seek(5,1))
    print('cursor position (seek from end):', fp.seek(-5,2))
    print(fp.read().decode())
```

```
cursor position (tell): 0
cursor position (seek from start): 5
cursor position (seek from current): 10
cursor position (seek from end): 188
ython
```

Lab10

March 17, 2025

Python Programming - 2301CS404

Lab - 10

24010101675 Dhruv R. Pithwa

1 Exception Handling

1.0.1 01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError #### Note: handle them using separate except blocks and also using single except block too.

[4]:
```
try:
    print(5/0)
except ZeroDivisionError as e:
    print(repr(e))

try:
    print(int('a'))
except ValueError as e:
    print(repr(e))

try:
    print('a'+5)
except TypeError as e:
    print(repr(e))
```

```
ZeroDivisionError('division by zero')
ValueError("invalid literal for int() with base 10: 'a'")
TypeError('can only concatenate str (not "int") to str')
```

1.0.2 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

[5]:
```
try:
    lst = [1, 2, 3]
    print(lst[3])
except IndexError as e:
    print(repr(e))

try:
    d = {1: 2}
    print(d[3])
except KeyError as e:
    print('Invalid key',repr(e))
```

```
IndexError('list index out of range')
KeyError(3)
```

1.0.3 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

[9]:
```
try:
    fp = open('unknown.txt')
except FileNotFoundError as e:
    print(repr(e))
else:
    fp.close()

try:
    import unknown
except ModuleNotFoundError as e:
    print(repr(e))
```

```
FileNotFoundError(2, 'No such file or directory')
ModuleNotFoundError("No module named 'unknown'")
```

1.0.4 04) WAP that catches all type of exceptions in a single except block.

[10]:
```
try:
    print(unknown)
except Exception as e:
    print(repr(e))
```

```
NameError("name 'unknown' is not defined")
```

1.0.5 05) WAP to demonstrate else and finally block.

[13]:
```
try:
    # print('unknown')
    print(unknown)
except Exception as e:
    print(repr(e))
else:
    print('else: This is executed if no exception is raised')
finally:
    print('finally: This is always executed')
```

```
NameError("name 'unknown' is not defined")
finally: This is always executed
```

1.0.6 06) Create a short program that prompts the user for a list of grades separated by commas.

1.0.7 Split the string into individual grades and use a list comprehension to convert each string to an integer.

1.0.8 You should use a try statement to inform the user when the values they entered cannot be converted.

[16]:
```
try:
    grades = input('Enter grades (comma seprated): ')
    grades = grades.split(',')
    grades = [int(grade) for grade in grades]
except (ValueError,Exception) as e:
    print('Invalid input Enter comma separated integers only')
```

```
invalid literal for int() with base 10: 'abc'
```

1.0.9 07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

[19]:
```
def divide(a,b):
    try:
        return a/b
    except ZeroDivisionError as e:
        print('Division by zero is not allowed')
        b = int(input('Enter a non-zero number: '))
        return divide(a,b)

print(divide(int(input('Enter base')),int(input('Enter divisor'))))
```

```
2.0
```

1.0.10 08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

[22]:
```
try:
    age = int(input('Enter age: '))
    if age < 18:
        raise ValueError('Enter age greater than 18')
except ValueError as e:
    print(e)
else:
    print('Age:',age)
```

```
Age: 20
```

1.0.11 09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

[30]:
```
class InvalidUsernameError(Exception):
    pass

try:
    username = input('Enter username: ')
    if len(username) < 5 or len(username) > 15:
        raise InvalidUsernameError
except InvalidUsernameError:
    print('Username must be between 5 and 15 characters long : Number of␣
 ↪characters',len(username),'Username:',username)
else:
    print('Username:',username)
```

```
Username: username
```

1.0.12 10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

[34]:
```
class NegativeNumberError(Exception):
    def __init__(self, msg='Cannot calculate the square root of a negative␣
 ↪number'):
        super().__init__(msg)
```

```
def square_root(num):
    try:
        if num < 0:
            raise NegativeNumberError
    except NegativeNumberError as e:
        print(e)
    else:
        return num**0.5

print(square_root(int(input('Enter a number: '))))
```

```
2.449489742783178
```

Lab11

March 17, 2025

Python Programming - 2301CS404

Lab - 11

24010101675 Dhruv R. Pithwa

1 Modules

1.0.1 01) WAP to create Calculator module which defines functions like add, sub,mul and div.

1.0.2 Create another .py file that uses the functions available in Calculator module.

[ ]:
```python
from lab11 import Calculator

print(Calculator.__doc__)
print('-' * 50)

print('Add:', Calculator.add(1, 2, 3))
print('Sub:', Calculator.subtract(1, 2, 3))
print('Mul:', Calculator.multiply(1, 2, 3))
print(f'Div: {Calculator.divide(1, 2):.2f}')
```

```
Module: Calculator
This module provides a calculator with basic arithmetic operations.
add(*args) - returns the sum of all arguments
subtract(*args) - returns the difference of all arguments
multiply(*args) - returns the product of all arguments
divide(a, b) - returns the quotient of a and b

--------------------------------------------------
Add: 6
Sub: -4
Mul: 6
Div: 0.50
```

1.0.3 02) WAP to pick a random character from a given String.

[2]:
```
import random

def random_char(string):
    return random.choice(string)

print('Random char:', random_char('Hellow World'))
```

```
Random char: l
```

1.0.4 03) WAP to pick a random element from a given list.

[3]:
```
def random_element(iterable):
    return random.choice(iterable)

print('Random element:', random_element([1,2,3,4,5]))
```

```
Random element: 1
```

1.0.5 04) WAP to roll a dice in such a way that every time you get the same number.

[4]:
```
def roll_dice():
    random.seed(1)
    return random.randint(1, 6)

print('Rolling dice:', roll_dice())
```

```
Rolling dice: 2
```

1.0.6 05) WAP to generate 3 random integers between 100 and 999 which is divisible by 5.

[5]:
```
def random_num():
    return tuple(random.choice(range(100, 1000, 5)) for _ in range(3))

print('3 Random between 100 and 999 / 5:', random_num())
```

```
3 Random between 100 and 999 / 5: (825, 180, 425)
```

1.0.7 06) WAP to generate 100 random lottery tickets and pick two lucky tickets from it and announce them as Winner and Runner up respectively.

[6]:
```
def lottery():
    ticket = [random.choice(range(10*5000, 50*5000)) for _ in range(100)]
    winner,runnerup = random.sample(ticket, 2)
    print('Winner:', winner)
    print('Runner-up:', runnerup)
```

```
lottery()
```

```
Winner: 167831
Runner-up: 137214
```

1.0.8 07) WAP to print current date and time in Python.

[7]:
```
import datetime

print('Current date:', datetime.datetime.now().strftime('%d-%m-%Y'))
```

```
Current date: 26-02-2025
```

1.0.9 08) Subtract a week (7 days) from a given date in Python.

[8]:
```
def subtract_week(date):
    return date - datetime.timedelta(weeks=1)

print('Subtract week:', subtract_week(datetime.datetime.now()).
 ↪strftime('%d-%m-%Y'))
```

```
Subtract week: 19-02-2025
```

1.0.10 09) WAP to Calculate number of days between two given dates.

[9]:
```
print('Date difference:', (datetime.datetime(2025, 1, 1) - datetime.
 ↪datetime(2019, 1, 1)).days, 'days')
```

```
Date difference: 2192 days
```

1.0.11 10) WAP to Find the day of the week of a given date.(i.e. wether it is sunday/monday/tuesday/etc.)

[10]:
```
print(datetime.datetime.now().strftime('%d %A, %b-%Y'))
```

```
26 Wednesday, Feb-2025
```

1.0.12 11) WAP to demonstrate the use of date time module.

[11]:
```
print(datetime.date(day=15, month=1, year=2025))
print(datetime.time(hour=10, minute=30, second=15))
print(datetime.datetime(year=2025, month=1, day=15, hour=10, minute=30,␣
 ↪second=15))
```

```
2025-01-15
10:30:15
2025-01-15 10:30:15
```

1.0.13 12) WAP to demonstrate the use of the math module.

[12]:
```
import math

print(f'Pi: {math.pi:.2f}')
print(f'e: {math.e:.2f}')
print('Mod:', math.remainder(10, 3))
print('Factorial:', math.factorial(5))
print('GCD:', math.gcd(10, 15))
print('Exp:', math.exp(0))
print('pow:', math.pow(2, 10))
print('sqrt:', math.sqrt(25))
```

```
Pi: 3.14
e: 2.72
Mod: 1.0
Factorial: 120
GCD: 5
Exp: 1.0
pow: 1024.0
sqrt: 5.0
```

Lab12

March 17, 2025

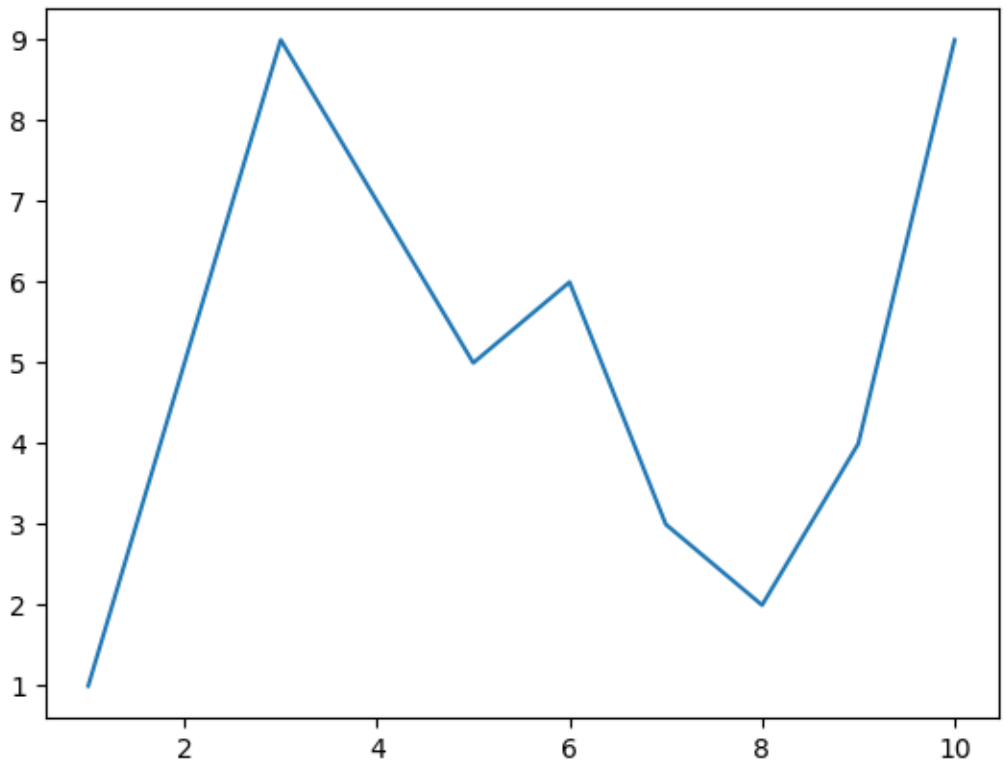
Python Programming - 2301CS404

Lab - 12

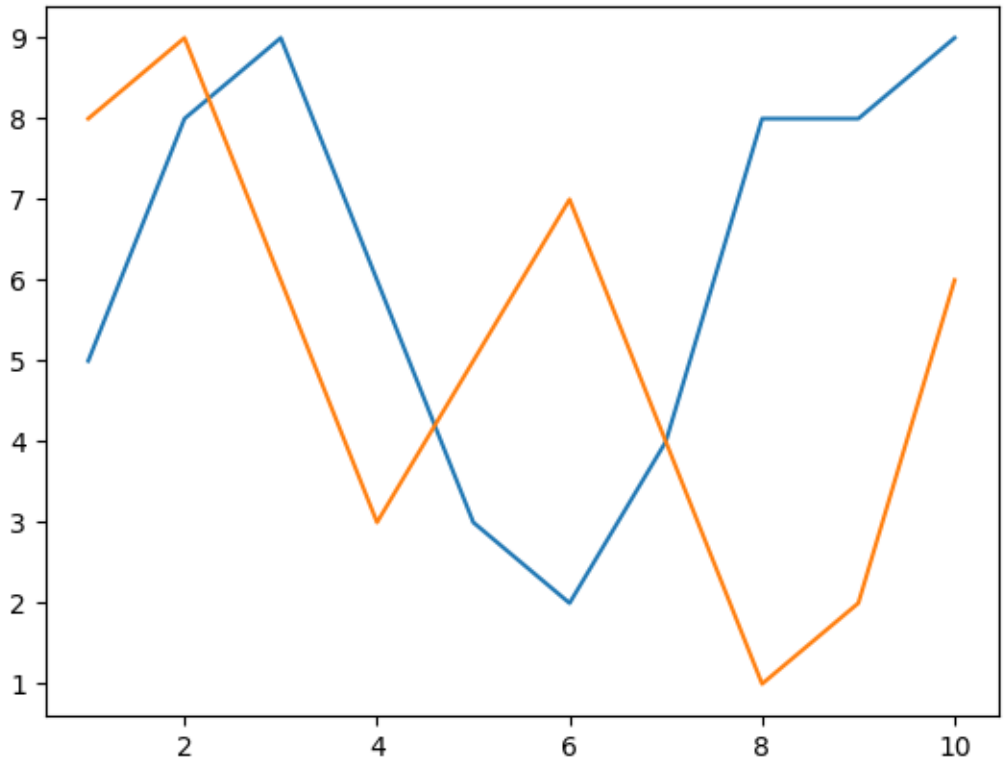
24010101675 Dhruv R. Pithwa

[1]:
```
#import matplotlib below
import matplotlib.pyplot as plt
```

[2]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

# write a code to display the line chart of above x & y
plt.plot(x,y)
plt.show()
```



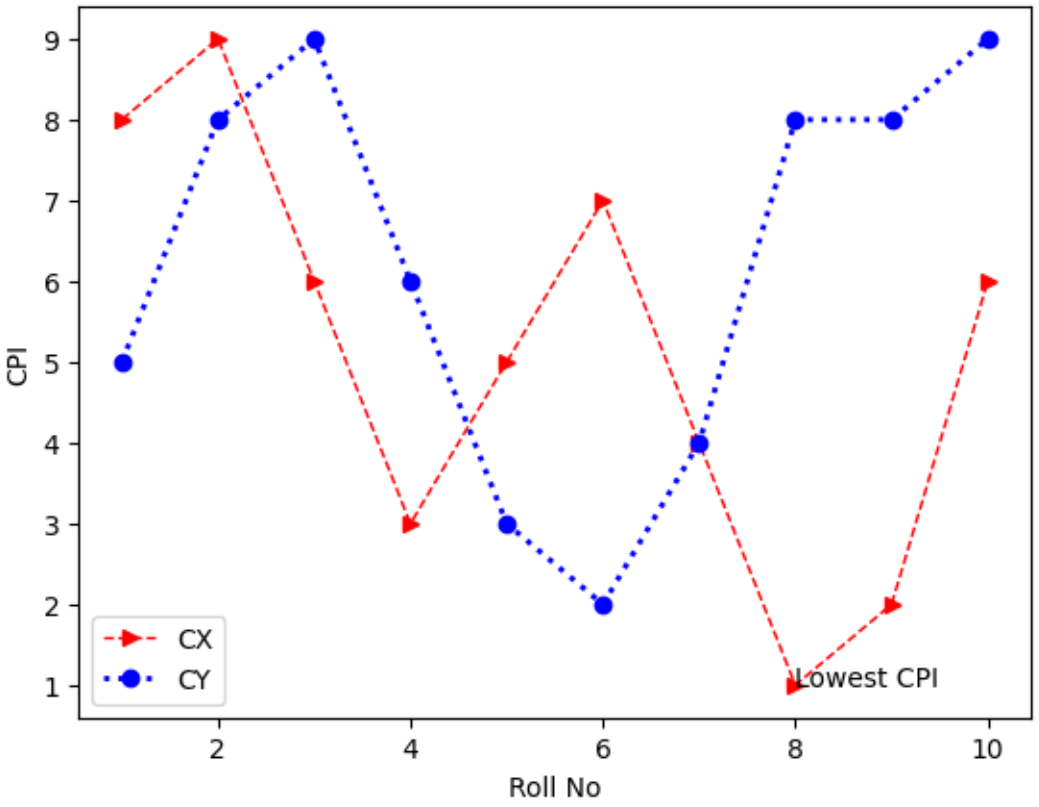
[3]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]
# write a code to display two lines in a line chart (data given above)
plt.plot(x,cxMarks)
plt.plot(x,cyMarks)
plt.show()
```



[4]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

# write a code to generate below graph
```

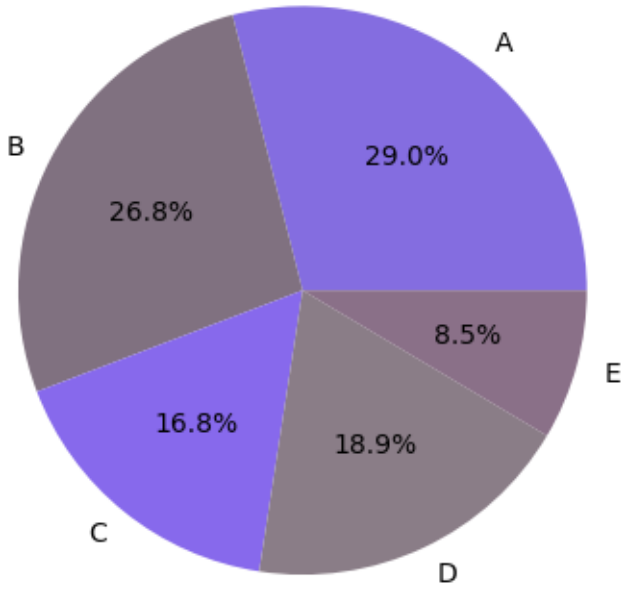
[5]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks, label='CX' , ls='--', color='r',linewidth=1, marker='>')
plt.plot(x,cyMarks, label='CY' ,ls=':',color='b',linewidth=2, marker='o')
```

```
plt.xlabel('Roll No')
plt.ylabel('CPI')
plt.text(8,1,'Lowest CPI')
plt.legend()
plt.show()
```



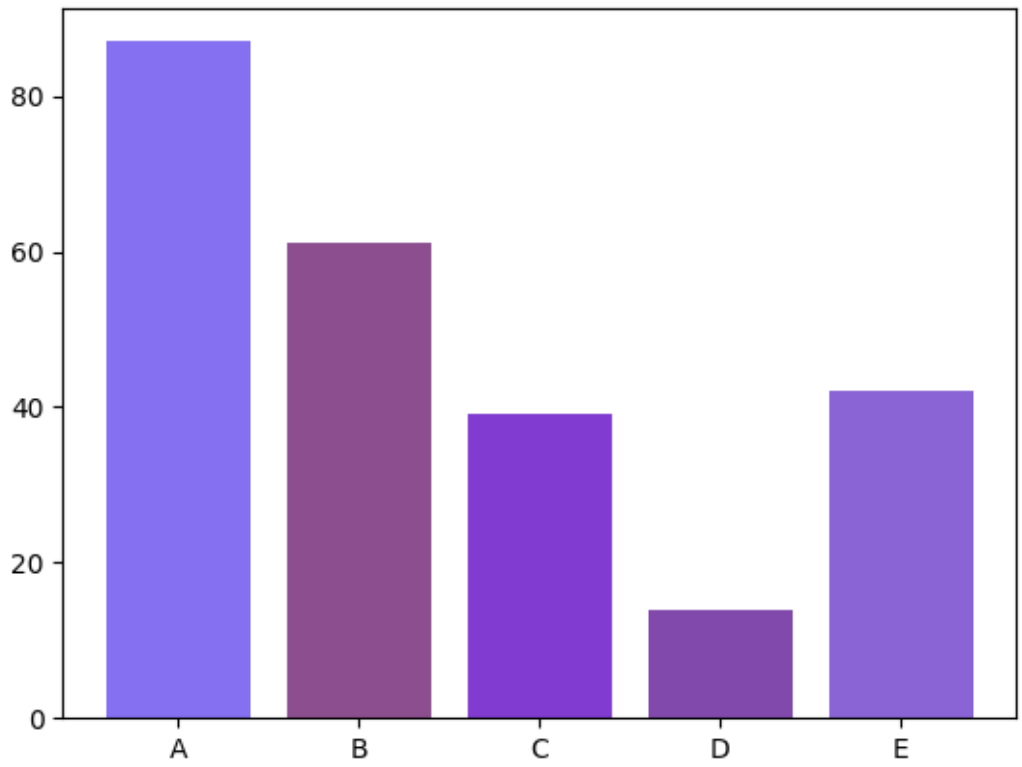
0.0.1 04) WAP to demonstrate the use of Pie chart.

[6]:
```
import random as r
colors = [f'#{r.randint(128, 140):02x}{r.randint(50, 128):02x}{r.randint(128,␣
 ↪255):02x}' for _ in range(5)]
plt.pie([r.randint(1, 100) for i in range(5)], labels=['A', 'B', 'C', 'D',␣
 ↪'E'], colors=colors, autopct='%1.1f%%')
plt.show()
```



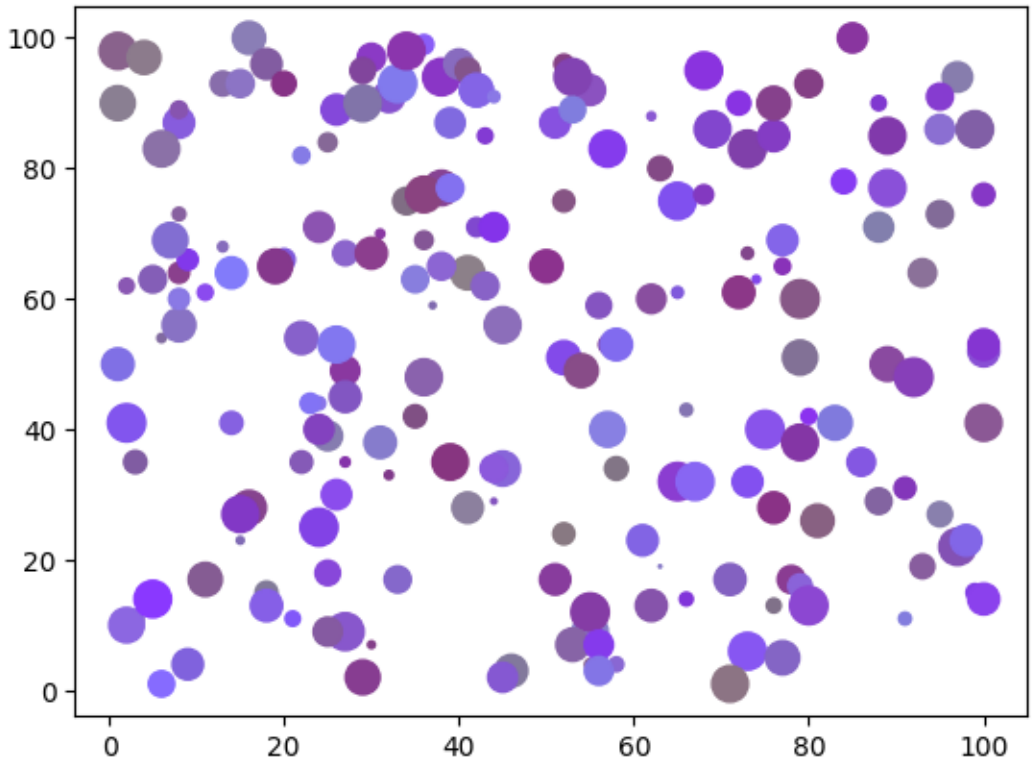
0.0.2 05) WAP to demonstrate the use of Bar chart.

[7]:
```
# colors = [f'#{r.randint(0, 0xFFFFFF):06x}' for _ in range(5)]
colors = [f'#{r.randint(128, 140):02x}{r.randint(50, 128):02x}{r.randint(128,␣
 ↪255):02x}' for _ in range(5)]

plt.bar(['A', 'B', 'C', 'D', 'E'], [r.randint(1, 100) for i in range(5)],␣
 ↪color=colors)
plt.show()
```



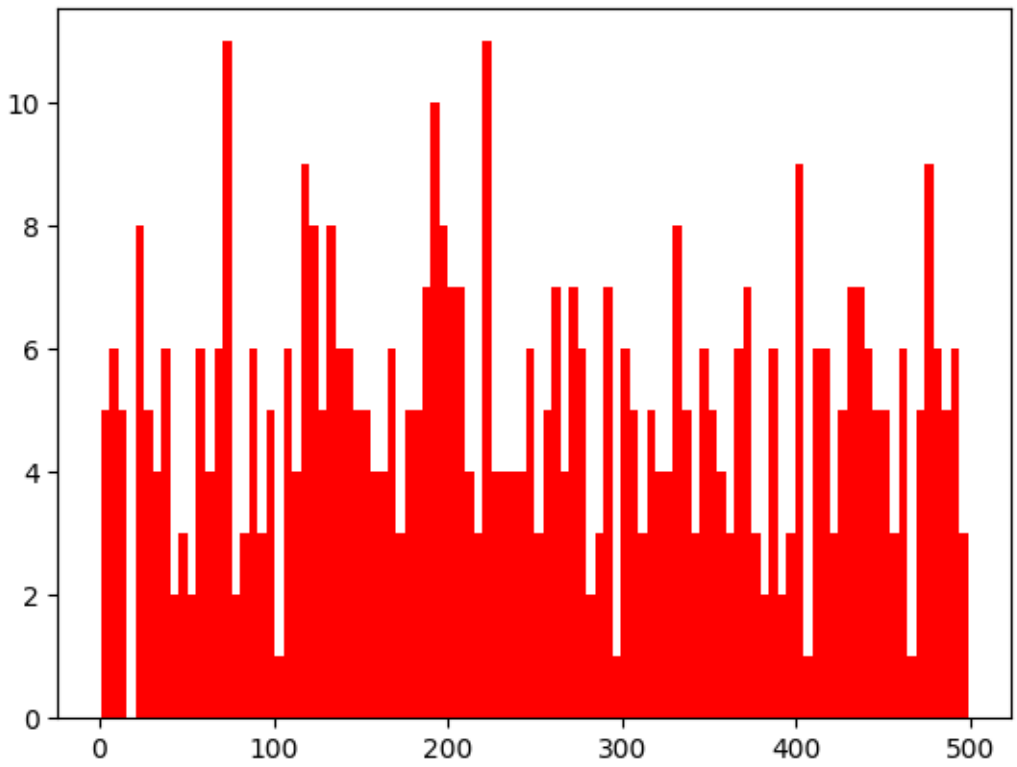
0.0.3 06) WAP to demonstrate the use of Scatter Plot.

[8]:
```
n = 200
colors = [f'#{r.randint(128, 140):02x}{r.randint(50, 128):02x}{r.randint(128,␣
 ↪255):02x}' for _ in range(n)]
plt.scatter([r.randint(1, 100) for i in range(n)], [r.randint(1, 100) for i in␣
 ↪range(n)], s=[r.randint(1, n) for i in range(n)], c=colors)
plt.show()
```



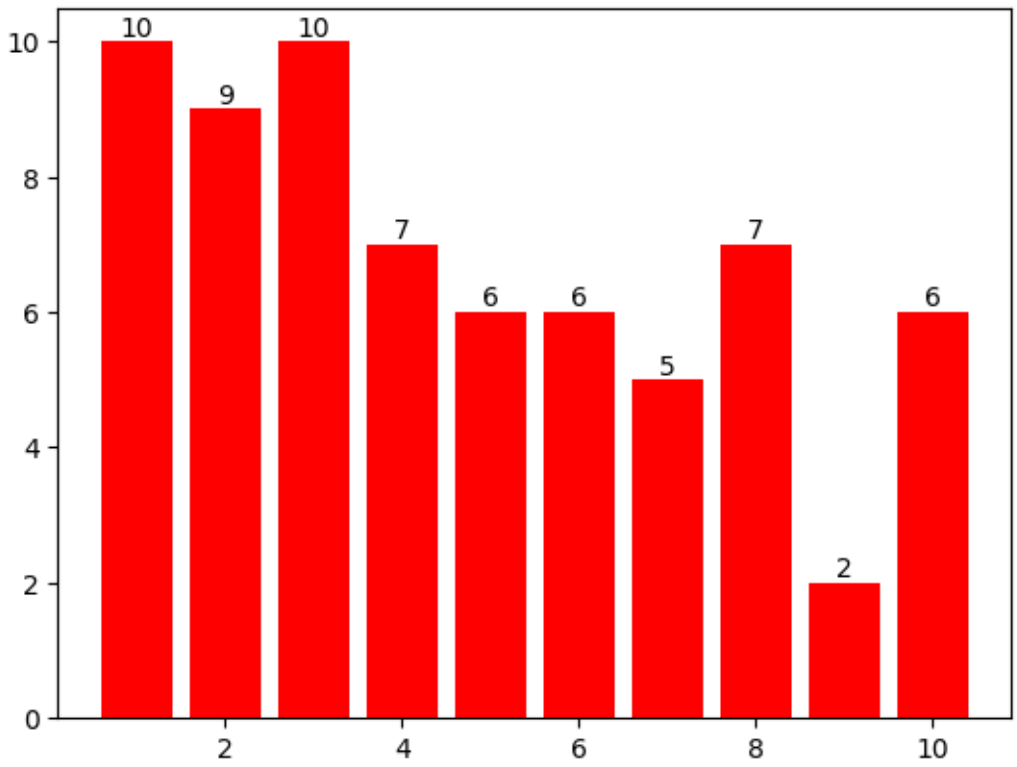
0.0.4 07) WAP to demonstrate the use of Histogram.

[9]:
```
plt.hist([r.randint(1, 500) for i in range(500)], bins=100, color='r')
plt.show()
```



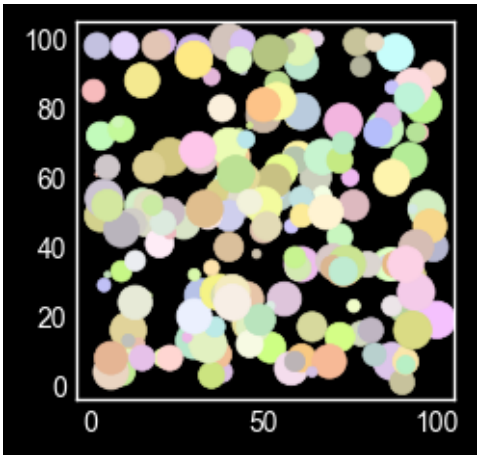
0.0.5 08) WAP to display the value of each bar in a bar chart using Matplotlib.

[10]:
```
x = range(1,11,1)
cxMarks= [r.randint(1,10) for i in range(10)]
bars = plt.bar(x,cxMarks , color='r')


for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), ha='center',␣
 ↪va='bottom')

plt.show()
```



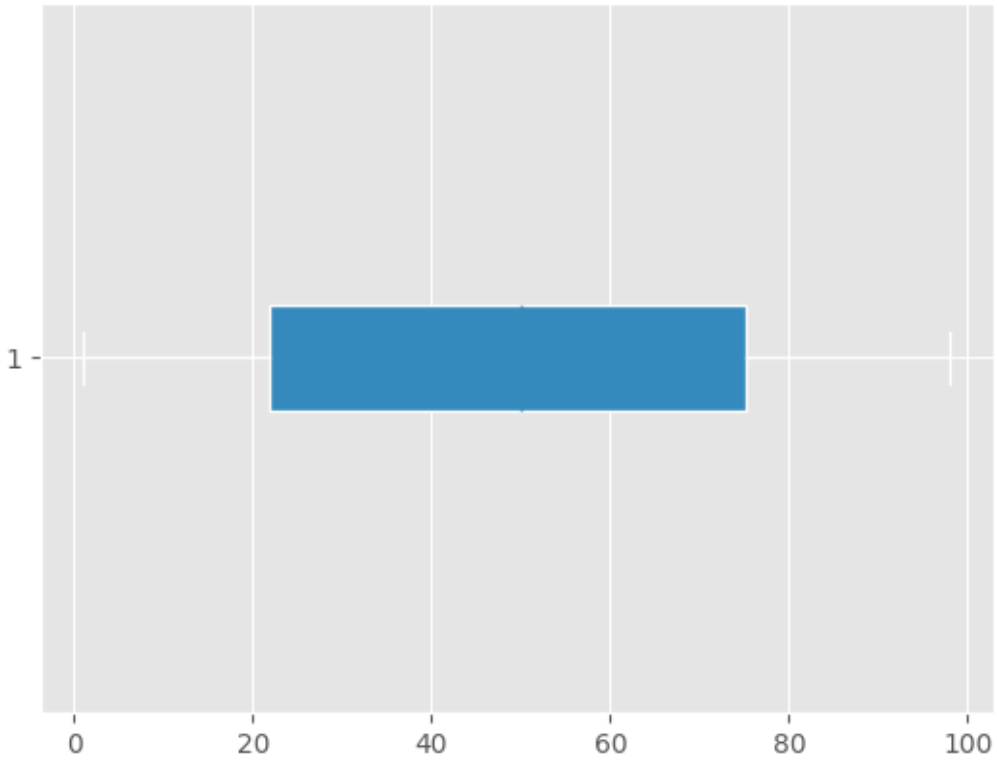
0.0.6 09) WAP create a Scatter Plot with several colors in Matplotlib?

[20]:
```
n = 200
colors = [f'#{r.randint(180, 255):02x}{r.randint(180, 255):02x}{r.randint(128,␣
↪255):02x}' for _ in range(n)]
x = [r.randint(1, 100) for i in range(n)]
y = [r.randint(1, 100) for i in range(n)]
plt.style.use('dark_background')
plt.scatter(x,y, s=[r.randint(1, n) for i in range(n)], c=colors)
plt.show()
```



0.0.7 10) WAP to create a Box Plot.

[12]:
```
plt.style.use('ggplot')
plt.boxplot([r.randint(1, 100) for i in range(100)], vert=False,␣
 ↪patch_artist=True)
plt.show()
```



Lab13-1

March 17, 2025

Python Programming - 2301CS404

Lab - 13

24010101675 Dhruv R. Pithwa

1 OOP

1.0.1 01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

[3]:
```
class Students:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

Student = Students('John', 25, 9.1)
print(Student.name, Student.age, Student.grade)
```

```
John 25 9.1
```

1.0.2 02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

[4]:
```
class Bank_Account:
    Account_No = None
    User_Name = None
    Email = None
    Account_Type = None
    Account_Balance = None

    def GetAccountDetails(self, Account_No, User_Name, Email, Account_Type,␣
 ↪Account_Balance):
        self.Account_No = Account_No
        self.User_Name = User_Name
        self.Email = Email
```

```
        self.Account_Type = Account_Type
        self.Account_Balance = Account_Balance

    def DisplayAccountDetails(self):
        print('Account No:', self.Account_No)
        print('User Name:', self.User_Name)
        print('Email:', self.Email)
        print('Account Type:', self.Account_Type)
        print('Account Balance:', self.Account_Balance)

account = Bank_Account()
account.GetAccountDetails(1001, 'John','john@gamil.com', 'Savings', 10000)
account.DisplayAccountDetails()
```

```
Account No: 1001
User Name: John
Email: john@gamil.com
Account Type: Savings
Account Balance: 10000
```

1.0.3 03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

[5]:
```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def Area(self):
        return 3.14 * self.radius * self.radius

    def perimeter(self):
        return 2 * 3.14 * self.radius

circle = Circle(5)
print('Area:', circle.Area())
print('Perimeter:', circle.perimeter())
```

```
Area: 78.5
Perimeter: 31.400000000000002
```

1.0.4 04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

[6]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary
```

```
    def updateEmployee(self, name = None, age = None, salary = None):
        self.name =  self.name if name is None else name
        self.age = self.age if age is None else age
        self.salary = self.salary if salary is None else salary

    def displayEmployee(self):
        print('Name:', self.name)
        print('Age:', self.age)
        print('Salary:', self.salary)

employee = Employee('John', 25, 10000)
employee.displayEmployee()
employee.updateEmployee('Smith')
employee.displayEmployee()
```

```
Name: John
Age: 25
Salary: 10000
Name: Smith
Age: 25
Salary: 10000
```

1.0.5 05) Create a bank account class with methods to deposit, withdraw, and check balance.

[7]:
```
class Bank_Account:
    Ammount = 0
    def __init__(self, Ammount):
        self.Ammount = Ammount

    def Deposit(self, Ammount):
        self.Ammount += Ammount

    def Withdraw(self, Ammount):
        if self.Ammount < Ammount:
            print('Insufficient Balance')
        else:
            self.Ammount -= Ammount

    def DisplayBalance(self):
        print('Account Balance:', self.Ammount)

account = Bank_Account(10000)
print('Initial Balance:')
account.DisplayBalance()
print('Deposit 5000:')
account.Deposit(5000)
```

```
account.DisplayBalance()
print('Withdraw 2000:')
account.Withdraw(2000)
account.DisplayBalance()
print('Withdraw 20000:')
account.Withdraw(20000)
```

```
Initial Balance:
Account Balance: 10000
Deposit 5000:
Account Balance: 15000
Withdraw 2000:
Account Balance: 13000
Withdraw 20000:
Insufficient Balance
```

1.0.6 06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

[8]:
```
class Inventory:
    Products = []

    def AddProduct(self, name, price, quantity):
        self.Products.append({'name': name, 'price': price, 'quantity':␣
 ↪quantity})

    def DisplayProducts(self):
        for product in self.Products:
            print('ID', self.Products.index(product))
            print('Name:', product['name'])
            print('Price:', product['price'])
            print('Quantity:', product['quantity'])

    def DisplayProduct(self, id):
        product = self.Products[id]
        print('Name:', product['name'])
        print('Price:', product['price'])
        print('Quantity:', product['quantity'])

    def UpdateProduct(self,id, name = None, price = None, quantity = None):
        product = self.Products[id]
        product['name'] = product['name'] if name is None else name
        product['price'] = product['price'] if price is None else price
        product['quantity'] = product['quantity'] if quantity is None else␣
 ↪quantity

    def DeleteProduct(self, id):
```

```
        self.Products.pop(id)

inventory = Inventory()
inventory.AddProduct('Laptop', 50000, 10)
inventory.AddProduct('Mobile', 10000, 20)
inventory.AddProduct('Tablet', 20000, 5)
inventory.DisplayProducts()
inventory.DisplayProduct(1)
inventory.UpdateProduct(1, 'Smart Phone', 15000)
inventory.DisplayProduct(1)
inventory.DeleteProduct(2)
inventory.DisplayProducts()
```

```
ID 0
Name: Laptop
Price: 50000
Quantity: 10
ID 1
Name: Mobile
Price: 10000
Quantity: 20
ID 2
Name: Tablet
Price: 20000
Quantity: 5
Name: Mobile
Price: 10000
Quantity: 20
Name: Smart Phone
Price: 15000
Quantity: 20
ID 0
Name: Laptop
Price: 50000
Quantity: 10
ID 1
Name: Smart Phone
Price: 15000
Quantity: 20
```

1.0.7 07) Create a Class with instance attributes of your choice.

[9]:
```
class A:
    class_variable = 'Class Variable'
    def __init__(self):
        self.instance_variable = 'Instance Variable'
```

```
print(A.class_variable)
a = A()
print(a.instance_variable)
```

```
Class Variable
Instance Variable
```

1.0.8 08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

[10]:
```
class Student_kit:
    principal_name = 'Mr ABC'

    def __init__(self, name):
        self.name = name
        self.attendance = 0

    def update_attendance(self, numOfDays):
        self.attendance += numOfDays

    def certificate(self):
        print('This is to certify that', self.name, 'has attended', self.
 ↪attendance, 'days of the class.', 'Principal:', self.principal_name)

student1 = Student_kit('John')
student1.update_attendance(10)
student1.certificate()
```

```
This is to certify that John has attended 10 days of the class. Principal: Mr
ABC
```

1.0.9 09) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

[11]:
```
class Time:
    def __init__(self, hours, minutes):
        self.hours = hours
        self.minutes = minutes

    def addTimes(self, time1, time2):
        self.hours = time1.hours + time2.hours
```

```
        self.minutes = time1.minutes + time2.minutes
        if self.minutes >= 60:
            self.hours += 1
            self.minutes -= 60

    def displayTime(self):
        print('Time:', self.hours, 'hours', self.minutes, 'minutes')

time1 = Time(2, 30)
time2 = Time(3, 45)
time3 = Time(0, 0)
time3.addTimes(time1, time2)
time3.displayTime()
```

```
Time: 6 hours 15 minutes
```

Lab13-2

March 17, 2025

Python Programming - 2301CS404

Lab - 13

24010101675 Dhruv R. Pithwa

0.1 Continued..

0.1.1 10) Calculate area of a ractangle using object as an argument to a method.

[ ]:
```
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def calculate_area(self, rect_obj):
        return rect_obj.length * rect_obj.breadth


rect1 = Rectangle(10, 5)
area = rect1.calculate_area(rect1)

print(f"The area of the rectangle is: {area}")
```

```
The area of the rectangle is: 50
```

0.1.2 11) Calculate the area of a square.

0.1.3 Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

[6]:
```
class Square:
    def __init__(self, side):
        self.side = side

    def area(self):
        self.output(self.side ** 2)

    def output(self,ptr):
        print('Area :',ptr)
```

```
square = Square(5)
square.area()
```

```
Area : 25
```

0.1.4 12) Calculate the area of a rectangle.

0.1.5 Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

0.1.6 Also define a class method that compares the two sides of reactangle. An object is instantiated only if the two sides are different; otherwise a message should be displayed : THIS IS SQUARE.

[17]:
```
# Improved Rectangle with Factory Method
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def area(self):
        print('Area:', self.length * self.breadth)

    @classmethod
    def create_rectangle(cls, length, breadth):
        if length == breadth:
            print("THIS IS A SQUARE")
            return None
        return cls(length, breadth)

rec1 = Rectangle.create_rectangle(5, 5)  # This will print "THIS IS A SQUARE"
rec2 = Rectangle.create_rectangle(5, 10)

if rec2:
    rec2.area()
```

```
THIS IS A SQUARE
Area: 50
```

0.1.7 13) Define a class Square having a private attribute “side”.

0.1.8 Implement get_side and set_side methods to accees the private attribute from outside of the class.

[ ]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
```

```
        return self.__side

    def set_side(self, new_side):
        self.__side = new_side

sqr = Square(5)
print(sqr.get_side())
sqr.set_side(10)
print(sqr.get_side())
sqr.__side = 20
print(sqr.get_side())
```

```
5
10
10
```

0.1.9 14) Create a class Profit that has a method named getProfit that accepts profit from the user.

0.1.10 Create a class Loss that has a method named getLoss that accepts loss from the user.

0.1.11 Create a class BalanceSheet that inherits from both classes Profit and Loss and calculates the balanace. It has two methods getBalance() and printBalance().

[25]:
```
class Profit:
    def getProfit(self, profit):
        self.profit = profit

class Loss:
    def getLoss(self, loss):
        self.loss = loss

class BalanceSheet(Profit, Loss):
    def getBalance(self):
        self.balance = self.profit - self.loss

    def printBalance(self):
        print("Net Balance:", self.balance)

bs = BalanceSheet()
bs.getProfit(5000)
bs.getLoss(2000)
bs.getBalance()
bs.printBalance()
```

```
Net Balance: 3000
```

0.1.12 15) WAP to demonstrate all types of inheritance.

[28]:
```
# Single Inheritance
class Parent:
    def show(self):
        print("This is a Parent class.")

class Child(Parent):
    def display(self):
        print("This is a Child class.")

obj1 = Child()
obj1.show()
obj1.display()
```

```
This is a Parent class.
This is a Child class.
```

[32]:
```
# Multiple Inheritance
class A:
    def methodA(self):
        print("Class A method.")

class B:
    def methodB(self):
        print("Class B method.")

class C(A, B):
    def methodC(self):
        print("Class C method.")

obj2 = C()
obj2.methodA()
obj2.methodB()
obj2.methodC()
```

```
Class A method.
Class B method.
Class C method.
```

[35]:
```
# Multilevel Inheritance
class Grandparent:
    def greet(self):
        print("Hello from Grandparent.")

class Parent(Grandparent):
    def greet_parent(self):
        print("Hello from Parent.")
```

```
class Child(Parent):
    def greet_child(self):
        print("Hello from Child.")

obj3 = Child()
obj3.greet()
obj3.greet_parent()
obj3.greet_child()
```

```
Hello from Grandparent.
Hello from Parent.
Hello from Child.
```

[37]:
```
# Hierarchical Inheritance
class Person:
    def introduce(self):
        print("I am a human.")

class Student(Person):
    def study(self):
        print("I am studying.")

class Teacher(Person):
    def teach(self):
        print("I am teaching.")

student = Student()
teacher = Teacher()
student.introduce()
student.study()
teacher.introduce()
teacher.teach()
```

```
I am a human.
I am studying.
I am a human.
I am teaching.
```

[39]:
```
# Hybrid Inheritance
class X:
    def methodX(self):
        print("Class X method.")

class Y(X):
    def methodY(self):
        print("Class Y method.")
```

```
class Z:
    def methodZ(self):
        print("Class Z method.")

class Hybrid(Y, Z):
    def methodHybrid(self):
        print("Hybrid class method.")

obj4 = Hybrid()
obj4.methodX()
obj4.methodY()
obj4.methodZ()
obj4.methodHybrid()
```

```
Class X method.
Class Y method.
Class Z method.
Hybrid class method.
```

0.1.13 16) Create a Person class with a constructor that takes two arguments name and age.

0.1.14 Create a child class Employee that inherits from Person and adds a new attribute salary.

0.1.15 Override the init method in Employee to call the parent class's init method using the super() and then initialize the salary attribute.

[40]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Employee(Person):
    def __init__(self, name, age, salary):
        super().__init__(name, age)
        self.salary = salary

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

emp = Employee("Alice", 30, 60000)
emp.display()
```

```
Name: Alice, Age: 30, Salary: 60000
```

0.1.16 17) Create a Shape class with a draw method that is not implemented.

0.1.17 Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors.

0.1.18 Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

[ ]:
```python
class Shape:
    def draw(self):
        pass

class Rectangle(Shape):
    def draw(self):
        print("Drawing a Rectangle.")

class Circle(Shape):
    def draw(self):
        print("Drawing a Circle.")

class Triangle(Shape):
    def draw(self):
        print("Drawing a Triangle.")

shapes = [Rectangle(), Circle(), Triangle()]
for shape in shapes:
    shape.draw()
```

```
Drawing a Rectangle.
Drawing a Circle.
Drawing a Triangle.
```