Practical 1

Aim:	You are given a string.Your task is to count the frequency of letters of the string and print the letters in descending order of frequency. If the following string is given as input to the program: aabbbccde Then, the output of the program should be: b 3 a 2 c 2 d 1 e 1

Input:	`def char_frequency(s1):`
	`  dict = {}`
	`  for n in s1:`
	`  keys = dict.keys()`
	`  if n in keys:`
	`  dict[n] += 1`
	`  else:`
	`  dict[n] = 1`
	`  return dict`
	`str = input("Enter a string: ")`
	`print(char_frequency(str))`

Output:	`E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python -u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\prac1.py"`
	`Enter a string: Hello World`
	`{'H': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'W': 1, 'r': 1, 'd': 1}`

Practical 2

Aim:	Write a procedure to find min, max, mean, standard deviation, variance of number list. Example Input: 10 50 80 70 49 23 11 4 output: 4 80 37.13 27.25 848.70
Input:	

```
import statistics
import pandas as pd

sr = pd.Series([10, 25, 3, 25, 24, 6])

mean = sr.mean()
median = sr.median()
mode = sr.mode()
range1 = sr.max() - sr.min();
stdeviation = sr.std(axis=0, skipna=True)

print(mean)
print(median)
print(mode)
print(range1)
print(stdeviation)
print("Variance of sample set is % s"
      % (statistics.variance(sr)))
```

Output:

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>pyt
hon -u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assign
ment\prac2.py"
15.5
17.0
0    25
dtype: int64
22
10.290772565750348
Variance of sample set is 105.9
```

Practical 3

Aim:	**You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container.** **Input: height = [1,8,6,2,5,4,8,3,7]** **Output: 49** **Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49. Example 2:** **Input: height = [1,1]** **Output: 1**

Input:	`def maxArea(A, Len):`
	`  area = 0`
	`  for i in range(Len):`
	`  for j in range(i + 1, Len):`
	`  area = max(area, min(A[j], A[i]) * (j - i))`
	`  return area`
	
	`a = [int(n) for n in input("Enter an array: ").split()]`
	
	`len1 = len(a)`
	`print(maxArea(a, len1))`

Output:	`E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python -u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\prac3.py"`
	`Enter an array: 1 8 6 2 5 4 8 3 7`
	`49`

Practical 4

Aim:	**You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container.** **Input: height = [1,8,6,2,5,4,8,3,7]** **Output: 49** **Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49. Example 2:** **Input: height = [1,1]** **Output: 1**

Input:	`from itertools import combinations`
	
	`num = [int(n) for n in input("Enter an array: ").split()]`
	`k = int(input("Enter the sumation you want to check combination about: "))`
	`cnt = 0`
	`for i in range(1, len(num)+1):`
	`  for c in combinations(num, i):`
	`  if sum(c) == k:`
	`  cnt += 1`
	`print(cnt)`

Output:	`E:\College\Sem 4\CE259_Pytho`
	`hon -u "e:\College\Sem 4\CE2`
	`ment\prac4.py"`
	`Enter an array: 1 2 3 4 5 7 5`
	`Enter the sumation you want to check combination about: 10`
	`8`

Practical 5

Question:

Explain about the different types of Exceptions in Python with suitable example.

Answer:

Some of the basic inbuilt exceptions are:

1. exception ArithmeticError
   This class is the base class for those built-in exceptions that are raised for various arithmetic errors such as:
   - OverflowError
   - ZeroDivisionError
   - FloatingPointError

```
try:
    a = 10/0
    print (a)
except ArithmeticError:
        print ("This statement is raising an arithmetic exception.")
else:
    print ("Success.")
```

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c5_1.py"
This statement is raising an arithmetic exception.
```

2. exception LookupError
   This is the base class for those exceptions that are raised when a key or index used on mapping or sequence is invalid or not found. The exceptions raised are :
   - KeyError
   - IndexError

```
try:
    a = [1, 2, 3]
    print (a[3])
except LookupError:
    print ("Index out of bound error.")
```

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\tem
pCodeRunnerFile.py"
Index out of bound error.
```

3. exception AttributeError
   An AttributeError is raised when an attribute reference or assignment fails such as when a non-existent attribute is referenced

```
class Attributes(object):
    pass


object = Attributes()
print(object.attribute)
```

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c5_3.py"
Traceback (most recent call last):
  File "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment
\prac5_3.py", line 6, in <module>
    print(object.attribute)
AttributeError: 'Attributes' object has no attribute 'attribute'
```

4. exception FloatingPointError
   A FloatingPointError is raised when a floating point operation fails. This exception is always defined, but can only be raised when Python is configured with the–with-fpectl option, or the WANT_SIGFPE_HANDLER symbol is defined in the pyconfig.h file.

```
import math

print(math.exp(1000))
```

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c5_4.py"
Traceback (most recent call last):
  File "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment
\prac5_4.py", line 3, in <module>
    print(math.exp(1000))
OverflowError: math range error
```

5. exception IndexError
   An IndexError is raised when a sequence is referenced which is out of range.

```
array = [ 0, 1, 2 ]
print (array[3])
```

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c5_5.py"
Traceback (most recent call last):
  File "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment
\prac5_5.py", line 2, in <module>
    print (array[3])
IndexError: list index out of range
```

Practical 6

6.	**Co mpl ete django t ut orial (part 1 to part 7) from t he official document. htt ps:// docs. djangoproject. co m/ en/ 4. 0/**
Code:	

```python
class Parent():
    def __init__(self):
        self.value = "Inside Parent"
    def show(self):
        print(self.value)
class Child(Parent):
    def __init__(self):
        self.value = "Inside Child"
    def show(self):
        print(self.value)
obj1 = Parent()
obj2 = Child()
obj1.show()
obj2.show()
```

Output:

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c6.py"
Inside Parent
Inside Child
```

Practical 7

AIM:	Write a Jango code to send an email with attachment.
Code:	

```
from django.shortcuts import render
from .forms import ContactForm
from django.core.mail import send_mail

def contactview(request):
    name=''
    email=''
    comment=''

    form= ContactForm(request.POST or None)
    if form.is_valid():
        name= form.cleaned_data.get("name")
        email= form.cleaned_data.get("email")
        comment=form.cleaned_data.get("comment")


        comment= name + " with the email, " + email + ", sent the
following message:\n\n" + comment;
        send_mail('The title of this post', comment,
'admin@gmail.com', ['admin@gmail.com'])


        context= {'form': form}
        return render(request, 'contact/contact.html', context)
    else:
        context= {'form': form}
        return render(request, 'contact/contact.html', context)
```

Practical 8

Aim:	**Program to demonstrate the Overriding of the Base Class method in the Derived Class.**
Input:	

```
class P1_class():

    def show(self):
        print("Inside Parent Class 1")


class P2_class():

    def display(self):
        print("Inside Parent Class 2")


class Child_class(P1_class, P2_class):

    def show(self):
        print("Inside Child Class")


obj = Child_class()

obj.show()
obj.display()
```

Output:

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c8.py"
Inside Child Class
Inside Parent Class 2
```

Practical 9

Aim:	Write Pythonic code to create a function named move_rectangle() that takes an object of Rectangle class and two numbers named dx and dy. It should change the location of the Rectangle by adding dx to the x coordinate of corner and adding dy to the y coordinate of corner.
Input:	

```
class Point(object):
    pass

class Rectangle(object):
    pass

rectangle = Rectangle()

bottom_left = Point()
bottom_left.x = 8.0
bottom_left.y = 3.0

top_right = Point()
top_right.x = 9.0
top_right.y = 6.0

rectangle.corner1 = bottom_left
rectangle.corner2 = top_right

dx = 15.0
dy = 16.0

def move_rectangle(rectangle, dx, dy):
    print(f"The rectangle started with bottom left corner at
({rectangle.corner1.x},{rectangle.corner1.y})"
          f"\nTop right corner at
({rectangle.corner2.x},{rectangle.corner2.y})"
          f"\ndx is {dx} and dy is {dy}")
    rectangle.corner1.x = rectangle.corner1.x + dx
    rectangle.corner2.x = rectangle.corner2.x + dx
    rectangle.corner1.y = rectangle.corner1.y + dy
    rectangle.corner2.y = rectangle.corner2.y + dy
    print(f"It ended with a bottom left corner at
({rectangle.corner1.x},{rectangle.corner1.y})"
          f"\nTop right corner at
({rectangle.corner2.x},{rectangle.corner2.y})")

move_rectangle(rectangle, dx, dy)
```

Output:

```
E:\College\Sem 4\CE259_Python\Assignments\Programming Assignment>python
-u "e:\College\Sem 4\CE259_Python\Assignments\Programming Assignment\pra
c9.py"
The rectangle started with bottom left corner at (8.0,3.0)
Top right corner at (9.0,6.0)
dx is 15.0 and dy is 16.0
It ended with a bottom left corner at (23.0,19.0)
Top right corner at (24.0,22.0)
```