## 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:

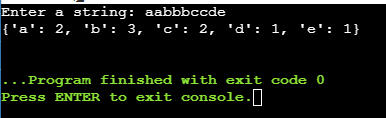
dict[n] = 1

return dict

str = input("Enter a string: ")

print(char\_frequency(str))

**Output:**



## 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:**

from re import A

import statistics

import pandas as pd

sr = pd.Series([10,50, 80, 70, 49, 23,11,4])

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)

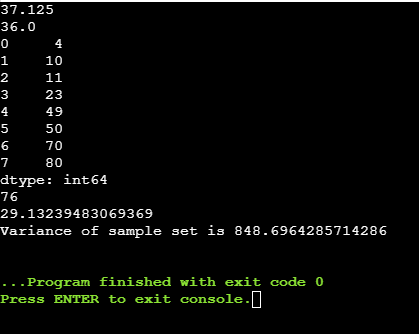
print(range1)

print(stdeviation)

print("Variance of sample set is % s"

%(statistics.variance(sr)))

**Output:**



## 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))

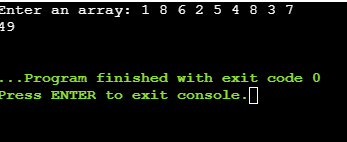
return area

a = [int(n) for n in input("Enter an array: ").split()]

len1 = len(a)

print(maxArea(a, len1))

**Output:**



## Practical 4

**Aim:**

**Given a list of integers, write a program to print the count of all possible unique combinations of numbers whose sum is equal to K. Input The first line of input will contain space-separated integers. The second line of input will contain an integer, denoting K. Output The output should be containing the count of all unique combinations of numbers whose sum is equal to K.**

**Sample Input 1 2 4 6 1 3 6**

**Sample Output 1 3**

**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:**

****

## 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

**INPUT:**

**try:**

**a = 10/0**

**print (a)**

**except ArithmeticError:**

**print ("This statement is raising an arithmetic exception.")**

**else: https://www.onlinegdb.com/online\_python\_compiler#editor\_1**

**print ("Success.")**

**OUTPUT:**

****

1. 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

**INPUT:**

**try:**

**a = [1, 2, 3]**

**print (a[3])**

**except LookupError:**

**print ("Index out of bound error.")**

**OUTPUT:**

****

1. exception AttributeError

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

**INPUT:**

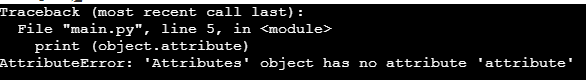
class Attributes(object):

pass

object = Attributes()

print (object.attribute)

**OUTPUT:**

****

1. exception FloatingPointError

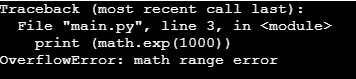
A FloatingPointError is raised when a floating point operation fails. This exception is always defined, but can only be raised when Python is configured with the–with-fpectl option, or the WANT\_SIGFPE\_HANDLER symbol is defined in the pyconfig.h file.

**INPUT:**

**import math**

**print (math.exp(1000))**

**OUTPUT:**

****

1. exception IndexError

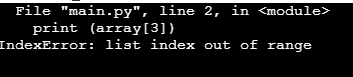
An IndexError is raised when a sequence is referenced which is out of range.

**INPUT:**

**array = [ 0, 1, 2 ]**

**print (array[3])**

**OUTPUT:**

****

## 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):**

**A print("Inside Child Class")**

**obj = Child\_class()**

**obj.show()**

**obj.display()**

**OUTPUT:**

****

## 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**

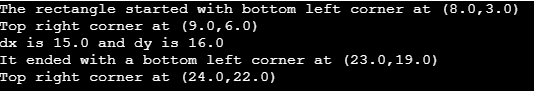
**rectangle.corner2.y = rectangle.corner2.y + dy**

**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:**

****