Practical List

**Q12** File Handling

***Code***

f = open("text.txt", "w")

f.write("Neither apple nor pine are in pineapple. Boxing rings are square.\nWriters write, but fingers don't fing. Overlook and oversee are opposites.\nA house can burn up as it burns down. An alarm goes off by going on.\n")

f.close()

f = open("text.txt", "a+")

f.seek(0)

print(f.read())

f.write("People in Poland are called Poles but people in Holland are not called Holes.\n")

f.seek(0)

lineno = 1

for i in f:

print(lineno,":",i)

lineno += 1

f.seek(0)

print("Last line:",f.readlines()[-1])

f.seek(9)

print(f.readline())

lineno = int(input("Enter line number: "))

f.seek(0)

print(f.readlines()[lineno-1])

d = {}

f.seek(0)

for i in f.readlines():

for j in i.split():

j = j.lower()

if j[0] not in d:

d[j[0]] = 1

else:

d[j[0]] += 1

for i in d.items():

print("Words beginning with",i[0],":",i[1])

***Output***





**Q13** File Handling

***Code***

a = input("Enter filename: ")

def isvowel(f1):

f1 = open(f1)

f2 = open("file2.text","w")

 for i in f1.readlines():

for j in i.split():

if j[0] not in "aeiouAEIOU":

f2.write(j+" ")

f1.close()

f2.close()

isvowel(a)

**Q14** File Handling

***Code Output***

f = open("data.txt", "r")

l = f.readlines()

for i in range(len(l)):

l[i] = tuple(l[i].split())

def sortNew(n):

 return n[2]

l.sort(key = sortNew)

d = {}

print("People with Experience less than 3 years:")

for i in l:

if int(i[3]) < 3:

print(i[0], i[1])

if i[4] in d:

d[i[4]] += 1

else:

d[i[4]] = 1

print()

for i in d.items():

print(i[0],":",i[1])

**Q15** File Handling

***Code Output***

f = open("myfile.txt", "r")

d = {}

words = 0

for i in f.readlines():

for j in i.split():

 if j.lower() in d:

d[j.lower()] += 1

else:

d[j.lower()] = 1

for i in d.values():

words += i

print("Total number of words:", words)

print("Number of different words:", len(d.keys()))

maxword, maxcnt = "", 0

for i in d.items():

if i[1] > maxcnt:

maxcnt = i[1]

maxword = [i[0]]

elif i[1] == maxcnt:

maxword.append(i[0])

print("Most common word(s) is/are:", maxword)

d1 = {}

for i in d:

if len(i) in d1:

d1[len(i)].append(i)

else:

d1[len(i)] = [i]

def find\_longest\_word():

return d1[max(d1)]

print("Longest word(s) is/are:",find\_longest\_word())

l2 = []

def filter\_long\_words(n):

print("Words longer than", n, "are:")

for i in d1:

if i > n:

l2.extend(d1[i])

print(d1[i])

filter\_long\_words(8)

f.seek(0)

l = f.read().lower().split()

f1 = open("newfile.txt","w")

for i in l:

if i not in l2:

f1.write(i)

f1.write(" ")

f1.close()

**Q16** File Handling

***Code Output***

shift = int(input("Enter shift value: "))

key = {}

x = input("Encode(e)/Decode(d)? ").lower()

inp = open("input.txt", "r")

out = open("output.txt", "w")

if x == "e":

for i in "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ":

 a = ord(i) + shift

if a > ord("z"):

a = ord(i) + shift - 26

elif a > ord("Z") and i.isupper():

 a = ord(i) + shift - 26

key[i] = chr(a)\

for i in inp.readlines():

for j in i:

if j.isalpha():

out.write(key[j])

elif j.isspace():

out.write(" ")

else:

pass

out.write("\n")

if x == "d":

for i in "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ":

a = ord(i) - shift

if a > ord("z"):

a = ord(i) - shift + 26

elif a > ord("Z") and i.isupper():

a = ord(i) - shift + 26

key[i] = chr(a)

for i in inp.readlines():

for j in i:

if j.isalpha():

out.write(key[j])

elif j.isspace():

out.write(" ")

else:

pass

out.write("\n")

inp.close()

out.close()

**Q1** Password Check

***Code Output***

password = input("Enter password: ").split(",")

lower, upper, number, special = 0, 0, 0, 0

for j in password:

for i in j:

if i.islower():

lower += 1

elif i.isupper():

upper += 1

elif i.isdigit():

number += 1

elif i in "@#$":

special += 1

if lower > 0 and upper > 0 and number > 0 and special > 0:

print("Valid password is", j)

break

else:

lower, upper, number, special = 0, 0, 0, 0

**Q2** Functions

***Code Output***

n = input("Enter number: ")

def count(n):

return len(n)

def reverse(n):

return n[::-1]

def hasdigit(n):

a = input("Enter digit to check: ")

return a in n

def show(n):

n = int(n)

result = []

divider = 10

while divider < n:

temp = n % divider

if temp != 0:

result.insert(0, str(temp))

n -= temp

divider \*= 10

result.insert(0, str(n))

return '+'.join(result)

print("No. of digits:", count(n))

print("Reverse of number:", reverse(n))

print(hasdigit(n))

print("Expanded form of number:", show(n))

**Q3 Modules**

***Code – perfect.py Output***

def GenerateFactors(n):

factors = []

for i in range(1, (n//2) + 1):

if n % i == 0:

factors.append(i)

 factors.append(n)

return factors

def isPrimeNo(n):

return (len(GenerateFactors(n)) == 2)

def isPerfectNo(n):

return (sum(GenerateFactors(n)) == n \* 2)

***Code – main.py***

import perfect

n = int(input("Enter number: "))

d = {1:perfect.GenerateFactors(n),2:perfect.isPrimeNo(n), 3:perfect.isPerfectNo(n)}

try:

choice = int(input("Choose 1 to see factors of number:\nChoose 2 to check if number is prime:\nChoose 3 to check if number is perfect:\n"))

except:

print("Invalid choice!")

print(d[choice])

**Q6** Population Pie Graph

***Code***

import matplotlib.pyplot as plt

states = ["Uttar Pradesh", "Madhya Pradesh", "Maharashtra", "Rajasthan", "Assam", "Karnataka", "Haryana", "West Bengal", "Tamil Nadu", "Arunachal Pradesh"]

pop = [820, 240, 370, 200, 397, 320, 573, 1029, 550, 17]

plt.pie(pop, labels=states)

plt.show()

***Output***



**Q7** Functions Line Graph

***Code Output***

x = np.arange(0,11,0.1)

y1 = np.exp(-1\*x/10)\*np.sin(math.pi\*x)

y2 = x\*np.exp(-1\*x/3)

plt.plot(x, y1, label="f(x)")

plt.plot(x, y2, label="g(x)")

plt.legend()

plt.xlabel('x')

plt.ylabel('y')

plt.savefig("Graph.jpg")

plt.show()

**Q4** Pascal’s Triangle

***Code Output***

import math

n = int(input("Enter number of rows: "))

def triangle(n):

if n == 0:

return []

elif n == 1:

return [[1]]

else:

new\_row = [1]

result = triangle(n-1)

last\_row = result[-1]

for i in range(len(last\_row)-1):

new\_row.append(last\_row[i] + last\_row[i+1])

new\_row += [1]

result.append(new\_row)

return result

for i in triangle(n):

for j in i:

print(j, end=" ")

print()

**Q5** Number Type Converter

***Code Output***