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()

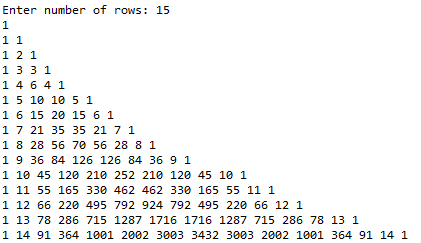
plt.xlabel('x')

plt.ylabel('y')

plt.savefig("Graph.jpg")

plt.show()

**Q4** Pascal’s Triangle

***Code Output***

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***

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

c = input("Choose B for binary, O for octal, H for hexadecimal: ").upper()

def toStr(n, base):

convertString = "0123456789ABCDEF"

if n < base:

return convertString[n]

else:

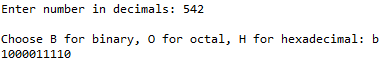
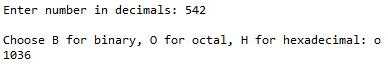
return toStr(n//base, base) + convertString[n%base]

if c == "B": c= 2

if c == "O": c = 8

if c == "H": c = 16

print(toStr(n,c))

***Output***

**Q8** Functions on Lists

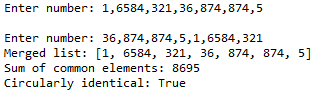
***Code***

import copy

l1 = [int(x) for x in input("Enter number: ").split(",") if x.isdigit()]

l2 = [int(x) for x in input("Enter number: ").split(",") if x.isdigit()]

def merge(l1,l2):

 l3 = copy.deepcopy(l1) ***Output***

for i in l2:

if i not in l1:

l3.append(i)

return l3

def commonSum(l1,l2):

sum = 0

for i in l1:

if i in l2:

sum += i

return sum

def isCircular(l1, l2):

l3, l4 = "", ""

l1.extend(l1)

for i in l1:

l3 += str(i)

for i in l2:

l4 += str(i)

if l4 in l3 or l4[::-1] in l3:

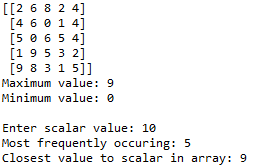
return True

print("Merged list:", merge(l1,l2))

print("Sum of common elements:", commonSum(l1,l2))

print("Circularly identical:", isCircular(l1,l2))

**Q9** NumPy Array

***Code Output***

import numpy as np

a = np.random.randint(10, size=(5,5))

max = 0

print(a)

print("Maximum value:", np.max(a))

print("Minimum value:", np.min(a))

for i in a:

if np.bincount(i).argmax()>max:

max = np.bincount(i).argmax()

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

def closestToScalar(array, value):

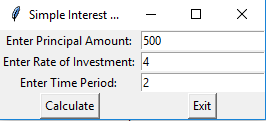
i = np.abs(array - value).argmin()

return a.flat[i]

print("Most frequently occuring:", max)

print("Closest value to scalar in array:", closestToScalar(a, value))

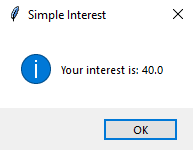
**Q10** Tkinter Simple Interest Calculator

***Code Output***

from tkinter import \*

from tkinter import messagebox

root = Tk()

root.title("Simple Interest Calculator")

P = IntVar()

R = IntVar()

T = IntVar()

Label(root, text="Enter Principal Amount: ").grid(column=0, row=0)

Entry(root, width=20, textvariable=P).grid(column=2, row=0)

Label(root, text="Enter Rate of Investment: ").grid(column=0, row=1)

Entry(root, width=20, textvariable=R).grid(column=2, row=1)

Label(root, text="Enter Time Period: ").grid(column=0, row=2)

Entry(root, width=20, textvariable=T).grid(column=2, row=2)

def clicked():

p = P.get()

r = R.get()

t = T.get()

si = (p\*r\*t)/100

messagebox.showinfo("Simple Interest", "Your interest is: "+str(si))

Button(root, text="Calculate", command=clicked).grid(column=0, row=4)

Button(root, text="Exit", command=root.destroy).grid(column=2, row=4)

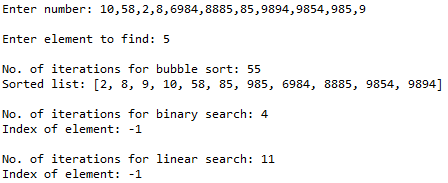
root.mainloop()

**Q11** Sorting/Searching Efficiency

***Code***

l = [int(x) for x in input("Enter number: ").split(",") if x.isdigit()]

a = int(input("Enter element to find: ")) ***Output***

def bubbleSort(l):

iter = 0

for i in range(len(l)):

for j in range(len(l)-i-1):

if l[j]>l[j+1]:

l[j], l[j+1] = l[j+1], l[j]

iter += 1

print("No. of iterations for bubble sort:", iter)

return l

def binarySearch(l, a):

low, high, iter = 0, len(l)-1, 0

while low <= high:

iter += 1

mid = (low + high)//2

if l[mid] == a:

print("No. of iterations for binary search:", iter)

return mid

elif l[mid] < a:

low = mid + 1

else:

high = mid - 1

print("No. of iterations for binary search:", iter)

return -1

def linearSearch(l, a):

iter = 0

for i in range(len(l)):

if l[i] == a:

print("No. of iterations for linear search:", iter)

return i

iter += 1

print("No. of iterations for linear search:", iter)

return -1

print("Sorted list:", bubbleSort(l))

print("Index of element:", binarySearch(l, a))

print("Index of element:", linearSearch(l, a))

**Q17** URLLib

***Code***

import urllib

u = urllib.request.urlopen("https://www.pythonforbeginners.com")

f = open("downloaded.htm","w") ***File***

for i in u.readlines():

print(i)

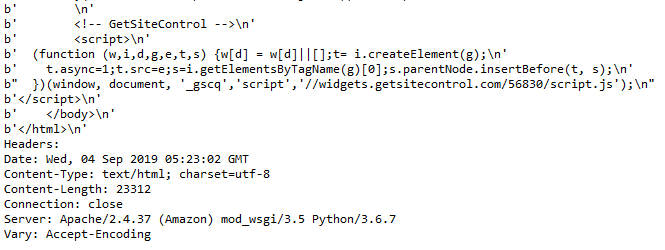
f.write(str(i))

print("Headers:")

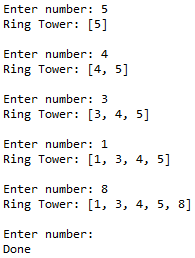
print(u.headers)

u.close()

f.close()

***Output***

**Q18** Ring Tower Stack

***Code Output***

class ringTower:

def \_\_init\_\_(self, ringCondition = False):

self.s = []

self.ring = ringCondition

def isEmpty(self):

if self.s == []:

return True

else:

return False

def push(self, item, tempTower):

if self.ring:

while item < self.peek():

tempTower.s.append(self.pop())

else:

self.s.append(item)

while not tempTower.isEmpty():

self.s.append(tempTower.pop())

else:

self.s.append(item)

def pop(self):

if not self.isEmpty():

return(self.s.pop())

else:

return False

def peek(self):

if not self.isEmpty():

return (self.s[-1])

else:

return 0

rings = ringTower(True)

temp = ringTower()

while True:

try:

rings.push(int(input("Enter number: ")), temp)

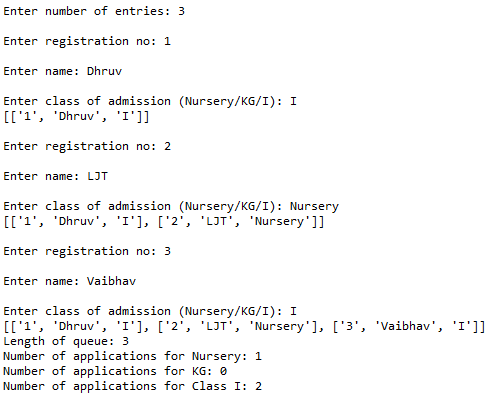
print("Ring Tower:", rings.s)

except:

print("Done")

break

**Q19** Admission Queue

***Code Output***

class queue:

def \_\_init\_\_(self):

self.q = []

def enqueue(self, admissionEntry):

self.q.append(admissionEntry)

def length(self):

return len(self.q)

def applicationByClass(self):

nur, kg, one = 0, 0, 0

for i in self.q:

if i[2] == "Nursery":

nur += 1

elif i[2] == "KG":

kg += 1

else:

one += 1

print("Number of applications for Nursery:", nur)

print("Number of applications for KG:", kg)

print("Number of applications for Class I:", one)

adm = []

q = queue()

i = 1

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

class incorrectDataEntry(Exception):

pass

while i <= n:

try:

l1 = input("Enter registration no: ")

l2 = input("Enter name: ")

l3 = input("Enter class of admission (Nursery/KG/I): ")

if l3 != "Nursery" and l3 != "KG" and l3 != "I":

raise incorrectDataEntry

except:

print("Data entry incorrect, try again!")

continue

q.enqueue([l1, l2, l3])

print(q.q)

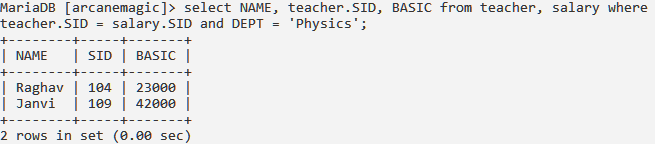
i += 1

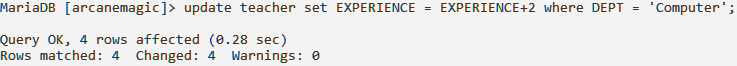
print("Length of queue:", q.length())

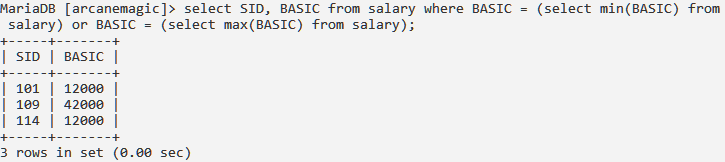
q.applicationByClass()

Database Management Q1

I.

II.

III.

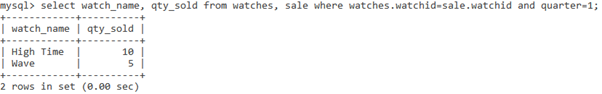


IV.

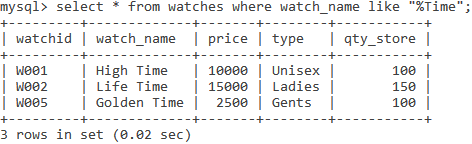


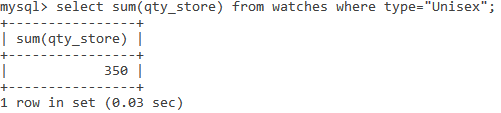
V.

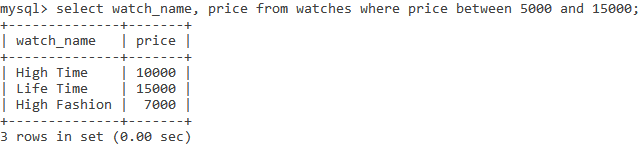
Database Management Q2

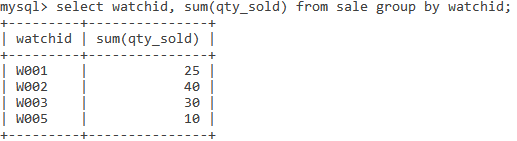


I.

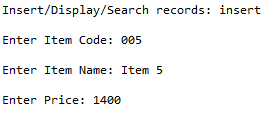
II. 

III. 

IV.

V. 

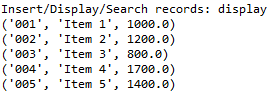
Database Management Q3

***Code Output***

import mysql.connector

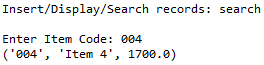
connection = mysql.connector.connect(host='localhost',

database='arcanemagic',

 user='root',

password='amity')

cursor = connection.cursor()

a = input("Insert/Display/Search records: ").lower()

def insertIntoTable():

itemCode = input("Enter Item Code: ")

itemName = input("Enter Item Name: ")

price = float(input("Enter Price: "))

query = "INSERT INTO item values('%s', '%s', '%f');" % (itemCode, itemName, price)

cursor.execute(query)

connection.commit()

def showRecords():

query = "SELECT \* FROM item;"

cursor.execute(query)

for i in cursor.fetchall():

print(i)

def searchRecords():

itemCode = input("Enter Item Code: ")

query = "SELECT \* FROM item WHERE Itemcode = '%s';" % (itemCode)

cursor.execute(query)

for i in cursor.fetchall():

print(i)

if a == "insert":

insertIntoTable()

elif a == "display":

showRecords()

elif a == "search":

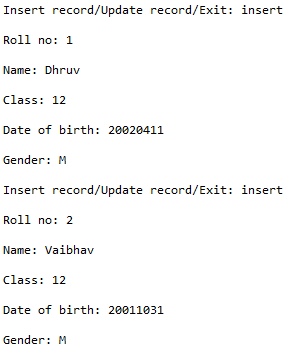
searchRecords()

else:

print("Unknown operation")

Database Management Q4

***Code Output***

import mysql.connector

connection = mysql.connector.connect(host='localhost',

database='arcanemagic',

user='root',

password='amity')

cursor = connection.cursor()

a = None

def insertIntoTable():

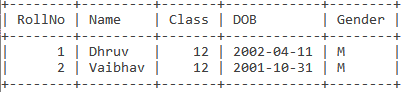
roll = int(input("Roll no: "))

name = input("Name: ")

grade = int(input("Class: "))

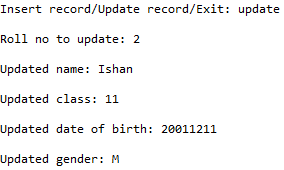
dob = int(input("Date of birth: "))

gender = input("Gender: ")

 query = "INSERT INTO student values('%d', '%s', '%d', '%d', '%s');" % (roll, name, grade, dob, gender)

cursor.execute(query)

connection.commit()

def updateRecords():

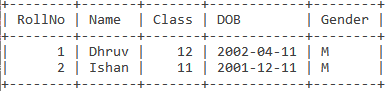
roll = int(input("Roll no to update: "))

name = input("Updated name: ")

grade = int(input("Updated class: "))

dob = int(input("Updated date of birth: "))

gender = input("Updated gender: ")

 query = "UPDATE student set Name='%s',Class='%d',DOB='%d',Gender='%s' where RollNo='%d';" % (name, grade, dob, gender, roll)

cursor.execute(query)

connection.commit()

while a != "exit":

a = input("Insert record/Update record/Exit: ").lower()

if a == "insert":

insertIntoTable()

elif a == "update":

updateRecords()