XII-CS-Practicals

**PYTHON**  
Sol 1: -

# PRACTICAL 1

"""

Writing a function to create a text file containing following data:

Neither apple nor pine are in pineapple. Boxing rings are square.

Writers write, but fingers don’t fing. Overlook and oversee are opposites.

A house can burn up as it burns down. An alarm goes off by going on.

AND

a)Reading back the entire file content using read() or readlines() and displaying.

b)Appending more text of your choice in the file and display the content of file with line numbers prefixed to line.

c)Displaying last line of file.

d)Displaying first line from 10th character onwards.

e)Reading and displaying a line from the file. Ask user to provide the line number to be read.

f)Finding the frequency of words beginning with every letter.

! The 'fing' typo was in the question, not done by me xD

“""

**def fl\_create\_ql(env\_path):**

**with open(env\_path, 'w+') as fl:**

**fl.writelines(['Neither apple nor pine are in pineapple. Boxing rings are square.\n’, 'Writers write, but fingers don\'t fing. Overlook and oversee are opposites.\n’, ‘A house can burn up as it burns down. An alarm goes off by going on.'])**

**fl.seek(0)**

**print(f'a)\n{fl.read().strip()}') # --> a)**

**with open(env\_path, 'a+') as fl:**

**fl.write(input('b)\nEnter text to be appended to the file:\n'))**

**fl.seek(0)**

**p = fl.read().split('\n')**

**print(\*[str(i + 1) + ' : ' + p[i] for i in range(len(p))], sep='\n')# --> b)**

**print(f'c)\nLast line of file is:\n{p[-1]}')# --> c)**

**print(f'd)\nFirst line from 10th character onwards:\n{p[0][10:]}’) # --> d)**

**with open(env\_path, 'r') as fl:**

**p = fl.readlines()**

**inp = int(input('e)\nEnter line no. to be read:\n'))**

**if inp > len(p) or inp < -len(p) or inp == 0:**

**print('The line no. you entered does not exist.')**

**elif inp > 0:**

**print(p[inp - 1].strip()) # --> e)**

**else:**

**print(p[inp].strip())**

**fl.seek(0)**

**p = (' '.join(fl.read().split('\n'))).split(' ')**

**freq = {}**

**for i in p:**

**if i[0].lower() in freq:**

**freq[i[0].lower()] += 1**

**else:**

**freq[i[0].lower()] = 1**

**print('f)\n')**

**print(\*['Words beginning with ' + \_ + ' : ' + str(freq[\_]) for \_ in freq], sep=‘\n') # --> f)**

**fl\_create\_ql(‘./file1.txt’)**

**# if we need to sort alphabetically, create var='abcd...xyz' and iterate**

**# over that printing <iter> : freq[<iter>]**

Sol 2: -

# PRACTICAL 2

"""

Taking file1.txt containing some text and writing a function named isvowel( ) that reads the file file2-1.txt and creates a new file named file2-2.txt, which shall contain only those words from the file file2-1.txt which don’t start with a vowel

"""

**with open(‘file2-1.txt’) as fl:**

**lines = fl.readlines()**

**print('Text read from file1:\n', \*lines)**

**for line in lines:**

**text = line.strip().split(' ')**

**nl = []**

**for char in text:**

**if char[0].lower() not in 'aeiou':**

**nl.append(char)**

**else:**

**continue**

**nl = ' '.join(nl) + '\n'**

**with open('file2-2.txt', 'w') as fl2:**

**fl2.write(nl)**

**with open('file2-2.txt') as fl2:**

**print('Text added to file2:\n', fl2.read())**

Sol 3: -

# PRACTICAL 3

"""

Each line in a csv file contains a first name, a second name, a registration number, no of years and a department separated by tabs.

a) Writing a program that copies the contents of the file into a list of tuples.

b) Displaying:   
  
- Full details of the student sorted by registration number. - The names of all students with no of year less than 3. - The number of people in each department.

"""

**with open(‘./studentdata.csv’, mode='r') as f:**

**data = csv.reader(f)**

**ls = []**

**for i in data:**

**for j in i:**

**entry = tuple(j.split(' '))**

**ls.append(entry)**

**print('a) Details copied to a list of tuples:\n')**

**print(ls) # --> a)**

**m = {}**

**dic = {}**

**for i in ls:**

**m[i[2]] = ls.index(i)**

**if i[-1] not in dic:**

**dic[i[-1]] = 1**

**else:**

**dic[i[-1]] += 1**

**k = sorted(m.keys())**

**for j in k:**

**k[k.index(j)] = ls[m[j]]**

**print('b) The details sorted by registration no. are as following:\n')**

**print(\*k, sep='\n') # --> b)**

**print('b) Names of students with no. of years less than 3:\n')**

**for \_ in ls:**

**if int(\_[3]) < 3:**

**print(f"{' '.join(\_[:2])}")**

**print('b) No. of people in each department:\n')**

**print(\*[i + ':' + str(dic[i]) for i in dic], sep='\n')# --> b)**

Sol 4: -

# PRACTICAL 4

"""

Writing a program that reads a file “myfile.txt” and builds a histogram

(a dictionary having key value pair as word: occurrence) of the words in the file.

a) Now use histogram to print-

i) Total number of words

ii) Number of different words

iii) The most common words

b) Using above text file “myfile.txt”, write a program that maps a list of words

read from the file to an integer representing the length of the corresponding words.

(use dictionary having key value pair as length : list of word )

Now using above dictionary design a function find\_longest\_word() to display a list of longest words from file.

Define a function filter\_long\_words(n) that takes an integer n and returns the list

of words that are longer than n from file.

"""

**\_PATH = ‘myfile.txt’   
def \_\_a\_\_():**

**with open(\_PATH) as f:**

**p = (' '.join(f.read().split('\n'))).strip().split(' ')**

**freq = {}**

**for i in p:**

**if i not in freq:**

**freq[i] = 1**

**else:**

**freq[i] += 1**

**print(freq)**

**print(f'a)\ni)Total no. of words \n{sum(freq.values())}') # -->a) i)**

**print(f'ii)No. of different words:\n{len(freq)}') # --> a) ii)**

**print('iii)Most common word:\n', \***

**[x for x in freq if freq[x] == max(freq.values())]) # -->**

**\_\_a\_\_()**

**def \_\_b\_\_():**

**freq = freq\_create()**

**print('Longest words in file:\n', \*[freq[x]**

**for x in freq if x == max(freq.keys())])**

**filter\_long\_words(**

**freq, input('Enter value for which you want to filter:')) # --> b)**

**def freq\_create():**

**with open(\_PATH) as f:**

**p = (' '.join(f.read().split('\n'))).strip().split(' ')**

**freq = {}**

**for i in p:**

**if len(i) not in freq:**

**freq[len(i)] = [i]**

**elif i not in freq[len(i)]:**

**freq[len(i)] += [i]**

**return freq**

**def filter\_long\_words(freq, n): # -->b)**

**print(\*[freq[\_] for \_ in freq if \_ > int(n)], sep='\n')**

**\_\_b\_\_()**

Sol 5: -

# PRACTICAL 5

"""

A dictionary Customer contains the following keys {roomno,name,duration}

A binary file “hotel.dat” contains details of customer checked in the hotel.

Write Code in python to perform the following using pickle module

(i) Read n dictionary objects and load them into the file

(ii) Read all the dictionary objects from the file and print them

(iii) Counts the number of customers present in the hotel.

(iv) Display those customers from the file, who have stayed more than 2 days in the hotel

"""

**import pickle**

**ROOM = []**

**def ui():**

**print('\nWelcome to pickle module example customer query system (using binary files)')**

**print('1- add dictionary objects \n\**

**2- Read and print all customer data(all dict objects) \n\**

**3- Count no. of customers in the hotel \n\**

**4- Display customers that have stayed longer than 2 days. \n\**

**5- Quit program')**

**index = input('Enter index no. corresponding to your desired action: ')**

**if index == '5':**

**print('Terminating...')**

**return**

**elif index == '1':**

**add()**

**elif index == '2':**

**print('\n All costumer data in the records: \n')**

**logd(2)**

**elif index == '3':**

**cnt()**

**elif index == '4':**

**stay2()**

**else:**

**print('invalid code')**

**ui()**

**def add(): # --> (i)**

**with open('hotel.dat', 'wb') as f:**

**n = int(input("No. of records to be added: "))**

**for \_ in range(n):**

**tup = input("Enter records one by one with fields seperated by spaces (room-no. name duration): ").split(' ')**

**# --> Unique room no. check not necessary, but added for error handling**

**global ROOM**

**if tup[0] not in ROOM:**

**ROOM.append(tup[0])**

**else:**

**print('ERROR... That room no. seems to be occupies')**

**return**

**tup[-1] = int(tup[-1])**

**customer = dict(zip(('Room No.', 'Name', 'Duration'), tuple(tup)))**

**pickle.dump(customer, f)**

**print('records added...')**

**def logd(x): # --> (ii)**

**if x == 0:**

**num = 0**

**elif x == 1:**

**L = []**

**with open('hotel.dat', 'rb') as f:**

**while True:**

**try:**

**d = pickle.load(f)**

**num = num + \**

**1 if x == 0 else L.append(d) if x == 1 else print(d)**

**except BaseException:**

**break**

**if x == 0:**

**return num**

**elif x == 1:**

**return L**

**def cnt():**

**print('\nNo. of customers in the hotel')**

**print(logd(0))**

**def stay2():**

**print("\nCustomers who have stayed longer than 2 days:")**

**L = logd(1)**

**L = [i for i in L if i['Duration'] > 2]**

**print(\_ for \_ in L)**

**ui()**

Sol 6: -

# PRACTICAL 6

"""

Sun Microsystems when held recruitment test. The file placement.csv containing the below format of data  
The marks are from 5 different tests conducted and each col is out of 5 marks

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SNO | NAME | MARKS1 | MARKS2 | MARKS3 | MARKS4 | MARKS5 |
| 1 | JOHN | 4 | 3 | 4 | 2 | 5 |
| 2 | PETER | 3 | 4 | 4 | 3 | 5 |

a) Read the above file and print the data

b) Write User Defined Function to find total no of people who came for the placement test   
c) Write the UDF to find the top n Names on basis of total Marks

"""

**import csv**

**f=open('placement.csv','r')**

**reader=csv.reader(f)**

**for line in reader:**

**print(line)**

**def total\_people():**

**i=0**

**f.seek(0)**

**for rec in reader:**

**if rec[0]=='SNO':**

**pass**

**else:**

**i=i+1**

**return i**

**def top():**

**n=int(input("Enter no. for top 'n' names: "))**

**d={}**

**l=[]**

**f.seek(0)**

**for lst in reader:**

**if lst[0]=='SNO':**

**pass**

**else:**

**total=0**

**for m in range (2,7):**

**total=total+int(lst[m])**

**l.append(total)**

**d[lst[1]]=total**

**l.sort(reverse=True)**

**l2=l[0:n]**

**print("\n")**

**print("Top",n,"people:")**

**for element in l2:**

**for key in d:**

**if d[key]==element:**

**print(key,'(marks:',d[key],')')**

**print("\n")**

**print("Total no. of people:",total\_people())**

**top()**

Sol 7: -

# PRACTICAL 7

"""

Write a program to input a number and then call the functions:

-count(n) which returns the number of digits

-reverse(n) which returns the reverse of a number

-hasdigit(n) which returns True if the number has a digit else False.

-show(n) to show the number as sum of place value

"""

**def reverse(n):**

**return str(n)[::-1]**

**def count(n):**

**return len(str(n))**

**def hasdigit(n):**

**check = '01234567890'**

**# If this was intended to check for some specific digit,**

**# REPLACE check with sting character of that number.**

**p = [True for i in str(n) if str(i) in check]**

**sets = True if True in p else False**

**return sets**

**def show(n):**

**n = str(n)**

**ls = ' + '.join([n[i] + '0' \* (len(n) - 1 - i) for i in range(len(n))])**

**return n + ' = ' + ls**

**# if you want to see output, call print() on these functions:**

**def ui():**

**num = int(input('Enter no. you want to analyse:'))**

**inp = int(input('For getting count press 1\n\**

**For getting reversed string press 2\n\**

**For checking for digits press 3\n\**

**For getting sum of digits press 4\n\**

**To quit, press 0:\n'))**

**if inp == 1:**

**print(count(num))**

**elif inp == 2:**

**print(reverse(num))**

**elif inp == 3:**

**print(hasdigit(num))**

**elif inp == 4:**

**print(show(num))**

**else:**

**return**

**ui()**

**ui()**

Sol 8: -

# PRACTICAL 8

"""

A Number is a perfect number if the sum of all the factors of the number (including 1) excluding itself is equal to number.

For example: 6 = 1+2+3 and 28=1+2+4+7+14  
Number is a prime number if it 's factors are 1 and itself.  
Write functions i) Generatefactors() to populate a list of factors  
ii) isPrimeNo() to check whether the number is prime number or not iii) isPerfectNo() to check whether the number is perfect number or not

Save the above as a module perfect.py and use in the program main.py as a menu driven program.

"""

**————————————————————————————————————  
# perfect.py**

**def generate\_factors(n):**

**ls=[]**

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

**if n%i==0:**

**ls.append(i)**

**ls.append(n)**

**return ls**

**def is\_prime\_no(n):**

**l= generate\_factors(n)**

**if len(l)==2:**

**return True**

**else:**

**return False**

**def is\_perfect\_no(n):**

**l=generate\_factors(n)[:-1]**

**sums=0**

**for i in l:**

**sums+=i**

**if sums==n:**

**return True**

**else:**

**return False**

**if \_\_name\_\_ == '\_\_main\_\_':**

**inp=int(input('Enter no: '))**

**print(f'Perfect No : {is\_perfect\_no(inp)}')**

**print(f'Prime No : {is\_prime\_no(inp)}')**

**print(f'Factors of {inp} are: {generate\_factors(inp)}')**

**————————————————————————————————————  
# main.py**

**from perfect import generate\_factors, is\_prime\_no, is\_perfect\_no**

**def main():**

**OPTIONS = [**

**"1- Get factors of number",**

**"2- Check if a number is prime",**

**"3- Check if a number is a perfect no.",**

**"4- Check if a number is prime and get factors for that no.",**

**"5- Check if a number is a pefect no. and get factors for that no.",**

**"6- Quit this program"**

**]**

**WELCOME\_MSG = "\nHello, welcome to pratical #8:\n\**

**Menu driven program for getting list of factors,\n\**

**checking if a number is prime,\n\**

**and to check if a number is a perfect no.\n\n"**

**try:**

**print(WELCOME\_MSG + '\n'.join(OPTIONS) + '\n')**

**choice = int(input('Enter action of your choice: '))**

**if choice == 6:**

**print('Bye...!')**

**return**

**else:**

**num = int(input('Enter number: '))**

**truth\_eval = lambda x: 'a' if x(num) else 'not a'**

**if choice == 1:**

**print(f'The factors of {num} are:\n{generate\_factors(num)}')**

**elif choice == 2:**

**print(f'The no. {num} is {truth\_eval(is\_prime\_no)} prime no.')**

**elif choice == 3:**

**print(f'the no. {num} is {truth\_eval(is\_perfect\_no)} perfect no.')**

**elif choice == 4:**

**print(f'The no. {num} is {truth\_eval(is\_prime\_no)} prime no.')**

**print(f'The factors of {num} are:\n{generate\_factors(num)}')**

**elif choice == 5:**

**print(f'The factors of {num} are:\n{generate\_factors(num)}')**

**print(f'The no. {num} is {truth\_eval(is\_perfect\_no)} perfect no.')**

**else:**

**print("That doesn't seem to be a valid option.\nPlease try again...")**

**retry()**

**except Exception as e:**

**print(f'The factors of {num} are:\n{generate\_factors(num)}')**

**print(f'An error occured: {e}\nPlease try again...')**

**retry()**

**def retry():**

**to\_reuse = str(input('Would you like to reuse this program? [Y/n]: '))**

**if to\_reuse == 'n':**

**print('Bye...!')**

**return**

**else:**

**main()**

**main()**

Sol 9: -

# PRACTICAL 9

**"""**

**pascal's triangle**

**Since, recursion is cut from syllabus, here is an iterative version.**

**However, I may also make a recursive one.**

**"""**

**n = int(input("Enter number of rows in the Pascaline Triangle-"))**

**p = []**

**for i in range(n):**

**p.append([])**

**p[i].append(1)**

**for j in range(1, i):**

**p[i].append(p[i - 1][j - 1] + p[i - 1][j])**

**if(n != 0):**

**p[i].append(1)**

**for i in range(n):**

**print(" " \* (n - i), end=" ", sep=" ")**

**for j in range(0, i + 1):**

**print('{0:6}'.format(p[i][j]), end=" ", sep=" ")**

**print()**

Sol 10: -

# PRACTICAL 10

"""

NUMBER BASE CHANGER

Since, recursion is cut from syllabus, here is an iterative version.

However, I may also make a recursive one.

"""

**def datacv():**

**inp = int(input('Enter the No. you want to convert-'))**

**print('Conversion type key-\n B-Binary \n O-Octal \n H-Hexadecimal')**

**typ = input("Which type of conversion do you want to execute-\n")**

**a = inp**

**ls = []**

**k = 1**

**if (typ == 'B'):**

**if (a == 0):**

**ls.append(0)**

**if (a == 1):**

**ls.append(1)**

**while(a >= 1):**

**b = a**

**if(a != 3):**

**a = a // 2**

**if (a >= 1):**

**l = b % a**

**ls.insert(0, l)**

**elif(a >= 0):**

**ls.insert(0, k)**

**else:**

**ls.insert(0, k)**

**a = 1**

**elif(typ == 'O'):**

**while(a > 0):**

**b = a % 8**

**a = (a - b) // 8**

**ls.insert(0, b)**

**elif(typ == 'H'):**

**while(a > 0):**

**b = a % 16**

**a = (a - b) // 16**

**if (b <= 9):**

**ls.insert(0, b)**

**else:**

**lst = ['A', 'B', 'C', 'D', 'E', 'F']**

**c = lst[b - 10]**

**ls.insert(0, c)**

**s = [str(i) for i in ls]**

**res = ["".join(s)]**

**print(res[0])**

**retry()**

**def retry():**

**A = input('''**

**Do you want to use the converter again?**

**Please type 1 for YES or 0 for NO.**

**''')**

**if (A == '1'):**

**datacv()**

**elif(A == '0'):**

**print('Bye...')**

**else:**

**retry()**

**datacv()**

Sol 11: -

# PRACTICAL 11

"""  
Write a program to input a list and write the function for the following:   
i) To sort list using bubble sort and find efficiency   
ii)  To search an element using binary search and find efficiency   
iii)  To search an element using linear search and find efficiency

"""

**lst=eval(input("Enter list of numbers: "))**

**def bubble\_sort(lst):**

**n=len(lst)**

**for i in range(n):**

**for j in range(0,n-i-1):**

**if lst[j]>lst[j+1]:**

**lst[j],lst[j+1]=lst[j+1],lst[j]**

**bubble\_sort(lst)**

**print("The sorted list is",lst)**

**def binary\_search(lst,value):**

**low=0**

**high=len(lst)-1**

**while low<=high:**

**mid=(low+high)//2**

**if lst[mid]==value:**

**return mid**

**elif lst[mid]<value:**

**low=mid+1**

**elif lst[mid]>value:**

**high=mid-1**

**else:**

**return "Not found"**

**value1=int(input("Enter element to be searched (using binary search): "))**

**print("Using binary search",value1,"is found in the list at",binary\_search(lst,value1))**

**def linear\_search(lst,value):**

**index=0**

**while index<len(lst) and lst[index]<value:**

**index+=1**

**if index>=len(lst) or lst[index]!=value:**

**return "Not found"**

**return index**

**value2=int(input("Enter element to be searched (using linear search): "))**

**print("Using linear search",value2,"is found in the list at",linear\_search(lst,value2))**

Sol 12: -

# PRACTICAL 12

"""

This might not work here as this is running on a server. (No Display)

However, this can be run on any other computer with python, tcl and the tkinter module.

"""

**from tkinter import messagebox, Tk, Label, Button, Entry**

**# the modules were listed seperately to improve execution time(less no. of**

**# imports)**

**wind = Tk()**

**wind.title('Simple Interest Calculator')**

**greet = Label(**

**wind,**

**text='Welcome to Simple Interest Calculator!').grid(**

**row=0,**

**column=0,**

**columnspan=2,**

**pady=6)**

**p\_l = Label(wind, text='Enter Principal: ').grid(row=1, column=0)**

**r\_l = Label(wind, text='Enter Rate: ').grid(row=2, column=0)**

**t\_l = Label(wind, text='Enter time: ').grid(row=3, column=0)**

**p = Entry(wind)**

**p.grid(row=1, column=1)**

**r = Entry(wind)**

**r.grid(row=2, column[?12; 4$y=1)**

**t= Entry(wind)**

**t.grid(row=3, column=1)**

**def si():**

**try:**

**si\_var = int(p.get()) \* int(r.get()) \* int(t.get()) / 100**

**m = messagebox.showinfo("Simple interest", f'Simple Interest: {si\_var}')**

**except Exception as e:**

**m= messagebox.showerror("ERROR!!!", e)**

**but= Button(wind, text= 'Calculate Simple Interest', command= si).grid(row= 4, column = 0, columnspan =2, pady= 5)**

**wind.mainloop()**

Sol 13: -

# PRACTICAL 13

"""

Use urllib3 module.

"""

**import urllib3**

**http = urllib3.PoolManager()**

**res = http.request('GET',**

**'https://www.pythonforbeginners.com/')**

**head = res.info()**

**# Print Headers-**

**print(f"Headers - {head}")**

**# Print Date-**

**print(f"Date- {head['date']}")**

**# Print Server-**

**print(f"Server- {head['server']}")**

**print(‘Writing data…’)**

**with open('downlaod.htm', 'w') as htm:**

**htm.write(str(res.data))**

Sol 14: -

# PRACTICAL 14

"""

RING GAME-

Create a stack to take in stack of numbers and then simulate a ring game.

A ring stand is such that only a ring of higher diameter can be placed on lower one.

The diameters are given by the user the program will compare the diameter of ring at

stack top with the diameter of ring to be placed if condition specified is true ring

is added to the stack otherwise keep popping and put them into temporary ring stand

to arrange them into specific order.

"""

**import time**

**s = []**

**top = None**

**def ui():**

**inp = int(input('\nPress 1 to play and add ring.\n\**

**Press 2 to Skip chance.\n\**

**Press 3 to Quit Game.\n'))**

**if inp == 1:**

**ring = int(input('Enter size of ring: '))**

**print(add\_ring(ring))**

**print(show\_stack())**

**elif inp == 2:**

**print('You have missed this chance.')**

**print(show\_stack())**

**elif inp == 3:**

**print('bye')**

**return**

**time.sleep(2)**

**ui()**

**def add\_ring(ring):**

**print('Adding ring...')**

**if isempty(s):**

**push(s, ring)**

**return 'RING HAS BEEN ADDED.'**

**substack = []**

**peeked\_ring = peek()**

**while ring < peeked\_ring or peeked\_ring == 'UNDERFLOW':**

**push(substack, stk\_pop(s))**

**peeked\_ring = peek()**

**if peeked\_ring == 'UNDERFLOW':**

**break**

**push(s, ring)**

**while not isempty(substack):**

**push(s, stk\_pop(substack))**

**return 'RING HAS BEEN ADDED.'**

**def isempty(stk):**

**if len(stk) == 0:**

**return True**

**else:**

**return False**

**def push(stk, el):**

**stk.append(el)**

**def stk\_pop(stk):**

**if len(stk) == 0:**

**return 'UNDERFLOW'**

**else:**

**p = stk.pop()**

**return p**

**def peek():**

**if len(s) == 0:**

**return 'UNDERFLOW'**

**else:**

**top = len(s) - 1**

**return s[top]**

**def show\_stack():**

**return f'The current stack is {s}\n'**

**ui()**

Sol 15: -

# PRACTICAL 15

"""

Create a program to take in a list reg\_no, Name,admission\_to\_class (Nursery, KG, I) and add member functions to

i) Add data to the queue.

ii) Display length of the queue.

iii)Print a report showing number of applications received for admission to each class

"""

**q = []**

**def ui():**

**opt = int(input('Press 1 to add entry to queue\n\**

**Press 2 to fetch the record in order\n\**

**Press 3 to show entire queue of records\n\**

**Press 4 to show total no. of applications\n\**

**Press 5 to show no. of applications per class\n\**

**Press 0 to QUIT.\n'))**

**if opt == 0:**

**print('Bye')**

**return**

**print()**

**if opt == 1:**

**reg = input('Enter registration no.: \n')**

**nm = input('Enter Name: \n')**

**cl = input('Enter class being admitted to(Nursery, KG OR I): \n')**

**dat = [reg, nm, cl]**

**enqueue(dat)**

**print('Entry added...\n')**

**elif opt == 2:**

**print(dequeue())**

**elif opt == 3:**

**print(show\_q())**

**elif opt == 4:**

**print(f'The no. of applications in record are: {len\_q()}')**

**elif opt == 5:**

**print('\n\n', fetch\_report(), '\n')**

**print()**

**ui()**

**def fetch\_report():**

**queue\_backup = q[::]**

**report = {}**

**for \_ in range(len\_q()):**

**entry = dequeue()[-1]**

**if entry in report:**

**report[entry] += 1**

**else:**

**report[entry] = 1**

**report = '\n'.join([f'{i}:{report[i]}' for i in report])**

**return report**

**def show\_q():**

**val = ''**

**for i in q:**

**val = val + str(i) + '\n'**

**return val**

**def len\_q():**

**return len(q)**

**def enqueue(el):**

**q.append(el)**

**def dequeue():**

**return q.pop(0)**

**ui()**

Database Management(**MySQL** + Python)

Q1:- Consider the following WATCHES and SALE table and Write the command in MYSQL for (i) to (v):

**Table:** WATCHES

 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Watchid** | **Watch\_Name** | **Price** | **Type** | **Qty\_Store** |
| W001 | High Time | 10000 | Unisex | 100 |
| W002 | LifeTime | 15000 | Ladies | 150 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W003 | Wave | 20000 | Gents | 200 |
| W004 | High Fashion | 7000 | Unisex | 250 |
| W005 | Golden Time | 2500 | Gents | 100 |

**Table:** SALE

|  |  |  |
| --- | --- | --- |
| **WatchId** | **Qty\_Sold** | **Quarter** |
| W001 | 10 | 1 |
| W003 | 5 | 1 |
| W002 | 20 | 2 |
| W003 | 10 | 2 |
| W001 | 15 | 3 |
| W002 | 20 | 3 |
| W005 | 10 | 3 |
| W003 | 15 | 4 |

i) To display watch name and their quantity sold in first quarter.   
ii) To display the details of those watches whose name ends with ‘Time’?   
ii) To display total quantity in store of Unisex type watches.  
iv) To display watch’s name and price of those watches which have price range in between 5000-15000.  
v) To display Quantity sold of all watches WatchId wise.

Sol 1:-i) SELECT Watch\_Name, Qty\_Sold FROM WATCHES W, SALES S WHERE W.WatchId = S.WatchId AND S.Quarter = 1;

ii) SELECT \* FROM WATCHES WHERE Watch\_Name LIKE ‘%Time’;  
 iii) SELECT SUM(Qty\_Store) FROM WATCHES WHERE Type LIKE ‘Unisex’;

iv) SELECT Watch\_Name, Price FROM WATCHES WHERE Price BETWEEN 5000 AND 15000;

v) SELECT WatchId, SUM(Qty\_Sold) FROM SALE GROUP BY WatchId;

Q2:- Consider the table “ITEM” having the following fields

**Itemcode varchar  
Itemname varchar  
Price float**i) Create the table ITEM in the mydb database  
ii) Create a menu driven program in python to have a) function for inserting records in the table   
b) function for displaying all the records from the table item  
c) Function for searching for a particular record on basis of Itemcode

Sol 2:-i)

import os  
import mysql.connector  
db\_user = os.environ.get(‘DB\_USER’)  
db\_pass = os.environ.get(‘DB\_PASS’)  
mydb = mysql.connector.connect(  
 host = ‘localhost’,  
 user = db\_user,  
 password = db\_pass,  
 database = ‘mydb’  
)  
cur = mydb.cursor()  
mycursor.execute(“CREATE TABLE ITEM (Itemcode VARCHAR(255), Itemname VARCHAR(255), Price FLOAT)”)

ii) #———menu-driven-system———

**def menu():  
 print(‘The following database actions can be performed:\n\  
1- Add a new item to the table\n\  
2- See a list of all items from the table\n\  
3- Search for records on the basis of Itemcode\n\  
4- EXIT’)  
 option = int(input(‘Select option[1/2/3/4]: ’))**   
  
  
  
**def insert\_record(table = ‘ITEM’):  
 QUERY = f“INSERT INTO {table}”  
 cur.execute(QUERY)**

Q3:-