Adham Ramoun

CU1901556

Doctor Appointment Organizing system

ORGANIZING ALL THE APPOINTMENTS DATA AND SORT THEM

Table of Contents

Introduction................................................................................................................................................. 2 Inputs........................................................................................................................................................... 2 Outputs........................................................................................................................................................ 3 Decisions and Assumptions ......................................................................................................................... 3 Data structures............................................................................................................................................ 3 Functions ..................................................................................................................................................... 4 Algorithm template ..................................................................................................................................... 4 Scenarios...................................................................................................................................................... 6

Conclusion.................................................................................................................................................... 6 Appendix..................................................................................................................................................... .7

# 

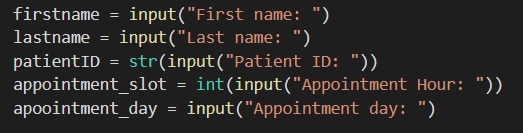
# Introduction

This code is designed to take inputs from the user about the doctors and their patients and their appointment data, and then sort them and filter them and print the data in the desired way by the user of the program. First the Doctor class and the patient class functions are written, then the program collects inputs from the users to fill in the information about the doctors and patients and assign each patient to a doctor

# Inputs

# 

First name, Last name, doctor ID and, the number of patients for this doctor. The first three inputs are set as strings, as names contain only letters, and the ID may be alpha numeric so it needs to be a string. Patients number is an integer, because it is used as a variable for setting for loops so it needs to be a number.



Details of the patients of the doctor is collected from the user



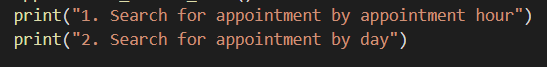
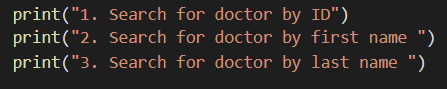
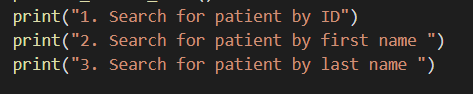
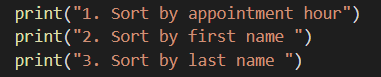
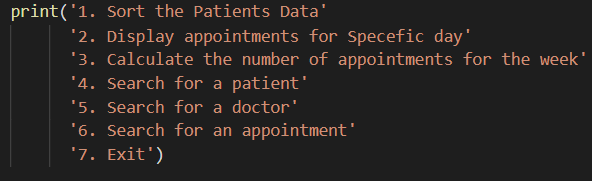








# Outputs



# Decisions and Assumptions

The user chooses the number of doctors and patients, and also assign the patients to the corresponding doctor. He can also choose how does he wants to sort in the list and can choose what any option from the menu as for example, if he wants to search of any doctor or patient.

## Data structures

The time of the appointment and the menu option was integers, to allow for the use of the menu option in the if statement of the menu, and the time of the appointment was integer to allow for a validation function to make sure it is not more than 24 or less than zero. The doc\_no which is the number of the doctor you choose is also integer

First name, last name, doctor ID, patient ID, and day of appointment was a string to make it easier to sort and store and avoid any errors

# Functions

**List the functions using the following template (table)**

|  |  |
| --- | --- |
| Function name | sortOnhour |
| arguments | self |
| return | \_list |

|  |  |
| --- | --- |
| Function name | sortOnName |
| arguments | Self, name |
| return | arr |

|  |  |
| --- | --- |
| Function name | populate\_doctor |
| arguments | None |
| return | doctor |

|  |  |
| --- | --- |
| Function name | populate\_patients |
| arguments | Doctor, NoOfDoctors |
| return | Doctor |

# Algorithm template

|  |  |
| --- | --- |
| Name | Selection sort |
| Input | The Unsorted list of first name, last name, patient ID, appointment day, and Appointment hour |
| Output | There is no a printed output, however, the data is stored sorted inside the program |
| Content | The algorithm sorts and unsorted list accordingly with other lists, whether list1 is first name, last name. Then it sorts the sorted list with the other lists accordingly. |
| Solution | def sortOnhour(self):          \_list = []          \_list = self.listOfPatients.copy()          for i in range(len(\_list)):              min\_idx = i              for j in range(i + 1, len(\_list)):                  min = self.returnPatient(\_list, min\_idx)                  a\_min = self.returnPatient(\_list, j)                  if min.appintmentslot > a\_min.appintmentslot:                      min\_idx = j              \_list[i], \_list[min\_idx] = \_list[min\_idx], \_list[i]          return \_list  def sortOnName(self, name):          arr = []          arr = self.returnPatient.copy()          n = self.NoOfPatients          for i in range(n):              min\_index = i              for j in range(i + 1, n):                  min = self.returnPatient(arr, min\_index)                  a\_min = self.returnPatient(arr, j)                  if name == 1:                      if min.firstName > a\_min.firstName:                          min\_index = j                  elif name == 2:                      if min.lastName > a\_min.lastName:                          min\_index = j              if min\_index != i:                  temp = arr[i]                  arr[i] = arr[min\_index]                  arr[min\_index] = temp          return arr |
| Analysis | O(n^2) :n stands for comparison of names alphabetically for the first name or last name or by score. I chose to use this method or algorithm over others since it is easier to use than others and is more efficient when using and is better with nested lists (2d lists) |
| Variations | Bubble sort, quick sort |

# Scenarios

First the user is asked in input how many doctors there are in the system, then enters the details of the doctor, Then the data of each patient is input one by one. The user chooses from the menu what does he want to sort or display,

If the chooses 1:

The list is sorted by first name and then the patient data is printed

If the user chooses 2:

Specific day appointments are printed

If the user chooses 3:

Total appointments of the week is printed

If the user chooses 4:

The user searches for a patient using first name or last name or ID

If the user chooses 5:

The user searches for a doctor using first name or last name or ID

If the user chooses 6:

The user searches for an appointment by either entering the hour or the day of the appointment

If the user chooses 7:

The loop closes and the program ends

# Conclusion

The possible improvement to the programme was to use Try and except blocks to try to minimize the number of errors through the execution of the program. Also, The day of the appointment could be chosen from the list to prevent the user from typing a wrong input. There was some un identified errors that prevent the code from running correctly that I could not figure it out.

# Appendix

import os

from typing import List, Any

class Patient:

    firstName = 'Adham'

    lastName = 'Ramoun'

    patientID = 0

    appintmentslot = 0

    appointmentDay = 'Sunday'

    def \_\_init\_\_(self):

        self.patientID = 0

    def patientID\_setter(self, id):

        self.patientID = id

    def firstName\_setter(self, name):

        self.firstName = name

    def lastName\_setter(self, name):

        self.lastName = name

    def appointmentDay\_setter(self, day):

        self.appointmentDay = day

    def appintmentslot\_setter(self, slot):

        self.appintmentslot = slot

    def firstName\_getter(self):

        return self.firstName

    def lastName\_getter(self):

        return self.lastName

    def appointmentDay\_getter(self):

        return self.appointmentDay

    def appintmentslot\_getter(self, number):

        return self.appintmentslot

    def patientID\_getter(self):

        return self.patientID

    def display\_patient\_data(self):

        print(f'ID: {self.patientID}  Patient Name: {self.firstName} {self.lastName} Appintment day: {self.appointmentDay} at {self.appintmentslot}')

class doctor:

    firstName = 'First'

    lastName = 'Last'

    doctorID = 1

    NoOfPatients = 0

    NoOfDoctors = 0

    listOfPatients = []

    def \_\_init\_\_(self):

        self.doctorID = 0

    def firstName\_setter(self,name):

        self.firstName = name

    def lastName\_setter(self,name):

        self.lastName = name

    def NoOfPatients\_setter(self, number):

        self.NoOfPatients = number

    def NoOfDoctors\_setter(self, number):

        self.NoOfDoctors = number

    def doctorID\_setter(self, id):

        self.doctorID = id

    def listOfPatients\_setter(self, \_list):

        self.listOfPatients = \_list.copy()

    def firstName\_getter(self):

        return self.firstName

    def lastName\_getter(self):

        return self.lastName

    def doctorID\_getter(self):

        return self.doctorID

    def NoOfPatients\_getter(self):

        return self.NoOfPatients

    def NoOfDoctors\_getter(self):

        return self.NoOfDoctors

    def Count\_Week(self, number):

        print("There are total of ", NoOfPatients, " Appointment for the week")

    def returnPatient(self, \_list, index):

        i = 0

        for Patient in \_list:

            if index == i:

                return Patient

            i = i + 1

    def sortOnhour(self):

        \_list = []

        \_list = self.listOfPatients.copy()

        for i in range(len(\_list)):

            min\_idx = i

            for j in range(i + 1, len(\_list)):

                min = self.returnPatient(\_list, min\_idx)

                a\_min = self.returnPatient(\_list, j)

                if min.appintmentslot > a\_min.appintmentslot:

                    min\_idx = j

            \_list[i], \_list[min\_idx] = \_list[min\_idx], \_list[i]

        return \_list

    def sortOnName(self, name):

        arr = []

        arr = self.returnPatient.copy()

        n = self.NoOfPatients

        for i in range(n):

            min\_index = i

            for j in range(i + 1, n):

                min = self.returnPatient(arr, min\_index)

                a\_min = self.returnPatient(arr, j)

                if name == 1:

                    if min.firstName > a\_min.firstName:

                        min\_index = j

                elif name == 2:

                    if min.lastName > a\_min.lastName:

                        min\_index = j

            if min\_index != i:

                temp = arr[i]

                arr[i] = arr[min\_index]

                arr[min\_index] = temp

        return arr

def main\_menu():

    print('1. Sort the Patients Data'

          '2. Display appointments for Specefic day'

          '3. Calculate the number of appointments for the week'

          '4. Search for a patient'

          '5. Search for a doctor'

          '6. Search for an appointment'

          '7. Exit')

def sort\_menu():

    print("1. Sort by appointment hour")

    print("2. Sort by first name ")

    print("3. Sort by last name ")

def patient\_search\_menu():

    print("1. Search for patient by ID")

    print("2. Search for patient by first name ")

    print("3. Search for patient by last name ")

def doctor\_search\_menu():

    print("1. Search for doctor by ID")

    print("2. Search for doctor by first name ")

    print("3. Search for doctor by last name ")

def appointment\_search\_menu():

    print("1. Search for appointment by appointment hour")

    print("2. Search for appointment by day")

def populate\_doctor():

    doctors = []

    for i in range(NoOfDoctors):

        firstName = str(input("First Name of the Doctor: "))

        lastName = str(input("Last Name of the Doctor: "))

        doctor\_id = str(input("Enter Doctor's ID: "))

        patients\_number = int(input("Enter number of Patients : "))

        doc\_patients = []

        doctor = doctors()

        doctor.doctorID\_setter(doctorID)

        doctor.firstName\_setter(firstName)

        doctor.lastName\_setter(lastName)

        doctor.NoOfPatients\_setter(patients\_number)

        doctor.append(doctors)

    return doctor

def populate\_patients(module, number\_of\_module):

    patients = []

    branch\_patients = []

    for i in range(NoOfDoctors):

        Patient.append(branch\_patients)

        Patient[i].clear()

    for i in range(doctor[0].NoOfPatients\_getter()):

        firstname = input("First name: ")

        lastname = input("Last name: ")

        patientID = str(input("Patient ID: "))

        appointment\_slot = int(input("Appointment Hour: "))

        apoointment\_day = input("Appointment day: ")

        Patient = patients()

        Patient.firstName\_setter(firstname)

        Patient.lastName\_setter(lastname)

        Patient.patientID\_setter(patientID)

        Patient.appointmentDay\_setter(apoointment\_day)

        Patient.appointmentslot\_setter(appointment\_slot)

        patient[index].append(patients)

    for i in range(number\_of\_module):

        Doctor[i].listOfPatients\_setter(patients[i])     #maybe an error

    return Doctor

if \_\_name\_\_ == '\_\_main\_\_':

    NoOfDoctors = int(input("Enter Number of Doctors: "))

    doctor = populate\_doctor()

    NoOfPatients = int(input('Number of Patients for this doctor: '))

    for number in range(NoOfDoctors):

        doctor[number].NoOfPatients\_setter(NoOfPatients)

    doctor = populate\_patients(doctor, NoOfDoctors)

    \_flag = True

    while \_flag:

        flag = True

        for doc in range(NoOfDoctors):

            print(f'{doc + 1}: ID: {doctor[doc].doctorID\_getter()} First Name: {doctor[doc].firstName\_getter()} Last Name: {doctor[doc].lastName\_getter()} ')

        print(f'{NoOfDoctors + 1 }: Close Program')

        doc\_no = int(input('Choose The Doctor: '))

        if doc\_no > 0 and doc\_no <= NoOfDoctors:

            while flag:

                main\_menu()

                option = int(input('Choose the option from above: '))

                elif option == 2:

                    while True:

                        d = input("Enter the day to print data for: ")

                        for doc in range(NoOfPatients):

                            if patient[appointment\_Day] = d

                            print(f'{doc + 1}: ID: {doctor[doc].doctorID\_getter()} First Name: {doctor[doc].firstName\_getter()} Last Name: {doctor[doc].lastName\_getter()} ')

                elif option == 7:

                    flag = False

        elif doc\_no == NoOfDoctors + 1:

            \_flag = False

os.system('cls')