**Raport proiect la Proiectarea Algoritmilor**

* **gestiunea pacienților unui medic de familie -**

**Craiova**

**2020**

**Python** este un limbaj de programare dinamic [multi-paradigmă](https://ro.wikipedia.org/wiki/Limbaj_de_programare_multiparadigm%C4%83), creat în [1989](https://ro.wikipedia.org/wiki/1989) de programatorul olandez [Guido van Rossum](https://ro.wikipedia.org/w/index.php?title=Guido_van_Rossum&action=edit&redlink=1). Van Rossum este și în ziua de astăzi un lider al comunității de dezvoltatori de software care lucrează la perfecționarea limbajul Python și implementarea de bază a acestuia, CPython, scrisă în [C](https://ro.wikipedia.org/wiki/C). Python este un limbaj multifuncțional folosit de exemplu de către companii ca [Google](https://ro.wikipedia.org/wiki/Google) sau [Yahoo!](https://ro.wikipedia.org/wiki/Yahoo!) pentru programarea aplicațiilor *web*, însă există și o serie de aplicații științifice sau de divertisment programate parțial sau în întregime în Python. Popularitatea în creștere, dar și puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de către programatori specializați și chiar și la predarea limbajului în unele medii universitare. Din aceleași motive, multe sisteme bazate pe Unix, inclusiv [Linux](https://ro.wikipedia.org/wiki/Linux), [BSD](https://ro.wikipedia.org/wiki/BSD) și [Mac OS X](https://ro.wikipedia.org/wiki/Mac_OS_X) includ din start interpretatorul CPython.

Python pune accentul pe curățenia și simplitatea codului, iar sintaxa sa le permite dezvoltatorilor să exprime unele idei programatice într-o manieră mai clară și mai concisă decât în alte limbaje de programare ca [C](https://ro.wikipedia.org/wiki/C). În ceea ce privește paradigma de programare, Python poate servi ca limbaj pentru software de tipul *object-oriented*, dar permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (*garbage collector*). Alt avantaj al limbajului este existența unei ample biblioteci standard de metode.

Python este un limbaj multi-paradigmă, concentrându-se asupra programării imperative, orientate pe obiecte și funcționale, ceea ce permite o flexibilitate mai mare în scrierea aplicațiilor. Din punctul de vedere al sintaxei, Python are un număr de contrucții și cuvinte cheie cunoscute oricărui programator, dar prezintă și un concept unic: nivelul de indentare are semnificație sintactică. Blocurile de cod sunt delimitate prin simplă indentare.

În C un astfel de blocuri sunt deseori desemnate prin acolade, {<cod>}, dar în Python nu este nevoie de astfel de construcții. Nivelele de indentare îndeplinesc această funcție. Această importanță a indentării este foarte suprinzătoare pentru mulți utilizatori noi ai limbajului Python, chiar dacă sunt programatori cu experiență. Dar o astfel de utilizare a indentării permite codului să fie mai ușor de citit și mai compact. Programatorii cu experiență vor indenta implicit codul sursă, oricare ar fi limbajul, fiindcă astfel se permite structurarea codului sursă și evidențierea funcționalității. Python face din această deprindere folositoare în acest sens o cerință strictă.

PyCharm este un mediu de dezvoltare Python pentru programatori profesionisti. PyCharm ofera completare automata a codului, inspectii cod, evidentierea erorilor si remedieri rapide, alaturi de refactorizarea automata de cod si abilitati bogate de navigare. Data Dimensional este partener JetBrains pentru solutia PyCharm in Romania.

Pentru realizarea acestui proiect am creat doua fisiere text, unul in care am scris codul propriu-zis si unul in care am salvat baza de date. Mai jos voi prezenta codul din primul fisier.

class pacienti:  
 def \_\_init\_\_(self,numeP,prenumeP,varsta,boala,medicamente,reteta,activitate):  
 self.NumeP = numeP  
 self.PrenumeP = prenumeP  
 self.Varsta = varsta  
 self.Boala = boala  
 self.Medicamente = medicamente  
 self.Reteta = reteta  
 self.Activitate = activitate  
  
#CISTESTE FISIERUL CARE CONTINE BAZA DE DATE SI CRREAZA O LISTA DE LISTE  
  
def citeste(filename):  
 bazadate =[]  
 fisierdes = open(filename, 'r')  
 for i in fisierdes:  
 lst = i.split(',')  
 #ELIMINAM SPATIILE ALBE  
 lst2 = []  
 for j in lst:  
 js=j.strip()  
 lst2.append(js)  
 pacient=pacienti(lst2[0], lst2[1], lst2[2], lst2[3], lst2[4], lst2[5], lst2[6])  
 bazadate.append(pacient)  
 fisierdes.close()  
 return bazadate  
  
#SALVEAZA BAZA DE DATE PE DISK  
  
def savebazadate(filename,bazadate):  
 fisierdes=open(filename,'w')  
 for pacient in bazadate:  
 str=pacient.NumeP+','+pacient.PrenumeP+','+pacient.Varsta+','+pacient.Boala+','+pacient.Medicamente+','+pacient.Reteta+','+pacient.Activitate+'\n'  
 fisierdes.write(str)  
 fisierdes.close()  
  
#AFISARE BAZA DE DATE  
  
def printbazadate(bazadate):  
 print('Numele pacientului-Prenumele pacientului-Varsta pacientului-Boala pacientului-Medicamentele pacientului-Retetele pacientului-Activitatea pacientului ')  
 for pacient in bazadate:  
 print(pacient.NumeP,',',pacient.PrenumeP,',',pacient.Varsta,',',pacient.Boala,',',pacient.Medicamente,',',pacient.Reteta,',',pacient.Activitate)  
  
#AFISARE DE MESAJE  
  
def afisaremesaj():  
 print('1->introducerea unui nou pacient: ')  
 print('2->afisarea tuturor pacientilor: ')  
 print('3->pacientii clasati in functie de varsta: ')  
 print('4->numarul de pacienti care sufera de o anumita boala: ')  
 print('5->pacientii cu acelasi tratament: ')  
 print('6->pacientii cu un numar de retete/luna mai mare decat un numar dat: ')  
 print('7->numarul de medicamente de un anumit tip: ')  
 print('8->pacientii in functie de domeniul de activitate: ')  
 print('0->iesire')  
  
  
#TESTAREA PROGRAMULUI  
  
#CRITERIUL DE SORTARE  
def crt(j):  
 return j.Varsta  
  
def test(filename):  
 dbx=citeste(filename)  
 z=1  
 while z!=0:  
 afisaremesaj()  
 z=int(input())  
  
 #INTRODUCEREA UNUI NOU PACIENT  
 if z==1:  
 nou=pacienti('Vlad','Catalin','20','Gripa','Paracetamol','1','student')  
 dbx.append(nou)  
 printbazadate(dbx)  
  
 #AFISAREA PACIENTILOR  
 if z==2:  
 printbazadate(dbx)  
  
 #CLASAREA PACIENTILOR IN FUNCTIE DE VARSTA  
 if z==3:  
 print('pacientii clasati de la cei tineri la varstnici: ')  
 dbx.sort(key=crt)  
 printbazadate(dbx)  
  
 #PACIENTII CARE SUFERA DE ACEEASI BOALA  
 if z==4:  
 numar=0  
 boala=input("introduceti boala cautata")  
 for i in dbx:  
 if i.Boala==boala:  
 numar=numar+1  
 print("numar de pacienti care au boala respectiva sunt in numar de :"+str(numar))  
  
 #PACIENTII CU ACELASI TRATAMENT  
 if z==5:  
 nr=0  
 tratament=input("introduceti tratamentul:")  
 for i in dbx:  
 if i.Medicamente==tratament:  
 print(" Numele pacientului : "+i.NumeP + " cu tratamentul : " + tratament)  
   
  
 #PACIENTII CU UN NUMAR DE RETETE/LUNA MAI MARE DECAT UN NUMAR DAT  
 if z==6:  
 numar=input("dati un numar:")  
 for i in dbx:  
 if int(i.Reteta)>int(numar):  
 print("pacientii cu un numar de retete mai mare decat numarul introdus sunt : "+ i.NumeP)  
  
 # 7 - MEDICAMENTELE DE UN ANUMIT TIP  
 if z==7:  
 tratament=input("dati tipul de medicament dorit:")  
 numar = 0  
 for i in dbx:  
 if i.Medicamente==tratament:  
 numar = numar+1  
 print("Numarul de medicamente : "+tratament+" este de : "+str(numar))  
  
  
 #PACIENTII IN FUNCTIE DE DOMENIUL DE ACTIVITATE4  
 if z==8:  
 domeniu=input("dati activitatea pacientului:")  
 for i in dbx:  
 if i.Activitate==domeniu:  
 print(" Numele pacientilor cu activitatea de "+ str(domeniu) + " sunt: " + i.NumeP)  
 if z==0:  
 break  
 savebazadate(filename,dbx)

Am creat clasa „Pacienti” in care am utilizat constructorul clasei pentru a initializata toate variabilele de care am nevoie pentru a realiza lista cu pacientii medicului de familie.

Mai jos sunt prezentate cateva metode utile in lucrul cu fisiere:

def citeste(filename):

def savebazadate(filename,bazadate):

def printbazadate(bazadate):

def afisaremesaj():

Apoi, dupa ce am facut aceste operatii cu fisiere in care citim fisierul, salvam, afisam etc., am realizat si sortarea elementelor componente in functie de varsta pacientilor:

def crt(j):  
 return j.Varsta

Apoi, cu metoda urmatoare am realizat toate actiunile aplicate asupra bazei de date, care apar in meniul de afisare creat anterior.

def test(filename):

Programul va executa o anumita comanda atunci cand de la tastatura se introduce un numar, astfel pentru tasta:

* 1 se va introduce un nou pacient
* 2 se vor afisa toti pacientii
* 3 se vor ordona pacientii in fucntie de varsta
* 4 se vor afisa doar pacientii care sufera de oa numita boala
* 5 se vor afisa doar pacientii care au acelasi tratament
* 6 se vor afisa doar pacientii care au un numar de retete pe luna mai mare decat o valoare data
* 7 se vor afisa medicamentele de un anumit tip
* 8 se vor afisa pacientii in functie de domeniul de activitate
* 0 se incheie programul

Cel de-al doilea fisier pe care l-am creat contine stocate toate elementele bazei de

date, cu toti pacientii:

Miron,Valentin,12,varicela,Odihna,1,elev

Vlad,Catalin,20,Gripa,Paracetamol,1,student

Vlad,Catalin,20,Gripa,Paracetamol,1,student

Ion,Cosmin,23,enterocolita,Nospa,1,student

Marin,Andrei,26,enterocolita,Nospa,1,vanzator

Mirel,Catalin,32,cancer,Citostatice,3,economist

Marcu,Gabriel,45,pneumonie,Augumentin,2,inginer

Intr-un fisier separat, am facut apel la adresa de memorie unde sunt salvate cele doua fisiere, astfel in consola dupa actiunea de lansare in executie a programului se afiseaza toate operatiile pe care vrem sa le facem asupra bazei de date

**Concluzii**

Avand in vedere cele prezentate mai sus, proiectul acesta dezvolta capacitatea de lucru cu operatii de fisiere, blocuri de instructiuni, structuri de date si algoritmi, precum si concepte de programare orientata pe obiecte efectuate in limbajul de programare Python.