**Raport proiect la Proiectarea Algoritmilor**

**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 Apartament:  
 def \_\_init\_\_(self,bloc,scara,numarApartament,numeLocatar,numarPersoane,intretinere,restanta):  
 self.bloc = bloc  
 self.scara = scara  
 self.numarApartament = numarApartament  
 self.numeLocatar = numeLocatar  
 self.numarPersoane = numarPersoane  
 self.intretinere = intretinere  
 self.restanta = restanta  
   
  
#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)  
 apartament=Apartament(lst2[0], lst2[1], lst2[2], lst2[3], lst2[4], lst2[5], lst2[6])  
 bazadate.append(apartament)  
 fisierdes.close()  
 return bazadate  
  
#SALVEAZA BAZA DE DATE PE DISK  
  
def savebazadate(filename,bazadate):  
 date=open(filename,'w')  
 for apartament in bazadate:  
 apartament=str(apartament.bloc)+','+str(apartament.scara)+','+str(apartament.numarApartament)+','+str(apartament.numeLocatar)+','+str(apartament.numarPersoane)+','+str(apartament.intretinere)+','+str(apartament.restanta)+'\n'  
 date.write(apartament)  
 date.close()  
  
#AFISARE BAZA DE DATE  
  
def printbazadate(bazadate):  
 print('Bloc -Scara -Numar apartament- Locatar - Numarul de persoane - Intretinere (medie) - Restanta')  
 for apartament in bazadate:  
 print(apartament.bloc,',',apartament.scara,',',apartament.numarApartament,',',apartament.numeLocatar,',',apartament.numarPersoane,',',apartament.intretinere,',',apartament.restanta)  
  
#AFISARE DE MESAJE  
  
def afisaremesaj():  
 print('1->introducerea unui nou apartament ')  
 print('2->Sortarea locatarilor dupa nume ')  
 print('3->Adaugarea unui nou locatar ')  
 print('4->Cautarea unui locatar ')  
 print('5->Mai mult de 5 persoane ')  
 print('6->Afisarea intretinerilor ')  
 print('7->Restanta mai mare ca suma dorita ')  
 print('8->Clasarea locatarilor in rai platnici sau buni platnici ')  
 print('0->iesire')  
  
  
#TESTAREA PROGRAMULUI  
  
#CRITERIUL DE SORTARE  
def crt1(j):  
 return j.numeLocatar  
  
def test(filename):  
 date=citeste(filename)  
 z=1  
 while z!=0:  
 afisaremesaj()  
 z=int(input())  
  
 # 1 - INTRODUCEREA UNUI NOU APARTAMENT  
   
 if z == 1:  
 lstp = [0,1,2,3,4,5,6]  
   
 lstp[0] = input("Introduceti blocul:")  
 lstp[1] = input("Introduceti scara:")  
 lstp[2] = input("Introduceti numarul apartamentului:")  
 lstp[3] = input("Introduceti numele locatarului:")  
 lstp[4] = input("Introduceti numarul de persoane:")  
 lstp[5] = input("Introduceti media pe intretinere:")  
 lstp[6] = input("Introduceti restanta actuala:")  
   
   
 apartament = Apartament(lstp[0],lstp[1],lstp[2],lstp[3],lstp[4],lstp[5],lstp[6])  
 date.append(apartament)  
 printbazadate(date)  
 print("\*" \*25)   
  
 # 2 - Sortarea locatarilor dupa nume  
 if z==2:  
 date.sort(key=crt1)  
 printbazadate(date)  
 print("\*" \*25)   
  
 # 3 - Adaugarea unui nou locatar  
 if z==3:  
   
 numar=input("introduceti numarul apartamentului pentru care se schimba locatarul : ")  
 for apartament in date:  
 if apartament.numarApartament == numar:  
 locatar = input("introduceti noul locatar : ")  
 apartament.numeLocatar = locatar  
 else:  
 print("Apartamentul nu a fost gasit !")  
 print("\*" \*25)   
  
 # 4 - Cautarea unui locatar  
 if z==4:  
   
 nume=input("introduceti numele locatarului :")  
 for apartament in date:  
 if apartament.numeLocatar == nume:  
 print(" Locatarul se afla in apartamentul : "+apartament.numarApartament)  
 print("\*" \*25)   
  
 # 5 - Mai mult de 5 persoane  
 if z==5:  
 for apartament in date:  
 if int(apartament.numarPersoane)>5:  
 print("In apartamentul : "+apartament.numarApartament + " locuiesc mai mult de 5 persoane")  
 print("\*" \*25)   
 # 6 - Afisarea intretinerilor  
 if z==6:  
 for apartament in date:  
 print("Apartamentul : "+apartament.numarApartament + " are o intretinere de : "+apartament.intretinere)  
 print("\*" \*25)   
   
 # 7 - Restanta mai mare ca suma dorita  
 if z==7:  
 suma = input("Introduceti suma dorita : ")  
 for apartament in date:  
 if int(apartament.restanta) > int(suma):  
 print("Apartamentul : "+apartament.numarApartament + " are de plata mai mult decat suma dorita ")  
 print("\*" \*25)   
   
 # 8 - Clasarea locatarilor in rai platnici sau buni platnici  
 if z == 8:  
 suma = input("Introduceti suma pentru care un locatar devine rau platnic : ")  
 for apartament in date:  
 if int(apartament.restanta) > int(suma):  
 print("Locatarul : "+apartament.numeLocatar + " este rau platnic")  
 else:  
 print("Locatarul : "+apartament.numeLocatar + " este bun platnic")  
 print("\*" \*25)   
   
 if z==0:  
 savebazadate(filename,date)  
 break

Am creat clasa „Apartament” in care am utilizat constructorul clasei pentru a initializata toate variabilele de care am nevoie pentru a realiza lista cu apartamentele unui bloc.

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 numele locatarilor.

#CRITERIUL DE SORTARE  
def crt1(j):  
 return j.numeLocatar

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 – introducerea unui nou apartament
* 2 – sortarea locatarilor dupa nume
* 3 – adaugarea unui nou locatar
* 4 – cautarea unui locatar
* 5 – verifica daca sunt mai mult de 5 persoane in apartament
* 6 – afisarea intretinerilor
* 7 – restanta mai mare decat suma dorita
* 8 – clasarea locatarilor in rai platnici sau buni platnici
* 0 – iesire

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

Bloc11,Sc2,Ap7,Tataru,4,200lei,0lei

BlocB1,Sc1,Ap5,Sisu,2,150lei,200lei

Bloc45,Sc3,Ap10,Popescu,1,400lei,1000lei

BlocF9,Sc5,Ap20,Anghel,3,780,200lei

Bloc21,Sc4,Ap19,ION,5,200lei,145lei

Bloc1,Sc2,Ap4,Voicu,4,600lei,0lei

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.

filename = r'E:\Download\proiect\Baza\_Date.txt'  
fileprog = r'E:\Download\proiect\program.py'  
exec(open(fileprog).read())  
test(filename)

**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.