



Sistem de gestionare a cursanților

Limbaaj: Python ***Nivel dificultate:*** 2/5 ***Domeniu:*** Programare ***Tehologii:-***

David Ana-Luiza

Iunie 2024

1.Descriere

Sistemul de gestionare a cursanților este un program interactiv în limbajul Python destinat gestionării informațiilor cursanților dintr-un curs. Utilizatorul poate introduce detalii personale pentru fiecare cursant, inclusiv prenumele, numele și CNP-ul. Programul include validări stricte pentru a asigura corectitudinea datelor introduse, precum verificarea existenței caracterelor speciale și a cifrelor în nume, și validarea formatului și corectitudinii CNP-ului.

2. Resurse

Aplicații necesare:

- Python 3.12 (sau orice altă versiune de Python mai nouă de 3.8)
- PyCharm

Cunoștințe minime:

- String-uri (https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022_T1/courseware/2ea0d1214cd749a380c9d079cffa1d2a/fdd90e0cf7f3474cb4cf799fb293776b/)
- Liste (https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022_T1/courseware/2ea0d1214cd749a380c9d079cffa1d2a/c2d5ccffaaf3466e89d2656537201076/)
- Dicționar (https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022_T1/courseware/256136442fd6439f8301cc4d64a33ece/066fbecddca64d11ab181b1b2066fe6d/)
- Funcții (https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022_T1/courseware/c82480aa92484b01be70b4003f6e0035/6d2ff2e3f0694fd689f7f36482798fb4/)

3. Implementare

3.1 Descrierea codului

Pentru o implemetare ușoară de scris și în același timp de înțeles/citit codul este format din partea de funcții unde au fost introduse subprograme pentru testele de validare a datelor și partea de citire de date de la tastatură. Mai departe vom detalia și explica functionalitatea subprogramelor.

```
def validare_pre_nume(cuv):  
    x = 0  
    for char in cuv:  
        if char.isdigit():  
            print("invalid:cifra introdusa")  
            x = x + 1  
            return False  
  
        if char in string.punctuation:  
            print("Invalid: caracter special introdus")  
            x = x + 1  
            return False
```

Funcția **validare_pre_nume(cuv)** este folosită atât pentru validarea numelui dar și a prenumelui. Pentru implementare au fost folosite doua if – uri:

- primul pentru a verifica dacă există cifre în numele introdus
- și cel de al doilea pentru a verifica dacă există caractere speciale (@, \$, # etc).

În felul acesta vor fi validate doar intrările care conțin litere.

```
def validare_cnp(cnp):  
    x = 0  
    if len(cnp) != 13:  
        print("Invalid: lungimea CNP-ului nu corespunde")  
        return False  
  
    if cnp.isdigit() != 1:  
        print("Invalid: nu ati introdus cifre")  
        return '!'  
        x = x + 1  
  
    if cifra_control(cnp) != int(cnp[12]):  
        print("Invalid: CNP invalid")  
        x = x + 1  
    return x == 0
```

Funcția **validare_cnp(cnp)** este folosită pentru verificarea codului numeric personal al cursantului. Pentru implementare au fost folosite 3 if – uri:

- primul pentru a verifica lungimea CNP-ului
- al doilea pentru a verifica formatul CNP-ului (trebuie să conțină doar cifre, nu și alte tipuri de caractere)
- al treilea verifică cifra de control (ultima cifră din CNP). Aici am ales crearea unei noi funcții numită **cifra_control(cnp)** pentru validarea CNP-ului conform algoritmului care poate fi găsit pe

<https://cnpgenerator.ro/verificare-cnp>

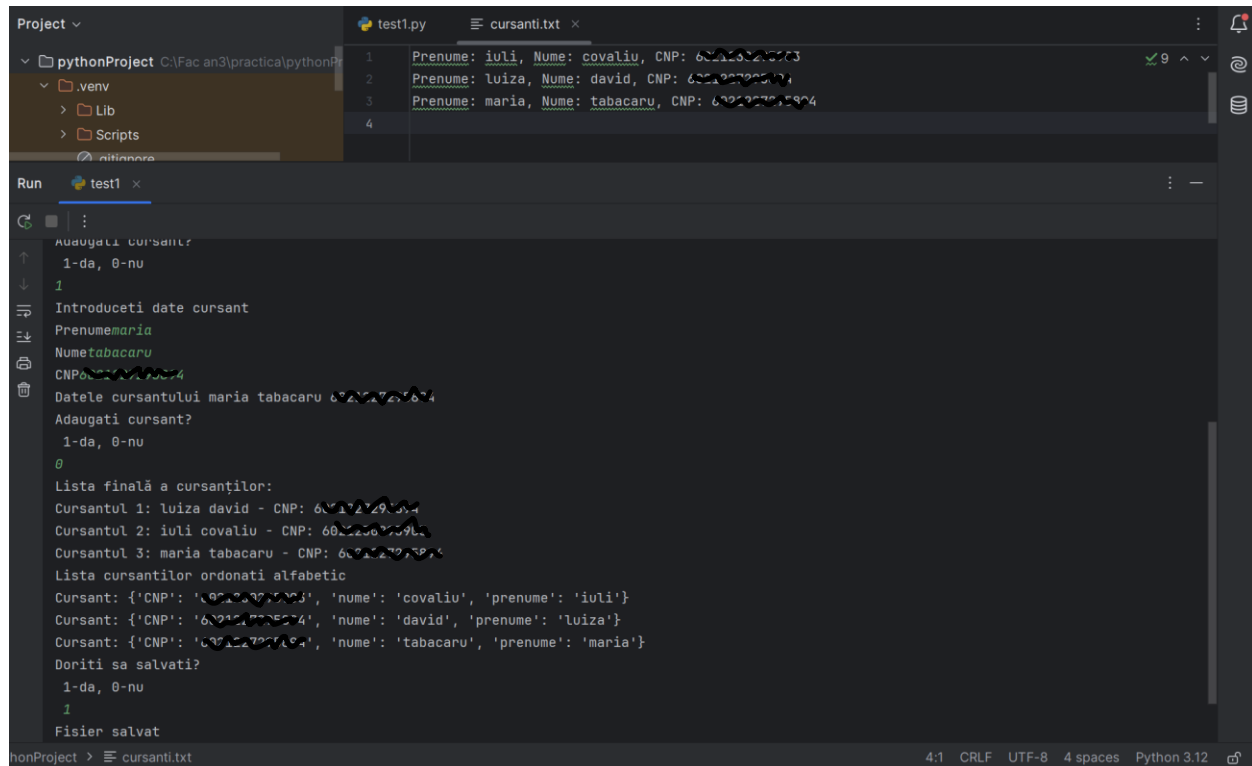
```
def validare_date(date):  
    for field in date:  
        if not field.strip():  
            print("Invalid: date incomplete")  
            return False  
        break  
    return True
```

Funcția **validare_date(date)** verifică dacă sunt toate cele 3 elemente introduse (nume, prenume, CNP) fără a exista intrari nule.

Pentru a putea stoca data cursanților am creat o listă numită **lista_cursanți** pe care am inițializat-o goală. Programul citește de la tastatură numele, prenumele și CNP-ul apoi fiind trecute prin procesele de validare. Dacă una dintre intrări nu este introdusă corect utilizatorul poate reintroduce datele respective de câte ori este necesar. De asemenea, este creat și un dicționar cu datele cursantului, pentru eventualele sortări. Utilizatorul are opțiunea de a adăuga cursanți noi cu ajutorul buclei **while i**. După fiecare introducere utilizatorul are de ales dintre a adăuga încă un cursant (1) sau nu (0), a doua opțiune oprind bucla de while.

Se listează apoi toți cursanții introduși, iar pentru o bună organizare a datelor se sortează cursanții alfabetic după nume. Cu ajutorul **lista_cursanti.sort** sortăm lista cursanților în mod alfabetic după nume, apoi vom sorta și dicționarul cu ajutorul listei mai sus menționată. Această implementare a fost aleasă pentru simplitate, față de o sortare după chei sau valori specifică dicționarilor. La final există alegerea de a salva sau nu datele introduse, salvarea făcându-se într-un fișier text cu numele “cursanti.txt”, unde datele vor fi salvate sortate.

3.2. Instanțe de rulare a programului



The screenshot shows an IDE with a project named 'pythonProject'. The file explorer on the left shows a directory structure with '.venv', 'Lib', and 'Scripts' folders. The main editor displays a file named 'cursanti.txt' with the following content:

```
1 Prenume: iuli, Nume: covaliu, CNP: 6021227294
2 Prenume: luiza, Nume: david, CNP: 6021227294
3 Prenume: maria, Nume: tabacaru, CNP: 6021227294
4
```

The 'Run' tab shows the execution of 'test1.py'. The output is as follows:

```
Adaugati cursant?
1-da, 0-nu
1
Introduceti date cursant
Prenume:maria
Nume:tabacaru
CNP:6021227294
Datele cursantului maria tabacaru 6021227294
Adaugati cursant?
1-da, 0-nu
0
Lista finală a cursanților:
Cursantul 1: luiza david - CNP: 6021227294
Cursantul 2: iuli covaliu - CNP: 6021227294
Cursantul 3: maria tabacaru - CNP: 6021227294
Lista cursantilor ordonati alfabetic
Cursant: {'CNP': '6021227294', 'nume': 'covaliu', 'prenume': 'iuli'}
Cursant: {'CNP': '6021227294', 'nume': 'david', 'prenume': 'luiza'}
Cursant: {'CNP': '6021227294', 'nume': 'tabacaru', 'prenume': 'maria'}
Doriti sa salvati?
1-da, 0-nu
1
Fisier salvat
```



The screenshot shows a terminal window with the following output:

```
Nume:covaliu8
invalid:cifra introdusa
Nume:covaliu#
Invalid: caracter special introdus
Nume:covaliu
CNP:602122729
Invalid: lungimea CNP-ului nu corespunde
CNP:602122dabs
Invalid: lungimea CNP-ului nu corespunde
CNP:gssks
Invalid: lungimea CNP-ului nu corespunde
CNP:6021227294
Datele cursantului iuli covaliu 6021227294
Adaugati cursant?
1-da, 0-nu
0
Lista finală a cursanților:
Cursantul 1: iuli covaliu - CNP: 6021227294
Lista cursantilor ordonati alfabetic
Cursant: {'CNP': '6021227294', 'nume': 'covaliu', 'prenume': 'iuli'}
Doriti sa salvati?
1-da, 0-nu
0
```

4. Îmbunătățiri

Pentru a îmbunătăți proiectul, posibilele modificări sunt:

- crearea unui meniu
- crearea unei interfețe
- stocarea datelor într-un fișier de tip CSV
- folosirea excepțiilor

Pentru partea de folosire a excepțiilor am atașat mai jos partea modificată din cod:

```
def validare_pre_nume(cuv):  
    x = 0  
    try:  
        for char in cuv:  
            if char.isdigit():  
                raise ValueError("Invalid: cifra introdusa")  
            if char in string.punctuation:  
                raise ValueError("Invalid: caracter special introdus")  
        return True  
    except ValueError as e:  
        print(e)  
        x += 1  
        return False  
  
def validare_cnp(cnp):  
    x = 0  
    try:  
        if len(cnp) != 13:  
            raise ValueError("Invalid: lungimea CNP-ului nu corespunde")  
        if not cnp.isdigit():  
            raise ValueError("Invalid: nu ati introdus cifre")  
        if cifra_control(cnp) != int(cnp[12]):  
            raise ValueError("Invalid: CNP invalid")  
        return True  
    except ValueError as e:  
        print(e)  
        x += 1  
        return False  
  
def validare_date(date):  
    try:  
        for field in date:  
            if not field.strip():  
                raise ValueError("Invalid: date incomplete")  
        return True  
    except ValueError as e:  
        print(e)  
        return False
```

5. Concluzii

Proiectul este un bun exercițiu pentru noii utilizatori de Python, fiind complex, trecând prin foarte multe concepte esențiale: listă, dicționar, funcții, prelucrarea datelor, stocarea datelor. În același timp este și un proiect destul de simplu, având mai puțin de 150 de linii de cod, făcându-l ușor de urmărit și înțeles.

Cele mai dificile părți ale proiectului au fost validarea CNP-ului fiind foarte multe condiții de îndeplinit, dar și ordonarea cursanților utilizând lista și dicționarul, acestea fiind tipuri avansate de date.

În concluzie, „Sistem de gestionare cursanți” este un proiect ideal pentru începători, oferind o bază solidă în programarea Python și acoperind aspecte esențiale ale dezvoltării de software. Acesta poate servi drept prim pas într-o călătorie lungă în învățarea și utilizarea Python.

6. Anexa: codul sursă

```
import string
def validare_pre_nume(cuv):
    x = 0
    for char in cuv:
        if char.isdigit():
            print("invalid:cifra introdusa")
            x = x + 1
            return False

        if char in string.punctuation:
            print("Invalid: caracter special introdus")
            x = x + 1
            return False

def validare_cnp(cnp):
    x = 0
    if len(cnp) != 13:
        print("Invalid: lungimea CNP-ului nu corespunde")
        return False
```

```
if cnp.isdigit() != 1:
    print("Invalid: nu ati introdus cifre")
    return '!'
    x = x + 1

if cifra_control(cnp) != int(cnp[12]):
    print("Invalid: CNP invalid")
    x = x + 1
return x == 0

def cifra_control(cnp):
    constanta = '279146358279'
    suma = 0

    for i in range(12):
        suma = suma + int(cnp[i]) * int(constanta[i])

    ctrl = suma % 11
    if ctrl == 10:
        ctrl = 1

    return ctrl

def validare_date(date):
    for field in date:
        if not field.strip():
            print("Invalid: date incomplete")
            return False
        break
    return True

lista_cursanti = []
i = 1
```



```
while i:

    print("Introduceti date cursant")
    prenume = input("Prenume")
    while (validare_pre_nume(prenume) == False ) :
        prenume = input("Prenume")

    nume = input("Nume")
    while (validare_pre_nume(nume) == False):
        nume = input("Nume")

    CNP = input("CNP")
    while (validare_cnp(CNP) == False ):
        CNP = input("CNP")

    date_cursant =[prenume, nume, CNP]
    while (validare_date(date_cursant) == False ):
        CNP = input("CNP")

    print("Datele cursantului", prenume, nume, CNP)

    cursant = {
        "prenume": prenume,
        "nume": nume,
        "CNP": CNP
    }

    lista_cursanti.append(cursant)

    i = int(input("Adaugati cursant? \n 1-da, 0-nu \n"))
    if i == 0:
        break

print("Lista finală a cursanților:")
for i, cursant in enumerate(lista_cursanti, start=1):
    print(f"Cursantul {i}: {cursant['prenume']} {cursant['nume']} - CNP: {cursant['CNP']}")
```

```
print("Lista cursantilor ordonati alfabetic")
lista_cursanti.sort(key=lambda cursant: cursant['nume'])
for i, cursant in enumerate(lista_cursanti, start=1):
    sorted_cursant = dict(sorted(cursant.items()))
    print(f"Cursant: {sorted_cursant}")

salvare = int(input("Doriti sa salvati? \n 1-da, 0-nu \n "))
if salvare:
    file = "cursanti.txt"

    with open(file, 'w') as f:
        for cursant in lista_cursanti:
            f.write(f"Prenume: {cursant['prenume']}, Nume: {cursant['nume']}, CNP: {cursant['CNP']}\n")

    print("Fisier salvat")
```