

# **Sistem de management a datelor unor cursanți**

*Proiect 1 Practică TelecomAcademy 2024*

Student: Covaliu Ioana Iuliana

Limbaj: Python

Nivel Dificultate: 2/5

Domeniu: programare

Tehnologii: -

## Descriere

Sistemul de management a datelor unor cursanti reprezinta o metoda inedită de familiarizare cu limbajul de programare Python, cuprinzând o sintaxă relativ simplă și o interfață prietenoasă cu utilizatorul.

Proiectul urmărește verificarea și validarea datelor cursanților TelecomAcademy și stocarea lor într-un fișier .txt după asigurarea corectitudinii acestora.

Pentru a realiza o interfață prietenoasă cu utilizatorii a fost implementat un meniu cu 4 opțiuni : 1) Adauga cursant, 2) Afiseaza lista cursantilor, 3) Salvează cursanții în fișier , insotite de cazul 0) Iesire. Aceasta abordare facilitează interacțiunea utilizatorului cu programul.

Funcționalitatea proiectului constă în verificarea acurateții datelor introduse de utilizator, astfel datele introduse trecând prin mai multe etape de verificare în funcție de tipul lor după cum urmează:

- ✚ Prima etapă de verificare se realizează în câmpurile de "Prenume" și "Nume" și urmărește ca numele și prenumele utilizatorului să conțină exclusiv litere(fara cifre sau caractere speciale) . În cazul în care acest lucru nu este îndeplinit, în consolă se va afișa mesajul "Invalid: cifra /caracter special introdus", iar utilizatorul va putea sa reintroduca datele.
- ✚ A doua etapa de verificare este reprezentata de valabilitatea CNP-ului, iar pentru aceasta s-au preluat criteriile de verificare a codului numeric personal de la nivelul statului român.

Dacă toate aceste criterii sunt îndeplinite, utilizatorul are posibilitatea de a introduce alte seturi de date pentru alți cursanți, iar mai apoi de a le salva și stoca într-un fișier .txt denumit cursanți.txt.

## Resurse:

Python versiune > 3.8

PyCharm versiuni 2023/2024

Cunostinte minime(in Python) : Liste, String-uri, Structuri de control, Functii

Curs Python Entry Level Telacad : [https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022\\_T1/courseware/c82480aa92484b01be70b4003f6e0035/61498918c7ab45b3ae4def2810394dd9/?child=first](https://cursuri.telacad.ro/courses/course-v1:TelecomAcademy+IntroducereinPython+2022_T1/courseware/c82480aa92484b01be70b4003f6e0035/61498918c7ab45b3ae4def2810394dd9/?child=first)

## Implementare:

Programul folosește o listă pentru stocarea datelor cursanților și diferite funcții pentru implementarea funcționalităților mai sus descrise.

Funcții de validare:

‘validare\_pre\_nume’

```
def validare_pre_nume(cuv):
    for char in cuv:
        if char.isdigit():
            print("Invalid: cifră introdusă")
            return False
        if char in string.punctuation:
            print("Invalid: caracter special introdus")
            return False
    return True
```

Funcția `validare_pre_nume` verifică dacă unul dintre prenume este compus doar din litere și folosește două secvențe `if` pentru detectarea cifrelor sau caracterelor speciale. Dacă în cuvânt există astfel de caractere, funcția va returna `False`, iar în consolă se va afișa mesajul de eroare “Invalid: cifra introdusă” sau “Invalid: caracter special introdus”.

‘validare\_cnp’ și ‘cifra\_control’

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

    if not cnp.isdigit():
        print("Invalid: nu ați introdus cifre")
        return False

    if cifra_control(cnp) != int(cnp[12]):
        print("Invalid: CNP invalid")
        return False

    return True
```

Cele două funcții compun etapa de verificare a CNP-ului cursantului înscris. `validare_cnp` verifică în primă instanță CNP-ul introdus să aibă lungimea standard de 13 cifre, urmat mai apoi de verificarea ca toate elementele introduse sunt cifre, iar în caz contrar se va afișa mesajul de eroare predefinit.

În cadrul aceleiași funcții, odată ce criteriile preliminare pentru CNP sunt îndeplinite, programul va calcula cifra de control și o va compara cu ultima cifră a CNP-ului introdus.

‘cifra\_control’ calculează cifra de control, denumită în cadrul funcției ‘ctrl’ și o compară cu ultima cifră a valorii primite de la tastatură. Preia constanta

'279146358279' , iar fiecare cifră din primele 12 cifre din C.N.P. este înmulțită cu corespondentul său din constantă

- rezultatele sunt însumate și totalul se împarte la 11
- dacă restul împărțirii este mai mic de 10, acela reprezintă valoarea variabilei ctrl
- dacă restul împărțirii este 10, valoarea lui ctrl este 1

```
- def cifra_control(cnp):  
    constanta = '279146358279'  
    suma = 0  
  
    for i in range(12):  
        suma += int(cnp[i]) * int(constanta[i])  
  
    ctrl = suma % 11  
    if ctrl == 10:  
        ctrl = 1  
  
    return ctrl
```

Funcția 'validare\_date' verifică dacă toate câmpurile din listă sunt completate, dar fără spații. Pentru asta se folosește funcția specifică python `field.strip()` care va afișa mesajul de eroare în caz de neconcordanțe.

Pentru stocarea datelor cursanților, programul utilizează o listă inițializată:

```
lista_cursanti = []
```

Meniul cuprinde funcționalitățile principale ale proiectului : adăugarea unui nou set de date , afișarea celor existente într-o listă, stocarea datelor introduse în fișier și opțiunea de a alege ce acțiune se va efectua.

```
while True:  
    print("\nMeniu Principal")  
    print("1. Adaugă cursant")  
    print("2. Afișează lista cursanților")  
    print("3. Salvează cursanții în fișier")  
    print("0. Ieșire")  
  
    optiune = input("Alegeți o opțiune: ")
```

Adăugarea unui cursant:

```
if optiune == '1':  
    # Adaugă cursant  
    print("Introduceți datele cursantului:")  
  
    # Colectare și validare prenume  
    prenume = input("Prenume: ")  
    while not validare_pre_nume(prenume):  
        prenume = input("Prenume: ")
```

```

# Colectare și validare nume
nume = input("Nume: ")
while not validare_pre_nume(nume):
    nume = input("Nume: ")

# Colectare și validare CNP
CNP = input("CNP: ")
while not validare_cnp(CNP):
    CNP = input("CNP: ")

# Crearea cursantului și validarea datelor
cursant = {
    "prenume": prenume,
    "nume": nume,
    "CNP": CNP
}

if not validare_date([prenume, nume, CNP]):
    continue

lista_cursanti.append(cursant)
print(f"Cursantul {prenume} {nume} a fost adăugat cu succes.")

```

Această opțiune apelează funcțiile de validare pentru cele 3 elemente, iar în cazul în care validarea nu este efectuată corect, utilizatorului i se permite să reintroducă datele neconforme, corectate prin secvențele de tipul: `while not validare_cnp(CNP)`

‘cursant’ este un dicționar care conține datele unui cursant: prenumele, numele și CNP-ul, colectate și validate anterior în cod.

```

# Crearea cursantului și validarea datelor
cursant = {
    "prenume": prenume,
    "nume": nume,
    "CNP": CNP
}

if not validare_date([prenume, nume, CNP]):
    continue

```

Funcția `lista_cursanti.append(cursant)` adaugă dicționarul ‘cursant’ la lista ‘lista\_cursanti’, iar folosirea funcției `append` adaugă dicționarul ‘cursant’ la sfârșitul listei

Afișarea listei cursanților:

```

elif optiune == '2':
    # Afișează lista cursanților
    if not lista_cursanti:
        print("Nu există cursanți în listă.")
    else:
        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']}")

```

Acest bloc afișează fiecare dicționar ca un element al listei.

## Salvarea cursantilor in fisier

```
elif optiune == '3':
    # Salvează cursanții în fișier
    nume_fisier = "cursanti.txt"
    with open(nume_fisier, 'w') as f:
        for cursant in lista_cursanti:
            f.write(f"Prenume: {cursant['prenume']}, Nume: {cursant['nume']}, CNP: {cursant['CNP']}\n")
    print(f"Datele cursanților au fost salvate în fișierul {nume_fisier}.")
```

Linia `with open(nume_fisier, 'w') as f:` deschide un fișier pentru scriere, iar `with` se asigură că fișierul va fi corect închis după ce blocul de cod este executat.

Dacă fișierul nu există, acesta va fi creat. Dacă fișierul există deja, conținutul său va fi înlocuit cu lista cursanților (suprascrisere).

“nume\_fisier” este definită anterior ca “cursanti.txt”

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

Acest `for` parcurge fiecare dicționar “cursant” din listă, iar `f.write` scrie informațiile despre fiecare cursant în fișierul deschis.

`\n` la sfârșitul șirului adaugă un caracter de linie nouă, asigurându-se că fiecare cursant este scris pe o linie nouă în fișier.

Ieșirea din program este realizată de blocul decizional, iar ultimul `else` gestionează opțiunile invalide.

```
elif optiune == '0':
    # Ieșire
    print("La revedere!")
    break

else:
    print("Opțiune invalidă. Încercați din nou.")
```

## Exemple de rulare a programului



## Meniu Principal

1. Adaugă cursant
2. Afișează lista cursanților
3. Salvează cursanții în fișier
0. Ieșire

Alegeți o opțiune: 1

Introduceți datele cursantului:

Prenume: maria

Nume: tabacaru

CNP: 667230905

Cursantul maria tabacaru a fost adăugat cu succes.

## Meniu Principal

1. Adaugă cursant
2. Afișează lista cursanților
3. Salvează cursanții în fișier
0. Ieșire

Alegeți o opțiune: 1

Introduceți datele cursantului:

Prenume: iuliana

Nume: covaliu

CNP: 667230905

Cursantul iuliana covaliu a fost adăugat cu succes.

Meniu Principal

1. Adaugă cursant
2. Afișează lista cursanților
3. Salvează cursanții în fișier
0. Ieșire

Alegeți o opțiune: 2

Lista finală a cursanților:

Cursantul 1: maria tabacaru - CNP: 601 000 000 3

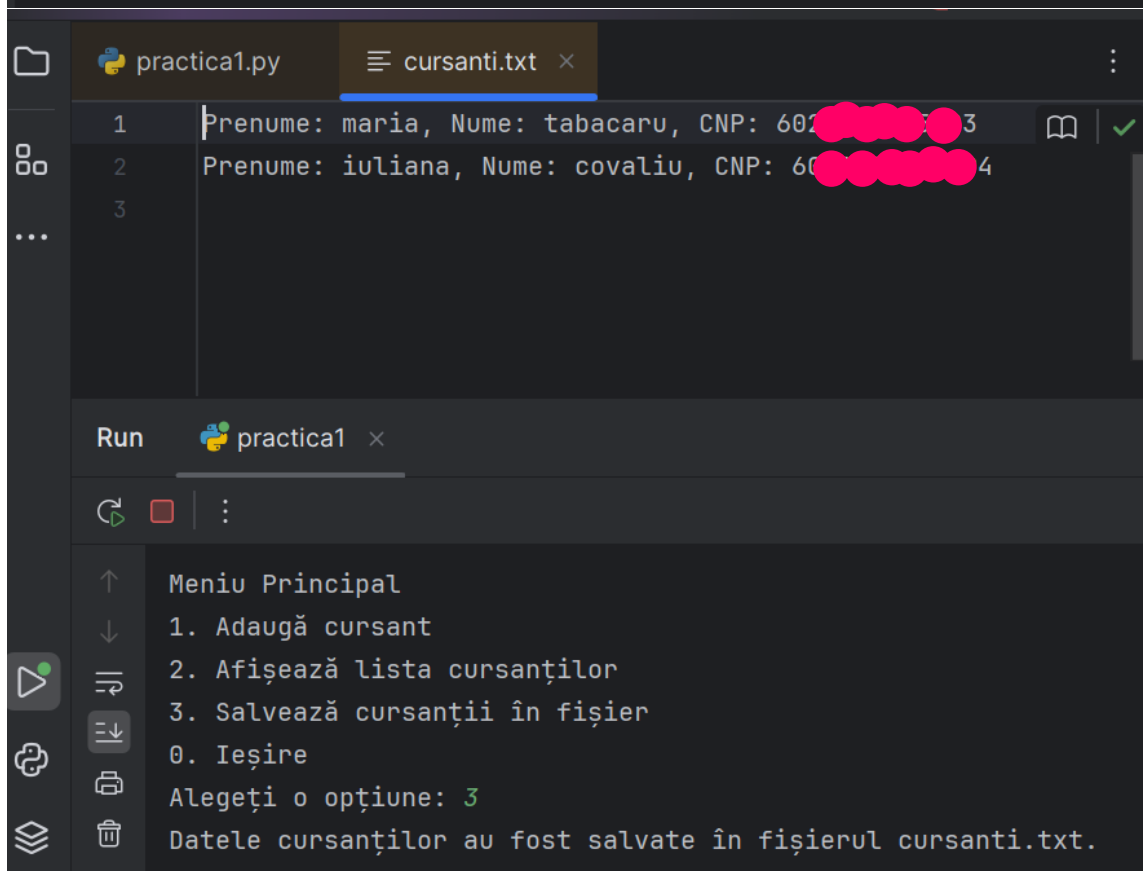
Cursantul 2: iuliana covaliu - CNP: 601 000 000 4

Meniu Principal

1. Adaugă cursant
2. Afișează lista cursanților
3. Salvează cursanții în fișier
0. Ieșire

Alegeți o opțiune: 3

Datele cursanților au fost salvate în fișierul cursanti.txt.



Verificarea cazurilor neconforme



```
Meniu Principal
1. Adaugă cursant
2. Afișează lista cursanților
3. Salvează cursanții în fișier
0. Ieșire
Alegeți o opțiune: 1
Introduceți datele cursantului:
Prenume: elena
Nume: vale5a
Invalid: cifră introdusă
Nume: valeria
CNP: 6045
Invalid: lungimea CNP-ului nu corespunde
CNP: 6046380246790
Invalid: CNP invalid
CNP: 6021227845690
Invalid: CNP invalid
CNP:
```

## Imbunatatiri: Utilizarea exceptiilor

```
import string

class PrenumeInvalid(Exception):
    pass

class CNPInvalid(Exception):
    pass

class DateIncomplete(Exception):
    pass

# Verificare prenume fără cifre
def validare_pre_nume(cuv):
    for char in cuv:
        if char.isdigit():
            raise PrenumeInvalid("Prenumele conține cifre.")
        if char in string.punctuation:
            raise PrenumeInvalid("Prenumele conține caractere speciale.")
    return True

# Verificare CNP
def validare_cnp(cnp):
```

```

if len(cnp) != 13:
    raise CNPInvalid("Lungimea CNP-ului nu corespunde.")

if not cnp.isdigit():
    raise CNPInvalid("CNP-ul trebuie să conțină doar cifre.")

if cifra_control(cnp) != int(cnp[12]):
    raise CNPInvalid("CNP-ul este invalid.")

return True

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

    for i in range(12):
        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(): # Verifică dacă field este gol sau conține spații
            raise DateIncomplete("Date incomplete.")
    return True

```

## Imbunatatiri posibile:

- Interfata grafica

- Clasificarea/sortarea cursantilor dupa diferite criterii

## Concluzii

Proiectul este o aplicație simplă, dar eficientă, dedicată gestionării cursanților ce ofera funcționalități esențiale pentru validarea prenumelor, numelor și CNP-urilor pentru a asigura introducerea corectă a datelor. Salvarea cursanților într-un fișier text permite stocarea informațiilor între sesiuni de utilizare, într-un format ușor de interpretat și de gestionat.

Gestionarea robustă a erorilor prin mesaje clare și prompte de re-introducere a datelor invalide contribuie la o experiență utilizator îmbunătățită, asigurând integritatea și acuratețea datelor stocate.

Toate acestea reprezinta o aplicatie ideala de implementat pentru un nivel de programare incepator intrucat aplica notiuni standard intr-un proiect cu functionalitate reala.

## Observatii

Cele mai dificile parti ale proiectului au fost reprezentate de functiile de validare ale CNP-ului, intampinandu-se dificultati in calculul cifrei de control, dar si in configurarea meniului.

# ANEXA

```
import string

# Verificare prenume fără cifre
def validare_pre_nume(cuv):
    for char in cuv:
        if char.isdigit():
            print("Invalid: cifră introdusă")
            return False
        if char in string.punctuation:
            print("Invalid: caracter special introdus")
            return False
    return True

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

    if not cnp.isdigit():
        print("Invalid: nu ați introdus cifre")
        return False

    if cifra_control(cnp) != int(cnp[12]):
        print("Invalid: CNP invalid")
        return False

    return True

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

    for i in range(12):
        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(): # Verifică dacă field este gol sau conține spații
            print("Invalid: date incomplete")
            return False
    return True

# Lista pentru a stoca toți cursanții
lista_cursanti = []

# Meniu principal
while True:
    print("\nMeniu Principal")
    print("1. Adaugă cursant")
    print("2. Afișează lista cursanților")
    print("3. Salvează cursanții în fișier")
    print("0. Ieșire")

    optiune = input("Alegeți o opțiune: ")
```

```

if optiune == '1':
    # Adaugă cursant
    print("Introduceți datele cursantului:")

    # Colectare și validare prenume
    prenume = input("Prenume: ")
    while not validare_pre_nume(prenume):
        prenume = input("Prenume: ")

    # Colectare și validare nume
    nume = input("Nume: ")
    while not validare_pre_nume(nume):
        nume = input("Nume: ")

    # Colectare și validare CNP
    CNP = input("CNP: ")
    while not validare_cnp(CNP):
        CNP = input("CNP: ")

    # Crearea cursantului și validarea datelor
    cursant = {
        "prenume": prenume,
        "nume": nume,
        "CNP": CNP
    }

    if not validare_date([prenume, nume, CNP]):
        continue

    lista_cursanti.append(cursant)
    print(f"Cursantul {prenume} {nume} a fost adăugat cu succes.")

elif optiune == '2':
    # Afișează lista cursanților
    if not lista_cursanti:
        print("Nu există cursanți în listă.")
    else:
        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']}")

elif optiune == '3':
    # Salvează cursanții în fișier
    nume_fisier = "cursanti.txt"
    with open(nume_fisier, 'w') as f:
        for cursant in lista_cursanti:
            f.write(f"Prenume: {cursant['prenume']}, Nume:
{cursant['nume']}, CNP: {cursant['CNP']}\n")
        print(f"Datele cursanților au fost salvate în fișierul
{nume_fisier}.")

elif optiune == '0':
    # Ieșire
    print("La revedere!")
    break

else:
    print("Opțiune invalidă. Încercați din nou.")

```

