Validare date personale

Stan Rareș-Gabriel

Limbaj : Python
Nivel dificultate : 2/5
Domeniu : Software
Tehnologii (Aplicații utilizate) : Pycharm, Excel

Cuprins

Descriere

Aplicația primește date personale despre un utilizator și stabilește pe baza unor reguli dacă acestea sunt valide sau nu. Odată validate, datele sunt introduse într-un fișier CSV. În caz contrar, se va afișa în consolă un mesaj de eroare specific ținând cont de ce reguli au fost încălcate.

Resurse

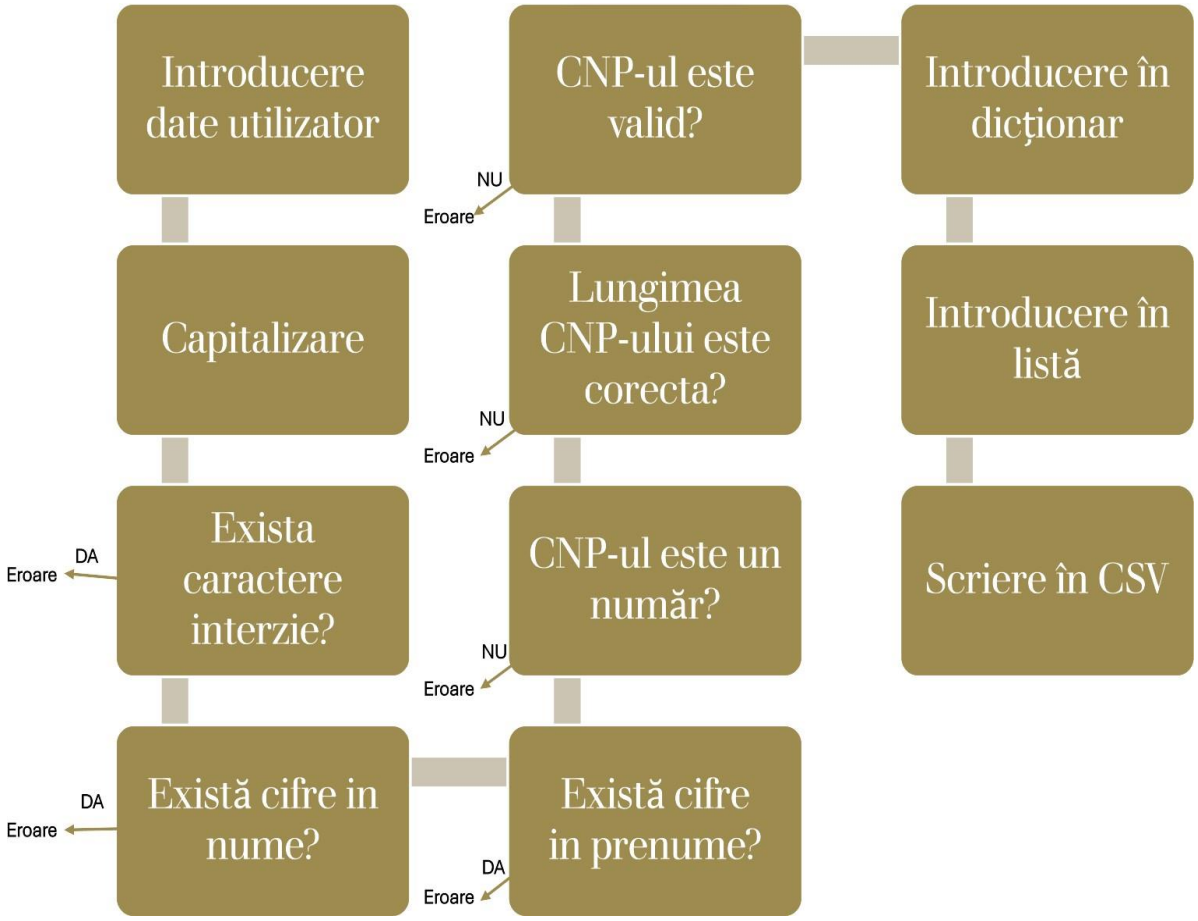
Aplicația a fost realizată cu ajutorul Python, un limbaj de programare de nivel înalt, interpretat și de uz general. Python este renumit pentru sintaxa sa clară și concisă, care favorizează lizibilitatea și ușurința în scrierea codului. S-au folosit o varietate de funcții specifice de tip Built-In pentru o înțelegere cât mai ușoară a programului, cât și pentru scurtarea propriu zisă a. lungimii acestuia. Funțiile menționate anterior sunt:

- **input -** Citește o linie de la tastatura și returnează ca un șir de caractere;
- **split -** Împarte un șir de caractere pe baza unui separator specificat. Dacă nu se specifică un separator, se folosește implicit spațiul;
- **append -** Adaugă un element la sfârșitul unei liste;
- **isdigit -** Verifică dacă toate caracterele dintr-un șir de caractere sunt cifre.
- **len -** Returnează lungimea (numărul de elemente) unui obiect. Poate fi folosit pe liste sau pe șiruri de caractere;
- **range -** Generează o secvență de numere între o valoare de început și o valoare de sfârșit specificată. Util pentru a itera printr-o secvență de numere;
- **sum -** Calculează suma elementelor unui obiect iterabil;
- **open -** Deschide un fișier și returnează un obiect fișier. Permite specificarea modului de deschidere;
- **csv.writer -** Creează un obiect de scriere CSV care convertește datele în format CSV și le scrie într-un fișier deschis;
- **capitalize -** Returnează o copie a șirului cu primul caracter transformat în literă mare și restul caracterelor în litere mici;
- **isnumeric** - Verifică dacă un obiect este de tip numeric sau nu

Pentru manipularea fișierelor de tip CSV s-a important biblioteca specială ‘csv’.

Implementare

Flowchart:



Programul se bazează în principal pe funția de validare a cărui scop este de a trece datele citite de la tastatură prin setul de reguli stabilite. Funcția are 2 argumente: datele introduse de către un cursant și dicționarul pe care trebuie să-l returneze dacă datele respective au fost validate cu succes.

Începem prin a declara o variabilă numită ‘ok’. Aceasta va fi inițializată cu 0 și valoarea acesteia se va schimba în 1 dacă oricare din regulile de validare nu este respectată. La finalul funcției, datele vor fi introduse în dicționar și ulterior în listă doar dacă valoarea lui ‘ok’ nu s-a modificat din 0, adică datele testate de funcție sunt valide.

Prima regulă de verificare este cea a caracterelor interzise. Variabila ‘caractere_interzise’ stochează un șir de caractere ce conține caracterele specificate. Toate caracterele din elementele listei de date primite ca argument sunt verificate să nu aparțină șirului de caractere ‘caractere_interzise’. În cazul în care se găsește un astfel de caracter, se afișează mesajul de eroare specific și se iese din bucla de verificare.

A doua și a treia regulă verifică daca numele și prenumele conțin cifre, caz în care se va afișa mesajul de eroare și se va ieși din buclă. Regula 2 verifică primul element din lista de date, deoarece se știe că acesta este numele cursantului, iar regula 3 verifică toate elementele din listă înafară de primul și ultimul (nume, CNP), deoarece nu se știe dacă persoana respectivă are unsinur prenume sau mai multe.

Ultimele 3 reguli de validare se bazează pe structura CNP-ului și formează un bloc de tip if-else. Dacă o regulă de validare afișează mesajul de eroare automat nu se mai verifică și restul regulilor. Regula 4 verifică dacă CNP-ul este de tip numeric, în caz contrar se afișează eroarea. Dacă regula 4 este îndeplinită, se verifică mai departe dacă acesta are 13 cifre. Această verficare se face în cadrul regulii 5. Regula 6 verifică daca ultima cifră a CNP-ului este validă cu ajutorul unui algoritm. În variabila ‘verif’ se stochează un șir de 12 cifre. Acestea se înmulțesc pe rând cu primele 12 cifre ale CNP-ului și rezultatele înmulțirilor sunt adunate. Restul împărțirii acestei sume la 11 trebuie să fie ultima cifră a CNP-ului valid. Dacă această condiție nu este îndeplinită CNP-ul nu este valid și se afișează mesajul de eroare specific.

Asa cum am menționat și în primul paragraf, dacă se trece prin toate regulile de validare fără eroare valoarea variabilei ‘ok’ rămâne 0 iar datele pot fi stocate mai departe într-un dicționar. Acest dicționar va avea 4 elemente.

Mai departe ne dorim împărțirea șirului de caractere introdus de la tastatură în elementele dorite pentru a forma lista de date folosită în funcția de validare.

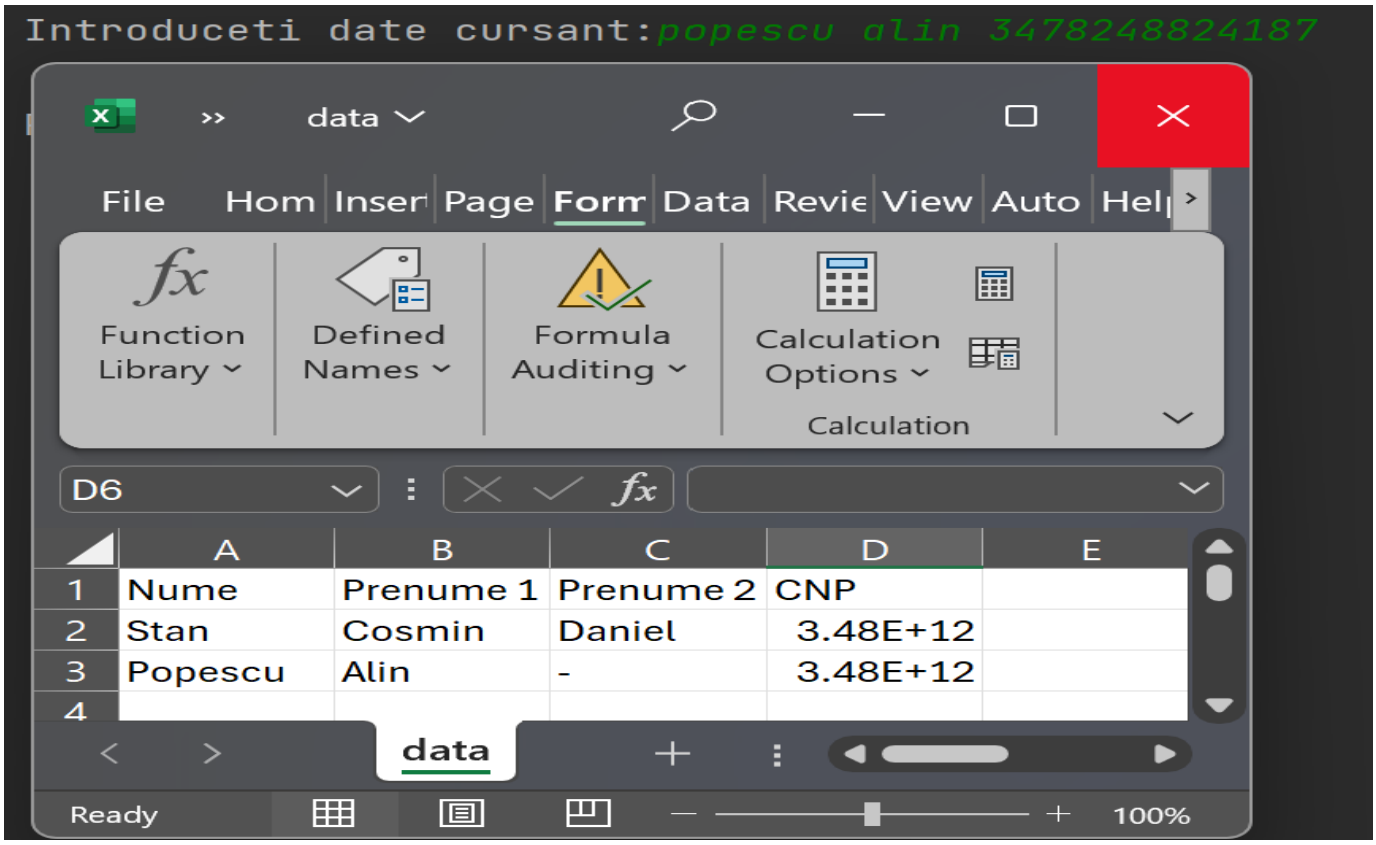
Șirul de caractere introdus de la tastatură este stocat în variabila ‘cursant’ care este impărțit în funție de caracterul ‘ ‘ într-o listă de cuvinte stocată la rândul ei în variabila ‘date_prime’. Daca valoarea cardinalului listei ‘date_prime’ este 3, există 2 posibilități: cursantul are un singur prenume sau 2 prenume unite prin caracterul ‘-’. Daca prenumele conține caracterul ‘-’, îl vom despărți mai departe în două. Odată ce toate datele sunt împărțite corect, le vom introduce în lista finală numită ‘date’.

Mai departe verificăm dacă numele și prenumele sunt scrise cu literă mare. În caz contrar le vom modifica înainte de a folosi lista de date în funcția de validare.

Se folosește funcția ‘validare’ pentru a trece datele prin setul de reguli, acestea urmând a fi sau nu introduse în fișierul de tip CSV.

Exemple de rulare

Date valide:



Date eronate:

```
Introduceti date cursant:s4an c0smin-daniel 2341673477621347
Numele nu este valid
Prenumele nu este valid
Lungimea CNP-ului este incorecta


Process finished with exit code 0
```

Îmbunătățiri

- Aplicația poate fi modificată pentru a permite stocarea datelor pentru persoane cu 3 prenume
- Lista de caractere interzise este relativ mică, poate fi modificată foarte ușor în funție de cerințe
- Tipul fișierului de stocare a datelor poate fi modificat în funcție de preferințe

Concluzii

- Aplicația în principiu nu va fi funcțională pentru persoane cu 3 prenume, deoarece dicționarele sunt făcute doar pentru maxim 2 prenume
- Este concepută pentru a corecta greșeli de scriere, nu pentru a recunoaste date false
- Există posibilitatea introducerii unui CNP inexistent care să fie trecut ca CNP valid în urma algoritmului de verificare a ultimei cifre
- Este o aplicație ușoară în teorie, dar care poate cauza probleme la implementare; este nevoie de o atenție sporită în folosirea listelor și iterarea prin șirurile de caractere
- Scrierea cu literă mică a numelui sau prenumelui nu este văzută ca o greșeală și este corectată automat de program

Anexă

```
import csv
#1
def caractere_interzise(c_i, date):
    for cuvant in date:
        for j in cuvant:
            if j in c_i:
                return 1
    return 0
#2
def validare_nume(nume):
    for c in nume:
        if c.isdigit():
            return 1
    return 0
#3
def validare_prenume(date):
    for cuvant in date:
        for c in cuvant:
                return 1
    return 0
#4
def cnp_numar(cnp):
    if not cnp.isnumeric():
        return 1
    return 0
#5
def lunigme_cnp(cnp):
    if len(cnp) != 13:
        return 1
    return 0
#6
def validare_ultima_cifra_cnp(verif, cnp):
    alg = []
    for nr in range(12):
        alg.append(int(cnp[nr]) * int(verif[nr]))
    suma = sum(alg)
    if suma % 11 != int(cnp[12]):
        return 1
    return 0
```

```
def validare(dict, date):
    ok = 0
    if caractere_interzise('+@!?\/', date) == 1:
        print("Datele contin caractere interzise")
        ok = 1
    if validare_nume(date[0]) == 1:
        print("Numele nu este valid")
        ok = 1
    if validare_prenume(date[1:-1]) == 1:
        print("Prenumele nu este valid")
        ok = 1
    if cnp_numar(date[-1]) == 1:
        print("CNP-ul nu este un numar")
        ok = 1
    elif lunigme_cnp(date[-1]) == 1:
        print("Lungimea CNP-ului este incorecta")
        ok = 1
    elif validare_ultima_cifra_cnp('279146358279', date[-1]) == 1:
        print("CNP-ul nu este valid")
        ok = 1
    if ok == 0:
        if len(date) == 3:
            dict.update({"Nume ": date[0],
                         "Prenume 1 ": date[1],
                         "Prenume 2": "-",
                         "CNP ": date[2]})
        elif len(date) == 4:
            dict.update({"Nume": date[0],
                         "Prenume 1": date[1],
                         "Prenume 2": date[2],
                         "CNP": date[3]})

def impartire_prenume(prenume, p):
    if '-' in prenume:
        impartire = prenume.split('-')
        for cuv in impartire:
            p.append(cuv)
    else:
        p.append(prenume)

def impartire_date(cursant, date):
```

```
    date_prime = cursant.split(' ')
    if len(date_prime) == 3:
        date.append(date_prime[0])
        impartire_prenume(date_prime[1], date)
        date.append(date_prime[-1])
    else:
        for cuv in date_prime:
            date.append(cuv)

dict = {}
date = []
cursant = input("Introduceti date cursant:")
impartire_date(cursant, date)
for i in range(len(date)-1):
    if date[i][0]!= date[i][0].upper():
        date[i]= date[i].capitalize()
validare(dict, date)
nume_col=dict.keys()
if dict:
    with open('data.csv', 'a', newline='') as file:
        w = csv.DictWriter(file, fieldnames=nume_col)
        if file.tell() == 0:
            w.writeheader()
        w.writerow(dict)
```