**MINISTERU EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLŢI**

**FACULTATEA DE ŞTIINŢE REALE, ECONOMICE ȘI ALE MEDIULUI**

**CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ**

**INTRODUCERE ÎN LIMBAJUL DE PROGRAMARE PYTHON**

**REFERAT LA CURSUL INFORMATICA GENERALĂ**

**Autor:**

Studenta grupei IS11Z

**Corina CERNOLEV**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conducător știintific:**

**Olesea SKUTNIȚKI**

magistru, asist. univ.

\_\_\_\_\_\_\_\_\_\_\_\_\_

**BĂLȚI, 2022**

CUPRINS

[INTRODUCERE 3](#_Toc102749036)

[1. LUCRU CU INTERPRETATORUL PYTHON 4](#_Toc102749037)

[2. SINTAXA LIMBAJULUI PYTHON 5](#_Toc102749038)

[2.1. Lucru cu variabilele în Python 5](#_Toc102749039)

[2.2. Tipurile de operatori ai limbajului 5](#_Toc102749040)

[2.3. Tipuri de date în limbajul Python 6](#_Toc102749041)

[3. TIPURI DE INSTRUCȚIUNI ÎN PYTHON 9](#_Toc102749042)

[4. DECLARAREA FUNCȚIILOR ÎN PYTHON 10](#_Toc102749043)

[5. AVANTAJELE LIMBAJULUI PYTHON 11](#_Toc102749044)

[CONCLUZII 13](#_Toc102749045)

[BIBLIOGRAFIE 14](#_Toc102749046)

INTRODUCERE

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 limbajului 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.

Implementarea de referință a Python este scrisă în C și poartă deci numele de CPython. Această implementare este software liber și este administrată de fundația Python Software Foundation.

Python este un limbaj de programare dinamic, de nivel înalt, ce pune accent pe expresivitatea și înțelegerea ușoară a codului. Sintaxa sa permite implementări echivalente cu alte limbaje în mai puține linii de cod. Datorită acestui fapt, Python este foarte răspândit atât în programarea de aplicații, cât și în zona de scripting.

Limbajul facilitează mai multe paradigme de programare, paradigma imperativa (C), orientată pe obiecte (Java) și funcțională. Spre deosebire de C, Python nu este un limbaj compilat, ci interpretat. Acest fapt are atât avantaje, cât și dezavantaje. Pe de-o parte, Python este mai lent decât C. Pe de altă parte, aplicațiile Python sunt foarte ușor de depănat, codul putând fi ușor inspectat în timpul rulării. De asemenea, este foarte ușor de experimentat cu mici fragmente de cod folosind interpretorul Python.

Sintaxa este gândită în așa fel încât programele Python să fie ușor de citit. Acest lucru este obținut prin folosirea de cuvinte în locul semnelor (de exemplu, and în loc de &&) și prin includerea indentării în limbaj. Astfel, în Python nu se folosesc acolade (ca în C/C++, Java), ci blocurile de cod se delimitează prin indentare. Programele Python sunt, de multe ori, foarte aproape de o „implementare” echivalentă în pseudocod.

1. LUCRU CU INTERPRETATORUL PYTHON

Limbajul Python este interpretat, nu compilat. Asta înseamnă că programele Python sunt transformate într-un limbaj intermediar. Acest lucru permite codului să fie ușor de portat pe diverse sisteme de operare și arhitecturi hardware.

Codul este executat linie cu linie. Astfel, dacă, de exemplu, se apelează o funcție care nu există, apare un mesaj de eroare abia când se încearcă executarea liniei respective. Erorile de sintaxă sunt raportate însă înainte de rularea programului.

Pentru a exersa sintaxa și elementele de bază ale limbajului, vom folosi direct interpretorul Python.

Interpretorul seamănă foarte mult cu o linie de comanda (cmd.exe, bash). Prompt-ul (>>>) apare când se așteaptă input din partea utilizatorului. După ce este introdusă o instrucțiune, aceasta este evaluată și dacă există output, este afișat în continuare, începând cu o linie nouă. Expresiile sunt evaluate și afișate (Fig. 1).

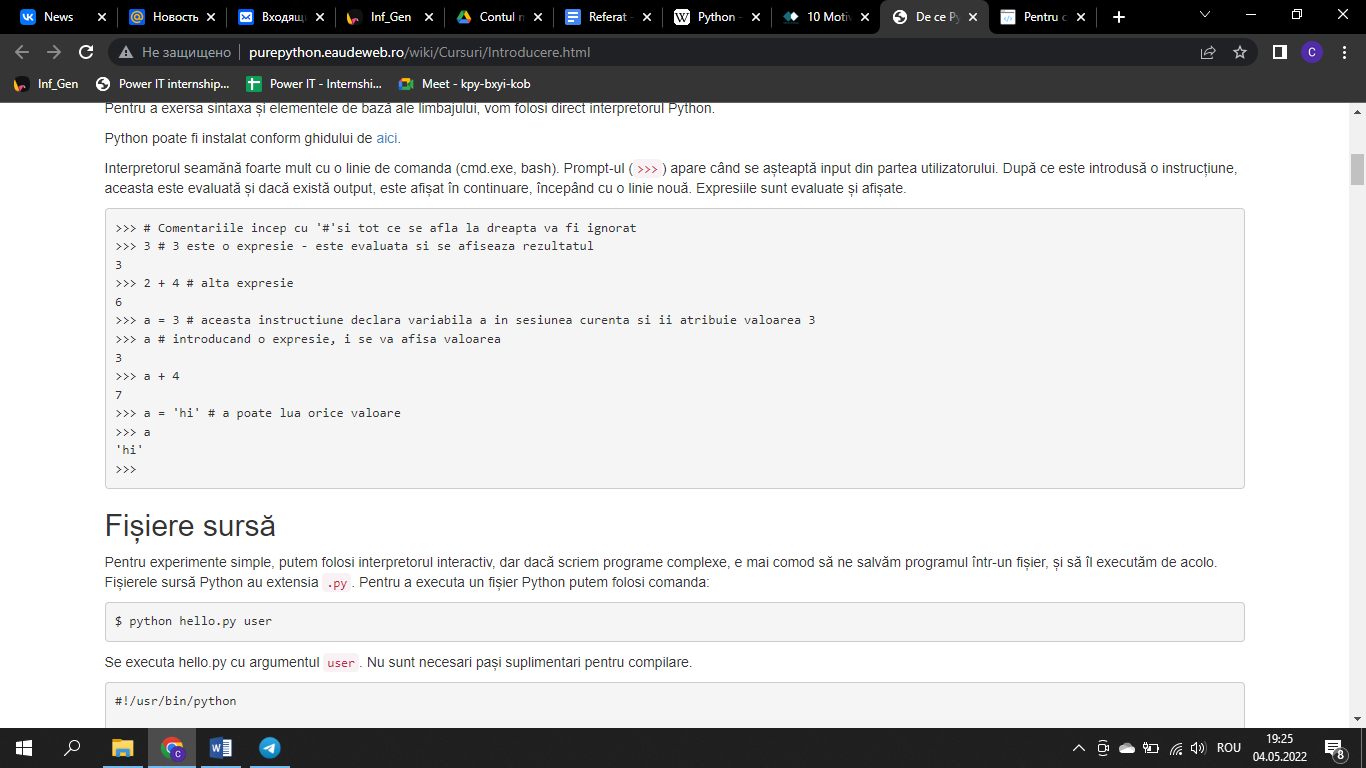


Fig. 1. Interpretatorul Python.

1. SINTAXA LIMBAJULUI PYTHON

# Lucru cu variabilele în Python

În Python nu este necesară folosirea operatorului „ ; ” la sfârșitul unei instrucțiuni. Se folosește o linie pentru fiecare instrucțiune. În cazul în care se dorește scrierea mai multor instrucțiuni pe aceeași linie, se poate folosi operatorul „ ; ” (Fig. 2).

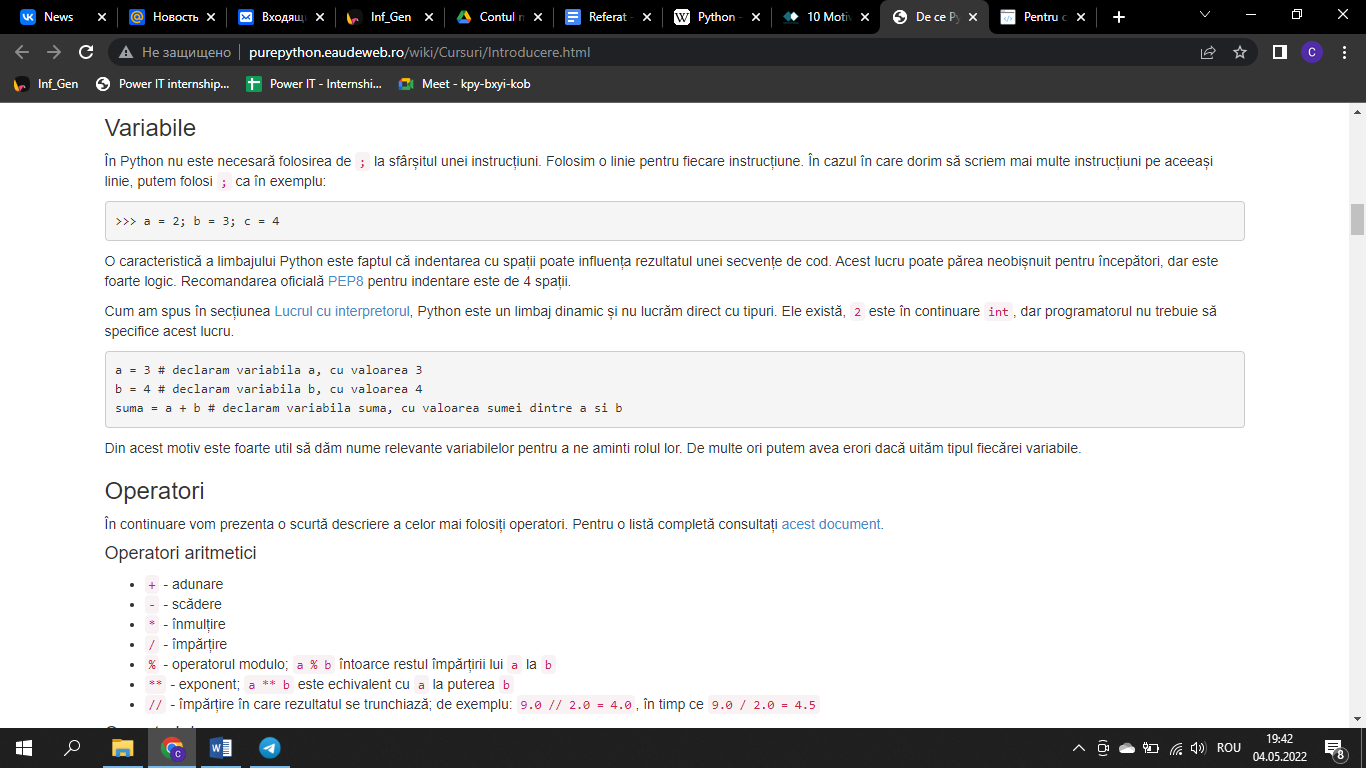


Fig.2. Instrucțiuni pe aceeași linie.

O caracteristică a limbajului Python este faptul că indentarea cu spații poate influența rezultatul unei secvențe de cod. Acest lucru poate părea neobișnuit pentru începători, dar este foarte logic. Recomandarea oficială pentru indentare este de 4 spații.

Python este un limbaj dinamic și nu lucrează direct cu tipuri. Ele există, dar programatorul nu trebuie să specifice acest lucru. (Fig. 3)

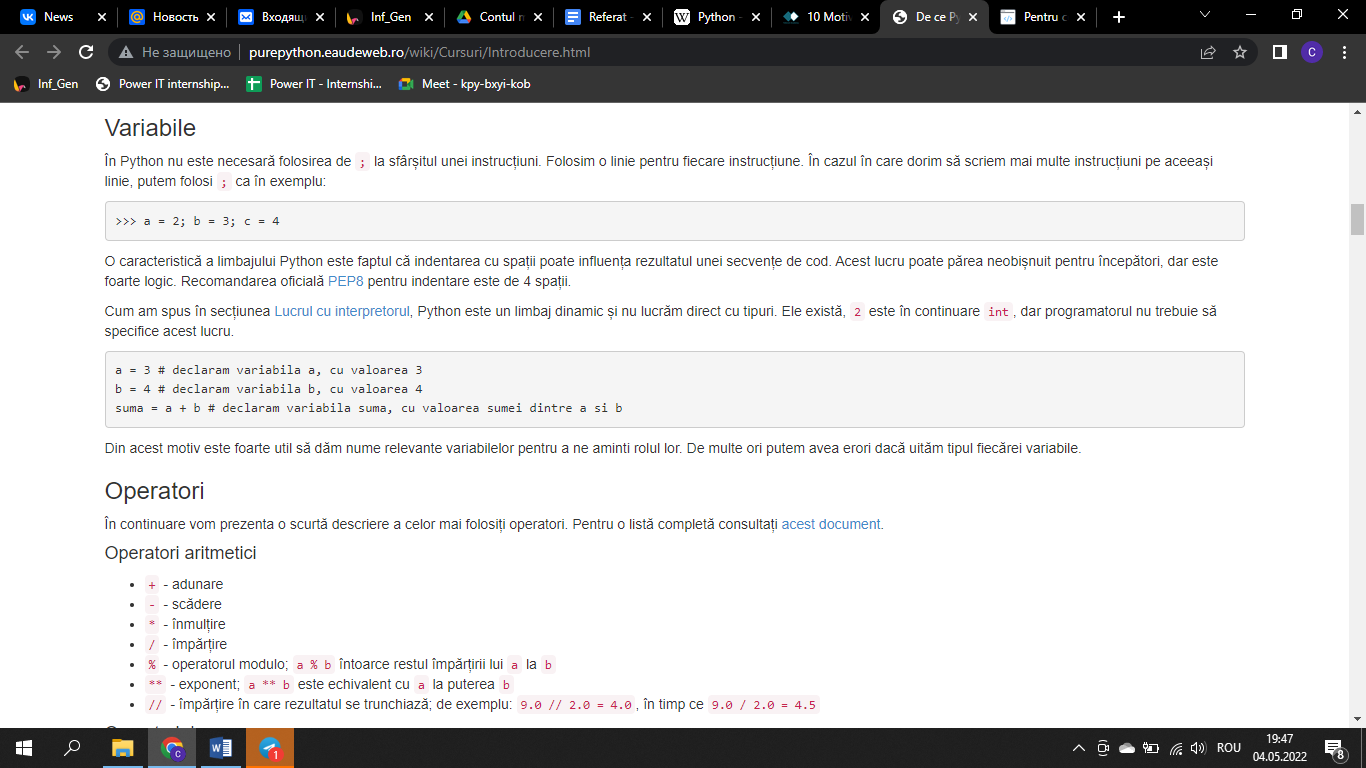


Fig.3. Declararea variabilelor.

Din acest motiv este foarte utilă numirea relevantă a variabilelor pentru a aminti rolul lor.

# Tipurile de operatori ai limbajului

1. Operatori aritmetici;
2. Operatori de comparare;
3. Operatori de atribuire;
4. Operatori logici;
5. Operatori de apartenență.

Operatorii aritmetici sunt:

* + - adunare;
* - - scădere;
* \* - înmulțire;
* / - împărțire;
* % - operatorul modulo; a % b întoarce restul împărțirii lui a la b;
* \*\* - exponent; a \*\* b este echivalent cu a la puterea b;
* // - împărțire în care rezultatul se trunchiază; de exemplu: 9.0 // 2.0 = 4.0, în timp ce 9.0 / 2.0 = 4.5;

Similar cu alte limbaje, operatorii de comparare sunt: ==, !=, <>, <, >, <=, >=.

Cel mai des folosit operator de atibuire este =, care atribuie variabilei din stânga lui valoarea expresiei din dreapta lui. Pe lângă acesta, s-au definit o serie de operatori compuși astfel: operatoraritmetic+ operatorul simplu de atribuire. Exemplu:

+= - realizează o adunare și o atribuire, astfel: a += b este echivalent cu a = a + b

%= - atribuie operandului din stânga restul împărțirii acestuia la operandul din dreapta, astfel: a %= b este echivalent cu a = a % b. Similar funcționează și operatorii: -=, \*=, \=, \*\*=, //=.

De remarcat că operatorii logici sunt foarte aproape de limbajul natural, ceea ce face codul ușor de citit și înțeles:

* and - operatorul ȘI;
* or - operatorul SAU;
* not - operatorul negație.

Operatorii de apartenență, verifică dacă un element aparține, respectiv nu aparține, unei secvențe: in, respectiv not in.(Fig.4).

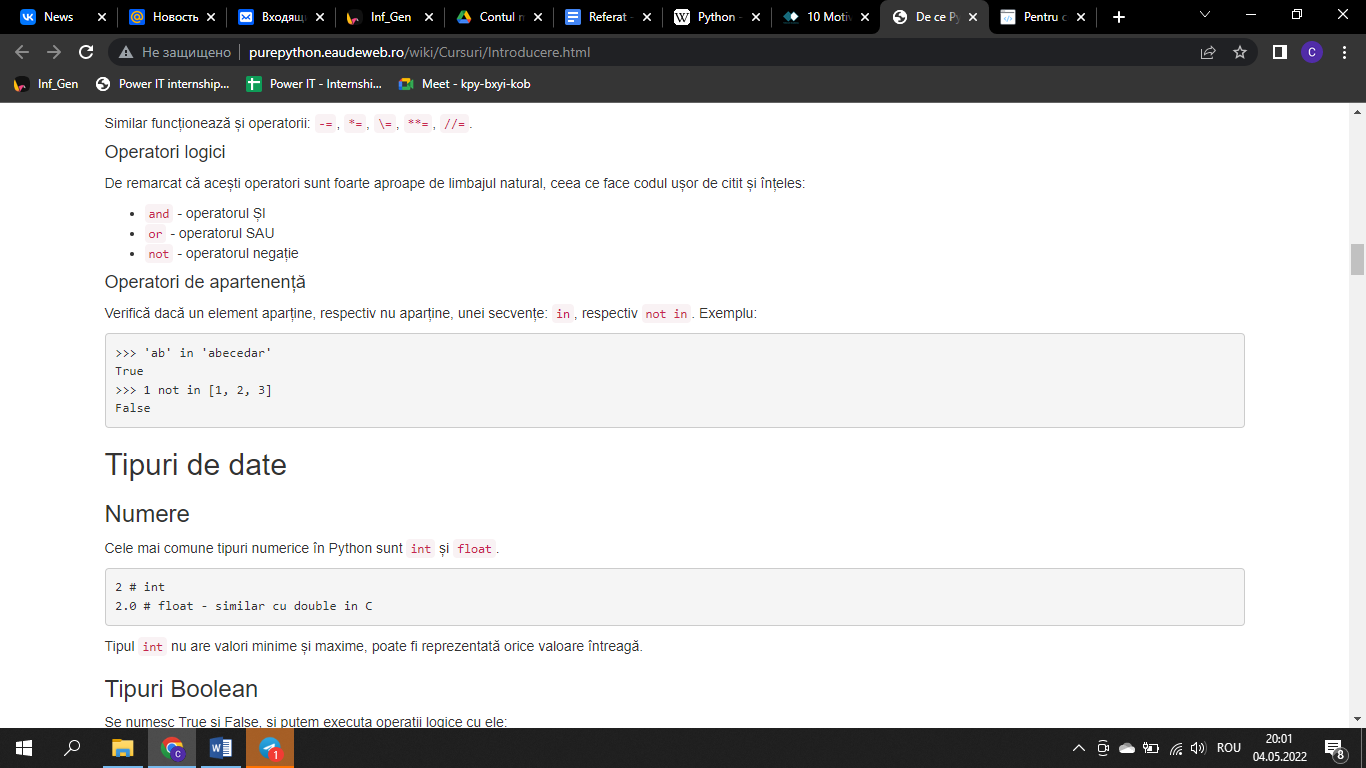


Fig.4. Operatori de apartenență.

# Tipuri de date în limbajul Python

1. Tipul de date numere;
2. Tipul de date boolean;
3. Tipul șir de caractere;
4. Liste (Vectori);
5. Dicționare.

Cele mai comune tipuri numerice în Python sunt int și float. (Fig.5)

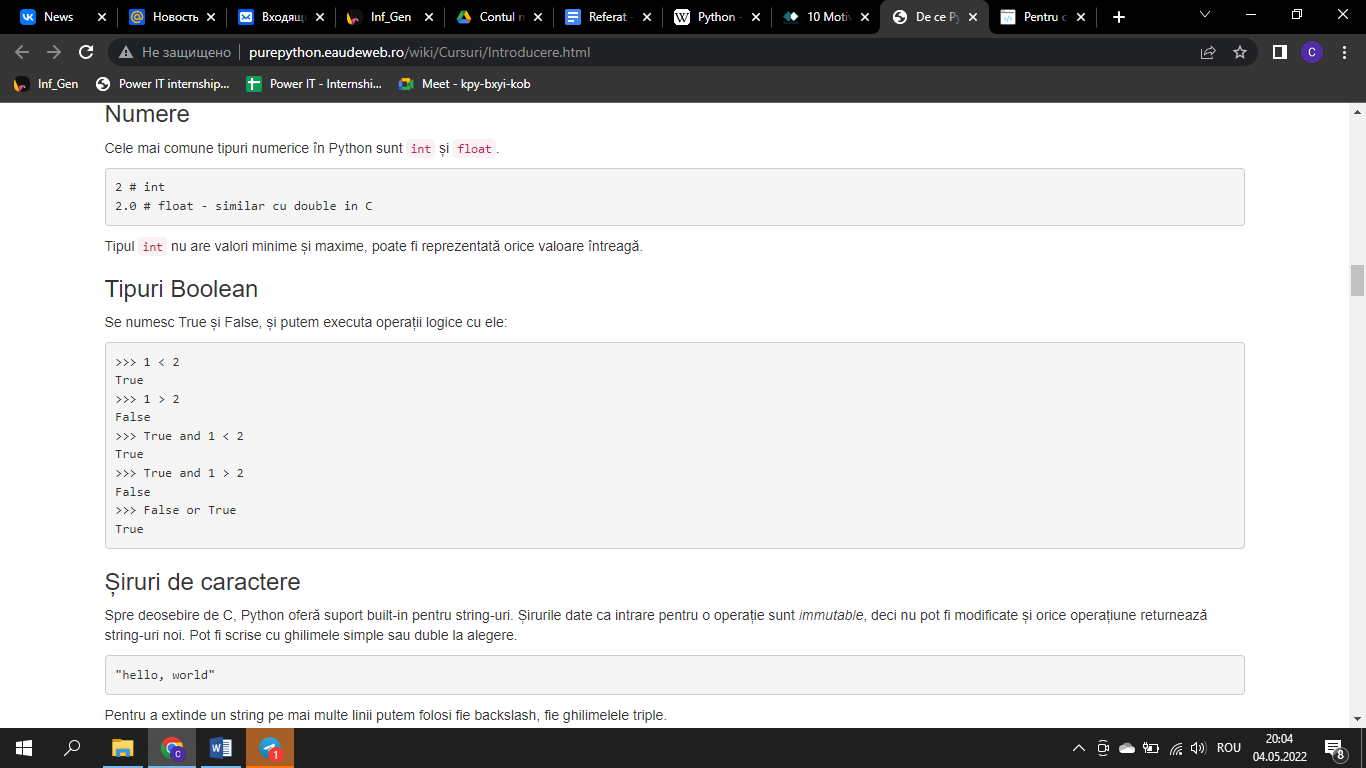


Fig.5. Tipul de date numere.

Tipul int nu are valori minime și maxime, poate fi reprezentată orice valoare întreagă.

Tipul Boolean, are două valori True și False, și se pot executa operații logice cu ele. (Fig.6)

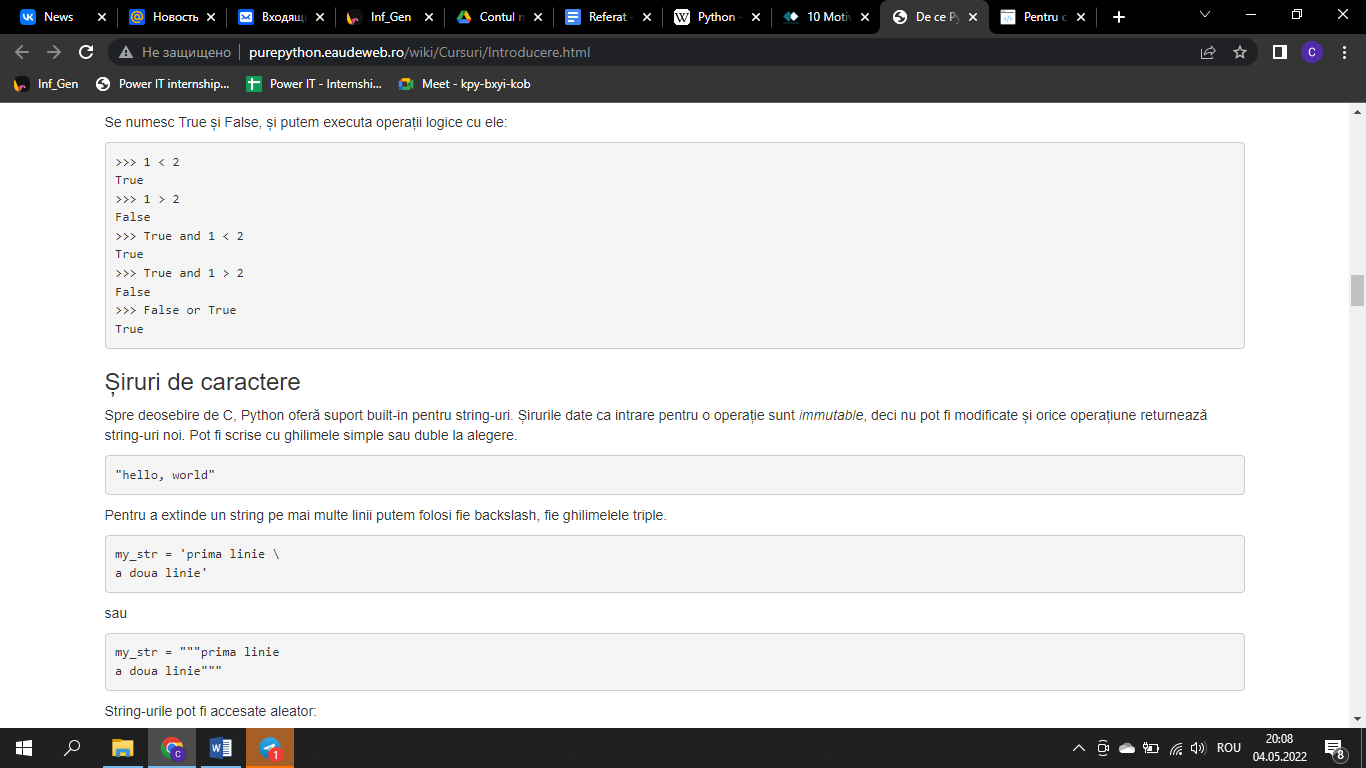


Fig.6. Tipul de date boolean.

Spre deosebire de C, Python oferă suport built-in pentru string-uri. Șirurile date ca intrare pentru o operație sunt immutable, deci nu pot fi modificate și orice operațiune returnează string-uri noi. Pot fi scrise cu ghilimele simple sau duble, la alegere.

Pentru a extinde un string pe mai multe linii putem folosi fie backslash, fie ghilimelele triple (Fig. 7).

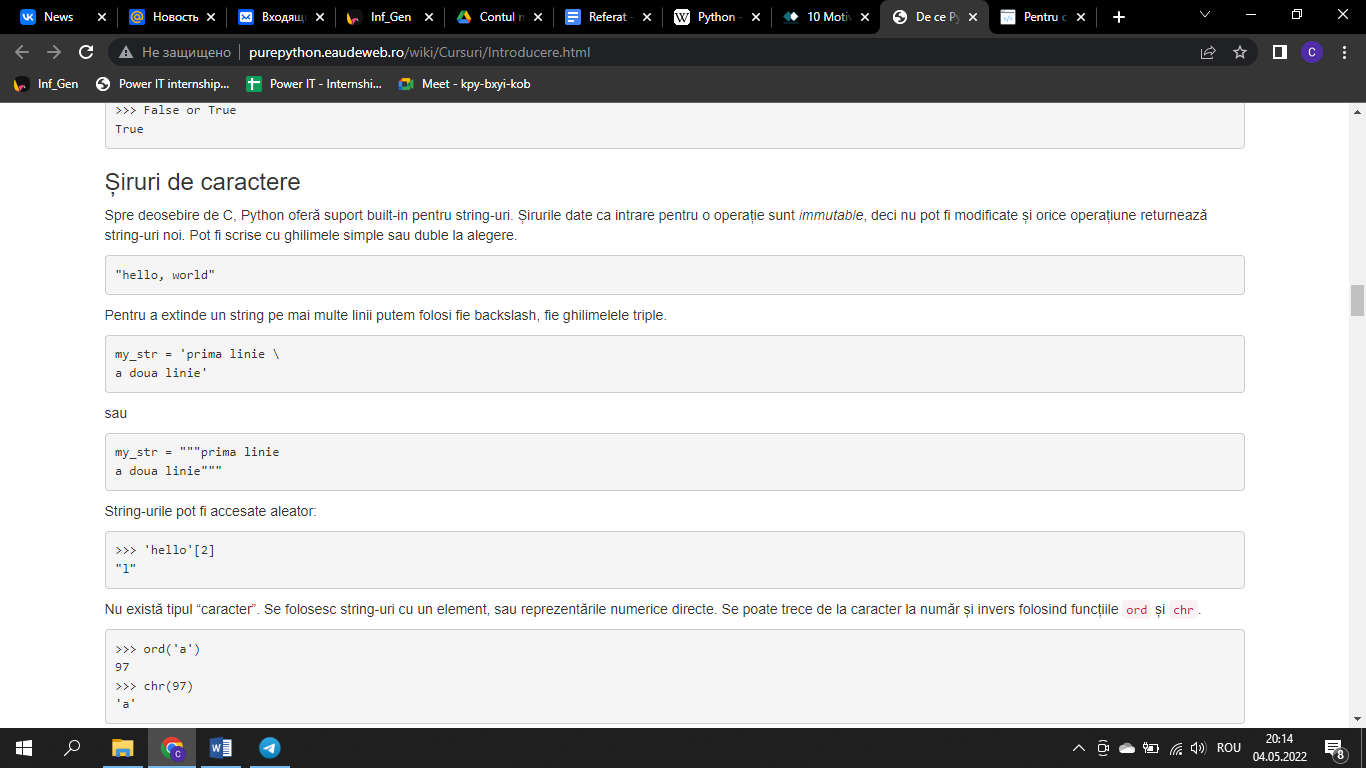


Fig.7. Tipul string pe mai multe linii.

String-urile pot fi accesate aleator. (Fig.8)

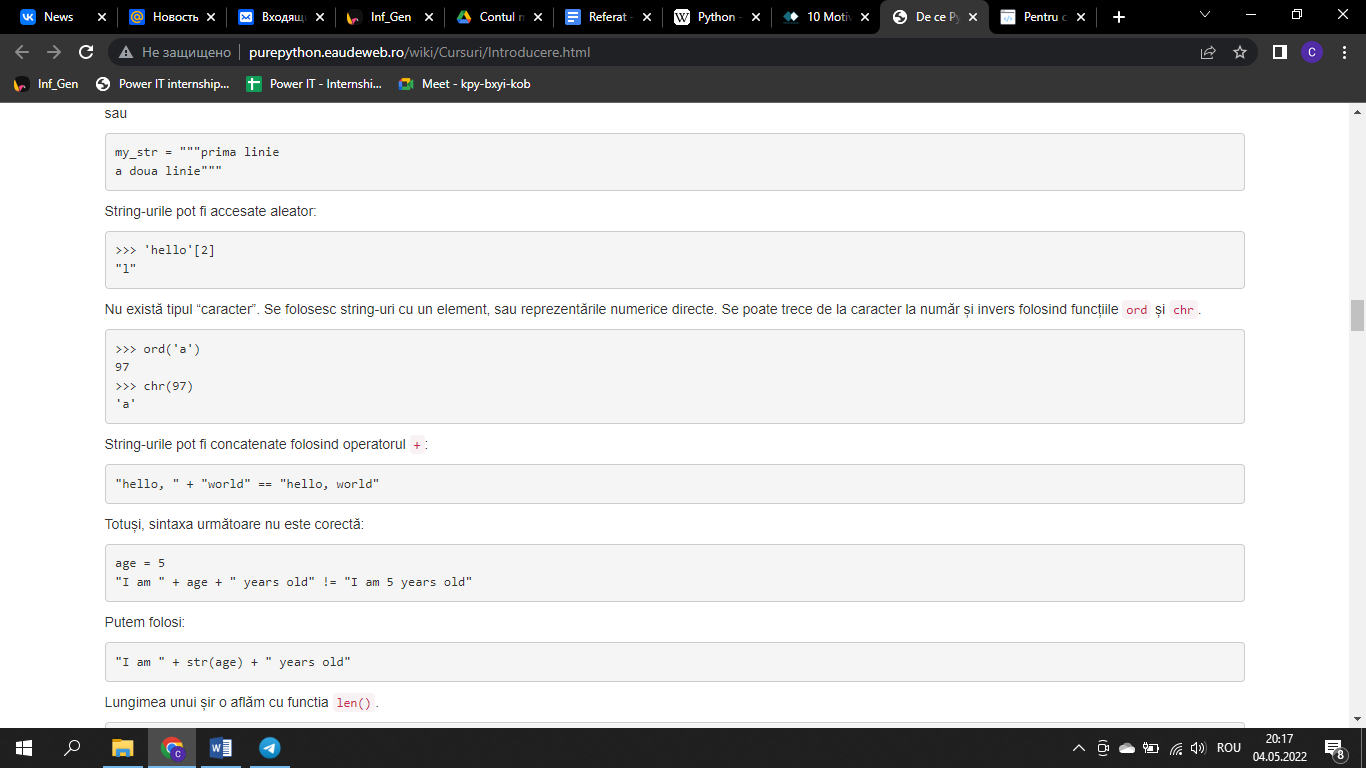


Fig.8. Accesarea.

Nu există tipul „caracter”. Se folosesc string-uri cu un element, sau reprezentările numerice directe. Se poate trece de la caracter la număr și invers folosind funcțiile ord și chr. (Fig.9)

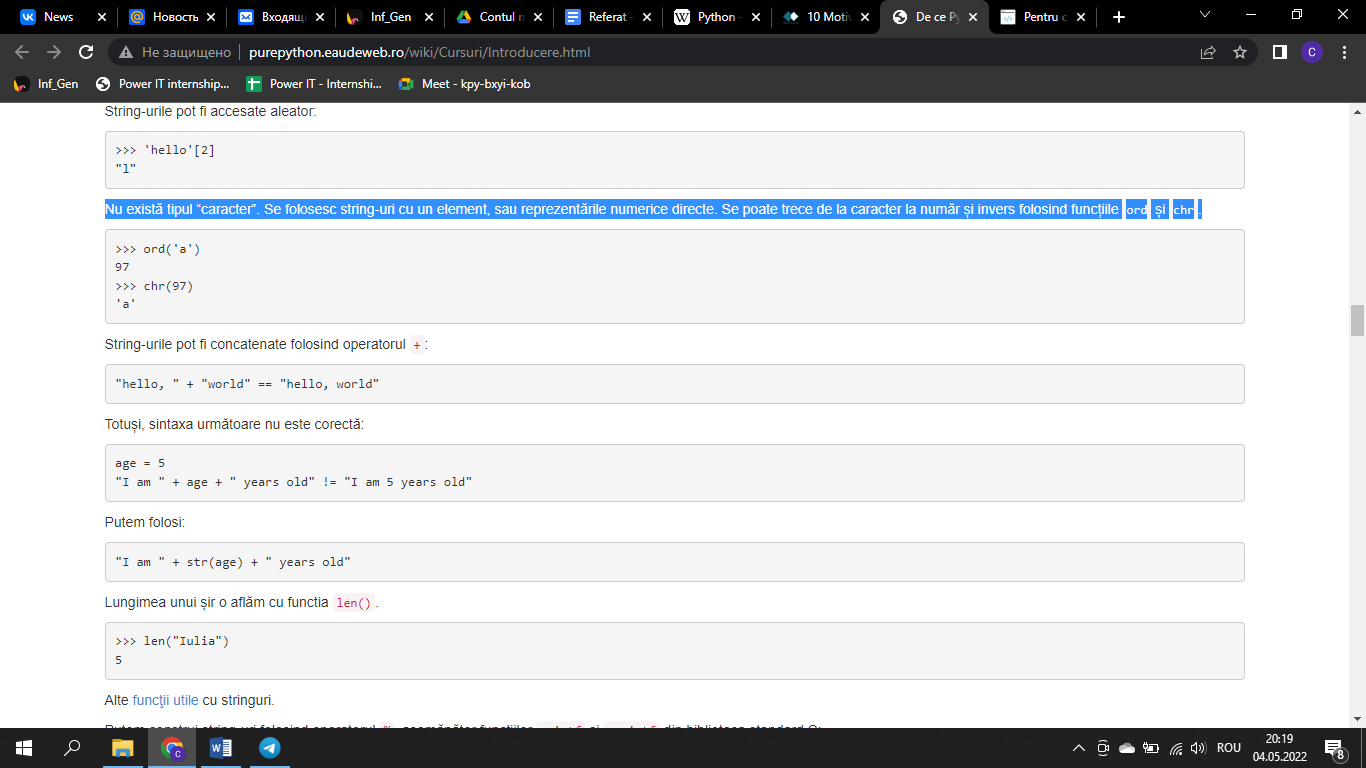


Fig.9. Funcțiile ord și chr.

String-urile pot fi concatenate folosind operatorul +. (Fig.10)

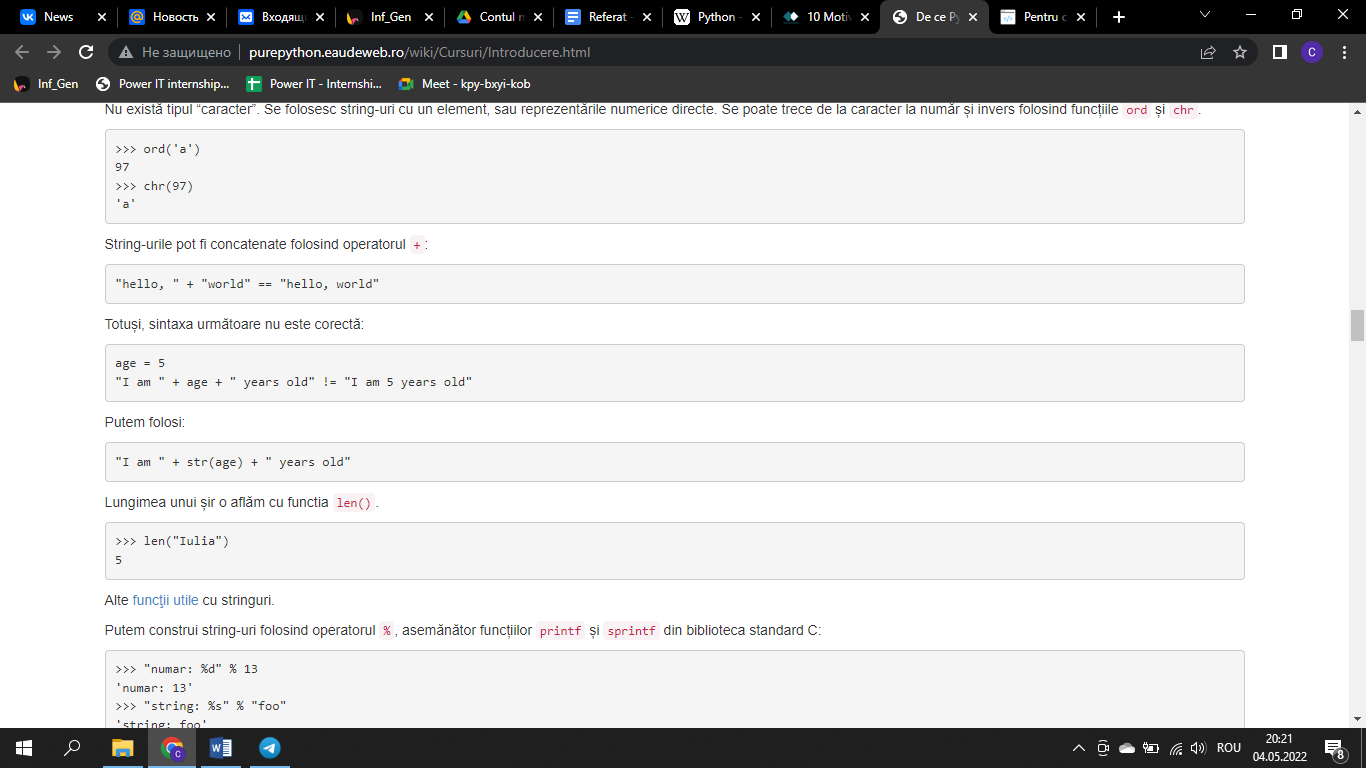


Fig.10. Concatenare.

Lungimea unui șir o aflăm cu functia len().(Fig.11)

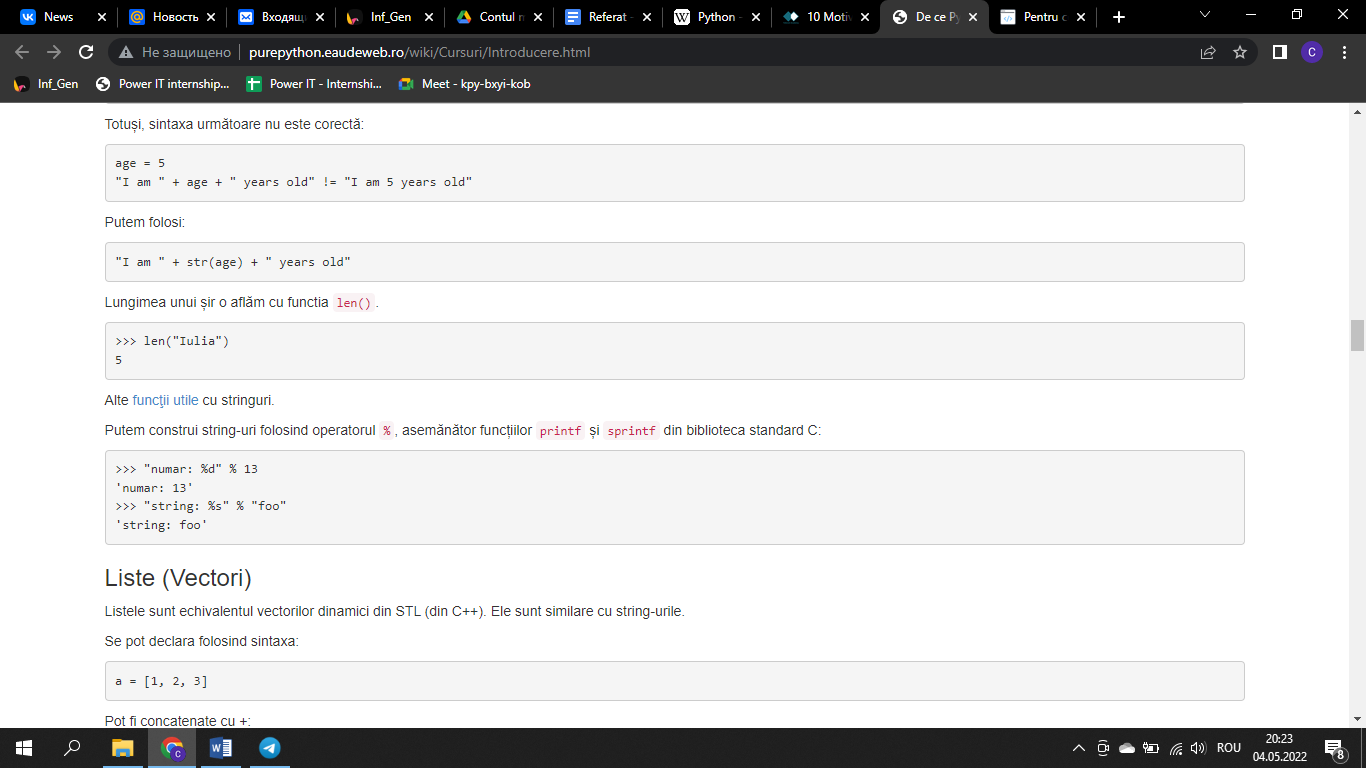


Fig.11. Funcția len().

Listele sunt echivalentul vectorilor dinamici din STL (din C++). Ele sunt similare cu string-urile. (Fig.12)

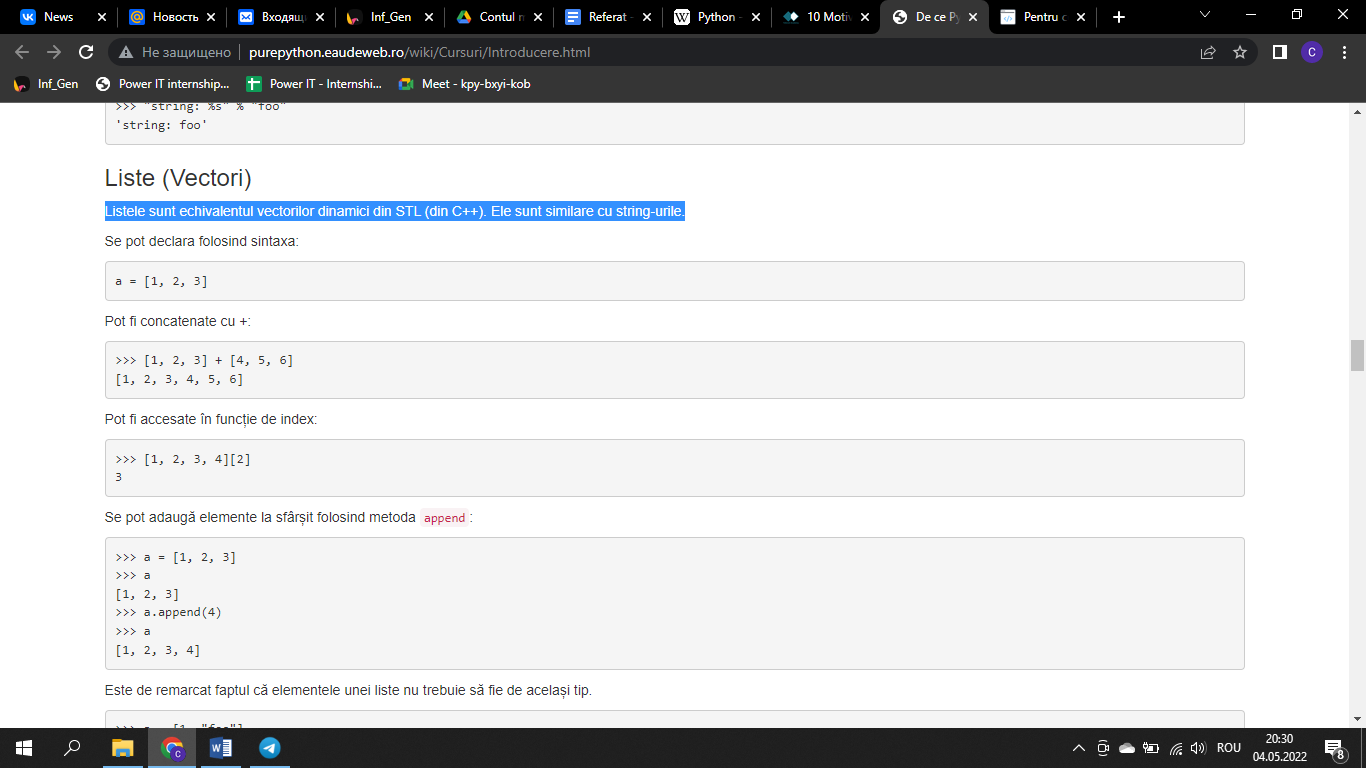


Fig.12. Declararea listei.

Pot fi concatenate cu +. (Fig.13)

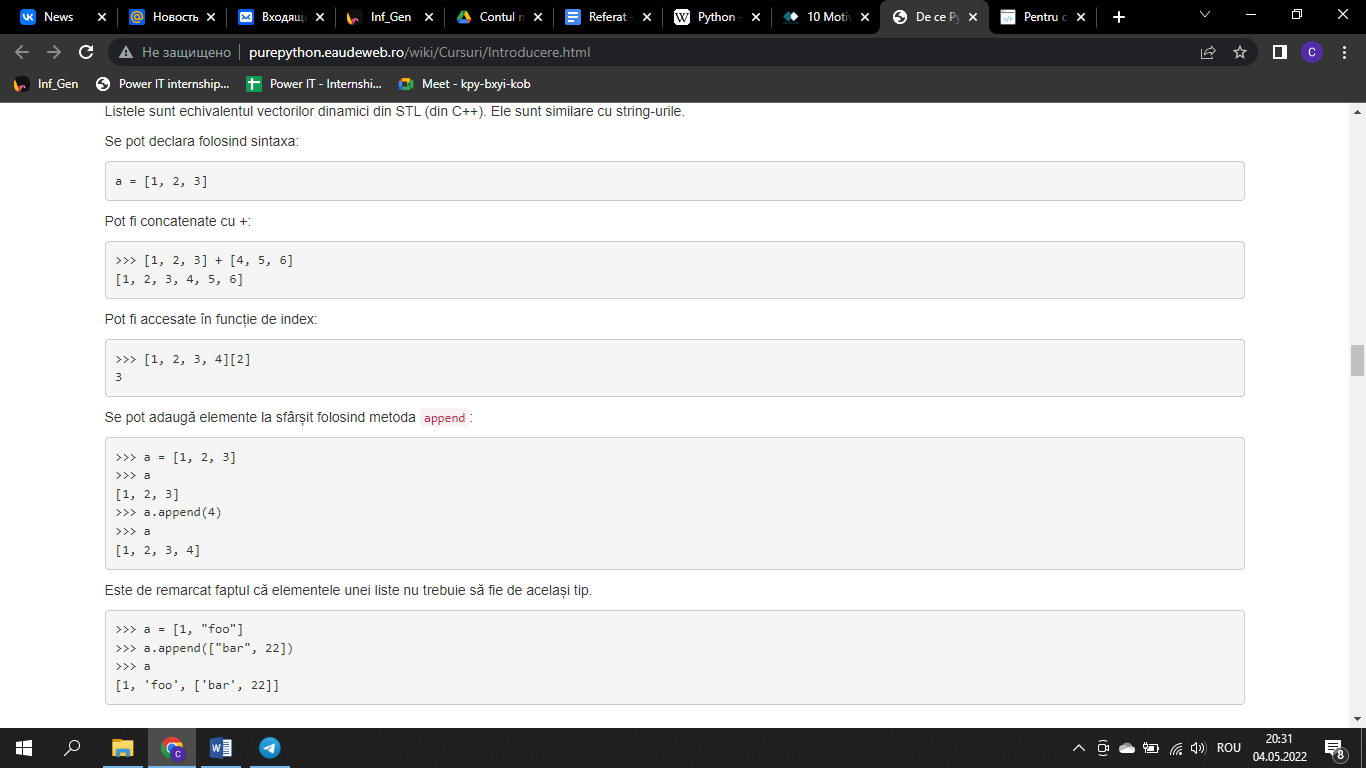


Fig.13. Concatenarea listelor.

Pot fi accesate în funcție de index (Fig.14)

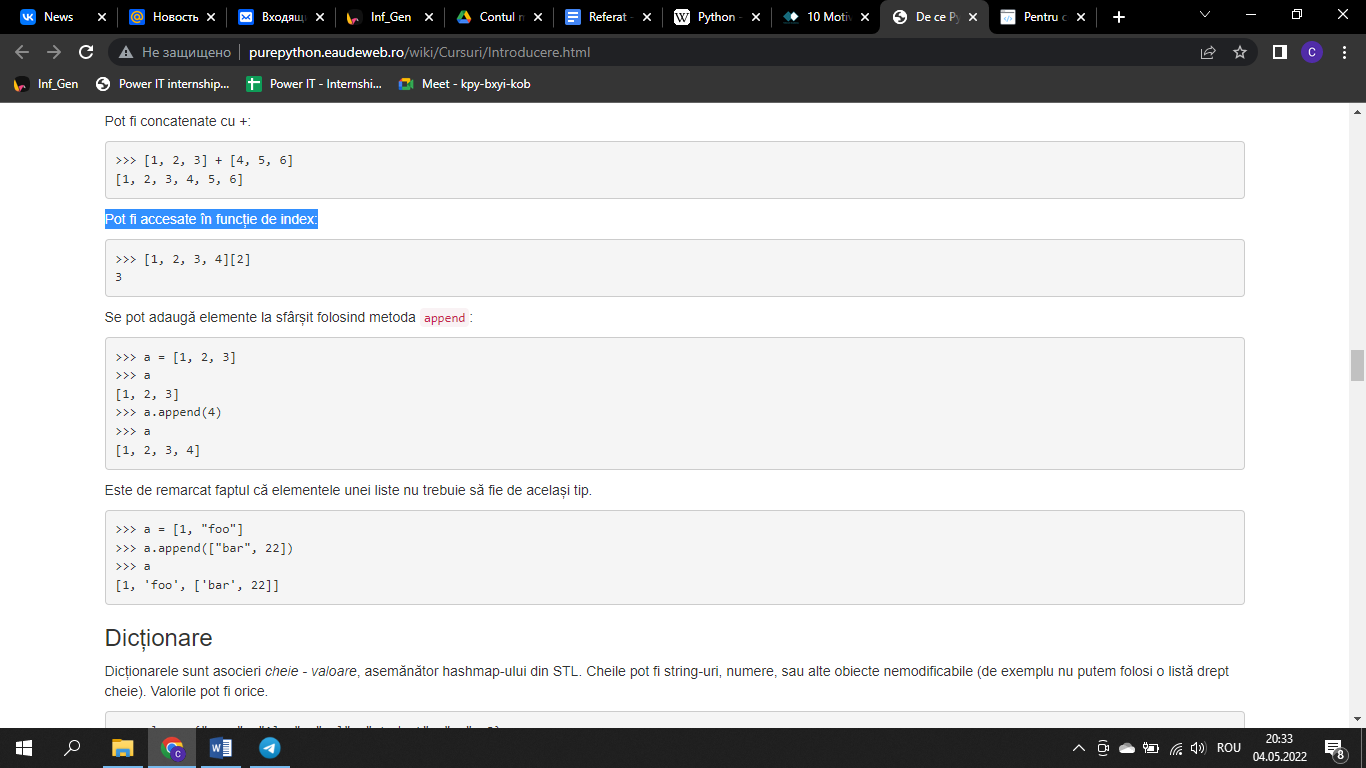


Fig.14. Accesarea componentelor listei după index.

Se pot adaugă elemente la sfârșit folosind metoda append.(Fig.15)

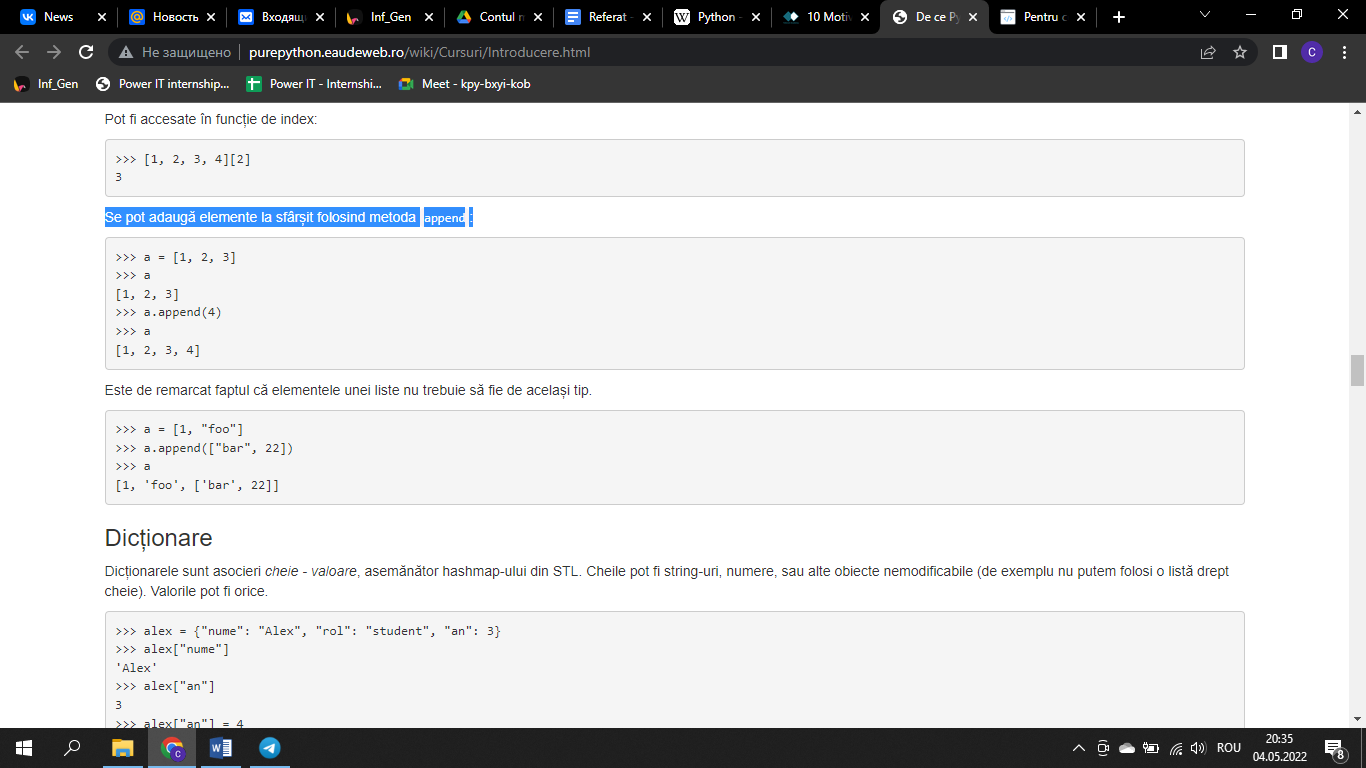


Fig.15. Adăugarea elementelor în listă.

Este de remarcat faptul că elementele unei liste nu trebuie să fie de același tip.

Dicționarele sunt asocieri cheie - valoare, asemănător hashmap-ului din STL. Cheile pot fi string-uri, numere, sau alte obiecte nemodificabile (de exemplu nu putem folosi o listă drept cheie). Valorile pot fi oricare. (Fig.16)

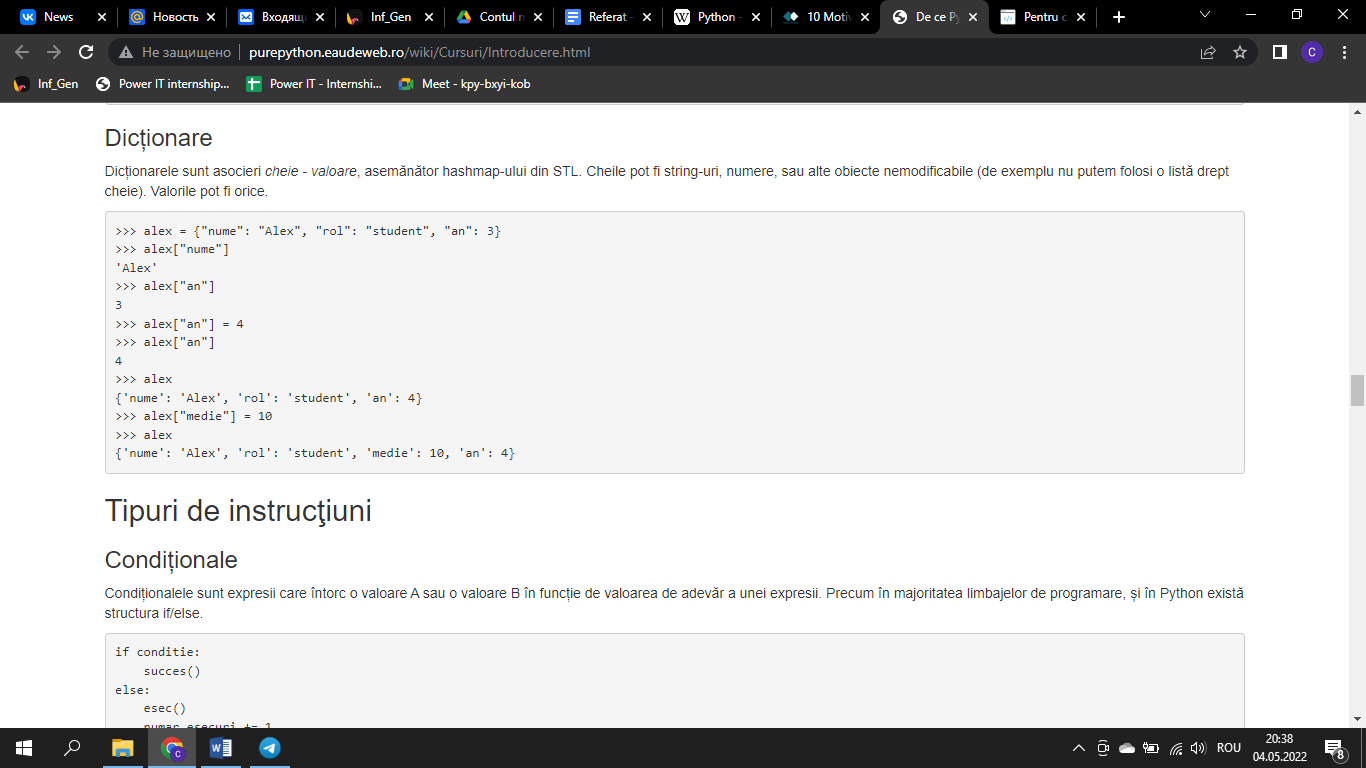


Fig.16. Dicționare.

1. TIPURI DE INSTRUCȚIUNI ÎN PYTHON
2. Instrucțiuni condiționale;
3. Instrucțiuni repetitive.

Condiționalele sunt expresii care întorc o valoare A sau o valoare B în funcție de valoarea de adevăr a unei expresii. Precum în majoritatea limbajelor de programare, și în Python există structura if/else. (Fig.17)

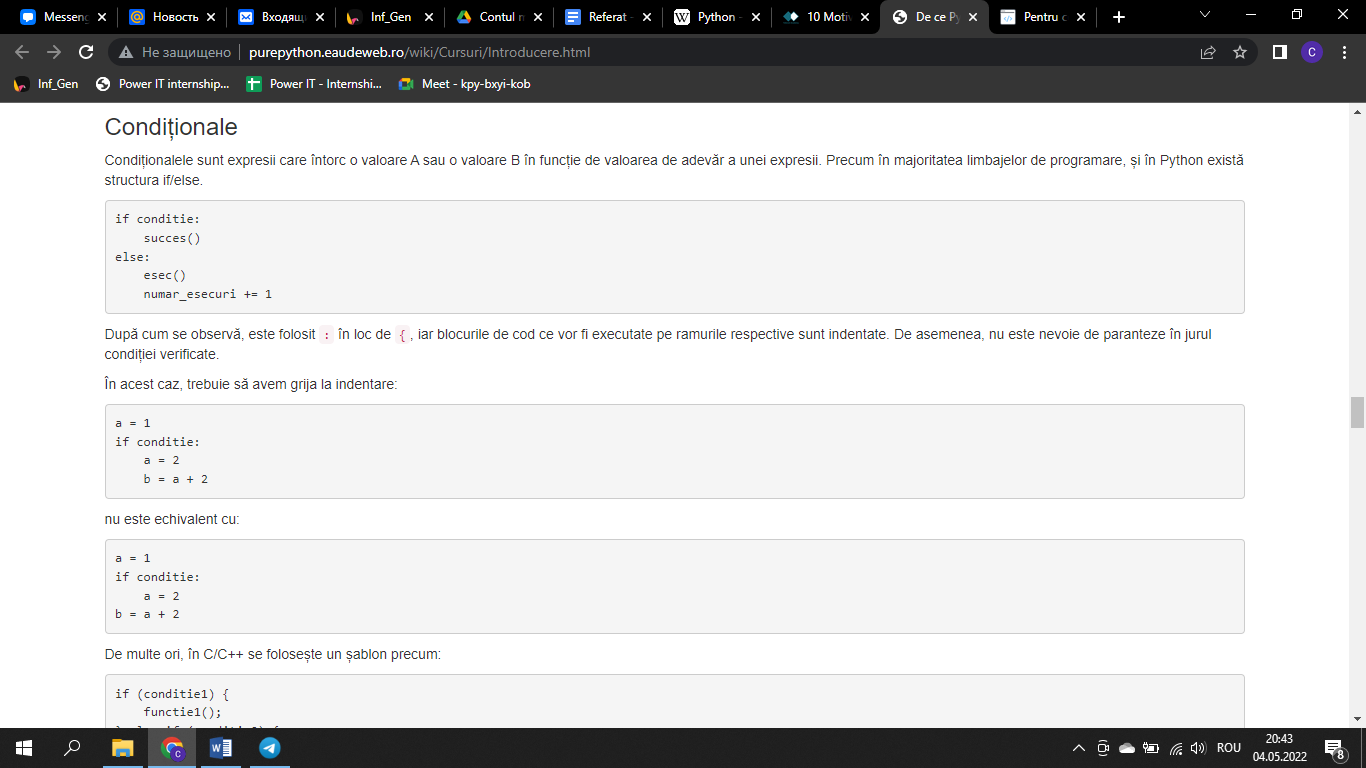


Fig.17. Instrucțiune condițională.

După cum se observă, este folosit : în loc de {, iar blocurile de cod ce vor fi executate pe ramurile respective sunt indentate. De asemenea, nu este nevoie de paranteze în jurul condiției verificate. În acest caz, trebuie de avut grijă la indentare.

Nu există else if în Python, ci elif.(Fig.18) Acest șablon de cod este aproape identic cu o structură switch. În Python nu există switch, folosindu-se blocuri if-elif-else.

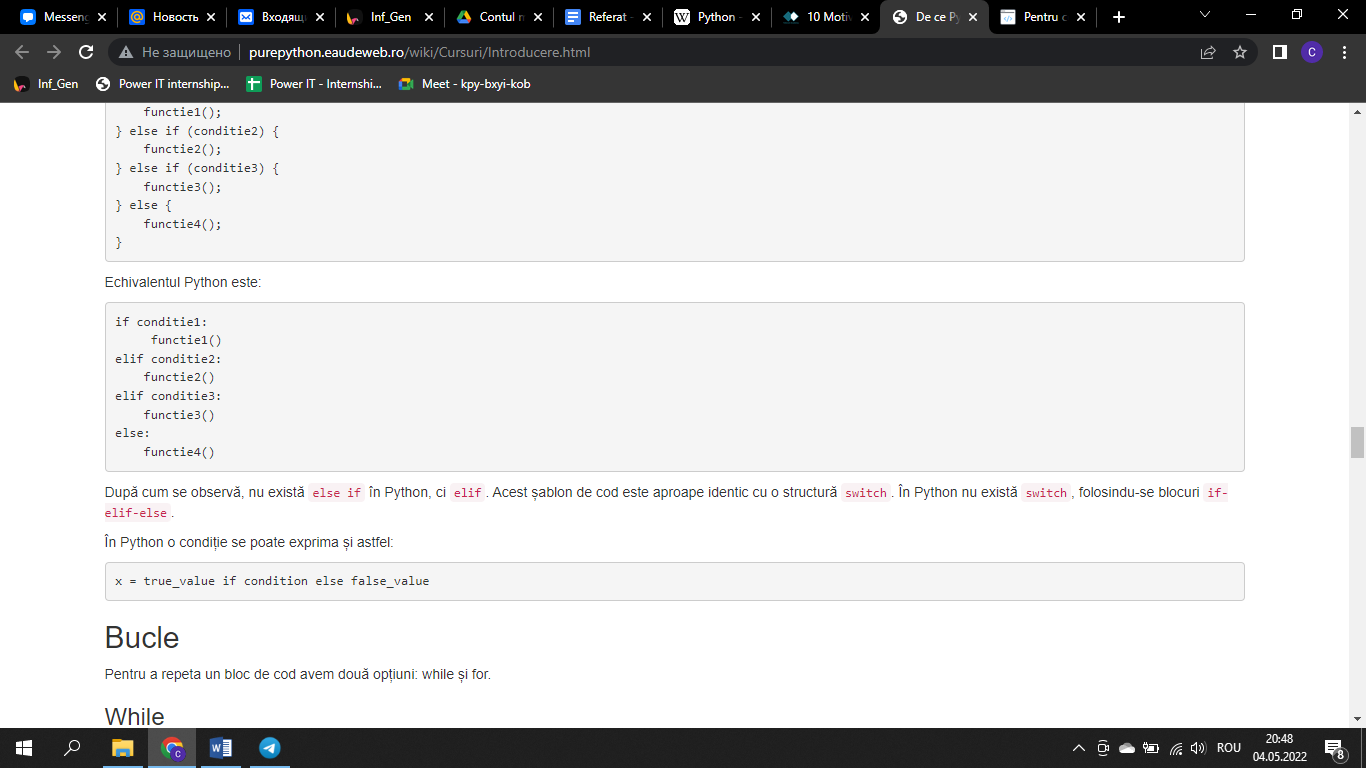


Fig.18. Instrucțiuni elif.

Pentru a repeta un bloc de cod avem două opțiuni: while și for.

Structura while din Python este la fel ca cea din C, ținând cont de diferențele de sintaxă.(Fig.19)

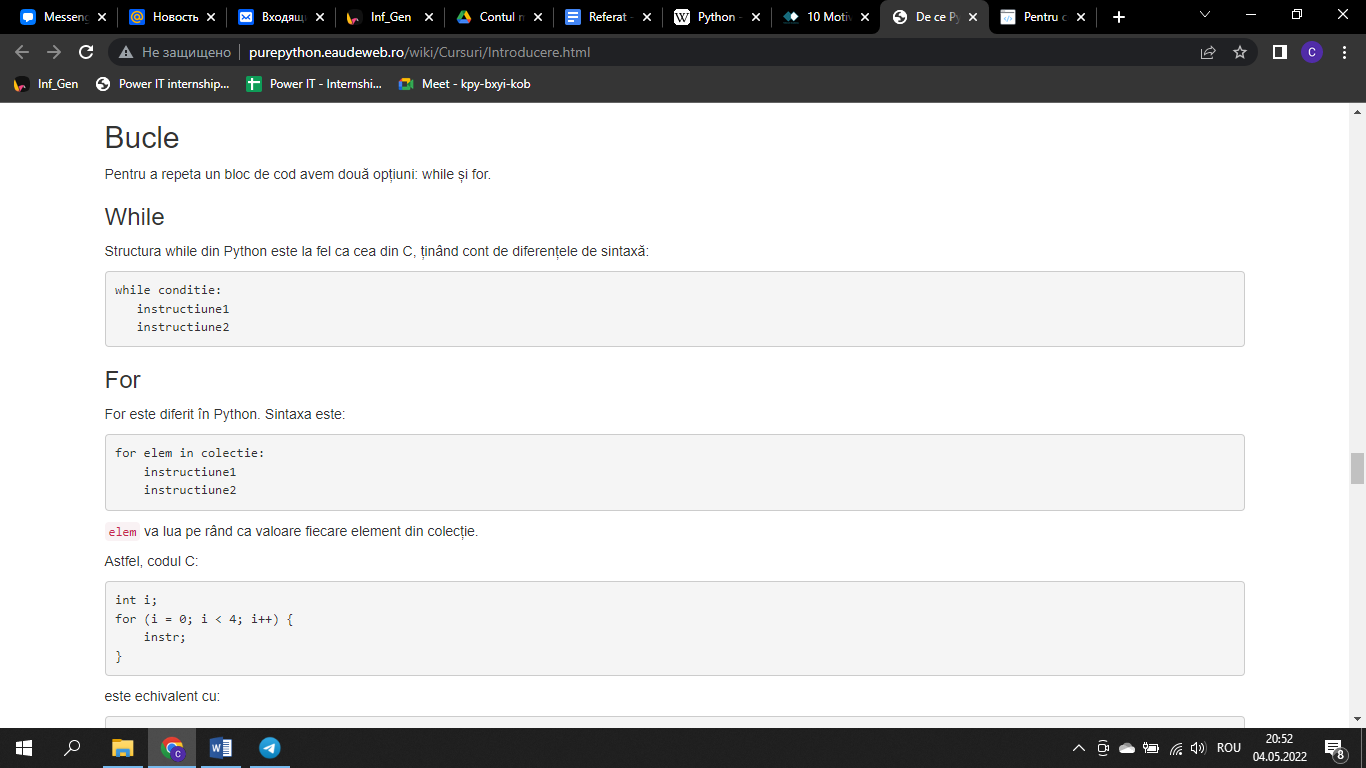


Fig.19. Bucla while.

For este diferit în Python. (Fig.20)

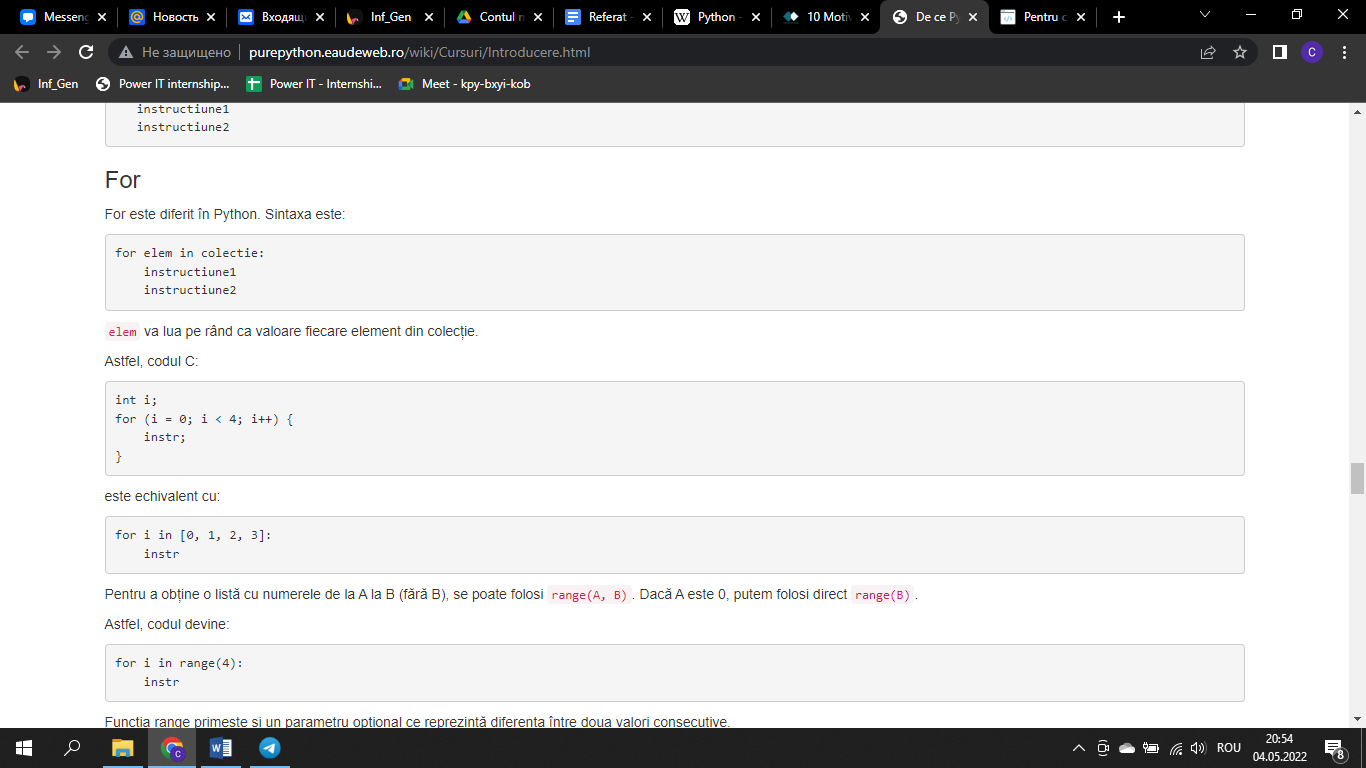


Fig.20. Bucla for.

Elem va lua pe rând ca valoare fiecare element din colecție.

1. DECLARAREA FUNCȚIILOR ÎN PYTHON

O funcție se declară folosind cuvântul cheie def, Documentarea unei funcții se face printr-un doc string: Corpul unei funcții se scrie indentat.(Fig.21)

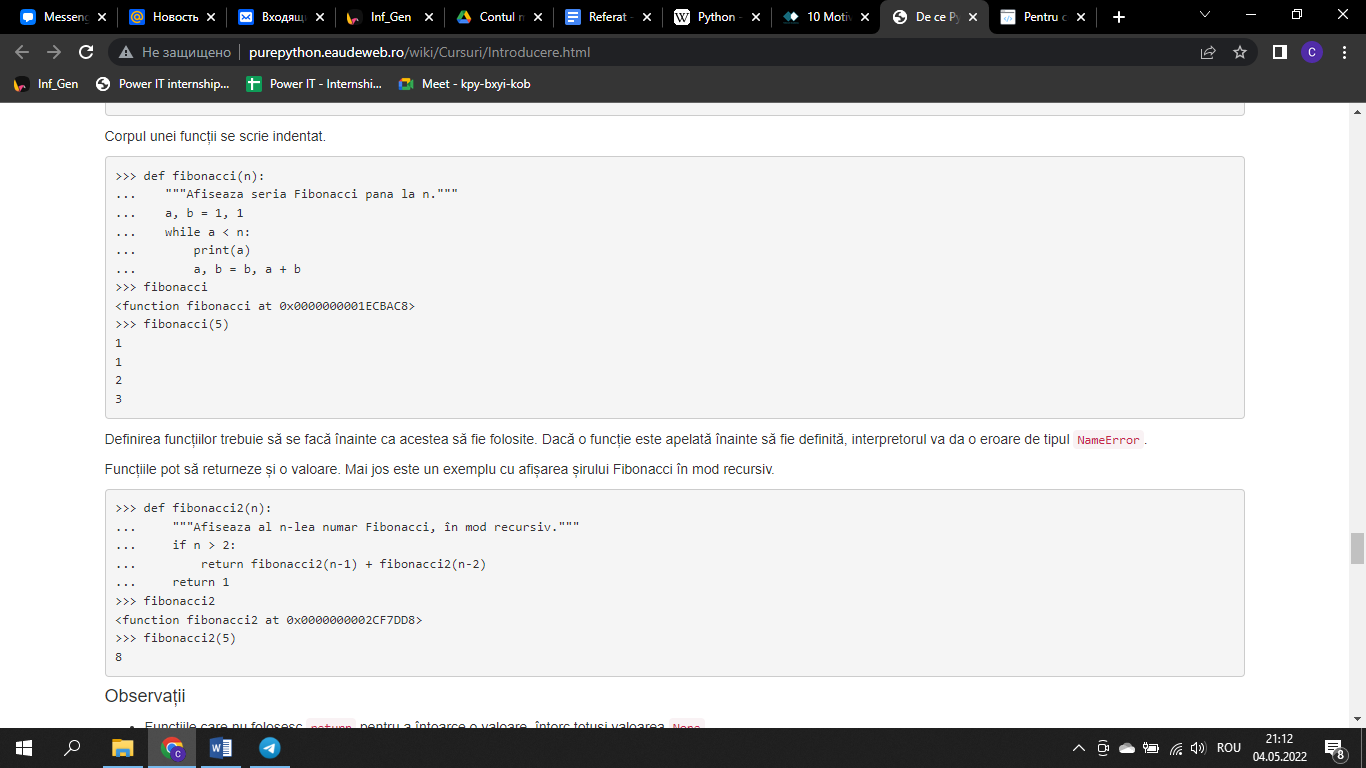


Fig.21. Funcția fibonacci.

Definirea funcțiilor trebuie să se facă înainte ca acestea să fie folosite. Dacă o funcție este apelată înainte să fie definită, interpretorul va da o eroare de tipul NameError.

Funcțiile pot să returneze și o valoare. Funcțiile care nu folosesc return pentru a întoarce o valoare, întorc totuși valoarea None. Dacă se folosește doar intrucțiunea return, valoarea întoarsă este tot None.

Variabilele declarate într-o funcție și argumentele acesteia, sunt numite variabile locale funcției. Variabilele definite la nivel global sunt numite variabile globale. Într-o funcție, putem să folosim valorile variabilelor globale: Variabilele globale pot fi modificate la nivel global. Însă, pentru a putea fi modificate într-o funcție, ele trebuie declarate folosind cuvântul cheie global.(Fig.22)

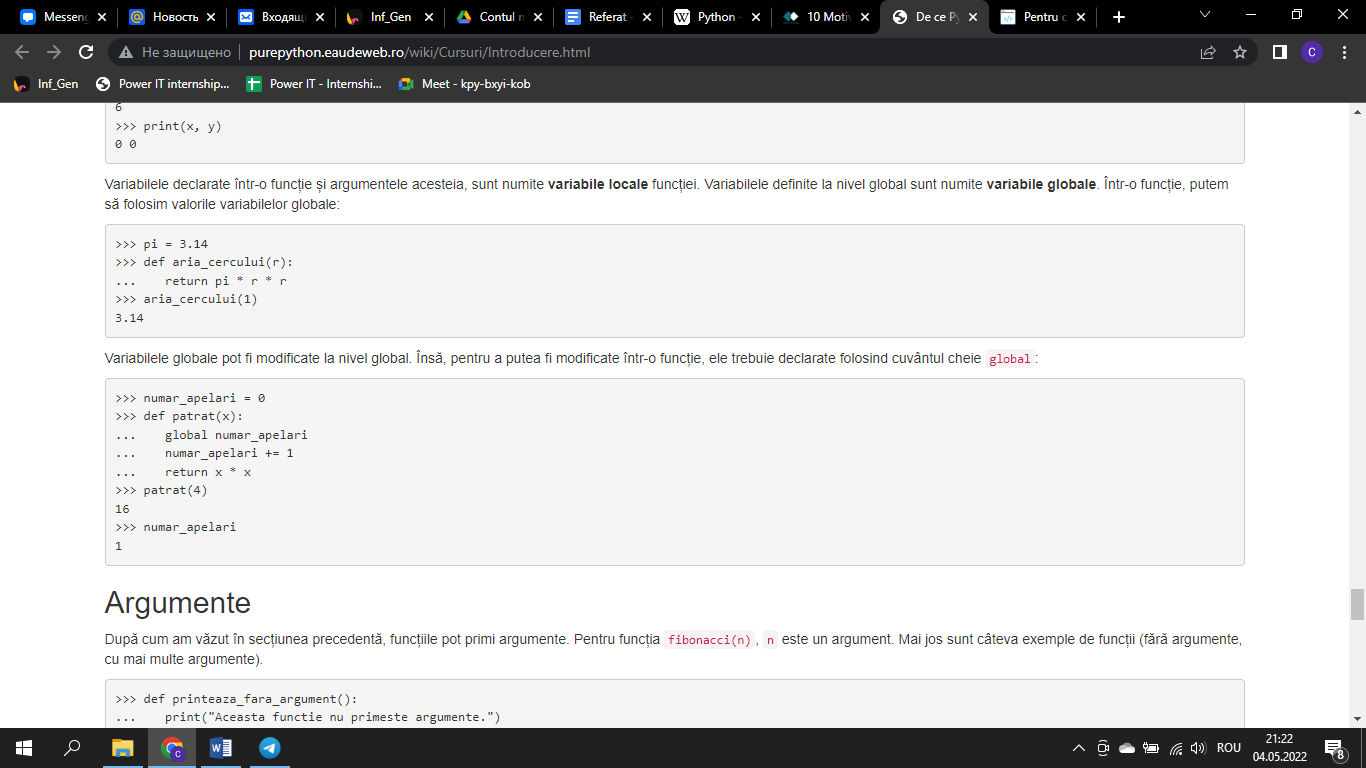


Fig.22. Declararea variabilelor globale.

1. AVANTAJELE LIMBAJULUI PYTHON
2. Sintaxa foarte simplă;
3. Este frecvent utilizat în Web Development;
4. Este foarte popular în scripting;
5. Scripturile Python pot fi utilizate pentru automatizarea sarcinilor;
6. Limbajul include numeroase biblioteci pentru data science;
7. Crește cererea pe piața muncii pentru cunoscătorii limbajului;
8. Foarte popular printre developerii seniori;
9. Are o gamă largă de librării;
10. Permite testarea facilă a aplicațiilor.

Sintaxa într-un limbaj de programare reprezintă regulile de scriere și pot fi văzute asemenea unor reguli gramaticale. În [Python](https://www.telacad.ro/cursuri/python-fundamentals/) se observă folosirea unei sintaxe ușor de urmărit, de înțeles și care este apropiată de limbajul natural. Încă de la începutul acestui limbaj, unul dintre principalele criterii pentru dezvoltarea sa a fost ușurința citirii și scrierii. Nu conține foarte multe reguli sau cazuri speciale ce trebuie tratate, lucru care încurajează scrierea codului pe cont propriu și rezolvarea cu ușurință a problemelor apărute pe parcurs.

Majoritatea aplicațiilor folosite în viața de zi cu zi au migrat în ultimii ani spre un format web, pentru a permite accesul utilizatorului de oriunde, oricând – fără a mai avea dependența de o mașină locală. De exemplu, într-o aplicație web bancară te poți autentifica cu ușurință de la birou, de acasă sau de pe telefon și să efectuezi tranzacții cu ușurință.

Python are un număr de framework-uri care au câștigat foarte multă popularitate în ultimii ani în acest domeniu, precum: Django, Flask, Pylons. Acestea sunt suficient de stabile și dau o viteză ridicată aplicațiilor, fiind în prezent folosite de firme precum: Spotify, Instagram, Dropbox, Airbnb, Netflix, Uber.

Python nu este folosit doar pentru dezvoltarea de aplicații complete, are de asemenea proprietățile necesare unui limbaj potrivit de scripting. Asta înseamnă că poate să fie folosit cu ușurință, datorită bibliotecilor sale pentru a rezolva diverse cerințe în mod rapid și a le automatiza. De ce contează asta? Pentru că bibliotecile conțin funcționalități deja implementate și pot fi reutilizate oricând. De aceea nu este necesară rescrierea aceluiași cod mereu.

Indiferent de domeniul în care activează, orice firmă își dorește să aibă automatizate procesele repetitive, deoarece eficientizează modul de lucru. Procese precum: crearea rapoartelor, colectarea datelor, a notificărilor, sau a testării. Python este limbajul perfect pentru a implementa orice tip de automatizare, având integrări foarte avansate precum: [Selenium](https://www.telacad.ro/cursuri/automated-testing-python-selenium/), behave, Robot Framework, guibot.

AI (inteligența artificială) este într-o creștere constantă și începe să fie integrată în din ce în ce mai multe aplicații pe care le utilizăm în fiecare zi. Nu se va ajunge la povești science-fiction ca „Terminator” în viitorul apropiat , dar pentru a ține pasul cu această nouă direcție, este recomandată învățarea un limbaj care oferă o bună integrare cu data science-ul.

Din fericire, în Python se regăsesc multiple biblioteci care au fost deja dezvoltate timp de mai mulți ani și testate cu succes într-un număr mare de contexte – de la procesare de imagini până la tehnici de deep-learning (scikit-learn, OpenCV, TensorFlow, FANN etc.).

Inițial nefiind atât de căutat, Python a ajuns totuși în prezent unul dintre limbajele preferate de către companii. Indiferent de domeniu (networking, automatizări, development, testare, administrare de sistem ș.a.) și indiferent de nivelul de experiență (junior, middle, senior) se observă că Python este cel mai popular.

Această tranziție a avut loc treptat, iar motivele enumerate mai sus sunt doar câteva care au influențat angajatorii să ceară acest limbaj.

Indiferent de clasamente – [StackOverFlow](https://insights.stackoverflow.com/survey/2020), TIOBE, PYPL – Python s-a bucurat de o creștere majoră a popularității și utilizării între developerii cu experiență, în mod special în ultimii 5-6 ani. În majoritatea clasamentelor se află chiar pe prima poziție, iar acolo unde nu o face, cu siguranță este în top 3.

Este important de știut acest lucru pentru că există o evoluție continuă a limbajelor de programare, iar faptul că acesta începe să fie preferat și de persoanele cu ani de experiență în domeniu arată stabilitatea și posibilitățile de dezvoltare pe care le pune la dispoziție Python.

Orice aplicație are nevoie să fie testată înainte să ajungă la clienți. Nicio firmă nu își dorește să livreze aplicații cu defecte iar testarea manuală poate să consume destul de mult timp. Datorită framework-urilor ce sunt integrate în cadrul Python, precum PyTest, Selenium, Faker, PyUnit, Unitest – testarea devine ușor de realizat pentru orice membru al unei echipe – de la developeri la product owners.

CONCLUZII

Python este un limbaj de programare util, întrucât este utilizat în aproape orice industrie și domeniu științific: știința datelor, învățarea autonomă, dezvoltarea web, educație în informatică, viziune computerizată și prelucrarea imaginilor, dezvoltarea jocurilor, medicină și farmacie, biologie și bioinformatică, neuroștiințe și psihologie, astronomie, robotică, etc. Are o sintaxă simplă, fiind asemănătoare cu pseudocodul. Învățarea Python, va deschide cu siguranță multe uși, el este aici pentru a rămâne, a transforma și a îmbunătăți lumea noastră actuală și va continua să o facă mulți ani.

BIBLIOGRAFIE

1. Allen Downey *Think Python. How to think like a computer scientist* Editura Copyright, 2012, disponibil online la : <https://www.greenteapress.com/thinkpython/thinkpython.pdf>;
2. Anand Chitipothu *Python practice book* Editura Copyright, 2019;
3. „Python.Sintaxa și implementarea.” [online] Disponibil la adresa: <https://ro.wikipedia.org/wiki/Python>;
4. „De ce să învăț Python?” [online] Disponibil la: <https://www.telacad.ro/10-motive-pentru-python/>;
5. „Introducere în Python” [online] Disponibil la: <http://purepython.eaudeweb.ro/wiki/Cursuri/Introducere.html>;
6. „Utilizarea Python” [online] Disponibil la: <https://ro.ilusionity.com/107-what-is-python-used-for-10-coding-uses-for-the-python-programming-language>;
7. „Sintaxa Python” [online] Disponibil la: https://ocw.cs.pub.ro/courses/programare/laboratoare/python.