LAB 1. INSTALAREA MEDIULUI PYTHON. CREAREA ȘI PORNIREA PROGRAMULUI PYTHON. TIPURI DE DATE. ȘIRURI DE CARACTERE. OPERAȚIUNI DE BAZĂ

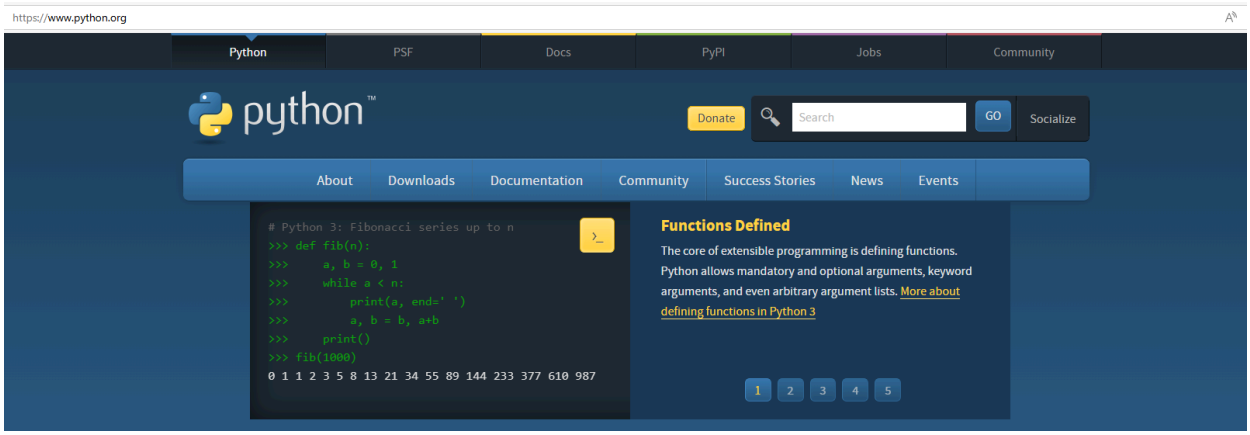
Obiective:

- Instalarea interpretatorului Python pe sistemul de operare Windows.
- Crearea și rularea unui program simplu Python.
- Identificarea și utilizarea tipurilor de date în Python.
- Lucrul cu șiruri de caractere și efectuarea operațiunilor de bază.

I. Instalarea mediului Python

Descărcarea Python:

Accesați site-ul oficial https://www.python.org/.



De pe pagina Downloads în zona **Download the latest version for Windows**, descărcați fișierul instalator (.exe), făcând click pe **Download Python 3.13.1** sau altă versiune mai recentă, dacă pe calculatorul dvs. este instalat sistemul de operare Windows.

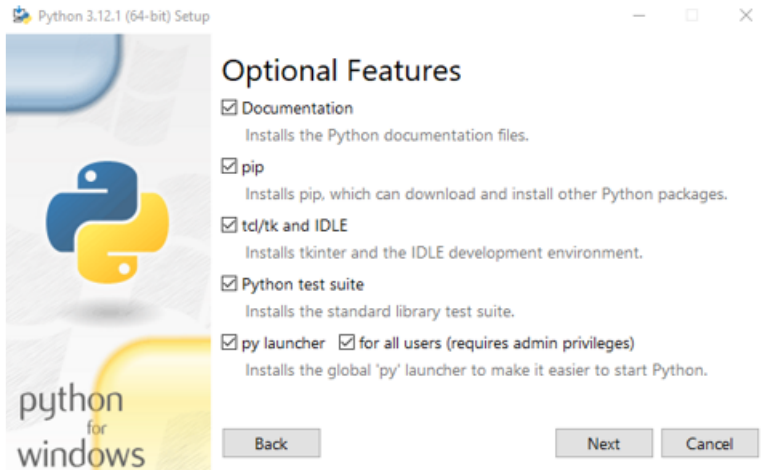
În caz contrar, selectați sistemul de operare instalat pe calculatorul dvs. din șirul de SO propuse în această zonă: **Linux/UNIX, MacOS, Other**.

Instalarea Python:

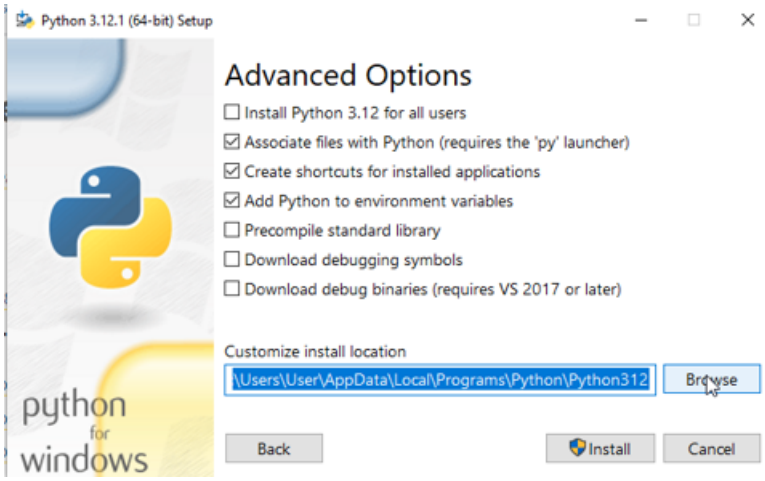
Rulați instalatorul descărcat.

Bifați opțiunile ”Use admin privileges when installing py.exe” și „Add Python to PATH”, faceți click pe opțiunea ”Customize installation”.

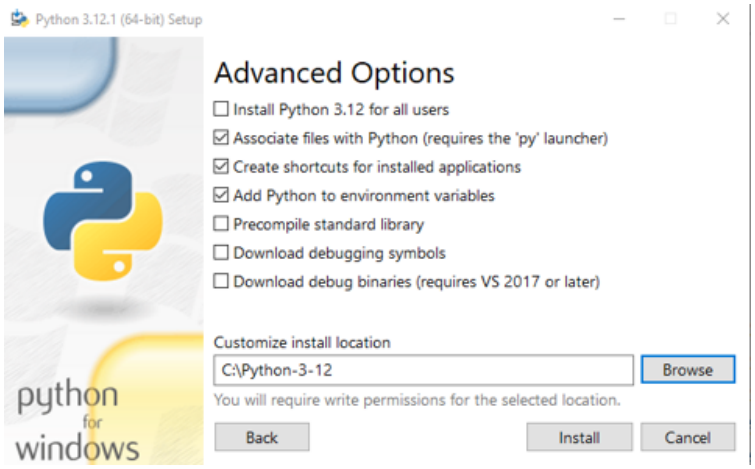
Asigurați-vă că toate opțiunile din fereastra deschisă sunt bifate.



Apăsați pe Next și indicați calea de instalare, folosind butonul Browse din dreapta câmpului unde se indică calea.



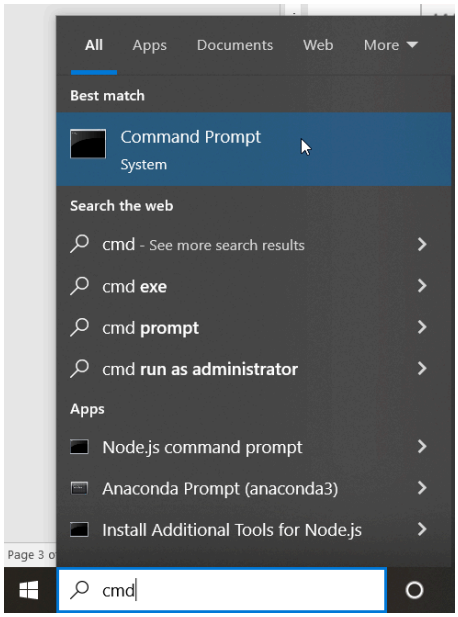
Se recomandă indicarea unei căi simple, de exemplu, pe discul C creați un director nou cu numele Python și indicarea versiunii.

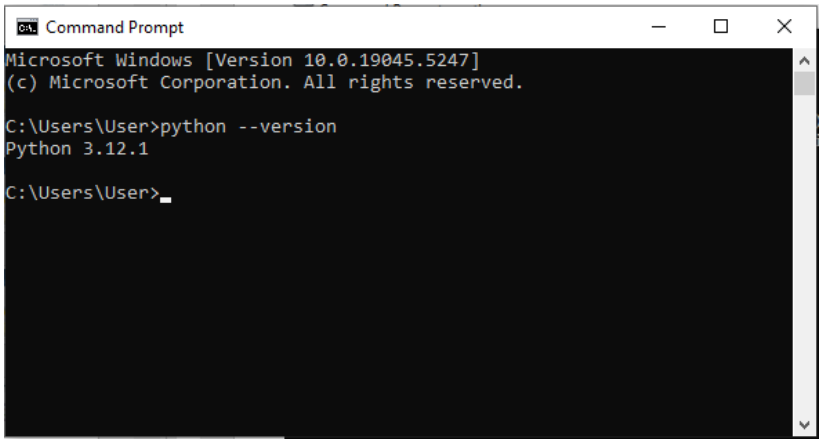


Apăsați „Install” și așteptați finalizarea instalării.

Verificarea instalării:

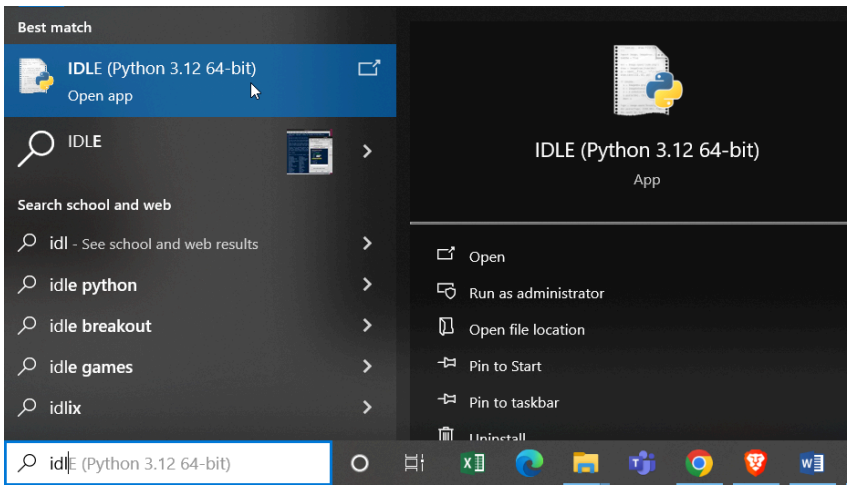
Deschideți linia de comandă (CMD) și tastați **python --version** pentru a verifica dacă Python a fost instalat corect. Ar trebui să vedeți versiunea instalată.


All
Apps
Documents
Web
More
Best match
Command Prompt
System
Search the web
cmd - See more search results
cmd exe
cmd prompt
cmd run as administrator
Apps
Node.js command prompt
Anaconda Prompt (anaconda3)
Install Additional Tools for Node.js
cmd

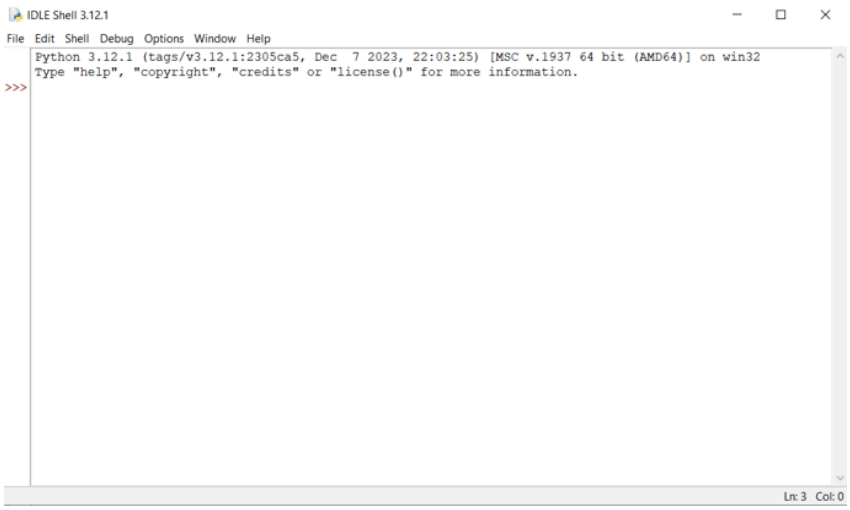

Command Prompt
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.
C:\Users\User>python --version
Python 3.12.1
C:\Users\User>

II. Crearea și pornirea programului Python

Deschideți meniul Start, căutați „IDLE” și deschideți-l.



Inițial se deschide **modul interactiv IDLE** (shell). Simbolul >>> este un prompt folosit de interpretorul Python pentru a indica că este gata. Acum putem scrie primul program.



Scrierea primului program:

Notă: Comentariile (codul care nu se execută, este ignorat de interpretor) se scriu în două moduri.

- Comentarii pe o singură linie: se folosește semnul diez (#) în fața textului

Exemplu:

```
# Acesta este un comentariu pe o singură linie
```

- Comentarii pe câteva linii: se folosesc șiruri de caractere triple, fie cu ghilimele simple (''), fie cu ghilimele duble (""), care deschid și închid comentariul.

Exemplu:

```
'''Acesta este un comentariu scris
pe doua linii'''
```

Mai întâi vom analiza o simplă instrucțiune care afișează un text. Scrieți:

```
print("Salutare, studenti UTM! :)")
```

Rulați instrucțiunea apăsând butonul Enter din tastatură. Ce s-a afișat în IDLE?

Observați, că rezultatul este returnat în următorul rând

Scrieți:

```
print(3 +4 )
```

Rulați instrucțiunea apăsând butonul Enter din tastatură. Care este rezultatul?

Observați, că IDLE returnează rezultatul sumării a două numere.

Scrieți:

```
x = [0,1,2]
print(x)
```

Rulați instrucțiunea apăsând butonul Enter. Ce s-a afișat în IDLE?

Observați, că atunci când valoarea este atribuită unei variabile, scriind numele variabilei obținem valoarea acesteia.

Modul interactiv al IDLE este convenabil, dar extrem de limitat în capacități. În această variantă codul nu poate fi reutilizat. În practică, codul programului este salvat într-un fișier și executat după salvarea fișierului. În acest caz e vorba de varianta sau **modul de execuție** a IDLE.

Pentru a intra în modul de execuție și a deschide un nou editor de cod, în fereastra IDLE selectați „File” > „New File”.

Scrieți:

```
print("Bun venit la primul laborator Python!")
```

Salvați fișierul cu numele welcome.py în directorul unde ați instalat Python sau într-un director creat pentru lucrările de laborator la BPC.

Executarea programului:

Puteți rula programul apăsând F5 sau selectând „Run” > „Run Module” din meniu.

III. Tipuri de date simple

1. **Numere întregi** (**int**) – numere atât pozitive, cât și negative, fără partea zecimală.

Exemple: 5, -3, 100

2. **Numere reale sau numere în virgulă mobilă** (**float**) – numere atât pozitive, cât și negative cu zecimale.

Exemple: 3.14, -0.003, 2.5e3 (care reprezintă 2500)

3. **Boolean** (**bool**) – valori logice de adevărat (True / 1) sau fals (False / 0). Sunt folosite frecvent pentru verificarea satisfacerii unei condiții (if, while etc.).
4. **Șiruri** (str) – secvențe de caractere textuale, delimitate de ghilimele simple (') sau duble ("). Exemple: "Hello", 'Python', "123", . "-0,003", '2.5e3'

Definirea variabilelor și verificarea tipului variabilei

Variabilele sunt nimic altceva decât memorie rezervată sub anumit nume, în care sunt stocate valori. Cu alte cuvinte, când este creată o variabilă, în memorie se alocă spațiu pentru valoarea acestei variabile.

Deschideți o nouă fereastră IDLE („File” > „New File”) și experimentați cu următoarele tipuri de date. Scrieți:

```
# int
numar_int = 10 # variabilei numar_int i s-a atribuit valoarea 10
print(numar_int)
```

Pentru a afla de ce tip este o variabila în Python utilizam funcția **type().**

```
# int
numar_int = 10 # variabilei numar_int i s-a atribuit valoarea 10
print(numar_int)
print(type(numar_int))
print(type(20))
# float
numar_real = 10.5
print(numar_real)
print(type(numar_real))
print(type(-87.7e10))
# string
mesaj = "Salut, lume!"
print(mesaj)
print(type(mesaj))
print(type("Salut"))
# bool
este_adevarat = True
print(este_adevarat)
print(type(este_adevarat))
print(type(False))
```

Rulați programul pentru a vedea rezultatele (F5 sau selectând „Run” > „Run Module” din meniu).

Atribuiri multiple

Python permite atribuirea unei valori simultan mai multor variabile. De exemplu:

```
a = b = c = 1 # valoarea 1 este atribuită la toate trei variabile.
```

Puteți, de asemenea, să atribuiți mai multor variabile diferite valori. De exemplu:

```
a, b, c = 1, True, "UTM"
```

În acest exemplu, variabilei ***a*** i s-a atribuit valoarea 1 (număr întreg – int), variabilei ***b*** – valoarea logică True, iar variabilei ***c*** – șirul "UTM".

IV. Operații de bază cu numere

1. Adunarea (+). Ex.: `suma = 5.2 + 3`
2. Scăderea (-). Ex.: `diferenta = 10 - 4`
3. Înmulțirea (*). Ex.: `produs = 7 * 6`
4. Împărțirea (/). Ex.: `impartire = 20 / 4`
5. Împărțirea întreagă (//). Returnează doar partea întreagă. Exemplu: `impartire_intreaga = 20 // 3` (rezultat – 6)
6. Restul împărțirii (%). Returnează Exemplu: `rest = 20 % 3` (rezultat – 2)
7. Puterile (**). Exemplu: `putere = 2 ** 3` (2 la puterea 3)

Deschideți o nouă fereastră IDLE („File” > „New File”) și efectuați câteva operații aritmetice și afișați rezultatele. Pentru aceasta, scrieți:

```
a = 5
b = 3
print("a =", a, "; " "b =", b)
suma = a + b
print("Suma:", suma)
diferenta = a - b
print("Diferența:", diferenta)
produs = a * b
print("Produsul:", produs)
impartire = a / b
print("Împărțirea:", impartire)
impartire_intreaga = a // b
print("Împărțirea întreagă:", impartire_intreaga)
rest_impartire = a % b
print("Restul împărțirii:", rest_impartire)
puterea = a ** b #a la puterea b
print("a la puterea b:", puterea)
```

Rulați programul pentru a vedea rezultatele (F5 sau selectând „Run” > „Run Module” din meniu).

V. Operații de bază cu șiruri

Deschideți o nouă fereastră IDLE („File” > „New File”) și efectuați câteva operații de bază cu șiruri.

```
nume = "John"
prenume = "Doe"
```

Concatenarea este operația de combinare a două sau mai multor șiruri de caractere (strings) pentru a forma un singur șir. În Python, acest lucru se realizează folosind operatorul +.

```
nume_complet = nume + " " + prenume
print(nume_complet)
```

Rulați programul pentru a vedea rezultatele (F5 sau selectând „Run” > „Run Module” din meniu).

VI. Funcția `input()`

În Python, funcția `input()` este folosită pentru a citi date de la utilizator. Aceasta permite programului să primească un text (șir/string) introdus de utilizator. Exemplu:

```
nume = input("Te rog, introdu numele tău: ")
print("Bun venit, " + nume + "!")
numar = input("Introdu un numar: ")
print("Numarul introdus este:", numar)
```

Pentru a putea efectua operații aritmetice cu numerele introduse de utilizator, acestea trebuie convertite în int sau float. Pentru aceasta sunt folosite funcțiile `int(), float()`. Exemplu:

```
numar = int(input("Introdu un numar: "))
print("Numarul dublat este:", numar * 2)
```

Sarcina 1

1. Scrieți un program, care cere utilizatorului să-și introducă numele și două numere de la tastatură.
2. Convertiți aceste două numere în valori întregi.
3. Efectuați cu aceste numere 7 operații descrise în **IV. Operații de bază cu numere**.
4. Returnați utilizatorului următorul text:

```
Stimate (numele utilizatorului),
Numerele introduse de dvs. sunt: ...
Mai jos sunt rezultatele operațiilor de bază cu aceste numere:
Adunarea: ...
Scăderea: ...
Etc.
```

5. Adăugați comentarii explicând pașii
6. Salvați fișierul sub numele nume-prenume_lab1-1.py.

Sarcina 2

Codul de mai jos convertește temperatura Celsius in Fahrenheit.

```
celsius = eval(input("What is the Celsius temperature? "))
    fahrenheit = (9/5) * celsius + 32
    print("The temperature is ",fahrenheit," degrees Fahrenheit.")
```

1. Scrieți un program, care convertește inci în centimetri folosing formula: cm = in * 2.54.
2. Programul va solicita utilizatorului introducerea numărului în inci și va imprima distanța în centrimetri, conform exemplului:

```
Care este distanța în inci? 10 # cifra 10 introdusă de utilizator!
10 inci este 25.4 cm
```

3. Adăugați comentarii explicând pașii
4. Salvați fișierul sub numele nume-prenume_lab1-2.py.

Asigurați-vă că lucrarea este efectuată individual și că respectați termenele de predare!