Școala informală de IT

Cuprins

1.1 Introducere

Fondatorul limbajului de programare Python este Guido van Rossum. Unul dintre avantajele majore ale acestuia este ca faciliteaza scrierea rapid a codului. Acesta este utilizat in aplicatii comerciale datorita faptului ca este un limbaj care nu este dependent de sistem de operare, asa cum sunt alte limbaje.

Subiectul OOP (object-oriented programming) este unul foarte dezbatut in viata de zi cu zi. Dar care e motivatia acestei cautari? OOP este un alt mod de a programa, te poate ajuta sa gasesti solutii pentru probleme intr-un mod scalabil si reprezinta cea mai indicata solutie pentru proiectele desfasurate pe termen lung. In Python programarea OOP este optionala, se poate scrie cod si procedural, acest lucru reprezentand un avantaj in anumite situatii, in comparatie cu limbaje de programare ca C# si Java.

Un avanataj al limbajului Python este ca permite realizarea procesului de debugging in fisiere pentru descoperirea unor probleme aparute la nivelul utilizatorului.

Ce inseamna un limbaj interpretabil?

Asa cum va asteptati, Python este un limbaj interpretabil, nu compilabil asa cum este C. Limbajul compilabil reprezinta limbajul care necesita compilarea fisierelor sursa in fisiere asemanatoare codului masina. In comparatie cu limbajele interpretabile, limbajele compilabile au o rapiditate stiuta, insa timpul de compilare este foarte mare. Pe langa faptul ca Python este un limbaj interpretabil, acesta este si byte-compiled, acesta reprezentand o forma intermediara fata de codul masina.

Gasim aplicativitatea limbajului Python in:

- Filme animate (Industrial Light & Magic, Sony Pictures Imageworks, Disney, Pixar)
- Youtube folosește Python pentru căutarea intre milioanele de filme.
- Realizarea de căutări pe internet (Google, Infoseek)

- Script-uri GIS pentru hărți (ESRI)
- Distribuirea de fișiere diverse pe Internet (BitTorrent)
- Prognoza meteo (U.S. National Weather Service, NOAA)
- Test computer hardware (Seagate, Intel, Hewlett-Packard, Micron, KLA)
- Analiza numerica (NASA, Los Alamos National Laboratory, Lawrence Livermore National Laboratory, Fermi)
- Criptografie și analiza financiara (NSA, Getco)
- Jocuri și grafica (Activision, Electronic Arts, Infogames, Origin, Corel, Blender, PyGame)
- Navigarea navetelor spațiale și experimente de control (Jet Propulsion Laboratory)
- Yahoo maps și cautare în directoare (Yahoo!)
- Ghid pentru instalarea Linux și mentenanță (Red Hat)
- Implementarea de siteuri web (Disney, JPL, Zope, Plone, Twisted, Instagram)
- Crearea de sisteme de apărare cu rachete (Lockheed Martin)
- Administrarea de liste de mail (Mailman)
- Postarea pe Facebook

Limbajul Python are si o poezie in spate. Aceasta se poate vedea utilizand comanda “import this”.

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1.2. Tipuri de fișiere

Daca incercam sa intelegeam tipurile de fisiere aflate in spatele limbajului, vom afla ca exista:

1. un fisier executabil python.exe cu rol de rulare in mod consola. Este utilizat daca scriptul nu adauga interfete grafice.
2. un fisier pythonw.exe care are rol de rulare in mod non-consola utilizat in construirea de servicii sau interfete.

in momentul in care dorim sa executam un program, fisierele de tip *.py si *.pyw sunt compilate in mod automat de catre interpretor care, apoi, este executata de sistemul de executie al limbajului numit python virtual machine, denumit si interpretor. Traducerea în byte code se realizează când dorim sa rulam. Byte code nu este cod mașină și este în cele din urma executat prin platforme de tip Python virtual machine, nu direct de către hardware-ul computerului.

Fisierele de tip pyc te poate ajuta sa accelerezi timpul de pornire al programelor, aceasta fiind denumita si versiune octet compilata. De exemplu: timpul de modificare al versiunii de sexemplu.py folosit pentru a crea exemplu.pyc este înregistrat în exemplu.pyc. În cazul în care timpul înregistrat în fișierul *.pyc nu se potrivește cu timpul ultimei modificări a fișierului *.py, fișierul pyc este ignorat si apoi rescris. Deci *pyc este creat automat la prima rulare a programului si se updateaza fara ca noi sa intervenim in acest proces. Trebuie amintit și faptul că fișierul cu extensia *pyo este un fișier pyc optimizat. Deoarece *.pyc este optimizat pentru o rulare cat mai eficienta intr-un mod automat, optimizarea manuala nu-și are rostul decât in cazul in care dorim sa realizam fișiere in C.

1.3. Șiruri de caractere

Lucrul cu sirurile de caractere este un lucru simplu si interesant in Python. Primul nostru program ne ajuta sa afisam sirul de stringuri (sir de caractere). Plecand de la premisa ca dorim sa il ghidam pe utilizator sa foloseasca un meniu dintr-un program,

In primul program "Salut Prieteni" am putut să afișam primul nostru șir de caractere (string în lb.en.). Dar șirurile de caractere pot deveni mult mai lungi și mai complexe. Spre exemplu dorim să dam indicații despre folosirea meniului unui program de tip consolă. Ar fi destul de obositor să scriem cate o linie, astfel ca folosim ghilimelele triple. Programul de mai jos exprima acest concept.

```
>>> print('Program "hai la tenis" la ora 18:00')
Program "hai la tenis" la ora 18:00
```

Dupa cum se poate observa pentru a putea afisa "hai la tenis" intre ghilimele am utiizat pentru delimitarea stringului ghilimelele simple. Regula ghilimelor in stringuri este urmatoarea: daca incepi cu un tip de ghilimele, sa zicem ghilimele simple, apoi poti utiliza ghilimele simple, apoi iar ghilimele duble. Dupa cum poti observa, ghilimele duble, in exemplu de mai sus, sunt tratate de catre interpretor ca un caracter normal.

```
>>> print('Program de voie intre orele """')
Program de voie intre orele """
```

```
>>> print('Program de voie intre orele:' 9-17'')
  File "<input>", line 1
    print('Program de voie intre orele:' 9-17'')
                                         ^
SyntaxError: invalid syntax
```

Un exemplu negativ in utilizarea ghilimelelor este cel prezent in imaginea de mai sus.

Cum putem scrie un program pe mai multe linii?

Mai jos regăsim un exemplu în care caracterul slash împarte un citat pe mai multe linii în afara și în interiorul șirului de caractere.

```
>>> print("""
... Ana are mere,
... Violeta I le cere
... """)

Ana are mere,
Violeta I le cere
```

O alta modalitate de scriere pe linii multiple este utilizarea ghilimelelor triple. Un exemplu in acest send se poate vedea in imaginea de mai sus.

Cum mutam un sir de caractere spre dreapta? Cu ajutorul caracterului tab in mod obisnuit, dar utilizand limbajul Python, cu ajutorul \t..

De asemenea, pentru a anula caractere se foloseste \ denumit in mod obisnuit backspace.

```
>>> print("\")
...
```

Se poate observa ca, in exemplu de mai sus, \ are rolul de a anula ghilimeaua dubla pentru afisare a caracterului.

```
>>> print("\"")
"
```

Pentru afisarea unei ghilimele duble intr-un string cu delimitator ghilimele duble se poate utiliza caracterul backspace si ghilimea.

\n are rolul de a afisa o linie nou.

```
>>> print("Ana are \n mere")
Ana are
 mere
```

Dar ce se intampla daca folosim semnul plus si semnul destinat operatiei de inmultire la nivelul stringurilor?

Semnul plus are rol de concatenare la nivelul stringurilor, iar semnul operatiei de inmultire, in Python * va produce o repetitie a caracterelor cu care acesta este inmultit.

```
>>> print("Ana are \n mere")
Ana are
 mere
```

1.4. Lucrul cu numere

In următorul program vedem cum putem să utilizam numerele și să efectuam operații numerice.

Python poate realiza toate operațiile matematice elementare cu foarte mare ușurința. Acestea se pot apela în mod direct prin comenzi date ca in exmeplul de mai sus.

+ reprezinta operatia de adunare
    - reprezinta operatia de scadere
    - * reprezinta operatia de inmultire

      ** reprezinta ridicare unui numar la o putere, a ** b, unde a e baza si b este exponentul

      % reprezintta operatia modulo, aceasta returnand restul unei impartiri

Discutand despre numere, la nivelul limbajului Python, acesta pot fi de 3 tipuri: de tip integer, float sau complex. Daca rezultatul unei impartiri are rest si tipul este intreg, restul urmeaza sa fie ignorat. Daca dorim o împărțire de precizie va trebui să folosim numere float. Acestea sunt reprezentate de numere care au cel puțin o zecimală după virgulă. În acest mod python diferențiază numerele float de cele integer. Daca cel puțin unul din elementele unei împărțiri este float (numitorul sau numărătorul) rezultat, rezultatul va fi float. Acest aspect ne ajuta în programele în care numărătorul nu-l introducem noi, dar dorim ca rezultatul să fie float.

```
>>> 2/3
0
>>> 2/3.0
0.6666666666666666
>>> 2.0/3
0.6666666666666666
>>> 2.0/3.0
0.6666666666666666
>>>
```

Orice operație matematică am realiza în python, unde unul dintre numere este float, rezultatul va fi float.

1.5. Variabile

Ce reprezinta variabilele si care este scopul lor?

Variabilele reprezinta obiectele prin intermediul carora informatia pot sa fie manipulate, dar si stocate.

```
>>> s = "As manca:"
>>> s += "mere"
>>> print(s)
As manca:mere
```

Dupa cum se poate vedea in exemplul de mai sus, acestea se pot utiliza pentru manipularea sirurilor de caractere si pentru stocarea unui sir concatenat ce urmeaza sa fie afisat utilizatorului.

O recomandare este ca denumirile variabilelor sa nu se aleaga intamplator. Intr-un program mic acestea nu par importante, dar pe masura ce programul se dezvolta, se observa ca acestea detin un rol foarte important.

Regula de denumire a variabilelor: acestea poate sa fie formate din litere, numere si undercore "_", dar nu poate sa inceapa cu cifra. Variabilele care incep cu "_" au o insemnatate in OOP la care revenim curand.

Se numește metoda de manipulare orice element ce se apelează cu paranteze si este de forma:

ceva.metoda_de_manipulare()

Se numește atribut orice element ce se apelează fără paranteze si este de forma:

ceva.atribut