

INTRODUCERE ÎN LUMEA PYTHON

Introducere Python

GOOGLE

"PYTHON HAS BEEN AN IMPORTANT PART OF GOOGLE SINCE THE BEGINNING, AND REMAINS SO AS THE SYSTEM GROWS AND EVOLVES. TODAY DOZENS OF GOOGLE ENGINEERS USE PYTHON, AND WE'RE LOOKING FOR MORE PEOPLE WITH SKILLS IN THIS LANGUAGE."

PETER NORVIG, DIRECTOR OF SEARCH QUALITY,
[GOOGLE, INC.](#)

Python este un limbaj de programare puternic, dar ușor de folosit dezvoltat de Guido van Rossum, primul lansat peste doua decenii în urmă, în 1991. Python facilitează o scriere rapidă și poate fi folosit pentru și aplicații comerciale, fiind cross-platform (platform independent). Acest lucru permite rularea aceluiași program pe diferite sisteme de operare. Deci același script python poate fi rulat pe sisteme de operare precum Windows, DOS, Macintosh sau Linux.

Dar de ce am folosi python și nu alt limbaj de programare?

Unul dintre motive este ca Python este ușor de folosit. Provocarea unui limbaj de programare este formata din sintaxa limbajului. Cu cat sintaxa este mai aproape de codul mașină, cu atât este mai greu de învățat și de a găsi răspunsuri când comportamentul nu este tocmai cel scontat.

Spre exemplu, programarea în limbajul C presupune un efort imens pt. găsirea unei soluții la un comportament defectuos. Limbaje precum C, C++, C# sau Java sunt considerate limbaje de nivel înalt, mult mai aproape ca sintaxa de codul mașină decât de limbajul uman.

Dar Python are reguli clare și simple fiind foarte aproape de limbajul uman, declarându-se pe buna dreptate ca Python este definit ca "programare la viteza gândului".

Utilizarea facilă a Pythonului crește productivitatea pentru programatorii avansați. Timpul de scriere a unui program ce va avea același comportament în mai multe limbaje de programare arată de ce pythonul este mult mai preferat decât alte limbaje. Un program în python se scrie de trei până la cinci ori mai rapid decât în limbajul Java, și de cinci până la zece ori mai scurt față de limbajul C++. Există anumite statistici care laudă limbajul Python afirmând că un singur programator Python ar putea face aceeași muncă cât doi programatori în C++.

Cum este și normal, există domenii în care limbajele de nivel înalt, precum C sau Java nu pot fi dispensate, cum ar fi programarea embedded, unde viteza și spațiul de stocare mic limitează posibilitatea de a folosi python.

Python este un limbaj foarte puternic și are toate facilitățile ca, la finalul acestui curs, să puteți să construiți interfețe grafice. Este atât de puternic că este folosit de cei de la youtube pentru căutarea filmulețelor în imensa bază de date. Companii precum Google, Hewlett-Packard, IBM, Industrial Light + Magic, Microsoft, NASA, Red Hat, Verizon, Xerox și Yahoo! îl folosesc. Activision, Electronic Arts și Infogrames au publicat jocuri ce au incorporat scripturi Python. Acest limbaj a fost integrat în Microsoft Studio, astfel că acum în spatele unui program .NET poate sta un script python. Python poate fi integrat și în alte limbaje precum C, C++ sau Java.

Dacă ai cunoștințe deja în programare probabil ai auzit de object-oriented programming sau OOP pe scurt. Este un subiect foarte dezbătut, și în general este ce îți dorește un programator în curriculum vitae. OOP este un alt mod de a programa, de a găsi soluții la rezolvarea problemelor din programare. Presupune o abordare scalabilă de a programa. Nu este singura metodă, dar pentru proiectele mari este cea mai bună soluție.

Limbajele precum C#, Java sau Python sunt toate object-oriented. Dar Python are o îmbunătățire. În C# și Java, OOP nu este opțional. Acest lucru face ca programele scurte să fie nejustificat de complexe. De asemenea în aceste limbaje de programare o persoană care dorește să învețe limbajul de programare va primi destul de multe informații înainte de a face ceva semnificativ. În schimb Python are o abordare diferită. În Python utilizarea de tehnici OOP este opțională. Ai toată puterea OOP la dispoziție, dar o poți folosi doar când ai nevoie, neîncărcând nejustificat programele mici.

Python are o comunitate bine dezvoltată și asta datorată în special simplității limbajului de programare. Există multe site-uri, bloguri etc.:

Site-ul oficial este : <http://www.python.org>

Există și un Google newsgroup : <https://groups.google.com/d/forum/comp.lang.python>

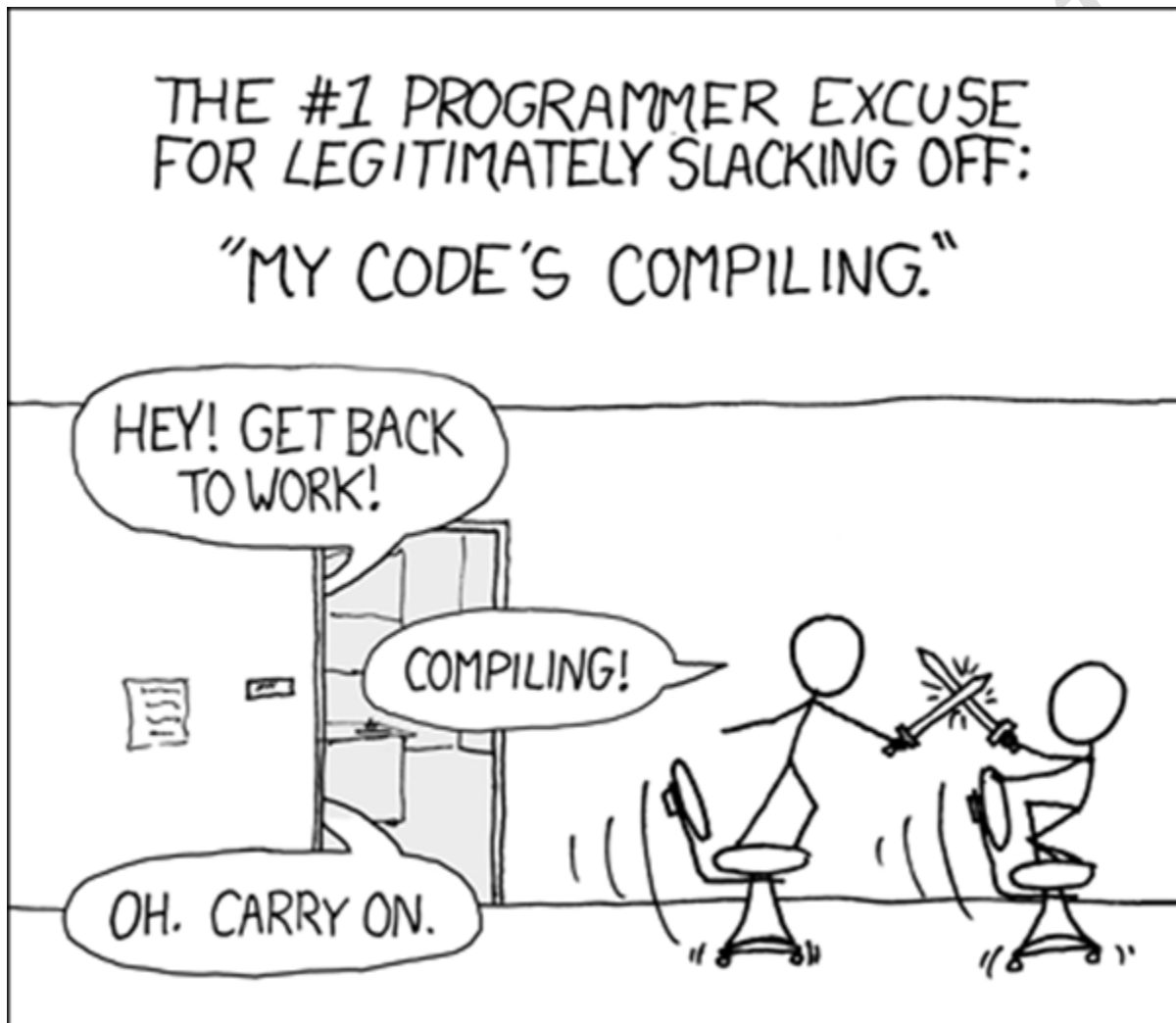
Python are de asemenea un mailing list numit "Python Tutor", un mod mai bun ca programatorii începători să pună întrebările care caută un răspuns:

<http://mail.python.org/mailman/listinfo/tutor>.

Un alt atu Python este că permite un debug ce include toate fișierele, funcțiile și clasele apelate până la eroarea aparută.

Python este considerat un limbaj interpretabil, spre deosebire de C care este un limbaj compilabil.

Un limbaj compilabil este un limbaj care trebuie să compileze fișierele sursă în fișiere care sunt foarte aproape de limbajul mașină. Din aceasta cauză, C este un limbaj foarte rapid. Din păcate timpul de compilare este câteodată foarte mare. Spre exemplu pentru un Router Avaya, compilarea fișierelor sursă într-o imagine executabilă pentru router din gama de top, după o operațiune de comentare a unei linii poate dura și trei ore...



În mod normal limbajele interpretate sunt de cele mai multe ori mai lente decât cele compilate. Totuși, precum Java, Python este și un limbaj "byte-compiled", rezultând o formă intermediară mai aproape de mașină. Acest aspect face ca pythonul să fie un limbaj rapid, dar să permită păstrarea tuturor avantajelor unui limbaj interpretabil.

Mai jos am adăugat câteva lucruri impresionante legate de Python, în speranță despre aplicabilitatea Python în diferite zone ale IT-ului:

- Filme animate (Industrial Light & Magic, Sony Pictures Imageworks, Disney, Pixar)
- Youtube folosește Python pentru căutarea între 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 financiară (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)
- Crearea de sisteme de apărare cu rachete (Lockheed Martin)
- Administrarea de liste de mail (Mailman)

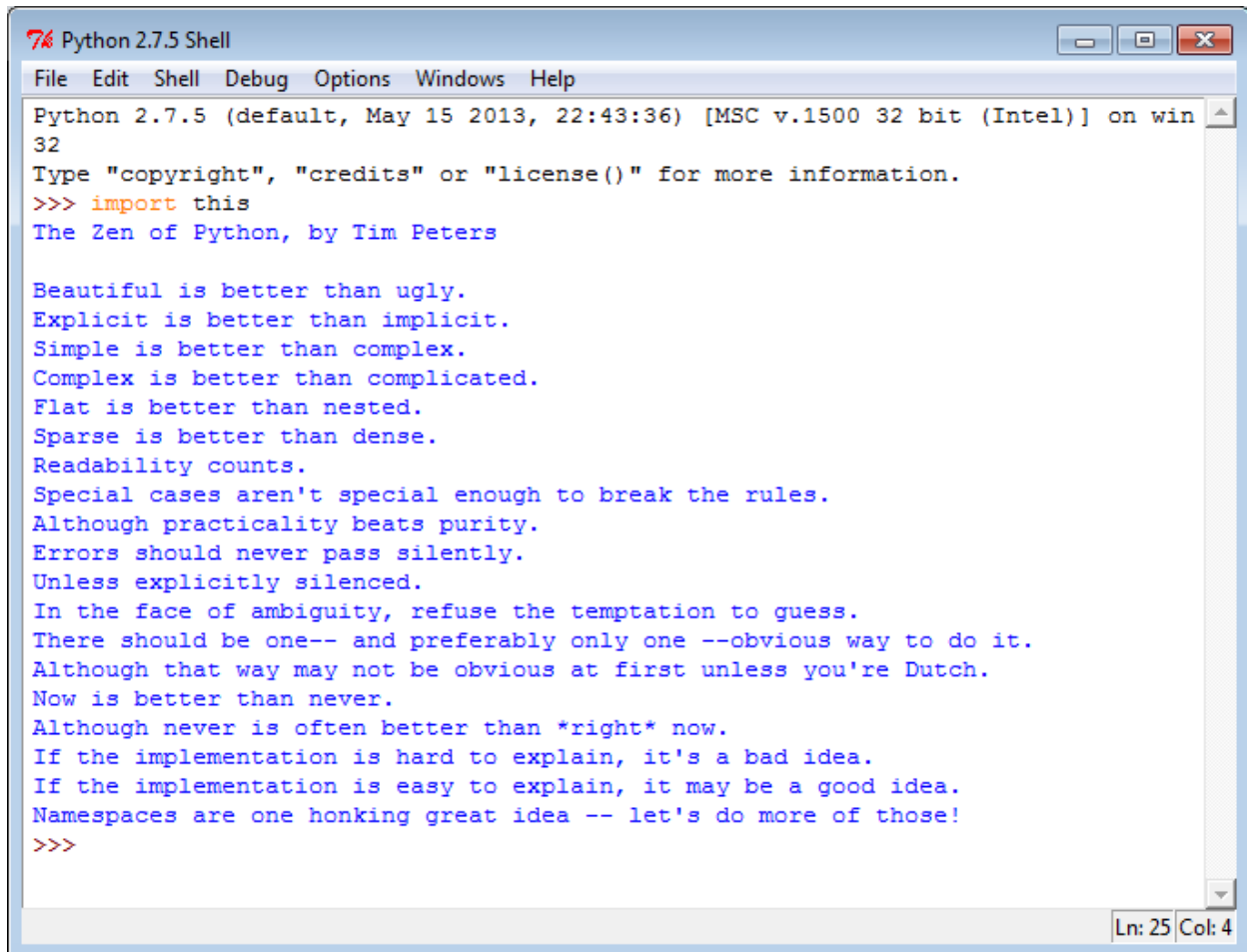
Aceste sunt doar câteva din domeniile de aplicabilitate a limbajului de programare Python. Aplicabilitatea se bazează pe module sau package-uri ce pot extinde funcționalitate standard a python-ului.

Alte păreri profesionale despre python puteți găsi pe pagina:

<http://www.python.org/about/quotes/>

În alt aspect interesant este filozofia Python scrisă de Python core developer, Tim Peters.

Prin comanda "import this" putem vedea o parte din această filozofie.



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 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!
>>>
```

Fig. 1

Instalare

Python este disponibil pe site-ul oficial dar și pe alte site-uri de profil:

<http://www.python.org>

După cum se poate vedea în fig. 1, Python se poate găsi sub diferite forme. La ora actuala exista doua versiuni. Versiunea 2.7 și versiunea 3.2.

Python pune la dispoziție adăugarea de funcționalitate în plus prin instalarea de module, module ce se găsesc în comunitatea python. Scopul este de a ușura munca programatorului.

Din păcate versiunea 3.2 vine cu mici schimbări incompatibile cu 2.7. Toate modulele adiționale au fost dezvoltate în versiuni mai vechi. Unele din aceste module au fost traduse, altele nu. Sunt module care nu au avut nevoie de o mapare, în funcție de

conținutul modulului. Exista un script numit 2to3 ce traduce codul din python 2.x in python 3.x

De asemenea, ar trebui să ținem cont că la apariția acestui limbaj de programare, toate mașinile ofereau o arhitectura pe 32 de biți. Acum situația s-a schimbat, mașinile ce au o arhitectura pe 32 de biți devenind o raritate. Din păcate, unele module au fost create doar pentru 32 de biți, neexistând o tranziție a acestora pe 64 de biți. Pentru o compatibilitate cat mai buna recomandam Python 2.7 pe 32 biți. Puteti accesa acest [link](#) pentru a descarca python.

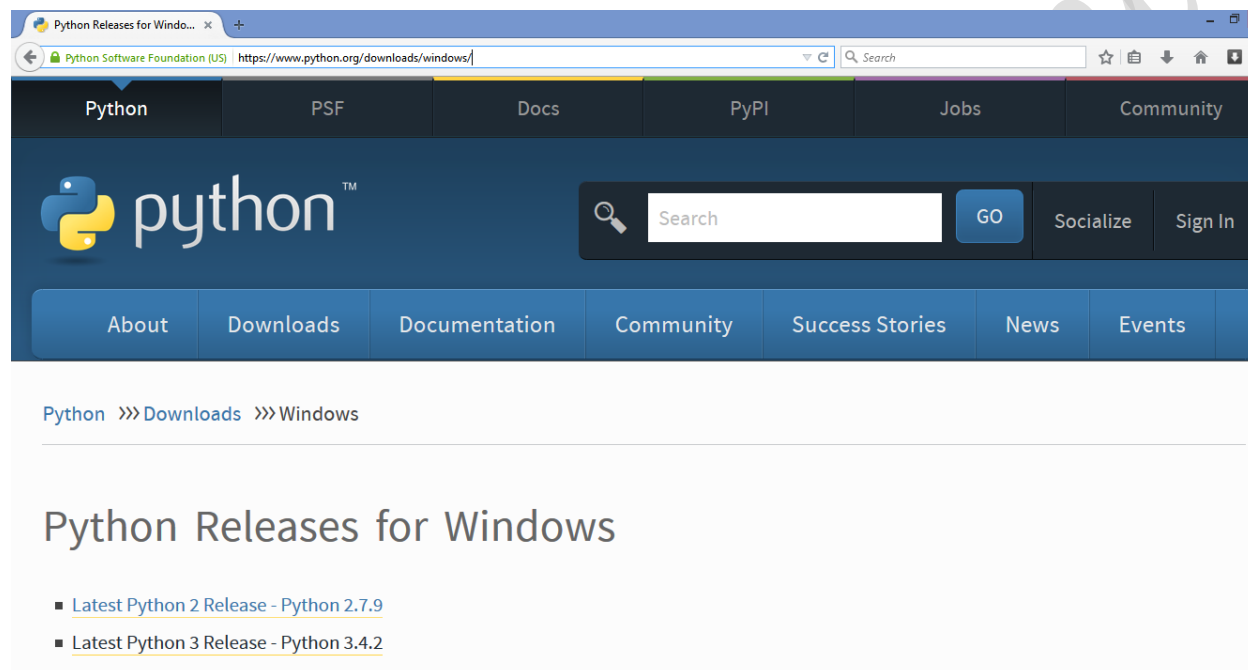


Fig. 2

Instalarea pe Windows presupune rularea executabilului pe care l-ați copiat local de pe site-ul python.org pe un sistem de operare Windows. În cazul care acceptați setările standard, python se va instala în directorul C:/Python27/



Fig. 3

Odată instalat Python include în interiorul fișierelor instalate și un program de interpretare numit IDLE. Acesta are doua moduri de funcționare: modul Interactiv și modul Script.

Daca apelam secțiunea de descărcări alternative a site-ului (<https://www.python.org/download/alternatives/>) putem găsi alte site-uri care lucrează pentru a aduce un surplus de funcționalitate Python-ului standard, deci care conține module adiționale, care în mod normal ar trebui să fie instalate după ce setup-ul python-ului a fost rulat.

Pe acest site găsim și instalări alternative cum ar fi [ActiveState-ActivePython](#). Active State este o companie ce pune la dispoziție package-uri (limbaje de programare cu librării/module incluse) pentru diverse limbaje de programare, printre altele și pentru python. Este o companie ce încearcă să realizeze un profit de pe urma softurilor gratuite prin adăugarea de funcționalitate fie gratuită, fie realizată de inginerii companiei. ActivePython include modulele lui Mark Hammond pywin32 dar și module de GUI precum PyQt sau wxPython, module pentru baza de date etc. Din păcate acestea se regăsesc doar în variantele cu plată, cu o remunerație destul de consistentă – varianta Business regăsindu-se la 999\$ pe an pentru o instanță OS.

Python se poate instala pe multe sisteme de operare precum:

- Python pentru AIX

Se va utiliza fie Python 2.1 pentru AIX ,fie o varianta cu plată de la ActivePython.

- Python pentru AROS

Se poate utiliza PyAROS pentru Amiga Research OS.

- Tim Ocock mentine AmigaPython..
- Python pentru AS/400 (OS/400)

Puteti apela <http://www.iseriespython.com> portat de Per Gummedal.

- Python pentru BeOS

Ultima varianta este BeBits.

- Python pentru MorphOS
- Python pentru MS-DOS
- Python pentru OS/2

Aceasta varianta este menținută de Andrew MacIntyre

1. Python pentru OS/390 si z/OS

Puteți apela <http://www.teaser.fr/~jymengant/mvpython/mvsPythonPort.html> pentru mai multe detalii

2. Python pentru Palm OS

Puteți apela <http://pippy.sourceforge.net/> pentru mai multe detalii.

3. Python pentru PlayStation

Erwin Coumans a portat o parte de Python către Sony PlayStation 2.

4. Python pentru Psion

Exista o portare pentru EPOC (Psion) Python port (v2.1) pe site-ul <http://sourceforge.net/projects/epocpython/> pentru mai multe detalii.

5. Python pentru QNX

Puteți apela <http://sourceforge.net/projects/pyqnx> pentru mai multe detalii.

6. Python pentru (Cea fost inainte Acorn) RISC OS

Citeste te rog RISCOS/README în directorul sursa Python 2.x . Suportul pentru RISC OS a fost eliminat în Python 3.x.

7. Python pentru Symbian 60/ Nokia

Nokia a portat Python 2 către platforma smartphone Symbian OS 60. Puteți apela <http://sourceforge.net/projects/pys60/> pentru mai multe detalii. Cel mai de succes proiect este QT.

8. Python pentru Solaris

Poți achiziționa ActivePython (versiuni comerciale – contra cost) . Sunfreeware.com este o pagina web care are o varietate de versiuni Python pentru o varietate de versiuni Solaris. Acestea utilizează pachete de instalare standard Sun cu extensia pkgadd.

9. Python pentru Windows CE sau Pocket PC

10. Python for HP-UX

11. Python for Linux

12. Python pentru Android

Puteți apela <http://code.google.com/p/python-for-android/> pentru mai multe detalii.

13. Python pentru MAC

Puteți apela <http://homepages.cwi.nl/~jack/macpython/index.html> pentru mai multe detalii.

Pentru mai multe detalii privind suportul oferit de Python în a fi rulat pe diferite sisteme de operare accesați acest link <https://www.python.org/download/other/>

În concluzie python poate fi utilizat pe orice sistem de operare unde există un compilator pentru a compila sursele și unde este cazul portarea codului. Prin urmare unde există interes financiar în a dezvolta acesta portare Python a fost deja dezvoltat, după cum se poate vedea și mai sus.

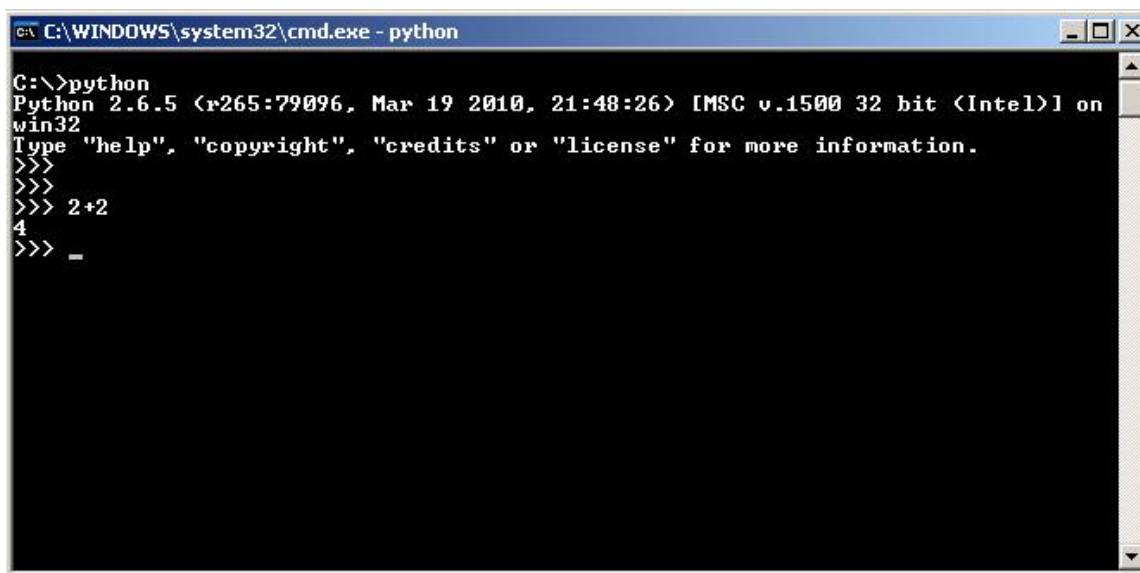
În ceea ce privește sistemele de operare uzuale încă se dorește un limbaj de programare cross-platform. Ceea ce înseamnă că multe din proiecte cum ar fi PyQt sau Kivy pot oferi un program ce poate fi rulat pe mai multe platforme. PyQt este un proiect prin care se poate realiza construcția de interfețe grafice, poate fi rulat pe cele cinci sisteme de operare cele mai uzuale din lume (Windows, Linux, Symbian, Apple - iOS, Android).

În ceea ce privește Linux, multe din distribuții vin cu Python deja instalat. Unele din cele mai uzuale distribuții, cum ar fi Ubuntu și Fedora vin cu Python 2.7 preinstalat pentru o compatibilitate maximă.

Pentru a testa dacă există instalată o versiune Python în distribuția Linux pe care o aveți instalată, tastați în cadrul unei ferestre terminal cuvântul cheie “python” apoi apăsați enter.

După cum se poate vedea putem deja să lucrăm cu python-ul, calculând spre exemplu o adunare a două numere întregi. Acest lucru este valabil și în Windows.

Pentru a ieși din acest mod de lucru trebuie să tastați "exit()" apoi să apăsați enter.



```
C:\>python
Python 2.6.5 <r265:79096, Mar 19 2010, 21:48:26> [MSC v.1500 32 bit <Intel>] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>> 2+2
4
>>> _
```

Fig. 4

În cazul în care doriți să aveți mai multe instalări de Python în sistemul de operare Windows va trebui să modificați sau adăugați în variabilele de sistem și o variabilă numită „Path” ce indică unde este directorul sursă Python.

Iată o procedură cum puteți face asta în cazul în care compania a achiziționat un ActivePython cu module speciale, adaptate la nevoile companiei.

1. Click dreapta pe My Computer apoi alegem din meniu Properties
2. Selectăm tabul Advanced
3. Selectăm butonul Environment Variables (poziționat în partea de jos stânga)
4. În secțiunea System Variables cauți "Path" și selectezi această variabilă
5. Apeși pe Edit și adaugi ca variabilă " C:\Python27_active\; "

Mai jos se poate regăsi în Figura 5 un exemplu de modificare a variabilei de cale pentru ActivePython.

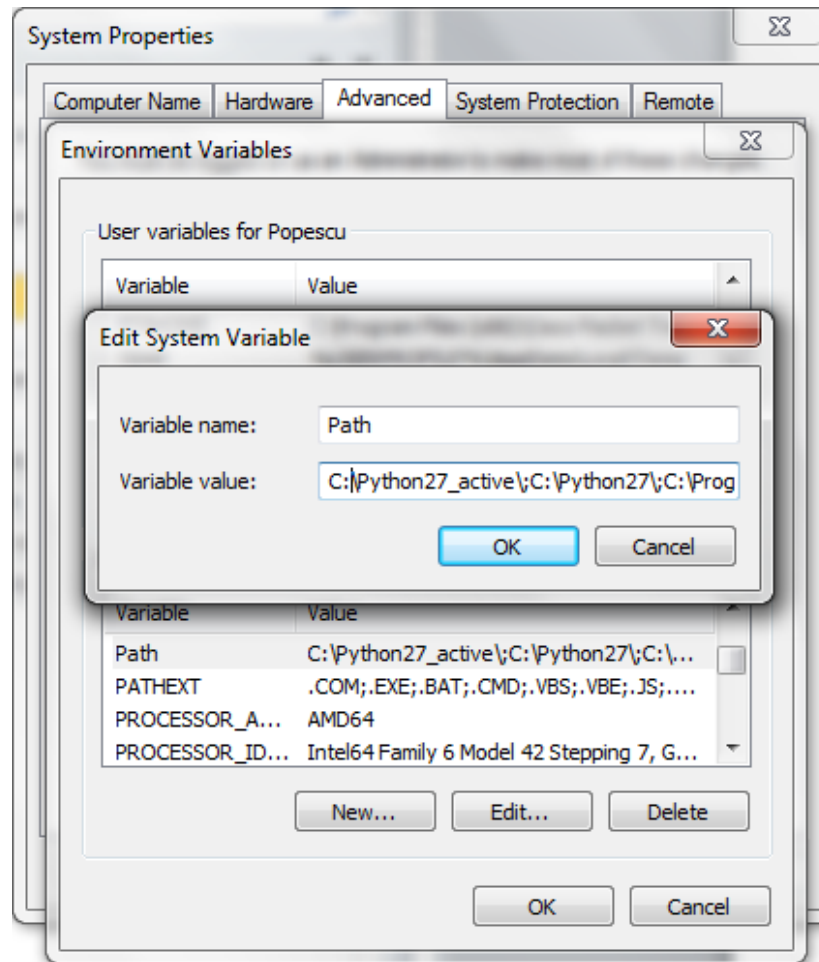


Fig. 5

Tipuri de fișiere

În directorul sursa Python (exemplu C:\Python27) se regăsesc două fișiere executabile:

- ➔ python.exe – Programul Python ce rulează în modul consola. Acest lucru determină apariția unei console pentru orice rulare a unui script python. De multe ori acest lucru este benefic, mai ales dacă scriptul nu creează interfețe grafice.
- ➔ pythonw.exe – Programul Python ce rulează în modul non-consola. Are rolul de evita ferestrele urate de tip DOS când nu lucrăm cu input/output-ul standard. Acest executabil este des utilizat la construirea de interfețe grafice sau la construirea de daemon-uri/servicii.

Când Python a fost instalat, a fost asociat cu anumite tipuri de fișiere, ce vor fi văzute în Windows ca fiind executabile. Prin urmare dacă dam dublu click pe unul din aceste fișiere din cadrul Explorer-ului va avea ca urmare rularea acelui fișier. Aceste fișiere au extensia py, .pyw, .pyc, .pyo.

Python este "interpretat" în aceeași manieră ca și Java: codul sursa python ce se găsește în fișierele *.py sau *.pyw sunt automat compilate (translatate) de interpretor într-o formă intermediară și independentă de platformă, care ulterior este executată de Python virtual machine. Python virtual machine este sistemul de execuție python sau motorul de rulare al limbajului. Python virtual machine mai poartă și numele de interpretor. Traducerea în byte code se realizează când dorim să rulăm. Byte code nu este cod mașină și este în cele din urmă executat prin platforme de tip Python virtual machine, nu direct de către hardware-ul computerului.

O modalitate importantă de accelerare a timpului de pornire pentru programele Python ce utilizează module este fișierul pyc. În cazul în care avem un fișier numit spam.pyc ce există în directorul în care se află spam.py, fișierul spam.pyc reprezintă o versiune deja "octet-compilată" (already-"byte-compiled") a modulului spam. Timpul de modificare al versiunii de spam.py folosit pentru a crea spam.pyc este înregistrat în spam.pyc. În cazul în care timpul înregistrat în fișierul *.pyc nu se potrivește cu timpul ultimei modificări a fișierului *.py, fișierul pyc este ignorat și apoi rescris. Deci *pyc este creat automat la prima rulare a programului și se actualizează fără ca noi să intervenim în acest proces. Trebuie amintit și faptul că fișierul cu extensia *.pyo este un fișier pyc optimizat. Deoarece *.pyc este optimizat pentru o rulare cât mai eficientă într-un mod automat, optimizarea manuală nu-și are rostul decât în cazul în care dorim să realizăm fișiere în C.

Un program nu rulează mai repede atunci când este citit dintr-un fișier ".pyc" sau ".pyo" decât atunci când este citit dintr-un fișier ".py". Singurul lucru care este mai rapid când folosim ".pyc" sau ".pyo" este viteza de încărcare a fișierelor din module.

Ar trebui să discutăm un pic despre diferența dintre *.py și *.pyw. În general vom salva un fișier sursă ca având extensia pyw în loc de py când dorim să apeleze pythonw.exe. În urma acestei modificări nu vom vedea decât elementele grafice, consola fiind suprimată. Această facilități este des întâlnită în interfețele grafice sau la programe de tip server.

Putem întâlni mai rar fișierul cu extensia ".pyd", fișier ce face parte tot din suita de fișiere python. Acest tip de fișier este de obicei un fișier de tip librărie dinamică Windows (având extensia .dll - Dynamic-link library), fiind redenumit într-un fișier cu extensia .pyd. În primul rând ar trebui să știm că un fișier dll poate cuprinde resurse, executabile scripturi și drivere (prin urmare orice tip de fișier). Aceste fișiere sunt stocate într-un dll pentru a modulariza codul pentru o eventuală reutilizare, eficientizarea memoriei și reducerea spațiului de pe hard disk. Prin urmare fișierele de tip dll oferă o încărcare mai rapidă, rulează mai rapid, și ocupă mai puțin spațiu pe disc.

Mai multe detalii despre fișierele de tip dll puteți găsi pe site-ul Microsoft:

<http://support.microsoft.com/kb/815065>

Am putea să folosim aceste fișiere care sunt deja implementate în python. Bineînțeles că putem utiliza direct fișierul cu extensia dll, dar majoritatea fișierelor dll aparțin

dezvoltatorilor de aplicații contra cost. Un exemplu bun ar fi crearea unui program de arhivare cu diferite extensii. Python vine cu anumite module de arhivare, dar recunoașterea fișierelor arhivate sub extensia 7z este mai dificilă. Aici se poate utiliza redenumirea fișierului „7z.dll” în fișierul „7z.pyd” pentru a utiliza resursele dll-ului în programul python.

Tool-uri pentru Python

Pentru a dezvolta programe Python putem apela la două mari soluții. Aceste tool-uri se numesc IDE (Integrated Development Environment) și reprezintă un set de aplicații ce formează o platformă cu scopul de a ajuta programatorul să dezvolte aplicații Python. Primul IDE este instalat cu Python2.7.x. și se numește IDLE. Cel de-al doilea IDE se numește Eclipse și este un program dezvoltat de un consorțiu de firme.

IDLE este un IDE ce poate fi folosit în două moduri: modul Interactiv și modul Script.

Modul Interactiv poate fi apelat prin deschiderea programului IDLE. Este cel mai rapid mod de a studia și testa o parte de cod. Acest mod ne oferă posibilitatea de a interacționa interactiv cu limbajul de programare deoarece orice comandă apelezi el o va rula imediat.

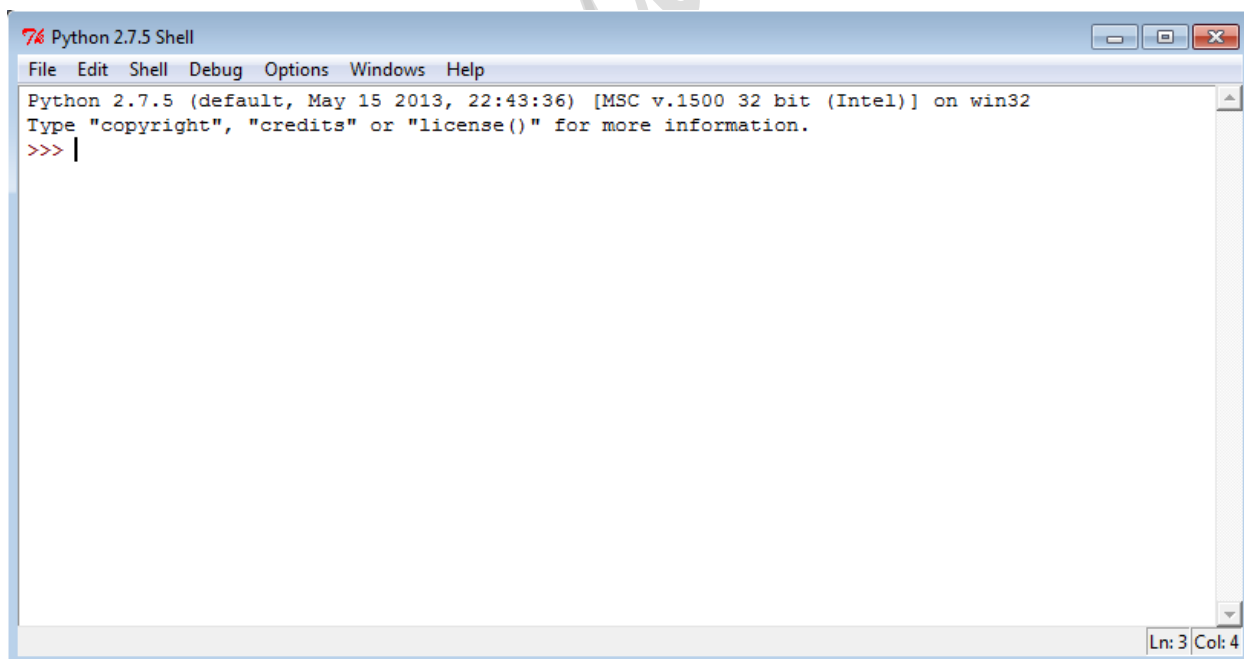


Fig. 6

Putem să realizăm același lucru ca și în consola, având în plus suport pentru debug și un meniu grafic cu toate facilitățile de care avem nevoie. Aceasta fereastră se mai numește și Python Shell. La command prompt (>>>) introdu următoarea secvență:

```
print "Salut Python! Salut PRIETENI ! "
```

Interpretorul ne returnează <<Salut Python! Salut PRIETENI ! >>. Acesta este primul program, Probabil ați ghicit ce realizează aceasta comandă, și anume va afișa pe ecran mesajul :Salut Python! Salut PRIETENI !

Din punct de vedere al limbajului programatorilor aceasta linie se numește statement și este o instrucțiune completă. Aceasta instrucțiune este formată din două părți. Prima parte “print” este o comandă. Spune interpretorului să facă o acțiune și ce acțiune. Python este sensibil la diferențele de litere mari sau mici. Deci “print” e diferit de “Print” sau “PRINT”. Se poate vedea în Fig. 7 diferența dintre ele.

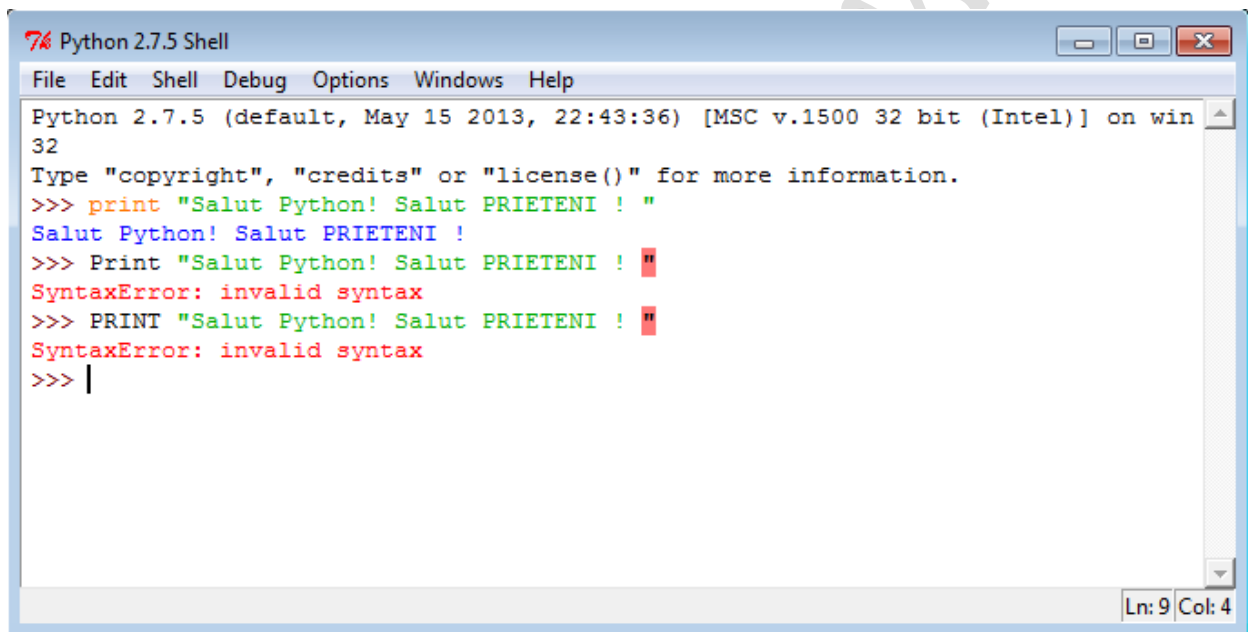


Fig. 7

Putem vedea că Python IDLE recunoaște cu ușurință cuvintele cheie cum ar fi comenzile; prin urmare în timp ce le scriem sintaxa va schimba culoarea în portocaliu.

Expresiile cum ar fi acest șir de caractere, pe care noi l-am printat anterior, sunt afișate în verde.

Output-ul returnat de interpretor se regăsește în culoarea albastră. Încet și cu practica codurile culorilor vor fi învățate fără să vă dați seama.

SyntaxError este un mesaj al interpretorului care ne indica ca sintaxa nu este corecta. Acest mesaj se regăsește afișat în culoarea roșie.

De reținut ca Python are un interpretor foarte puternic, astfel ca în cazul unei erori va indica linia și eroarea. În cazul în care eroarea apare sub diferite fișiere ce conțin apelări multiple în acestea, va arata tot parcursul apitelor în fiecare fișier pana la eroare. Acest aspect este superior altor limbaje care afișează doar eroarea finală.

Să vedem cum putem utiliza programarea IDLE în script mode.

Utilizarea modului interactiv returnează output imediat si nu putem salva programul nostru. Acest mod este foarte bun, mai ales când înveți lucruri noi sau încerci o parte de cod care nu-i înțelegi comportamentul. Dar acest mod nu a fost conceput pentru crearea de programe unde poți salva și rula ulterior. Modul IDLE script oferă scrierea, editarea încărcarea și salvarea programului tău. Este ca Notepad sau Word. Dar de ce nu folosim notepad pt. a edita *.py? Deoarece nu vom avea culorile ce ne ajuta să recunoaștem codul, nu vom putea rula in mod direct etc.

Poți deschide o fereastră în script mode din interactive mode , apelând din File>New Window.

Daca scriem în interiorul ferestrei o sintaxa de print, vom putea rula acest program din Run>Run Module. Pentru că acest program nu este salvat încă va trebui sa-l salvăm înainte de a putea rula. Fig. 8 și 9 va vor îndruma cum să executați corect rularea programului.

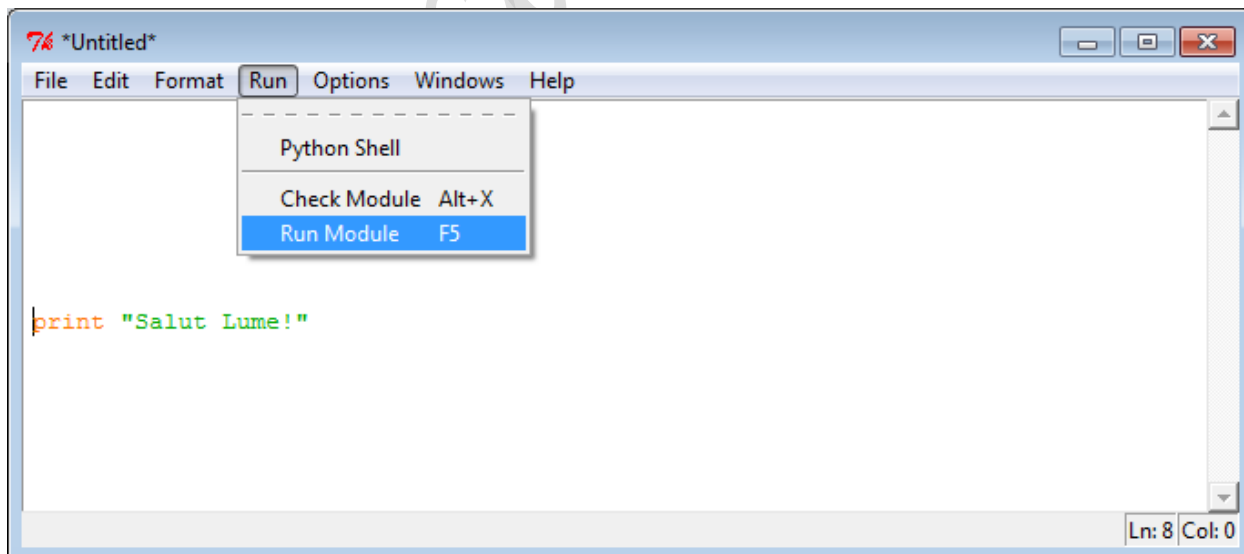
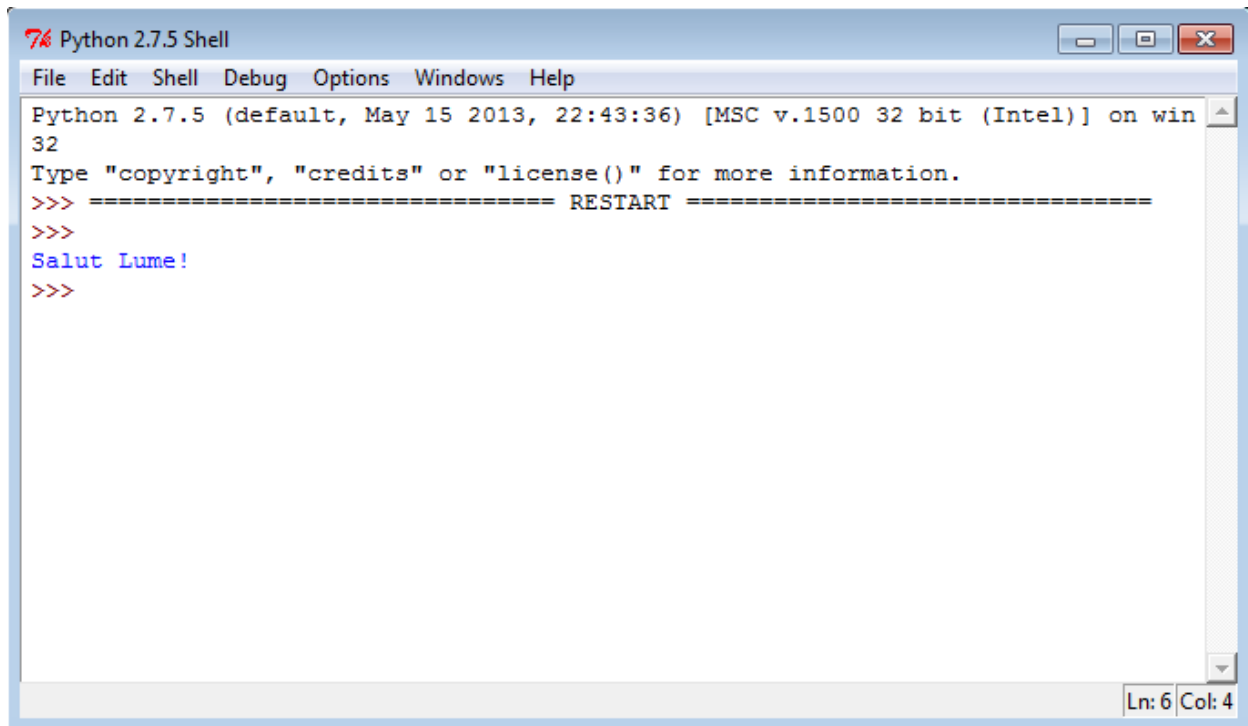


Fig. 8

Putem observa ca liniile libere nu afectează în nici un mod rularea programului.



The screenshot shows a Python 2.7.5 Shell window. The title bar reads "Python 2.7.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area displays the following content:

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Salut Lume!
>>>
```

The status bar at the bottom right indicates "Ln: 6 Col: 4".

Fig. 9

Avem posibilitatea de a scrie comentarii în programul nostru, comentarii care nu au nici un efect în rularea acelui program. Aceste comentarii au rolul de a ne îndruma, de a adăuga informații utile sau de a elimina funcționalitate prin comentarea anumitor linii.

Poate comentariile nu par un lucru util pentru programele mici scrise de o singura persoana, dar și în acest caz după câteva luni va fi destul de greu să va aduceți aminte cum ați conceput acel program. Sa ne gândim un pic la programele scrise și menținute de mai mulți dezvoltatori fiecare cu experiența și obiceiurile lui. Conform unui studiu aproximativ 70 % din timpul alocat programării este alocat pentru menținerea codului. Prin urmare orice comentariu este foarte util.

O linie comentata începe cu caracterul # . Putem să folosim caracterul # și la jumătatea liniei comentând o parte din aceasta linie.

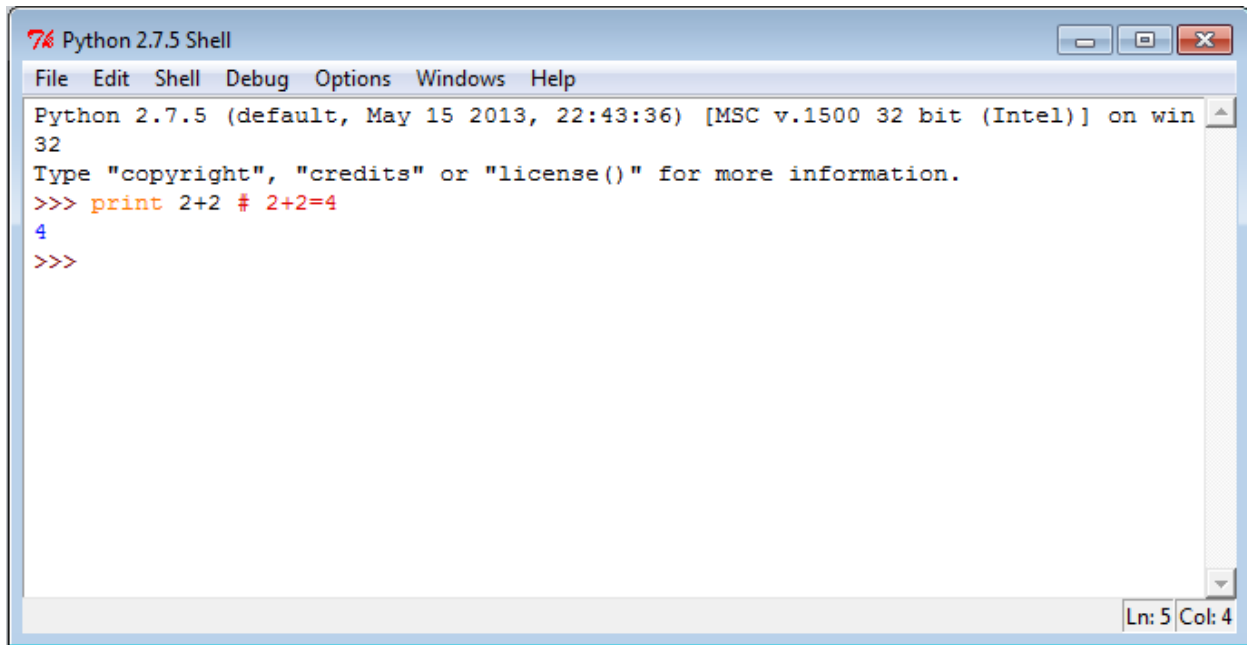


Fig. 10

Exista un cod de buna conduita în ceea ce privește partea superioara a unui curs. Astfel primele trei linii ar trebui să reprezinte informații despre program, ce rol are acest fișier în cadrul programului, cine a lucrat la acest fișier, versiune și data ultimei editări. Aceasta este doar o recomandare. Interpretorul nu va returna eroare daca nu intalneste primele trei linii comentate.

```
# Salut Prieteni
# Demonstreaza comanda print
# Ion Studentul - 12/09/13 - vers 6
```

În cazul în care rulăm prin dublu click acest program, putem vedea doar o fereastră neagră pentru o fracțiune de secundă, și asta datorita faptului că interpretorul Python va rula codul, apoi va ieși.

Pentru a putea vedea și rularea acestui cod vom adăuga la final o linie cu statut special.

Acesta linie se regăsește în general în toate programele python de tip non grafic și are scopul de a opri rularea pana la apăsarea tastei enter.

```
raw_input("Apasa <Enter> pentru a iesi.")
```

În acest mod putem vedea ce a rulat pana la ultima linie. Dupa ce utilizatorul apasa enter , programul se va incheia.

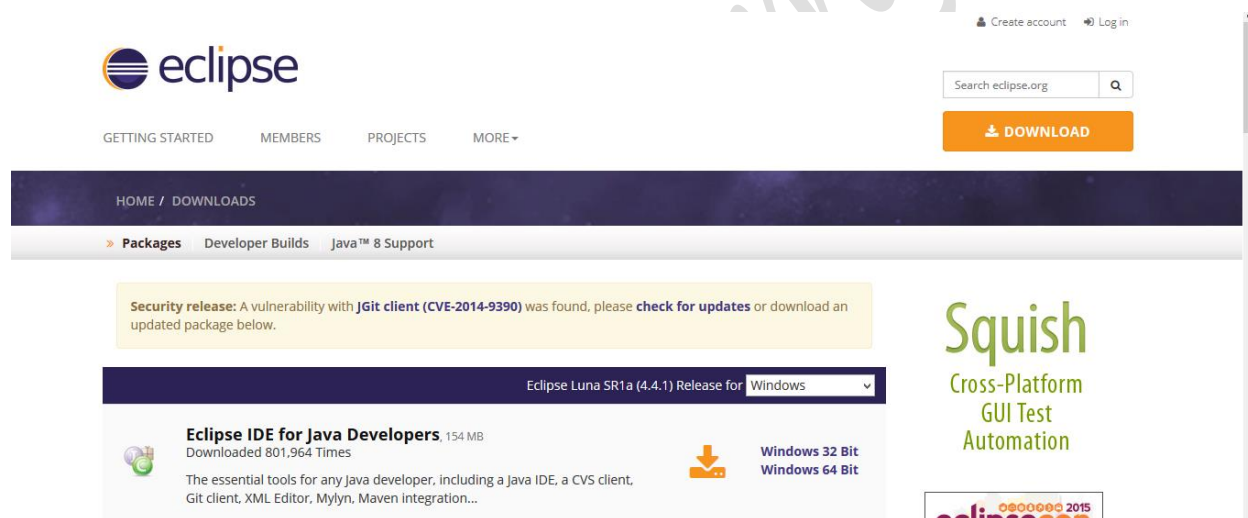
Eclipse este un mediu de dezvoltare open-source scris preponderent în Java. Acesta poate fi folosit pentru a dezvolta aplicații Java, C, C#, Python, TCL și, prin intermediul unor plug-in-uri, în alte limbaje, cum ar fi C, C++, COBOL, Python, Perl, și PHP. De dezvoltarea să se ocupe Fundația Eclipse : <http://www.eclipse.org/>

Acesta are un debugger incorporat foarte bun și vine cu tot felul de posibilități de a modifica și personaliza interfața utilizatorului. Din acest motiv devine foarte ușor de folosit și este disponibil pentru mai multe tipuri de sisteme de operare. Eclipse este programul recomandat pentru curs, dar nu este obligatoriu. Astfel se poate opta pe care varianta o considerați mai ușoară.

În următoarea secțiune vom vedea cum se instalează și se setează Eclipse pentru a dezvolta programe Python.

Copierea Eclipse se poate face accesând acest link:

<http://www.eclipse.org/downloads/>



Alegeți ca variantă standard (Eclipse IDE for Java developers) pentru sistemul de operare de 32 de biți.

Dezarhivați fișierul într-o locație cunoscută cum ar fi C:\Program Files (x86)\Eclipse

Deschideți executabilul eclipse.exe (puteți să faceți un shortcut către desktop pentru a fi mai ușor de apelat).

După deschiderea programului acesta cere o cale către workspace, directorul unde va salva toată munca. De preferat să alegeți o cale pe o partiție separată de cea a sistemului de operare pt. a vă putea salva munca în caz de virusare sau corupere a sistemului de operare.



Fig. 11

Odată instalat avem un Eclipse pe care nu l-am personalizat pentru python. Pentru a instala programele de care avem nevoie în python apelați din meniu: “Help>Install new software”. În Fig. 12 se poate vedea fereastra ce se deschide apelând această opțiune.

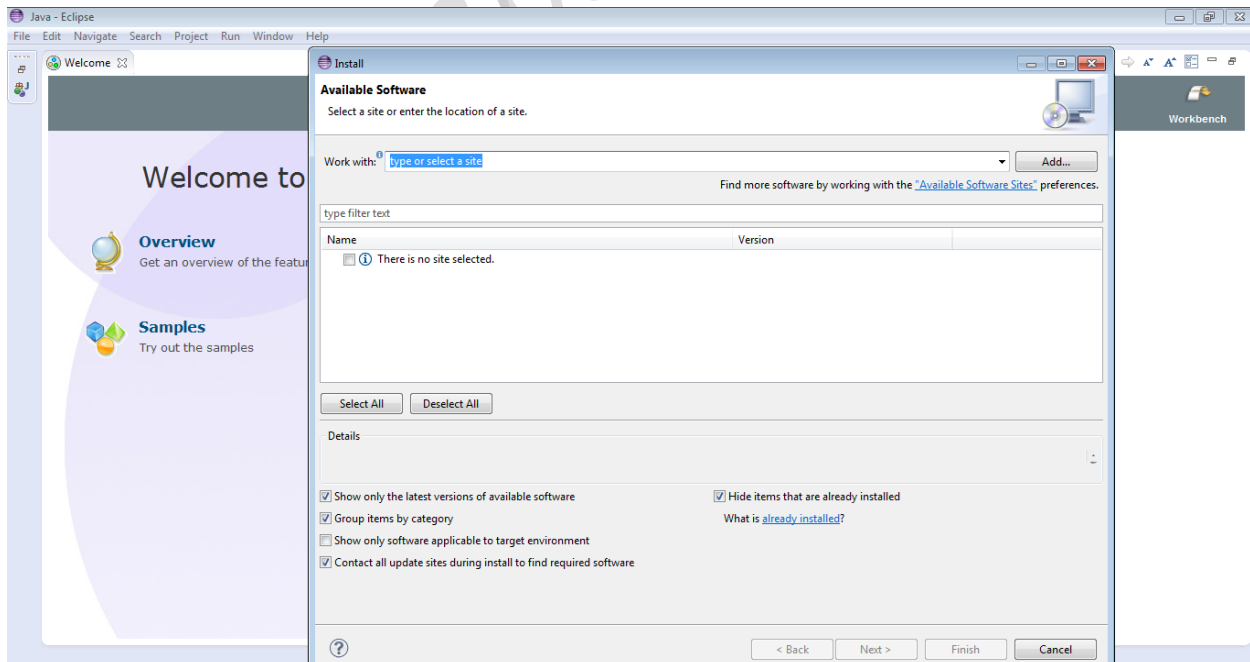


Fig. 12

În cele ce urmează vom instala o librărie PyDev care devine disponibilă dacă accesăm prin intermediul Eclipse site-ul:

<http://pydev.org/updates>

Pentru a adăuga un site (numit repository), va trebui să dăm click pe butonul de Add, apoi să completăm site-ul <http://pydev.org/updates> la secțiunea site și un nume sugestiv. Va trebui să alegem acest repository nou format din lista (pick-list-ul) pe care o avem la dispoziție în stanga butonului Add (triunghiul cu vârful în jos). În câmpul “type test here” putem să scriem PyDev. Vom avea la dispoziție două opțiuni: PyDev for Eclipse și PyDev MyLyn Integration. Le vom selecta pe amândouă așa cum se poate vedea și în Fig. 13.

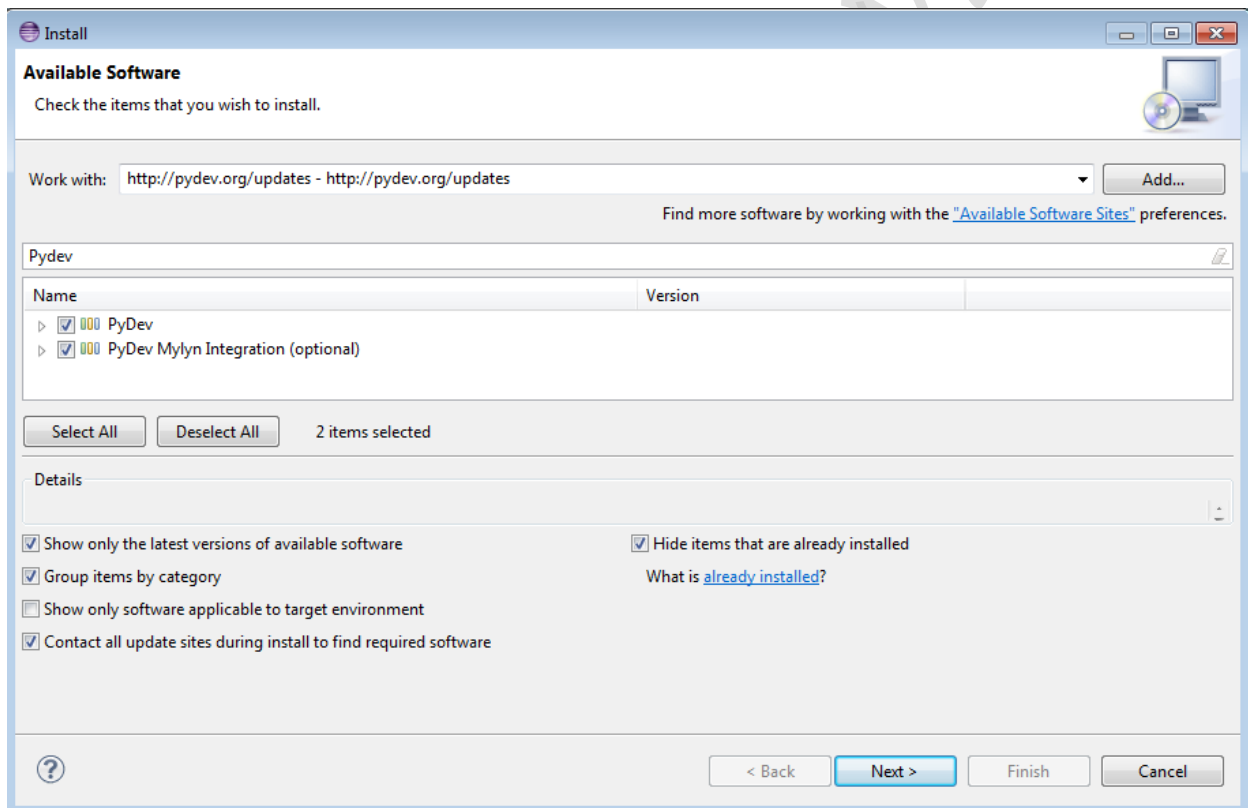


Fig. 13

Vom da click pe Next urmând instrucțiunile de instalare și acceptând condițiile în cazul în care suntem de acord. Atenție! În cadrul instalării există și o atenționare (warning) pe

care va trebui sa o acceptați pentru ca instalarea să fie făcută cu succes. După instalare, se va cere restartarea aplicației.

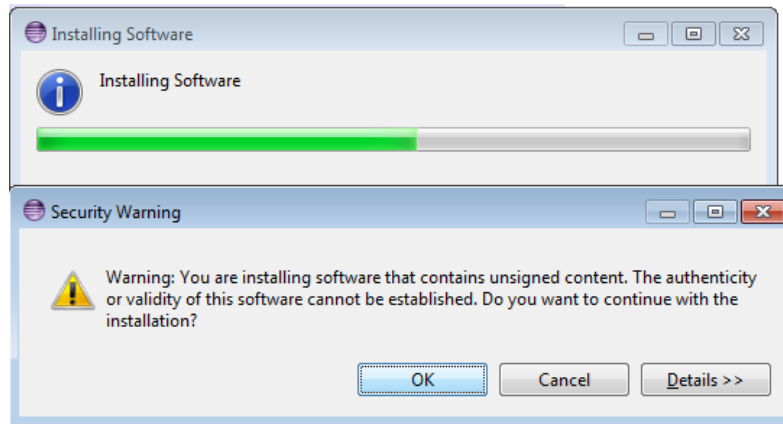


Fig. 14

După restart, vom putea crea proiecte Python. Crearea unui proiect se poate face apelând din meniul: File> New>Project.

Vom alege să cream un proiect de tipul PyDev așa cum este prezentat și în Fig. 15

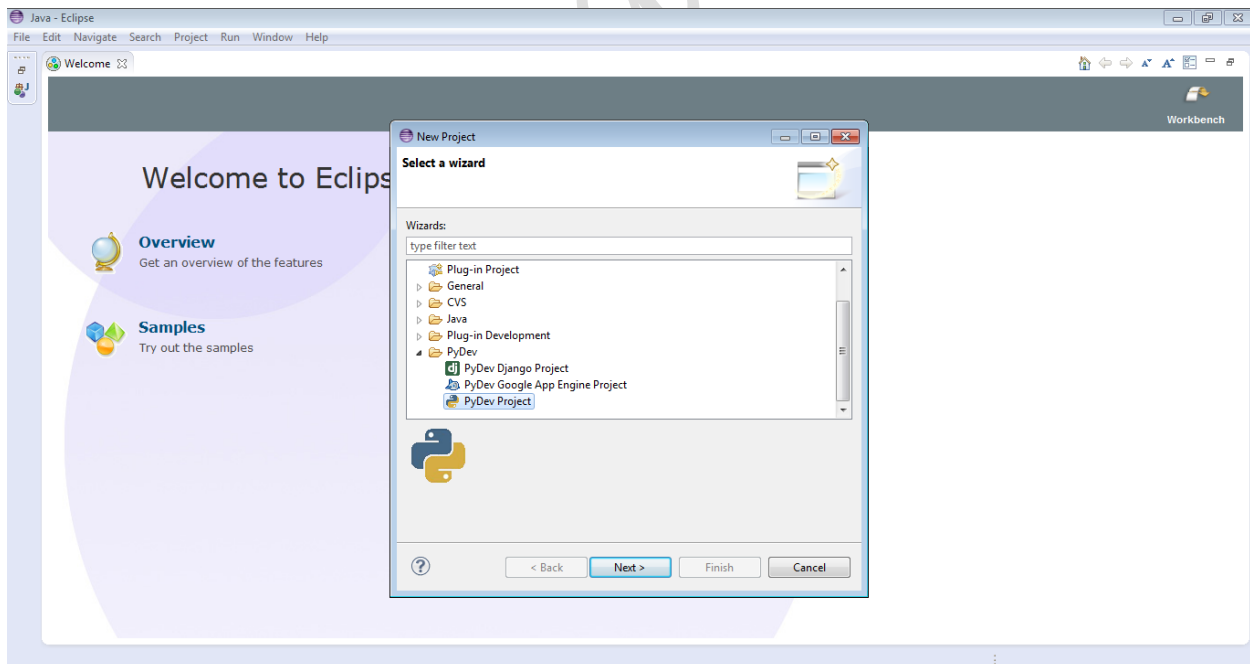


Fig. 15

După apelarea butonului de Next va trebui să facem anumite alegeri cu privire la numele proiectului, directorul rădăcină al proiectului, tipul de proiect(python, jython, IronPython), versiunea de Python (recomandat 2.7) și modul cum este atașat directorul proiectului la directorul rădăcină al proiectului. Jython este un modul de python care permite combinarea de python cu Java, iar IronPython este un modul de Python care permite dezvoltarea de Python în Visual Studio și combinarea cu .Net și C#. Pentru început, vom face alegerile ca în Fig. 16

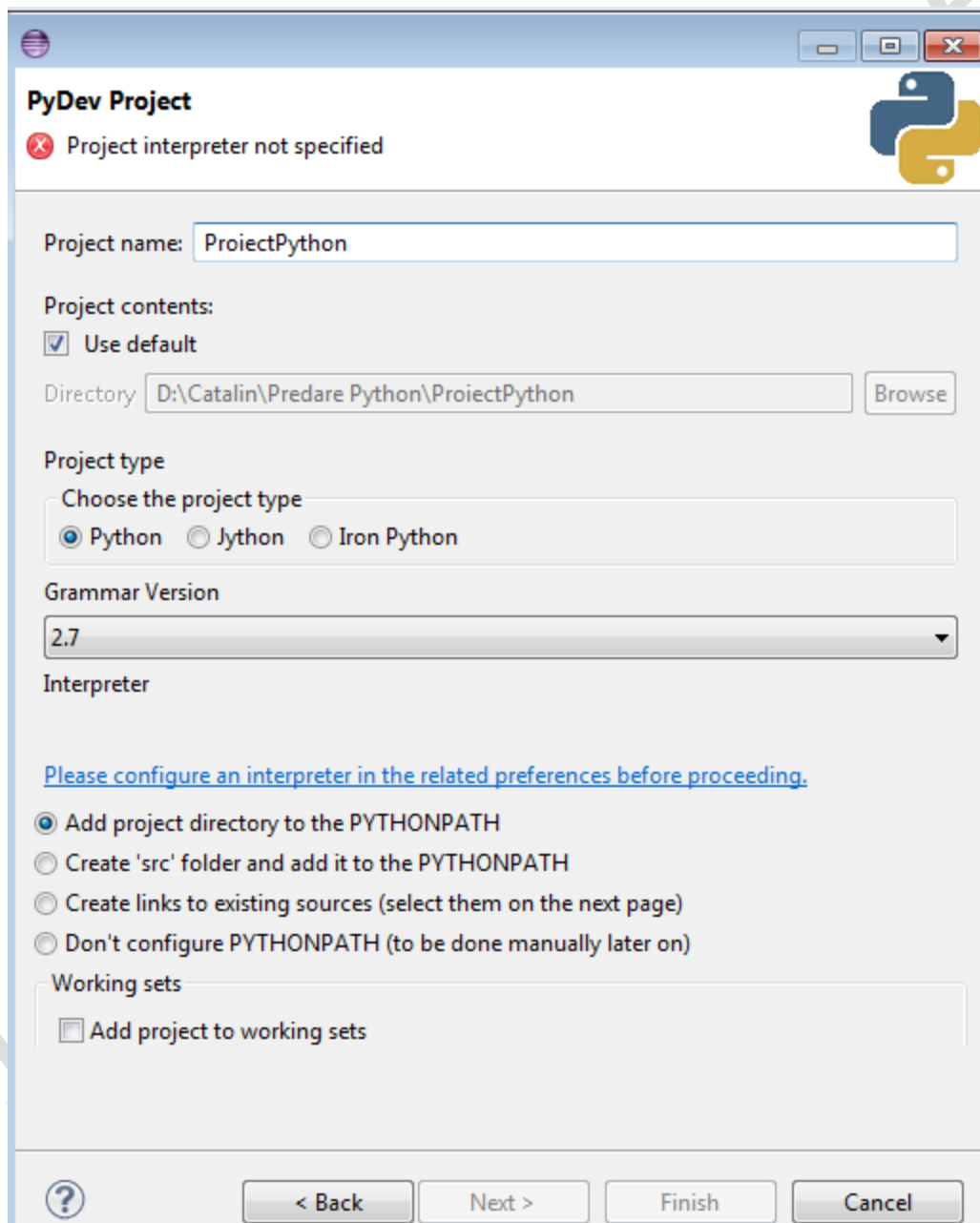


Fig. 16

Observăm ca butonul de Next sau Finish nu este disponibil, iar în partea superioară a ferestrei vedem și motivul: Project interpreter not specified. Acest warning se referă la verificarea și alegerea unei versiuni de python din cele instalate. Pentru a rezolva acest impediment, va trebui să apelăm link-ul albastru ce se găsește la mijlocul ferestrei:

[Please configure an interpreter in the related preferences before proceeding](#)

Alegem autoconfig și vom da click ok la căutarea și adăugarea interpretorului găsit.

După setarea interpretorului python vedem că este disponibil și butonul de Finish. Apăsând butonul de Finish vom fi întrebați dacă perspectiva vizuală ar trebui schimbată cu cea a proiectelor PyDev. Vom apăsa pe butonul de Yes.

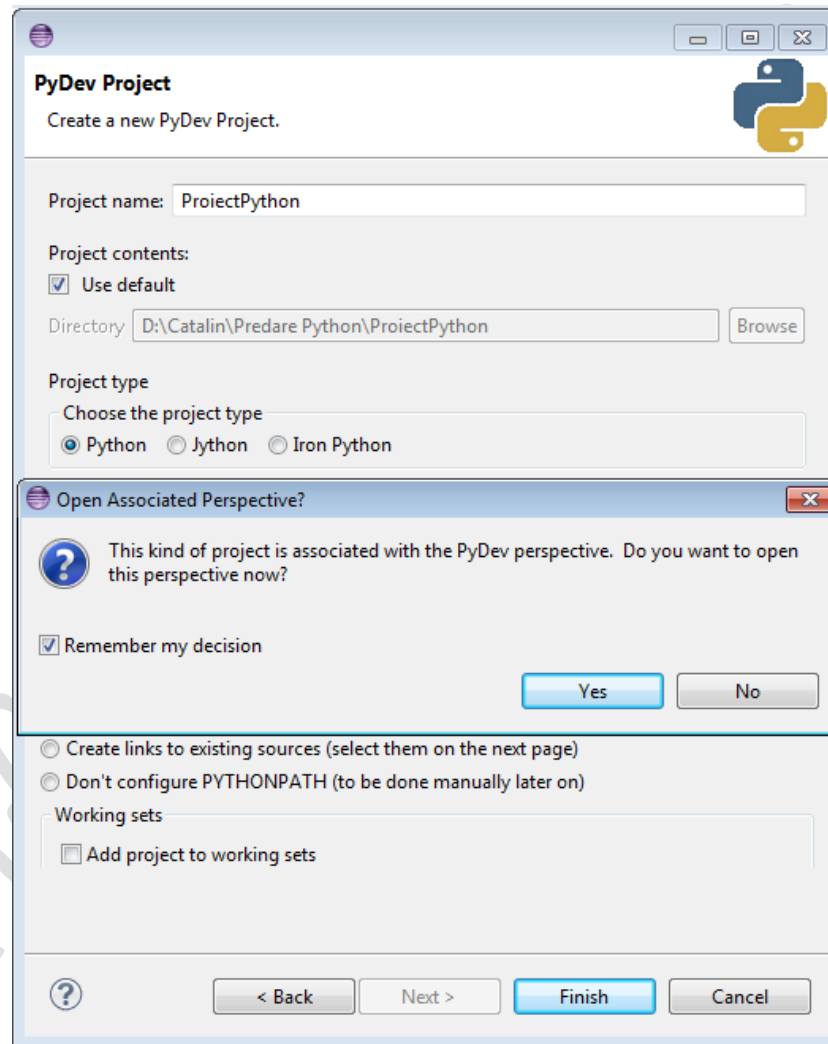


Fig. 17

Din acest moment Eclipse cu Python este disponibil. Putem crea fișiere sau directoare, putem rula și face operațiuni de debug.

Rularea unui program se poate face dand click dreapta pe acel fisier apoi alegand Run As >> Python Run. In partea dreapta unde regasim tab-ul Console vom putea vedea rularea acelui program.

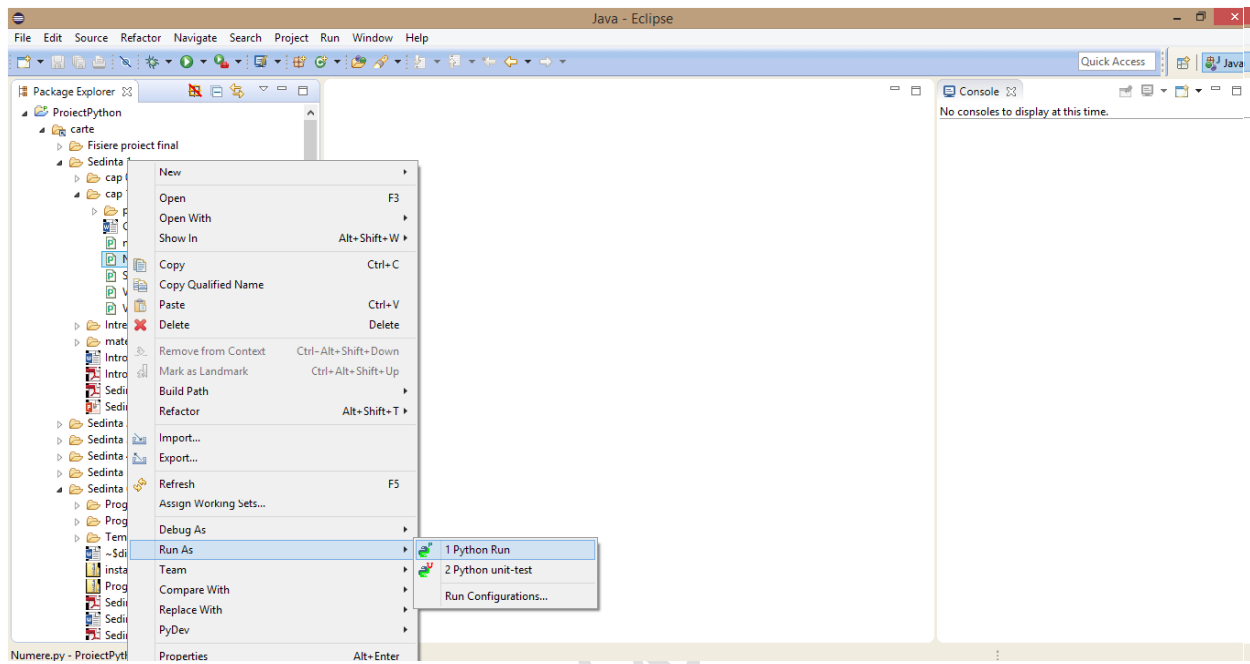


Fig. 18

SEDINTA 1 – VARIABLE ȘI OPERATORI DECIZIONALI

Lucru cu șiruri de caractere

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.

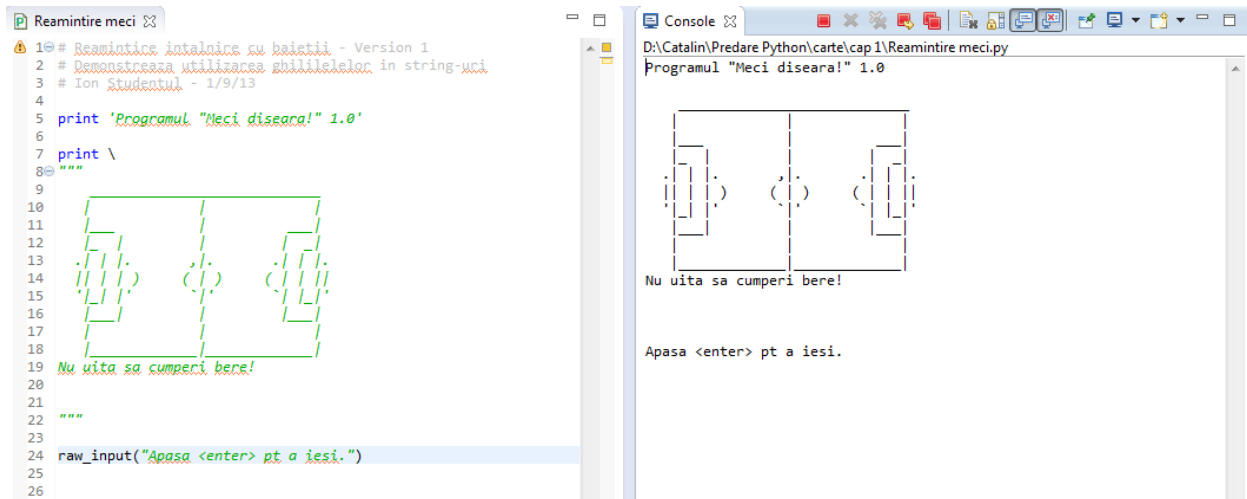


Fig. 1

Cum ați putut observa putem crea string-uri folosind ghilimele simple sau duble.

Astfel în primul statement de print vedem că am utilizat la început ghilimele simple, apoi ghilimele duble, după care iarăși ghilimele simple. Regula este simplă: dacă încep cu ghilimele simple să termin cu ghilimele simple. În acest caz ghilimelele duble vor fi tratate ca un caracter normal. În cazul în care folosesc ghilimele duble la început și la final, ghilimelele simple vor fi tratate ca un caracter normal.

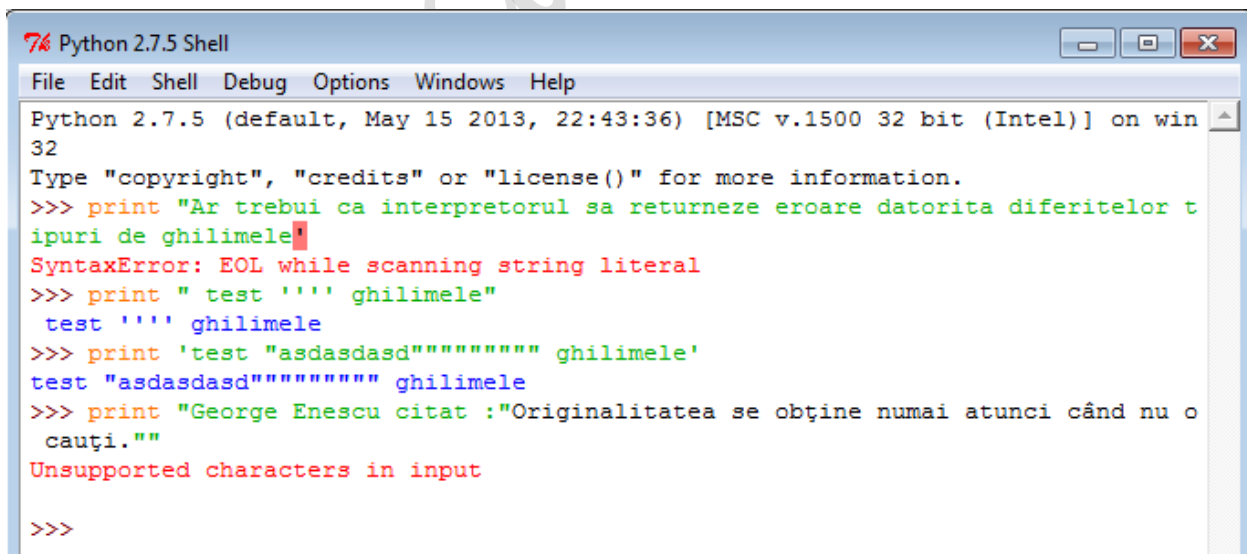


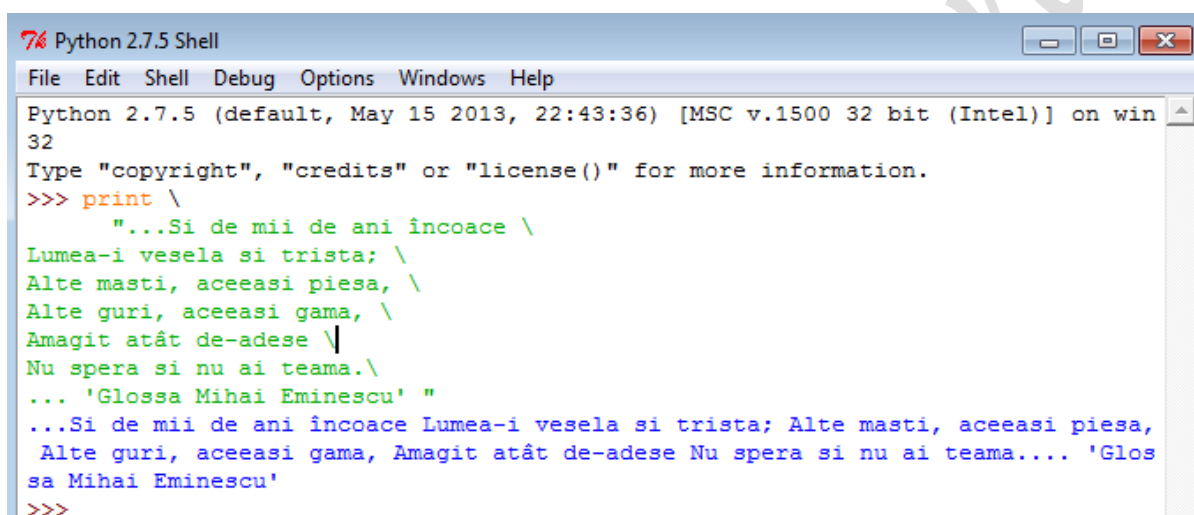
Fig. 2

Așa cum se poate vedea și în Fig. 2 puteți utiliza cate ghilimele doriți în interiorul șirului de caractere atât timp cat sunt diferite de ghilimelele de la început și final.

Daca aceasta regulă nu este respectată interpretorul va genera erori.

Revenind la programul Reamintire meci.py găsim sintaxa "print \" pe rândul al 7-lea. În cazul în care au o sintaxa care se întinde pe mai multe caractere, depășind 80 caractere (in general lățimea unui ecran) , va deveni greu de citit pt. dezvoltatorii python care au rolul să mențină codul. Cu caracterul slash \ putem obliga interpretorul să ignore enter-ul de după caracter și să interpreteze liniile ca o singura sintaxa.

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



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print \
    "...Si de mii de ani incoace \
Lumea-i vesela si trista; \
Alte masti, aceeasi piesa, \
Alte guri, aceeasi gama, \
Amagit atât de-adese \
Nu spera si nu ai teama.\
... 'Glossa Mihai Eminescu' "
...Si de mii de ani incoace Lumea-i vesela si trista; Alte masti, aceeasi piesa,
Alte guri, aceeasi gama, Amagit atât de-adese Nu spera si nu ai teama.... 'Glos
sa Mihai Eminescu'
>>>
```

Fig. 3

În programul Reamintire meci, începând de la linia 8 vedem un șir de caractere ce se întinde peste mai multe linii, folosind ghilimele triple. Fie ca folosim trei ghilimele simple la început și trei ghilimele simple la final, fie ca folosim trei ghilimele duble la început și trei ghilimele duble la final vom avea același rezultat. În cazul în care le combinăm interpretorul va genera o eroare.

Imaginea formata din caractere alpha și non-alpha numerice afișata în program, este denumita ASCII Art. ASCII Art este o pictura din caracterele ce se găsesc la tastatura. Acest tip de arta nu este noua și nu a apărut odată cu era calculatoarelor, așa cum era de așteptat. Primul desen înregistrat pe mașina de scris datează încă din 1898.

Deoarece unii au transformat aceasta pasiune într-o arta puteți accesa pagina de mai jos pentru a putea vizualiza, utiliza și modifica adevărate opere de arta ASCII:

<http://www.chris.com/ascii/> .

În următorul program vom putea învăța despre secvențele de evadare (escape sequence). Acestea sunt caractere speciale ce au rolul de a schimba comportamentul normal al afișării unui șir de caractere, oferind o flexibilitate avansată în afișare.

```
7% Escape Sequence.py - D:\Catalin\Predare Python\carte\cap1\Escape Sequence.py
File Edit Format Run Options Windows Help
# Informatii parola uitata
# Demonstreaza escape sequences
# Ion Studentul 1/11/13

print "\a Nu-ti amintesti parola ?\a"

print "\t\t\tInformatii despre parola"

print "\t\t\t\\ \\ \\ \\ \\ \\ \\ \\ \\ \"
print "\t\t\t\gasesti la"
print "\t\t\tIon Studentu';"
print "\t\t\t\\ \\ \\ \\ \\ \\ \\ \\ \"
print "\nPoaate fi contactat:"
print "Telefon :0722.222.222 sau \nprin e-mail a adresele \" mesaje_pt_ion@studentul.com\" \"nsi \"'resetpassword@studentul.com\"'."
raw_input("\n\nApasa <enter> pt a iesi.")
```

Fig. 4

```
*Python 2.7.5 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
• Nu-ti amintesti parola ?•
                        Informatii despre parola
                        \ \ \ \ \ \ \ \ \
                        \gasesti la
                        Ion Studentu';
                        \ \ \ \ \ \ \ \ \

Poaate fi contactat:
Telefon :0722.222.222 sau
prin e-mail a adresele " mesaje_pt_ion@studentul.com"
si 'resetpassword@studentul.com'.

Apasa <enter> pt a iesi.|
```

Fig. 5

Sa discutam despre elementele noi care se regăesc în cadrul acestui program.

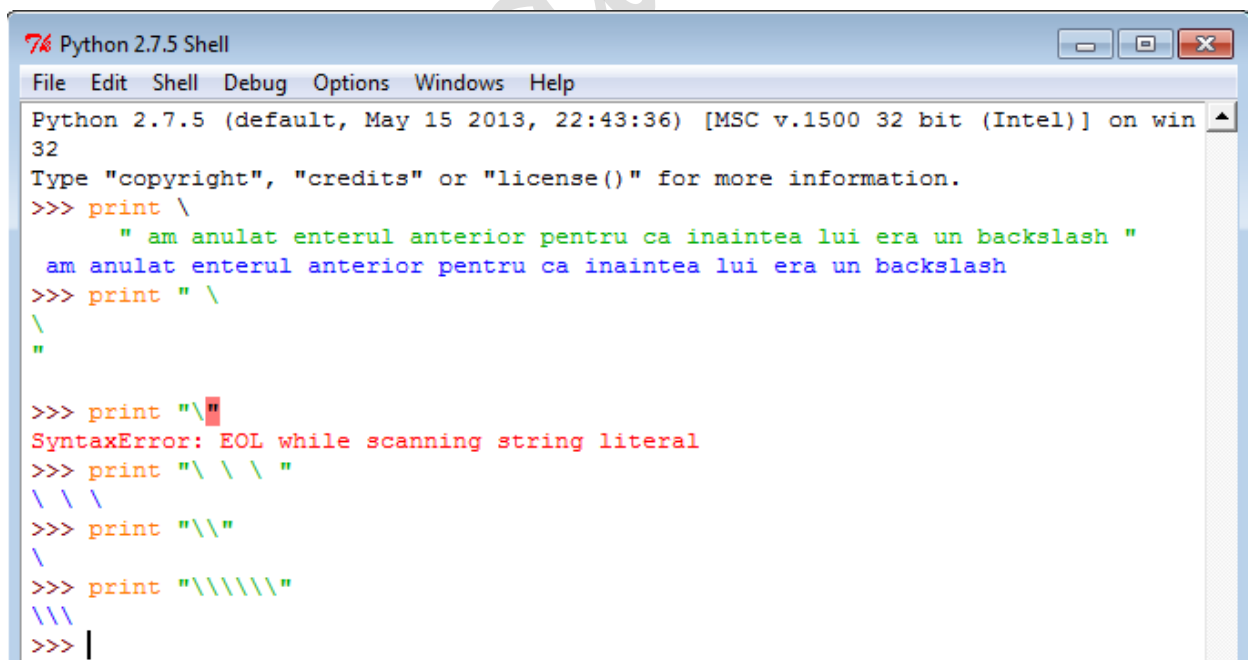
```
print "\a"
```

Aceasta sintaxa va genera un sunet de bip. Exista însă o condiție: rularea trebuie să se realizeze prin dublu click (Windows) sau prin comanda *python file.py* (linux). Prin urmare nu putem să rulăm programul în cadrul IDLE sau Eclipse pentru a declanșa un sunet de bip. În cadrul IDLE sau Eclipse va fi afișat în schimbul secvenței de evadare `\a` un caracter special.

Putem auzi în program rulat mai sus doua bip-uri deoarece de fiecare data când interpretorul întâlnește aceasta secvența `\a` va genera un sunet.

Secvența de evadare `\t` are ca efect mutarea sirului de caractere cu un tab. Un tab este reprezentat de 4 spatii. Deci, în cadrul programului găsim mai multe secvențe de evadare de acest tip care generează ca outputul programului să fie deplasat către dreapta corespunzător cu un nr. de taburi.

O alta secvența de evadare este caracterul backspace, utilizat și în programul anterior. Dacă data trecută am utilizat backslash pt. a putea anula caracterul enter. De această dată caracterul backslash anulează însemnătatea caracterului backslash pentru a putea fi afișat. În Fig. 6 se regăesc exemple care deserveșc la o înțelegere mai amplă a conceptului de anulare a următorului caracter dată de backslash.



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print \
    " am anulat enterul anterior pentru ca inaintea lui era un backslash "
    am anulat enterul anterior pentru ca inaintea lui era un backslash
>>> print " \
\
"

>>> print "\
SyntaxError: EOL while scanning string literal
>>> print "\ \ \ "
\ \ \
>>> print "\\ "
\
>>> print "\\\\\\\\"
\\ \ \
>>> |
```

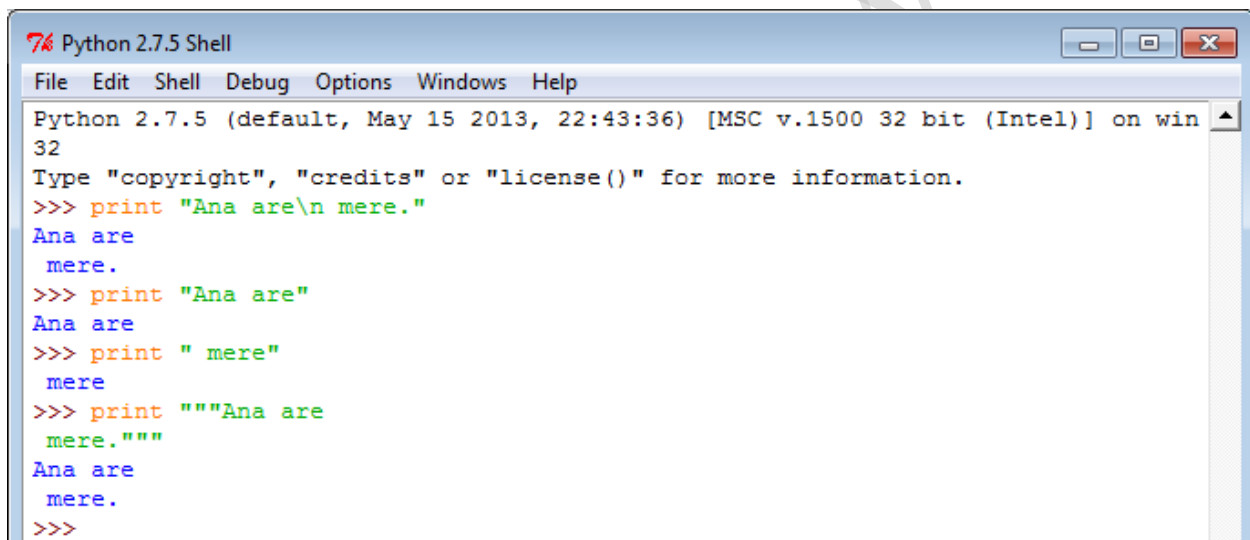
Fig. 6

Se poate vedea ca daca în cadrul unui string punem backslash și space având același efect ca atunci cand avem doua backslash-uri unul după altul. Printarea unui backslash în mod direct nu se poate efectua deoarece ar anula importanta data de ghilimele(fie simple fie duble). Un alt aspect important al backslash-ului este că poate să ne ajute la afișarea diferitelor tipuri de ghilimele, anulând însemnătatea ghilimelei și transforma ghilimeaua într-un caracter normal. Din această cauza vedem că interpretorul va returna o eroare de sintaxa care e similara cu eroarea data de sintaxa:

```
print "Lipseste ultima ghilimea.
```

Totuși în cadrul programului "Informatii parola uitata" putem afișa adresele de e-mail ca fiind încadrate în ghilimele fără ca interpretorul să returneze o eroare.

Secvența de evadare "\n" are scopul de a introduce enter – linie noua (new line) după acest caracter. Mai jos avem o figură în care se poate regăsi un echivalent al acestei secvențe de evadare.



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print "Ana are\n mere."
Ana are
mere.
>>> print "Ana are"
Ana are
>>> print " mere"
mere
>>> print """Ana are
mere."""
Ana are
mere.
>>>
```

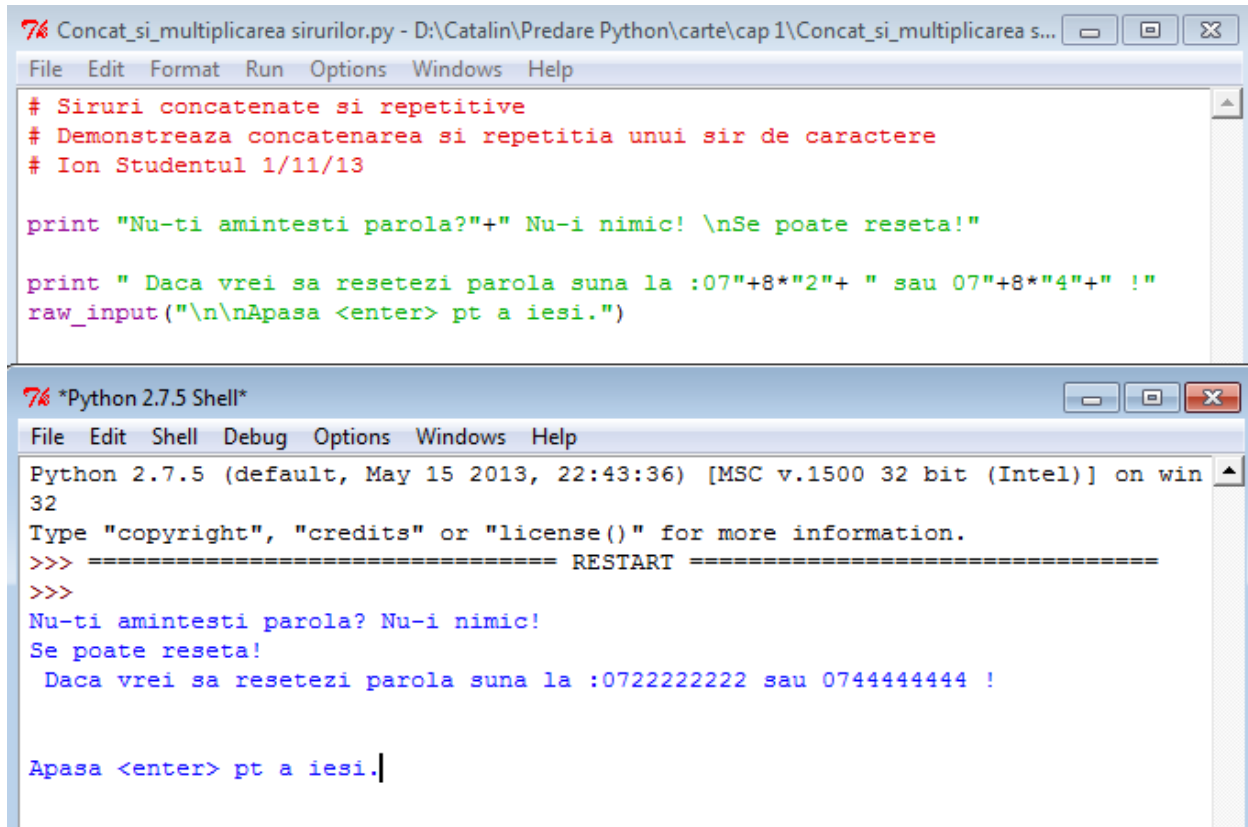
Fig. 7

Așa cum ne indica și Fig. 7 putem să utilizăm și alte metode de a afișa o noua linie, dar metoda secvenței de evadare "\n" este ce mai ușor de identificat în cod.

În secțiunea următoare vom discuta despre repetarea și concatenarea sirurilor.

Vi s-a întâmplat să auziți anumite discuții despre educație în cadrul școlilor, cum ar fi metodele de învățământ ce folosesc pedeapsa și răsplata. Una din metodele preferate de pedagogi este pedepsirea prin scrierea repetitivă a unui șir de caractere ce are o însemnătate deosebită. Un exemplu ar fi: "Promit să nu mai fur mâncare la colegi". Dacă am folosi python și am dori să scriem un șir de caractere de mai multe ori am putea utiliza chiar semnul înmulțirii, operație numită repetare. Dar în cazul în care am

dori să unim mai multe șiruri de caractere? Soluția se regăsește în programul de mai jos:



The image shows two windows from a Python IDE. The top window is a text editor titled 'Concat_si_multiplicarea sirurilor.py' containing Python code. The code uses '+' for concatenation and '*' for repetition. The bottom window is the 'Python 2.7.5 Shell' showing the execution of the code. The output matches the code's logic, displaying concatenated strings and repeated strings.

```
# Siruri concatenate si repetitive
# Demonstreaza concatenarea si repetitia unui sir de caractere
# Ion Studentul 1/11/13

print "Nu-ti amintesti parola?"+" Nu-i nimic! \nSe poate reseta!"

print " Daca vrei sa resetezi parola suna la :07"+8*"2"+ " sau 07"+8*"4"+ " !"
raw_input("\n\nApasa <enter> pt a iesi.")
```

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Nu-ti amintesti parola? Nu-i nimic!
Se poate reseta!
Daca vrei sa resetezi parola suna la :0722222222 sau 0744444444 !

Apasa <enter> pt a iesi.
```

Fig. 8

Dacă folosim semnul + pentru concatenare și semnul * pentru repetiție, nu ar trebui să privim șirurile de caractere ca fiind numere! Acestea nu funcționează cu semnul minus – 'sau cu cel de divizare '/.

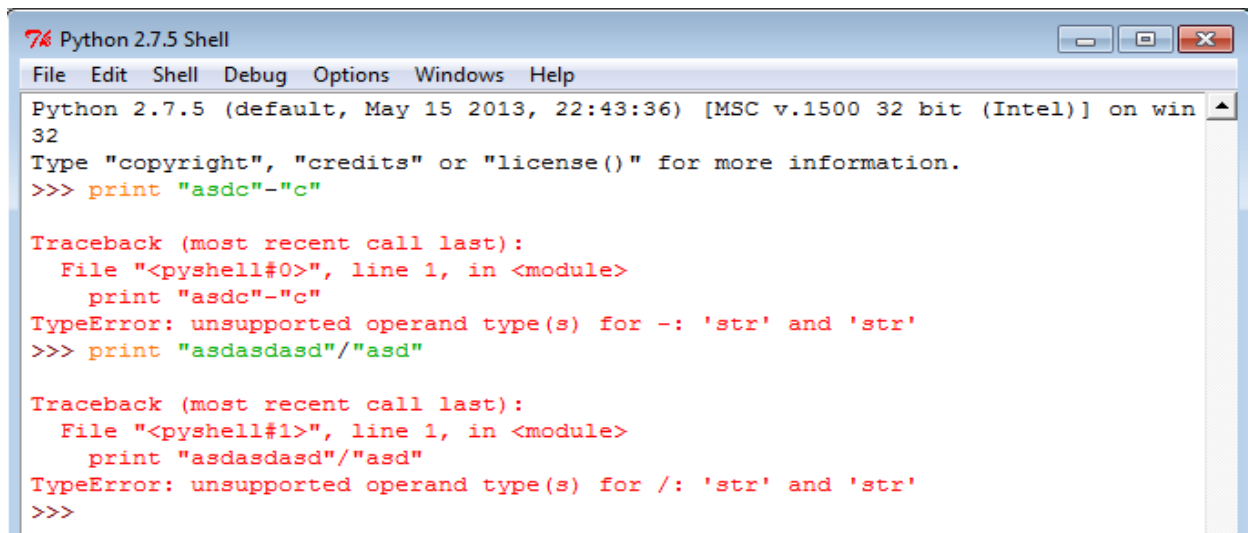


Fig. 9

Lucrul cu numere

În următorul program vedem cum putem să utilizăm numerele și să efectuăm operații numerice.

```
# Lucrul cu numere
# Demonstreaza lucrul cu numere
# Ion Studentul 1/11/13

print "\n\tTema matematica clasa a 4a\n"
print "Rezolva adunarea: 2+2"
print 2+2

print \
    """\nCat canteste o balena de 2000 kg care a nascut un pui de 100 kg."""
print "2000 - 100 = ",
print 2000 - 100

print "\nRezolva inmultirea 333*3? Raspunsul corect este", 333*3,"! "

print "24 / 6 = ",
print 24 / 6 ,
print "(impartire)\n"

print "107 % 4 = ",
print 107 % 4, "Aceasta operatie se numeste modulo!\n"

print \
    """Operatia matematica de mai jos aplica impartirea cu rest!"""
print "19 / 4 = ",
print 19 / 4
```

```

print "Gresit!"

print "Raspunsul corect este :",19.0 / 4

print "\nOperatie mateamatica complexa1: [(2+2)*3]/4 =" ,((2+2)*3)/4,"\n"
print "Operatie mateamatica complexa2: 2*2*2*2 =" ,2 ** 4,"\n"

raw_input("\n\nApasa <enter> pt a iesi.")

#####

>>>

    Tema matematica clasa a 4a

Rezolva adunarea: 2+2
4

Cat canteste o balena de 2000 kg care a nascut un pui de 100 kg.
2000 - 100 = 1900

Rezolva inmaltirea 333*3? Raspunsul corect este 999 !
24 / 6 = 4 (impartire)

107 % 4 = 3 Aceasta operatie se numeste modulo!

Operatia matematica de mai jos aplica impartirea cu rest!
19 / 4 = 4
Gresit!
Raspunsul corect este : 4.75

Operatie matematica complexa1: [(2+2)*3]/4 = 3

Operatie matematica complexa2: 2*2*2*2 = 16

```

Fig. 10

In Fig.10 se poate observa rulara programului 'Lucrul cu numere'.

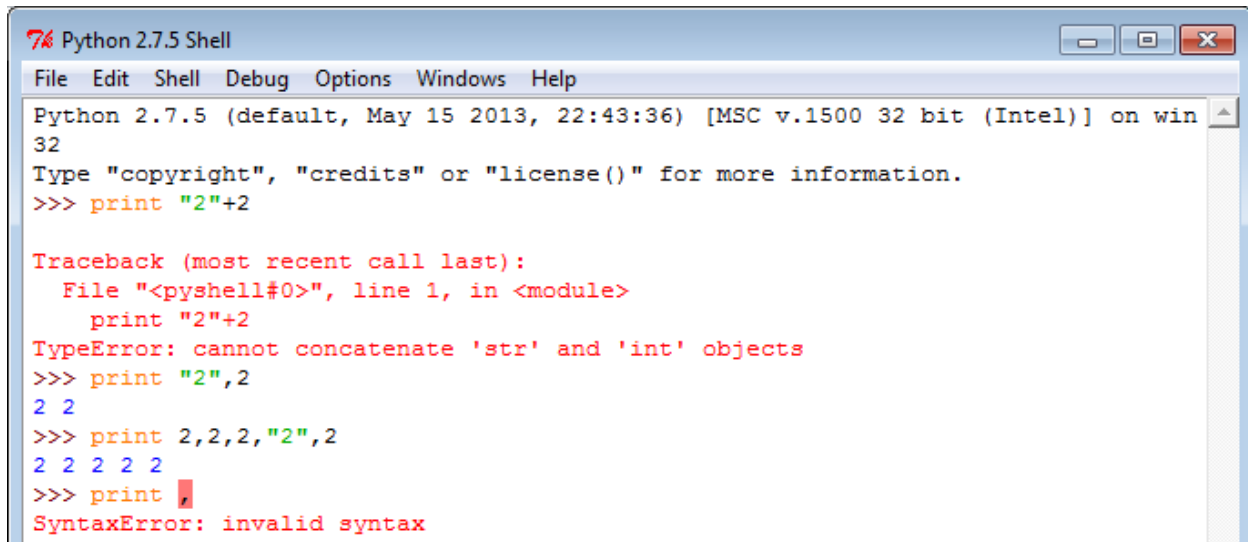
Să vedem ce elemente noi avem în acest program.

In primul rând Python poate realiza toate operațiile matematice elementare cu foarte mare ușurință. Acestea se pot apela în mod direct prin comenzi de genul :

```
print 2+2
```

Un alt element cheie în afișarea diferitelor tipuri de date este virgula ','. Virgula permite afișarea diferitelor structuri de date sub aceeași sintaxa, cum ar fi numerele și șirurile de

caractere. Cum acestea nu se pot concatena singura posibilitate ar fi virgula care face ca interpretorul să ia fiecare element în parte din acea sintaxa și să apeleze separat comanda print, apoi să unească outputul fiecărei comenzi, așa cum se poate vedea și în Fig. 11.



```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print "2"+2

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print "2"+2
TypeError: cannot concatenate 'str' and 'int' objects
>>> print "2",2
2 2
>>> print 2,2,2,"2",2
2 2 2 2 2
>>> print
SyntaxError: invalid syntax

```

Fig. 11

Virgula mai permite și unirea a mai multe linii pe o singura linie:

```

print "2000 - 100 = ",
print 2000 - 100

```

Cele doua sintaxe de mai sus vor fi afișate sub aceeași linie datorita virgulei ce se regăsește la finalul primei sintaxe. Se poate observa că comanda print exista si pe urmatoarea linie. Prin urmare extinderea comenzii care era validă prin utilizarea backslash-ului nu se aplică și aici datorita repetarii cuvântului cheie print. Deci aplicarea celor doua linii de mai jos va genera eroare:

```

print "2000 - 100 = "\
print 2000 - 100

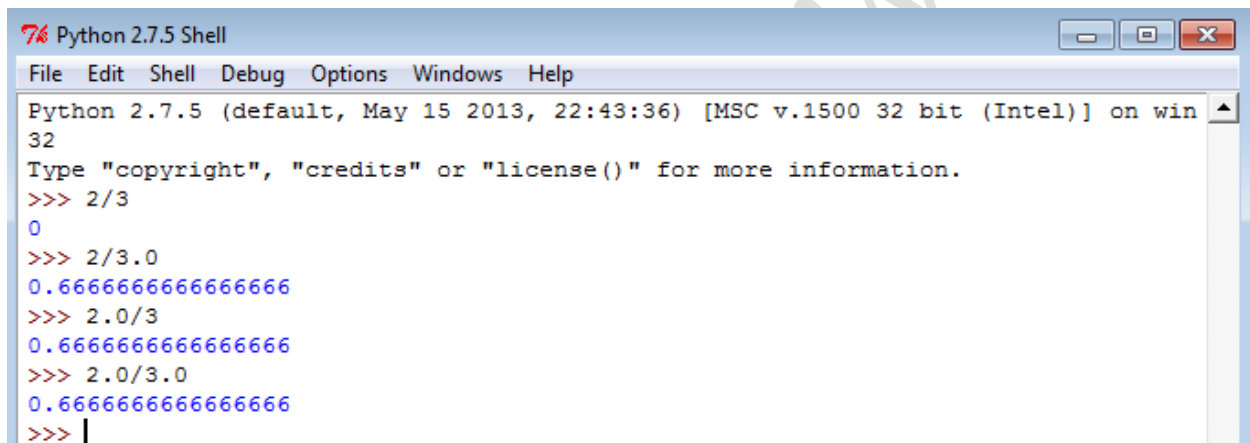
```

Pe lângă operațiile clasice de adunare, împărțire, înmulțire și scădere, python este capabil și de a face operații cu paranteze . Cum parantezele pătrate și acoladele au alt rol în python peste tot se vor folosi paranteze rotunde.

Putem să efectuăm operații matematice exponențiale folosind gruparea de simboluri `***`, unde primul număr este baza, iar al doilea număr este exponentul. Spre exemplu x la puterea y în python se scrie ca `x**y`.

O altă operație al cărui nume este modulo și realizându-se cu ajutorul `%` este valabilă în Python. Operația matematică `7%3` va fi egală cu 1 deoarece modulo returnează restul împărțirii lui 7 la 3. Sapte împărțit la 3 dau 2 rest 1.

În Python numerele pot fi de trei feluri: integer, float și complex. Numerele integer sunt numere întregi la care împărțirea nu se face pe baza restului, care va fi ignorat. Dacă 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. Dacă cel puțin unul din elementele unei împărțiri este float (numitorul sau numărătorul) rezultat, rezultatul va fi float. Acest aspect ne ajută în programele în care numărătorul nu-l introducem noi, dar dorim ca rezultatul să fie float.

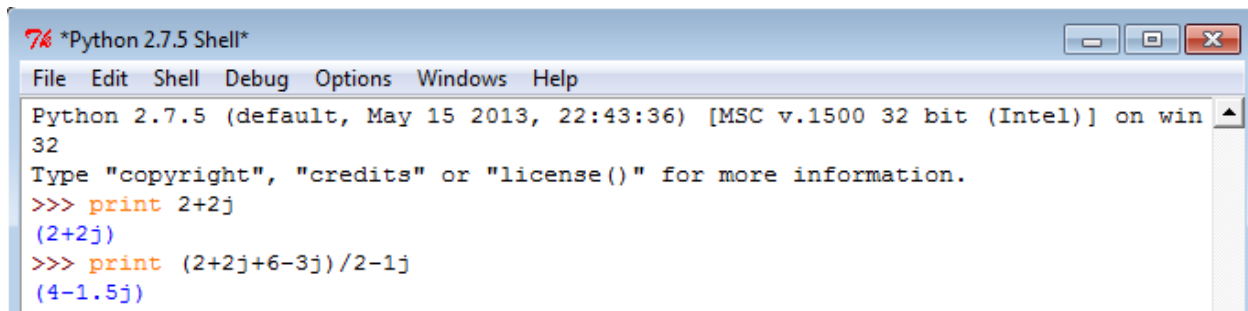


```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 2/3
0
>>> 2/3.0
0.6666666666666666
>>> 2.0/3
0.6666666666666666
>>> 2.0/3.0
0.6666666666666666
>>> |
```

Fig. 12

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

În python există și notiunea de numere complexe. Un exemplu de număr complex ar fi : $2+2i$. Datorită faptului că în America se folosește litera `<<j>>` în loc de `<<i>>` pentru a identifica partea complexă a unui număr complex, așa vom regăsi și noi numerele complexe în python.



```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print 2+2j
(2+2j)
>>> print (2+2j+6-3j)/2-1j
(4-1.5j)

```

Fig. 13

Variabile

Cu ajutorul variabilelor informația poate fi stocată și manipulată, un aspect fundamental în programare. Python permite crearea și manipularea de variabile. Dar care ar fi motivul pentru care am avea nevoie de variabile? Cel mai bun motiv este flexibilitatea de a schimba o anumită variabilă așa cum se poate vedea în programul următor.

```

# Stefan Cel Mare
# Demonstreaza lucrul cu variabile
# Ion Studentul 1/11/13

s = "Stefan cel Mare"
st = s + "și Sfânt"

print "\tLectie de istorie: Stefan cel Mare\n"
print s, "a domnit între între anii 1457 și 1504.\n"

print s, "a avut 36 de batalii, din care a castigat 34."
print s, "a fost canonizat in 1992 pentru eforturile supraomenești de "
print "a pastra identitatea nationala."
print "Astfel ", s, "devine", st, "."

raw_input("\n\nApasa <enter> pt a iesi.")

```

```

D:\Catalin\Predare Python\carte\cap 2\Variabile.py
|
Lectie de istorie: Stefan cel Mare

Stefan cel Mare a domnit intre intre anii 1457 si 1504.

Stefan cel Mare a avut 36 de batalii, din care a castigat 34.
Stefan cel Mare a fost canonizat in 1992 pentru eforturile supraomenești de
a pastra identitatea nationala.
Astfel Stefan cel Mare devine Stefan cel Mare si Sfânt .

Apasa <enter> pt a iesi.

```

Fig. 14

Prin urmare, variabila `s` va stoca șirul de caractere 'Stefan cel Mare'. Apoi de fiecare dată când apelăm variabila `s` putem utiliza conținutul (ce stochează) variabilei `s`. Adicional putem să folosim variabila `s` pentru a crea noi variabile cum ar fi crearea variabilei `<<st>>` din exemplu de mai jos. Ce credeți că o să afișeze printarea variabilei `<<st>>`?

```

# Stefan cel Mare
# Demonstreaza lucrul cu variabile
# Ion Studentul 1/11/13

s = "Stefan cel Mare"
st = s + " și Sfânt"
#redenumirea variabilei s
s = "Stefan CEL Mare"

print "\tLectie de istorie: Stefan cel Mare\n"
print s, " a domnit intre intre anii 1457 și 1504.\n"

print s, " a avut 36 de batalii, din care a castigat 34."
print s, " a fost canonizat in 1992 pentru eforturile supraomenești de "
print "a pastra identitatea nationala."
print "Astfel ", s, "devine", st, "."

raw_input("\n\nApasa <enter> pt a iesi.")

```

Așa cum probabil ați intuit, am putea să redefinim variabila `s` pe care o putem utiliza cu noua valoare. Prin urmare, interpretorul va returna un output ca în Fig. 6


```

D:\Catalin\Predare Python\carte\cap 2\Variabile.py
Lectie de istorie: Stefan cel Mare

Stefan CEL Mare a domnit intre intre anii 1457 si 1504.

Stefan CEL Mare a avut 36 de batalii, din care a castigat 34.
Stefan CEL Mare a fost canonizat in 1992 pentru eforturile supraomenești de
a pastra identitatea nationala.
Astfel Stefan CEL Mare devine Stefan cel Mare si Sfânt .

Apasa <enter> pt a iesi.

```

Fig. 15

Numele variabilei nu a fost ales întâmplător. Dacă stăm să ne gândim la un program mic aceasta variabilă nu ar încurca dezvoltatorul aplicației, dar în cazul în care programul este mare devine suficient de dificil să ne amintim ce stochează variabila `s`, neavând un nume sugestiv. Dar ce nume ar putea lua o variabilă ?

O variabilă poate fi formată din litere, numere și underscore `'_'`. Primul caracter din numele unei variabile poate fi doar o literă sau underscore. Variabilele ce încep cu underscore au altă însemnătate fiind folosite în programarea orientată pe obiecte.

Ceea ce înseamnă ca numele variabilei ar trebui să înceapă cu o literă, de obicei o literă mică, literele mari sunt definite pentru funcții. Trebuie ales un nume sugestiv care să indice ce conține variabila.

Spre exemplu variabila `s` ar trebui denumită `stefanMare` iar variabila `st` `stefan_Sfant`.

Prin aceste două exemple vreau să subliniez ca există două stiluri de a crea variabile:

Primul stil este cel în care alternezi literele mici cu cele mari pentru a desparti cuvintele, iar cel de-al doilea stil se regăsește prin separarea cuvintelor prin underscore. Fiecare programator va trebui să-și identifice propriul stil. E bine să existe o consistență a variabilelor pentru a fi ușor de citit codul.

În mod uzual o variabilă globală (validă pentru tot programul și accesabilă în fiecare ramură a programului) va fi declarată la începutul programului, imediat după comentariile ce descriu programul.

Capturarea user input

Python este capabil să captureze cuvintele introduse de la tastatură de către user. Prin urmare acestea se pot utiliza ulterior similar cu o variabilă definită de programator.

Următorul program va arata cum se poate realiza capturarea de caractere de la tastatura.

```
# Salut Personalizat
# Demonstreaza user input
# Ion Studentul 1/13/03

Nume_Utilizator = raw_input("Salut. Cum te numesti? \n\n")

print "\n",Nume_Utilizator,"?\n"

print "Salut " + Nume_Utilizator + "!"

raw_input("\n\nApasa <enter> pt a iesi.")
```

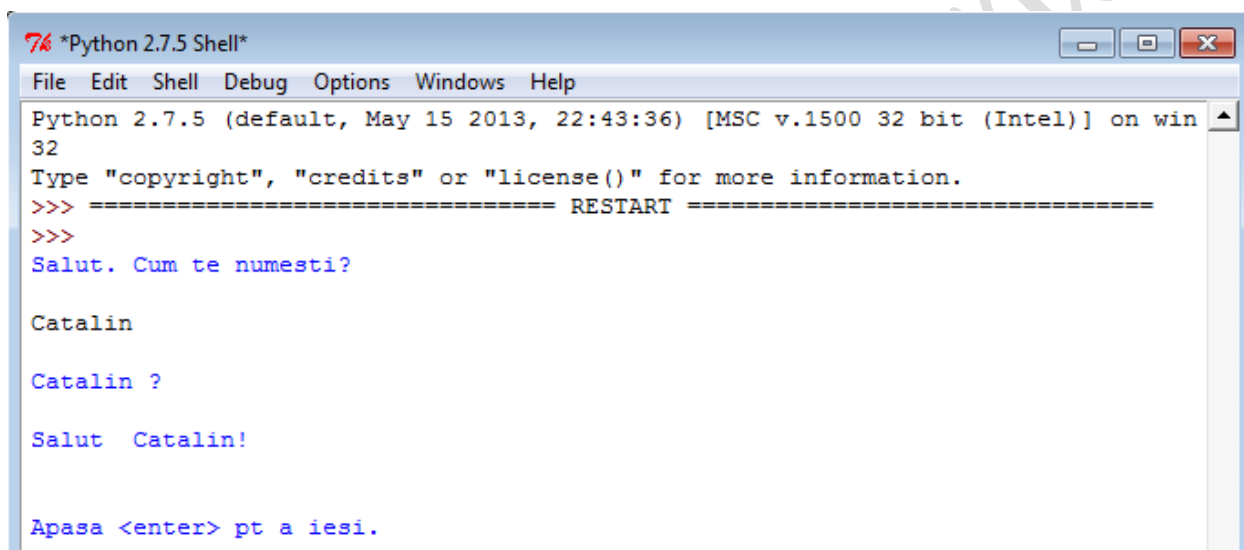
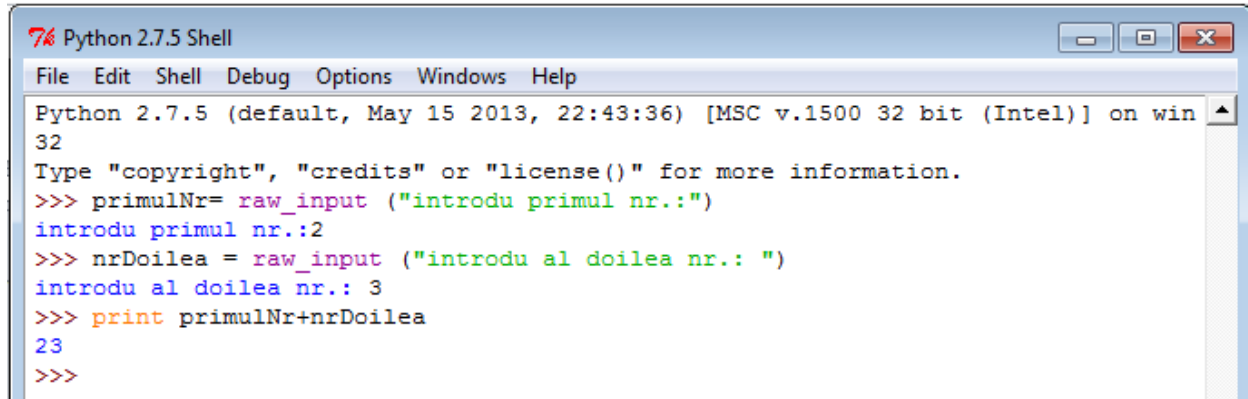


Fig. 16

În programul de mai sus se poate vedea că variabila `Nume_utilizator` va captura ce scrie utilizatorul de la tastatură. Odată capturată valoarea poate fi utilizată ca orice variabilă de tip șir de caractere. Astfel ea poate fi printată sau modificată.

După cum se poate vedea în programul de mai sus, vom utiliza comanda `raw_input` pentru a captura user input. Tot mereu comanda `raw_input` va interpreta ce capturează ca fiind un șir de caractere. Deci, dacă dorim să realizăm o sumă a două numere introduse de la tastatură folosind `raw_input` în mod direct fără a converti acele șiruri de caractere capturate în numere, programul va genera concatenarea celor două șiruri de caractere, după cum se poate vedea în Fig. 17.



```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> primulNr= raw_input ("introdu primul nr.:")
introdu primul nr.:2
>>> nrDoilea = raw_input ("introdu al doilea nr.: ")
introdu al doilea nr.: 3
>>> print primulNr+nrDoilea
23
>>>

```

Fig. 17

Ce introduce user-ul ar trebui verificat de cele mai multe ori. Spre exemplu ne așteptăm ca userul să introducă un număr . Acest lucru trebuie verificat. Acesta este următorul lucru pe care îl vom studia și anume manipularea de variabile de tip sir de caractere.

Manipularea șirurilor de caractere

În următorul program o să învățăm manipularea șirurilor de caractere în vederea verificării informației stocate de acest sir de caractere. Putem afișa 2 ca fiind un sir de caractere , deci avem nevoie de o metoda de verificare ce va indica ce conține un sir de caractere. Aceste metode de manipulare a unui sir de caractere se aplica la finalul sirului de caractere cu punct urmand structura de mai jos:

„Sir_caractere”.metoda_aplicata()

Iată și un exemplu de aplicare a unor metode de manipulare a sirurilor de caractere:

```

>>> "Imi place Python".upper()
'IMI PLACE PYTHON'
>>> x= "Imi place Python"
>>> x.lower()
'imi place python'
>>>

```

Fig . 18

Așa cum se poate observa în Fig.18 pot aplica o metoda de manipulare direct unui sir de caractere sau unei variabile ce stochează un sir de caractere. Metoda de manipulare upper() returnează toate caracterele de tip litera cu litera mare, pe când metoda de manipulare lower() va returna un sir de caractere cu litera mică.

Un alt aspect important intalnit in programul urmatoar este introducerea variabilei de tip boolean ce poate avea doar doua stări: True și False. Aceste stări se regăsesc în culoarea albastră și trebuie scrise ca atare (True sau False) fiind sensibile la diferente de caractere mici sau mari (eng.case sensitive) dacă le declaram manual.

Explicarea acestor metode dar si a variabilei de tip boolean o vom face pe baza unui exemplu ce se regaseste in sectiunea urmatoare:

```
# Manipularea sir caractere
# Demonstreaza manipularea variabilelor ce contin un sir de caractere
# Ion Studentul 1/13/03

# Citat Presedinte IBM, Thomas Watson, in 1943
citat = "I think there is a world market for maybe five computers."
print "Citat original:"
print citat

print "\nIn Litere mari:"
print citat.upper()

print "\nIn Litere mici:"
print citat.lower()

print "\nCa titlu:"
print citat.title()

print "\nCu o mica schimbare:"
print citat.replace("five", "millions of")

print "\nCitatul original este inca :"
print citat

print "\nVerificare daca variabila este un sir de caractere format numai \
din numere:"
print citat.isdigit()

print "\nVerificare daca variabila este sir de caractere format din \
caractere alpha:"
print citat.isalpha()

citat_modificat= citat.replace(" ", "")
citat_modificat= citat.replace(".", "")

print "\nVerificare daca variabila este sir de caractere format din caractere alpha:"
print citat_modificat.isalpha()

raw_input("\n\nApasa <enter> pt a iesi.")
```

Mai jos regăsim și rularea programului (Fig . 19).

```

D:\Catalin\Predare Python\carte\cap 2\manipularea_sir_caractere.py
Citat original:
I think there is a world market for maybe five computers.

In litere mari:
I THINK THERE IS A WORLD MARKET FOR MAYBE FIVE COMPUTERS.

In litere mici:
i think there is a world market for maybe five computers.

Ca titlu:
I Think There Is A World Market For Maybe Five Computers.

Cu o mica schimbare:
I think there is a world market for maybe millions of computers.

Citatul original este inca :
I think there is a world market for maybe five computers.

Verificare daca variabila este un sir de caractere format numai din numere:
False

Verificare daca variabila este sir de caractere format din caractere alpha:
False
I think there is a world market for maybe five computers asta e

Verificare daca variabila este sir de caractere format din caractere alpha:
False

Apasa <enter> pt a iesi.|

```

Fig. 19

Sa analizam ce a rezultat din urma rulării și ce face fiecare sintaxa.

Putem să observăm cu ușurință ca `nume_variabila.upper()` va avea ca rezultat toate caracterele cu litere mari așa cum am menționat și mai devreme.

Similar `nume_variabila.lower()` va transforma toate caracterele în litere mici.

În ceea ce privește afișarea ca titlu folosind metoda de manipulare `.title()` vom observa ca fiecare cuvânt din șirul de caractere va fi modificat astfel încât primul caracter să aibă literă mare, apoi celelalte caractere să fie litere mici.

Bineînțeles ca avem la dispoziție opțiunea de a modifica textul, și cu ajutorul expresiei `variabila.replace('valoare de înlocuit', 'valoare ce înlocuiește')` putem să modificăm ce dorim, așa cum se poate vedea și în Fig.19.

Sa analizam un pic cele patru linii de mai jos extrase din program:

```
print "\nCu o mica schimbare:"
print citat.replace("five", "millions of")

print "\nCitatul original este inca :"
print citat
```

Char daca am aplicat metoda de manipulare `.replace()`, vedem ca variabila 'citat' nu s-a modificat deoarece valoarea ce rezulta din acesta expresie a fost doar afişata, nefiind atribuita unei variabile care sa stocheze noau informatie. Daca ne doream să schimbam variabila 'citat' ar fi trebuit să avem o sintaxa de genul:

```
citat = citat.replace("five", "millions of")
```

Reamintim ca variabila de tip boolean ce poate avea doar doua stări: True şi False. Aceste stări se regăsesc în culoarea albastră şi trebuie scrise ca atare (True sau False) fiind sensibile la diferente de caractere mici sau mari (eng.case sensitive) dacă le declarăm manual. Metodele de manipulare a sirurilor `.isdigit()`, `.isalpha()`, `.islower()`, `.isupper()`, `.isspace()`, `.isalnum()`, `.istitle()` returneaza o variabila boolean.

Sintaxa `citat.isdigit()` va returna True daca toate caracterele sunt formate doar din numere, şi False daca cel puțin un caracter este diferit de un număr. Similar va fi rezultatul sintaxei `citat.isalpha()`, diferența o consta în faptul că verificarea o va face pentru caractere alpha.(litere). Spatiul este considerat un caracter ce nu este numar sau litera. Prin urmare un sir de caractere ce va avea stocat o propozitie cu spatiu sau semne de punctuatie va returna False la verificarea sirului cu ajutorul metodei de manipulare `.isalpha()`.

Deci ambele vor trebui să returneze False deoarece şirul nu este format doar din numere şi nu este format doar din litere (deoarece are spațiu şi ' ').

Astfel Programul propune să înlocuim caracterul spațiu cu nimic(deci ştergere) şi caracterul punct cu nimic(deci ştergere).

```
citat_modificat= citat.replace(" ", "")
citat_modificat= citat.replace(".", "")
```

Din păcate ceva nu merge bine deoarece variabila 'citat_modificat' rezulta tot False. Care credeți că este greșeala?

Pai 'citat_modificat' este rezultatul aplicarii metodei de manipulare `replace()` pe baza sirului original ce înseamnă ca va rămâne doar ultima forma de inlocuire. Astfel, pentru a funcționa, trebuie să modificăm în a doua sintaxa variabila căruia i se va aplica metoda replice din 'citat' în 'citat_modificat', variabila ce a fost modificata anterior înlocuindu-se spațiu cu punct. În forma actuala rezultatul afişării cu ajutorul comenzii `print` a variabilei modificat este:

I think there is a world market for maybe five computers

Iată astfel cum ar trebui să modificam pentru a înlocui:

```
citat_modificat= citat.replace(" ", "")
citat_modificat= citat_modificat.replace(".", "")
```

Rezultatul va fi acela așteptat:

Verificare dacă variabila este șir de caractere format din caractere alpha:
True

Similar și în cazul în care folosim metoda `variabila.isdigit()`; va returna True doar dacă toate caracterele șirului de caractere memorat de variabila este format exclusiv din numere.

Mai jos regăsim alte metode de manipulare a șirurilor:

`.islower()` - verifică dacă toate caracterele șirului de tip literă sunt mici; ignora alte caractere neafectând rezultatul returnat

`.isupper()` - verifică dacă toate caracterele șirului de tip literă sunt mari; ignora alte caractere neafectând rezultatul returnat

`.isspace()` - verifică dacă toate caracterele șirului sunt de tip spațiu exclusiv. Orice alt caracter exceptând spațiu ce face parte din șir va genera un False.

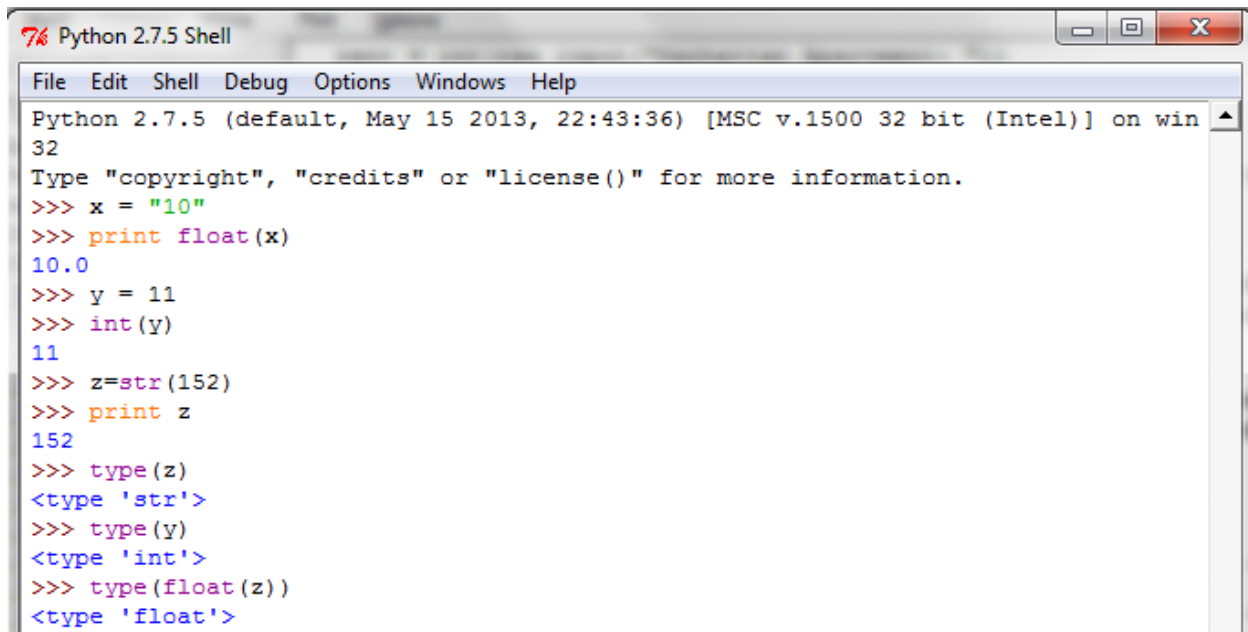
`.isalnum()` - verifică dacă toate caracterele șirului sunt de tip literă sau număr exclusiv. Orice alt caracter exceptând literă sau număr ce face parte din șir va genera un False.

`.istitle()` - verifică dacă toate cuvintele șirului încep cu litere mari; ignora alte caractere neafectând rezultatul returnat

Deci, avem metode prin care putem să demonstrăm ce am captat de la utilizator verificând dacă acel șir de caractere este explicit ceea ce noi căutăm. Totuși nu putem să facem operații matematice deoarece este un șir de caractere. Prin urmare avem nevoie de conversie.

Din păcate, dacă noi capturăm caractere de la tastatură, e greu să conținutul variabilei folosite ca parametru, dar depinde de noi să verificăm existența șirului de caractere înainte de a fi convertit.

Pentru a converti un șir de caractere în numere va trebui să folosim una din funcțiile ce se regăsesc în sintaxele de mai jos:



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> x = "10"
>>> print float(x)
10.0
>>> y = 11
>>> int(y)
11
>>> z=str(152)
>>> print z
152
>>> type(z)
<type 'str'>
>>> type(y)
<type 'int'>
>>> type(float(z))
<type 'float'>
```

Fig. 20

Sa discutam un pic despre sintaxele din IDLE ce se regăsesc în Fig. 20.

Primul pas este să definim o variabila de tip șir de caractere. Aceasta poate fi convertita într-o valoare float apelând comanda float. Aceasta ia ca și parametru(argument) o variabila de tip string sau integer. Un parametru este o valoare ce o punem între parantezele unei comenzi, iar comenzile se numesc de fapt funcții. Funcțiile le putem defini noi, dar există și funcții standard.

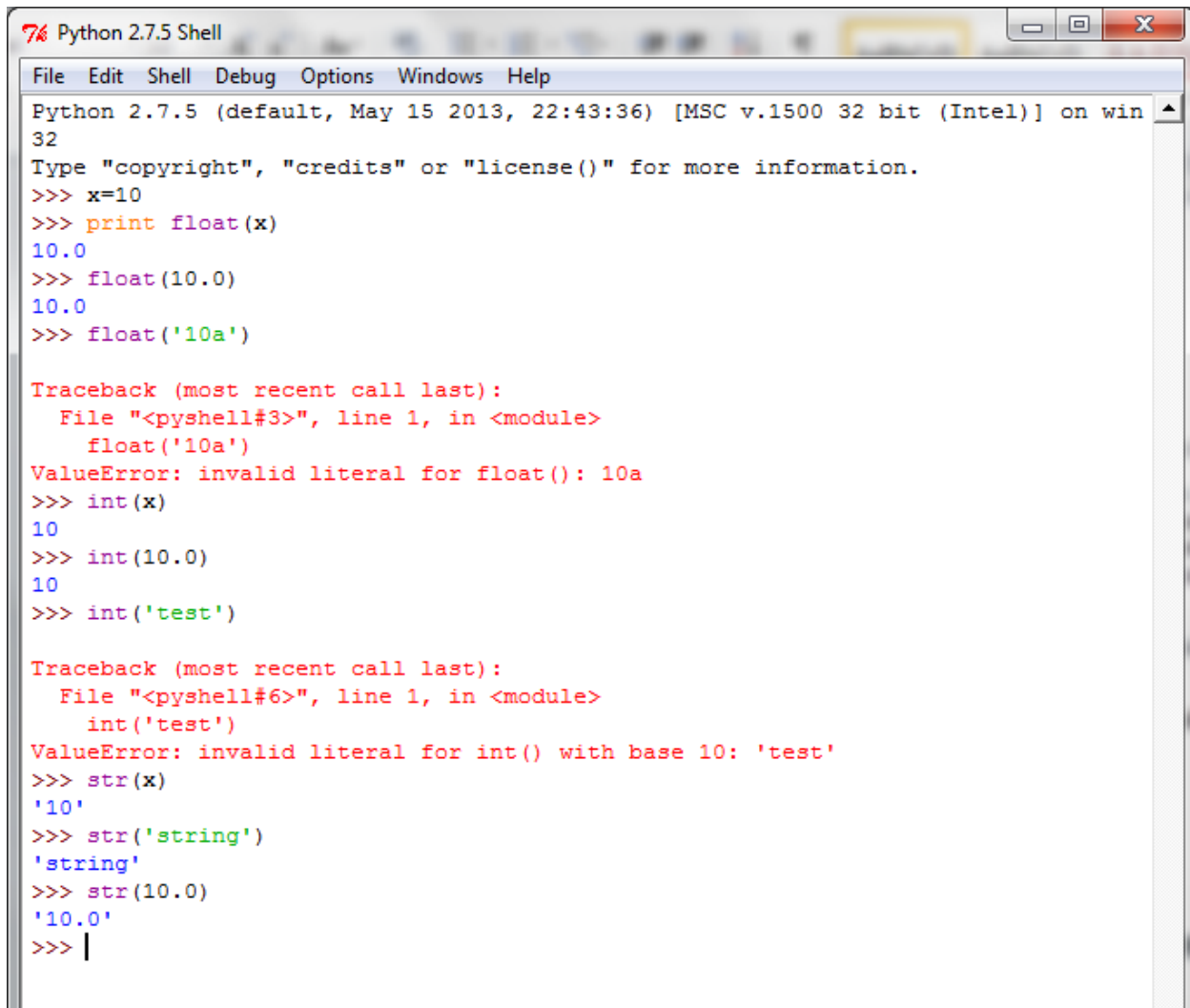
O altă funcție de conversie este str(x) unde x este o variabila ce susține un număr (integer sau float).

Avem la dispoziție și funcția int(x) unde x este o variabilă ce susține un șir de caractere.

Dar ce se întâmplă dacă parametru de intrare nu respectă regulile. Așa cum se poate vedea în Fig. 21, funcția float și funcția int sunt sensibile la un parametru ce este un șir de caractere format din caractere non-numerice. În acest caz interpretorul va returna o eroare indicând ce e greșit la această sintaxă.

Conversia unui număr float în float sau a unui număr integer în integer nu returnează o eroare, dar nu are nici un sens nici utilizarea acestei transformări.

Tot în Fig. 21 putem vedea că funcția str() nu returnează eroare în nici unul din cazuri.



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> x=10
>>> print float(x)
10.0
>>> float(10.0)
10.0
>>> float('10a')
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    float('10a')
ValueError: invalid literal for float(): 10a
>>> int(x)
10
>>> int(10.0)
10
>>> int('test')
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    int('test')
ValueError: invalid literal for int() with base 10: 'test'
>>> str(x)
'10'
>>> str('string')
'string'
>>> str(10.0)
'10.0'
>>> |
```

Fig. 21