Programarea calculatoarelor si limbaje de programare 1

Introducere

Universitatea Politehnica din Bucureşti

Sumar

- ***Syllabus & Curriculum***
- Avantajele Python
- Executia in Python
- Cum se ruleaza programele Python

Orar

- Curs: Miercuri saptamani impare, in sala AN 015
- Laborator: conform orar, sala EG 305/103a
  - Sg. 716b se muta vineri ora 14:00 in EG103a, iar sg. 711a se muta joi ora 10:00 in EG305, cu cealalta semigrupa.
- Site oficial curs: TBD
- Site de rezerva: http://bbftppro.myftp.org/PCLP1/

Mod de notare

Pe parcurs:

- 10p - minim 5 prezente curs.
- 30p - activitate laborator.
- 30p - lucrare de verificare (sapt 7-8).
- 20p - tema de casa (un program in Python care sa contina principalele instructiuni bine comentate; discutii).

Verificare finala:

- 20p - test

TOTAL: 110 puncte

Ce vom prezenta

- **Python, sem. 1**
  - Cum se executa programele
  - Tipuri de date si operatii
  - Tipuri numerice
  - Tipuri dinamice
  - Stringuri, liste, dictionare
  - Tuple, fisiere, instructiuni
  - Atribuiri, expresii, print, if
  - Instructiuni repetitive, while, for
  - Iteratii
  - Documentarea programelor
- **Python, sem. 2**
  - Functii si generatori
  - Argumente, notiuni avansate - functii
  - Colectii iterative si generatori
  - Benchmarking
  - Module si pachete - baza
  - Pachete si importari
  - Module - avansat
  - Clase si OOP
  - Clase – detalii, overloading
  - Proiectarea cu clase
  - Exceptii, obiecte de tip exceptie
  - Notiuni avansate, Unicode, atribute, decoratori, metaclase

Sumar

- Syllabus & Curriculum
- ***Avantajele Python***
- Executia in Python
- Cum se ruleaza programele Python

Avantajele folosirii Python

- Calitate superioara software – usor de citit/inteles.
- Productivitate – in dezvoltare
- Portabilitate – toate platformele de operare
- Biblioteci standard ample – ex. NumPy
- Integrare cu alte limbaje de programare

Python – limbaj pentru script-uri

Este un limbaj orientat obiect pentru script-uri, folosit la:

- Script-uri pentru sisteme de operare – *shell scripts.*
- Limbaj de control – al unor aplicatii complexe.
- Usor de utilizat – pentru implementari rapide.

Dezavantaje Python ?

- Viteza de executie – este interpretat nu compilat.
- Foloseste cod intermediar – *bytecode*.
- Versiunea *PyPy* extinde compilarea mai aproape de codul masina, rezultand viteza.
- Viteza de dezvoltare in Python e mai importanta.
- Se pot folosi extensii compilate pentru maximum de viteza.

Utilizatori ai limbajului Python

- Google – pentru cautare web
- YouTube – serviciul de partajare video
- BitTorrent – peer-to-peer partajare fisiere
- NSA – pentru criptare si analiza informatiilor
- Netflix si Yelp – pentru infrastructura software
- Intel, Cisco, HP, Qualcomm, IBM – teste hardware
- NASA, Los Alamos, Fermilab, JPL – pentru programare stiintifica.

Domenii de utilizare ale Python

- Programare de system – portabila
- Programare GUI – *tkinter*, interfata OO pentru Tk GUI API.
- Programare in Internet:
  - Client-server
  - HTML
  - Server-side – *mod_python* pt. Apache.
  - Creare de site-uri web – cu *web2py*
  - Aplicatii RIA (rich Internet apps)

Domenii…

- Integrarea software – componente C in Python sau cod Python integrat in alt sistem, ex. *Cython*.
- Programarea bazelor de date – relationale: Sybase, Oracle, ODBC, MySQL, PostgreSQL si non-SQL - *pickle*.
- Modelarea sistemelor – cu prototipuri, rapida.
- Programare stiintifica si numerica – *NumPy*, *SciPy*
- Jocuri – *pygame*, prelucrare imagini – *PyOpenGL*, Data Mining – *Orange*, Robotica – *PyRo*, inteligenta artificiala – *PyBrain*, monitorizare retea – *zenoss*, programarea telefoanelor mobile – pt. Android, IOS, etc.

Suport si dezvoltare in Python

- Este un sistem *open source*; se folosesc PEP – Python Enhancement Proposals pentru schimbari.
- PSF – Python Software Foundation, organizeaza conferinte – PyCon, OSCON (organizat de O'Reilly)
- Dezvoltare dinamica dar si uneori haotica; versiunea 2.X este stabila, 3.X in continuu progres.

Caracteristici tehnice ale Python

- Orientat Obiect dar si Functional (pe langa modelul procedural clasic)
- Free-ware, disponibil in sursa, dar bine suportat de comunitate. Autor initial: Guido van Rossum (BDFL pana in 2018)
- Portabil – scris in ANSI C, sub toate SO.
- Puternic:
  - tipuri de date dinamice
  - management automat al memoriei

Caracteristici…

- include module, clase, exceptii.
- tipuri de date predefinite – liste, dictionare, strings cu dimensiuni variabile, imbricate.
- operatii predefinite – concatenare, slicing, sortari, mapari.
- biblioteci extinse – expresii regulate, networking.
- utilitati extinse, provenite din contributii externe.

- Usor de mixat cu alte limbaje
- Usor de utilizat
- Usor de invatat.

Comparatie cu alte limbaje

- E mai puternic decat ***Tcl***.
- Mai usor de citit decat ***Perl***.
- Mai usor de folosit decat *Java* sau *C#*.
- Mai simplu decat *C++* (1/3 – 1/5 din marimea codului)
- Mai simplu dar si de nivel mai inalt decat *C*.
- Mai general decat ***PHP*** (website).
- Mai puternic si general decat ***JavaScript***.
- Are sintaxa mai simpla decat ***Ruby***.
- Mai accesibil decat ***Lisp***, ***Prolog***.

Sumar

- Syllabus & Curriculum
- Avantajele Python
- ***Executia in Python***
- Cum se ruleaza programele Python

Executia programelor in Python

Interpretorul de Python – format din interpretorul propriu-zis si biblioteci de support.

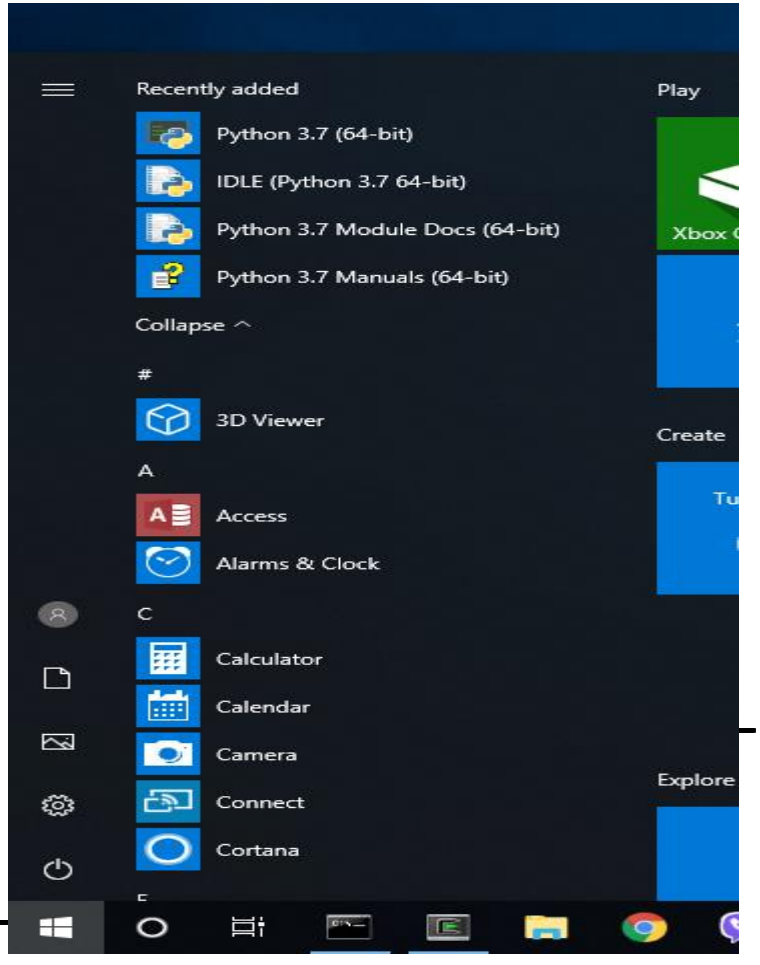
Se descarca (python-3.7.4-amd64.exe) de la www.python.org.

Sub Windows se instaleaza:

- Python 3.7 (64-bit) – interpretor in linie de comanda
- IDLE (Python 3.7 64-bit) – interpretor in mod GUI
- Module Docs – documentatie on-line
- Manuals – documentatie locala.

Asemanator, sub Linux exista pachete de instalare, ex: *python_2.7.16-1_amd64.deb* (cu *apt*, *yum*, *rpm*), sub Mac OS ex: *python-3.7.4-macosx10.6.pkg*

Poate fi preinstalat pe diverse platforme (Linux).



Primul program in Python

print('Hello, World!')

Instructiunea este cuprinsa intr-un fisier text, de obicei cu extensia **.py**

Se executa prin click pe numele fisierului, din IDLE (GUI) interactiv:

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

*>>> **print('Hello, World!')***

Hello, World!

>>>

sau linia de comanda:

*C:\Users\Dan\Desktop\code>**python first.py***

Hello, World!

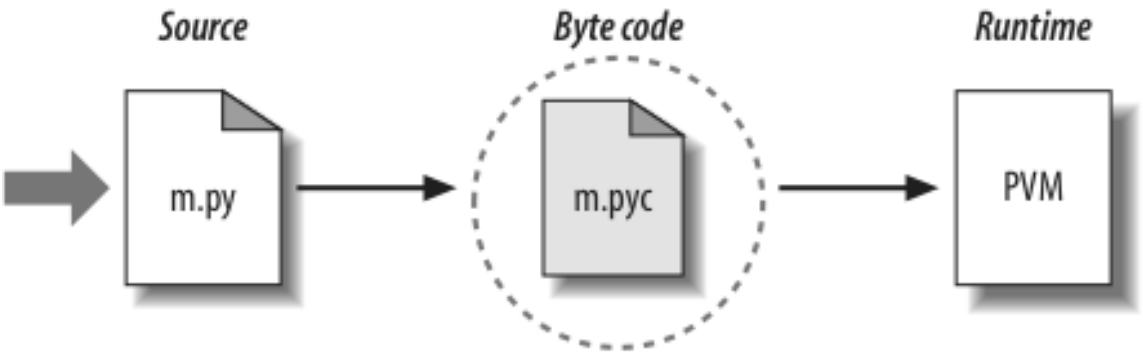
Modelul executiei in Python

Fazele executiei:

- Compilarea sursei (fisier .py), se obtine formatul ***byte code*** – intr-un fisier cu extensia **.pyc**, avand acelasi nume si locatie ca sursa, sau aflat in subdirectorul *__pycache__* (in 3.X), in afara de nume putandu-se adauga si versiunea de python ex: *first.cpython-37.pyc;* doar pentru fisierele importate!

Modelul…

- Executia propriu-zisa se face de catre **PVM – *Python Virtual Machine:***
  - Viteza de executie este intrucatva mai redusa
  - Dezvoltarea programelor este mai rapida – fara compilare, *make*.



Implementari ale Python

- **CPython** – implementarea standard (C); de la www.python.org
- **Jython** – se integreaza cu limbajul *Java* la nivel de Java byte code; de la http://jython.org
- **IronPython** – integrare cu Microsoft *.NET* framework; de la http://ironpython.net

Implementari…

- **Stackless** – Python pentru programe concurente, vezi http://stackless.com
- **PyPy** – Python de performanta, care foloseste compilare JIT (just in time), portiuni din *byte code* sunt inlocuite la rulare cu cod masina; excutie de 10-100X mai rapida a codului, vezi http://pypy.org.

Optimizari ale executiei cu:

- **Cython** – este un hibrid Python/C care permite compilarea completa via un Python/C API (application programming interface), vezi http://cython.org.
- **Shed Skin** – translator de la Python la C++
- **Psyco** – un compilator JIT specializat in generarea de cod masina bazat pe tipuri de date.

Frozen Binaries

Sunt imagini binare (executabile) care combina byte code cu PVM si alte fisiere necesare rezultand un pachet executabil (.exe) care nu necesita instalatii Python la executie.

Se creeaza cu:

- **py2exe** sub Windows
- **PyInstaller** sub Linux si Mac OS
- **p2app** pentru aplicatii Mac OS
- **freeze** – originalul.

Sumar

- Syllabus & Curriculum
- Avantajele Python
- Executia in Python
- ***Cum se ruleaza programele Python***

Cum se ruleaza programele Python

- Folosind o sesiune interactiva, in linia de comanda, intr-o fereastra DOS (*cmd*), consola.

C:\Users\Dan\Desktop\code>**python** *&rem dar **%PATH%** trebuie sa fie correct configurata*

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> **^Z** #asa se iese, Ctrl-Z urmat de Enter

C:\Users\Dan\Desktop\code>**py** *&rem **py** si **pyw** sunt deja in %PATH% de sistem.*

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> ^Z

C:\Users\Dan\Desktop\code>**py -3** *&rem se forteaza versiunea 3.X*

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> ^Z

Exemple de sesiune interactiva

- Comenzile introduse la promptul **>>>** se executa imediat:

```
>>> print( 'Hello, World!' )  #afisarea stringului dintre apostrofi
Hello, World!
>>> print( 2 ** 100 )          # 2 la puterea 100, cu operatorul – **
1267650600228229401496703205376
>>>
```

- Experimentarea instructiunilor noi:

```
>>> 'spam!' * 8               #repetitie a unui string prin multiplicare – *
'spam!spam!spam!spam!spam!spam!spam!spam!'
>>>
```

- Testarea codului din module – fisiere cu extensia .py:

```
>>> import os                 #importul modulului os
>>> os.getcwd()               #get current working directory
'C:\\Users\\Dan\\Desktop\\code'
```

Reguli ale sesiunii interactive

- Tipariti doar comenzi Python, nu comenzi ale sistemului de operare – >>> *os.system('dir')* #asa se executa un ***dir*** din Python
- Instructiunea *print()* poate fi omisa interactiv, dar e necesara in fisiere Python.
- Nu indentati instructiunile simple, ci doar pe cele compuse.
- Atentie la promptul **…** care marcheaza continuarea unei instructiuni pe mai multe linii
- Incheiati instructiunile compuse cu o linie alba, Enter/Enter:

```
>>> for x in 'spam':      #instructiune compusa, for
...     print( x )        #instructiune indentata, cu un tab sau spatiu (dupa cele 3 puncte)
...
s
p
a
m
```

Script-uri in Python

- Scriptul editat in fisierul **script1.py**:

```
# Un prim script Python
import sys                  # Incarca un modul de biblioteca, sys denumit.
print(sys.platform)
print(2 ** 32)              # Ridica 2 la putere(a 32)
x = 'Spam!'
print(x * 8)                # Repetarea unui string (de 8 ori)
```

- Contine comentarii (# marcheaza inceputul comentariului).
- Se poate executa in linia de comanda:

C:\Users\Dan\Desktop\code>**python script1.py**

win32

4294967296

Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!

Script-uri…

- Sub Windows, numele scriptului este destul pentru executie:

C:\Users\Dan\Desktop\code>**script1.py**

win32

4294967296

Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!

- Sub Unix, Linux, Mac OS dar si sub Windows (3.X), prima linie poate indica interpretorul – cu **#!**, iar scriptul poate fi facut executabil (*chmod +x script1.py*):

#!/usr/local/bin/python - cale/path completa

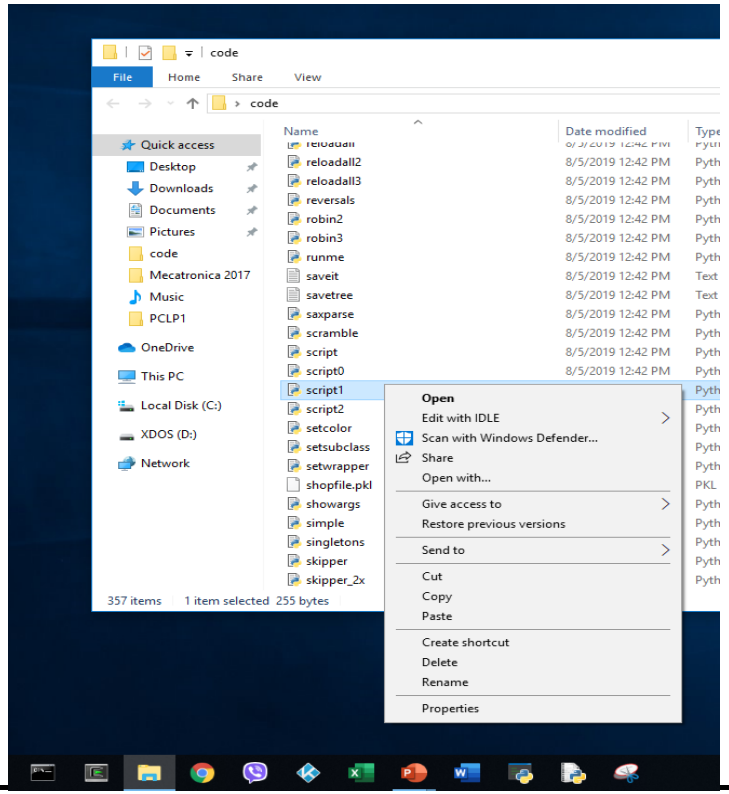
sau

#!/usr/bin/env python - cu cda. *env* care gaseste python pe path

Executia la promptul liniei de comanda: *$* ***script1.py***

Executia prin click pe icon

- Sub Windows, fisierele cu extensia .py (sau .pyc)se pot executa prin (double-)click de mouse, insa output-ul sau/si mesajele de eroare se pot pierde.



- Prin includerea instructiunii ***input()*** (3.X) sau ***raw_input()*** (2.X) executia se incheie doar la tastarea unui *Enter:*

```
>>> input( 'Apasati Enter pentru a termina ' )
Apasati Enter pentru a termina
''
>>> #exemplu cu mesaj, optional.
```

Dezavantaje click

- Output-ul se pierde din cauza executiei rapide.
- Mesajele de eroare se pierd inainte de a se ajunge la un *input()* final.
- Prin tratarea exceptiilor de executie (*try*) sau redirectarea output-ului (*print*) disparitiile se pot evita.

Importarea si reincarcarea modulelor

- Reprezinta metode de executie a programelor Python:

C:\Users\Dan\Desktop\code>python

>>> ***import script1***

win32

4294967296

Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!

- *reload()* este necesar, dupa modificari ale sursei, deoarece se executa doar un singur import per sesiune(operatie costisitoare):

>>> ***from importlib import reload***

>>> ***reload(script1)***

win32

126765060022822940149670320537 6

Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!

<module 'script1' from 'C:\\Users\\Dan\\Desktop\\code\\script1.py'>

Importarea…

- Prin importarea unui modul se acceseaza atributele sale:

```
>>> import os
>>> os.system( 'type myfile.py' )
titlu = "O scrisoare pierduta"
0
>>> import myfile
>>> myfile.titlu
'O scrisoare pierduta'
>>> #sintaxa este obiect.atribut
```

```
>>> os.system( 'type threenames.py' )
a = 'nou'                # Se definesc trei atribute
b = 'student'            # Exportabile catre alte fisiere
c = 'universitate'
print(a, b, c)           # Se executa aici si un print()
0
>>> import threenames
nou student universitate
>>> threenames.b, threenames.c
('student', 'universitate')
>>> from threenames import a, b, c
>>> b, c
(‘student', 'universitate’)
>>> #sunt tuple!
```

Importarea, continuare

>>> ***dir(threenames)*** #dir(), instructiune predefinita

['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'a', 'b', 'c']

>>> #***lista*** de attribute ale modulului *threenames, __X__ sunt attribute predefinite (dublu underscore)*

- Modulele reprezinta “spatii de nume” – *namespace* independente.
  - Modulele asigura separarea numelor de variabile, numite la fel in diferite module.
  - Combinatia *from/import* “infrange” aceasta separatie:

    >>> ***a = 33***

    >>> ***from threenames import a, b, c***

    nou student universitate

    >>> ***a***

    'nou'

Folosirea *exec()* pentru rulare

- **exec()** este o instructiune predefinita care executa versiunea curenta a unui modul:

```
>>> exec( open( 'script1.py' ).read() )
win32
126765060022822940149670320537 6
Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!
```

- Nu este import, deci variabilele pot fi inlocuite:

```
>>> x = 99
>>> exec( open( 'script1.py' ).read() )
win32
126765060022822940149670320537 6
Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!
>>> x
'Spam!'
```

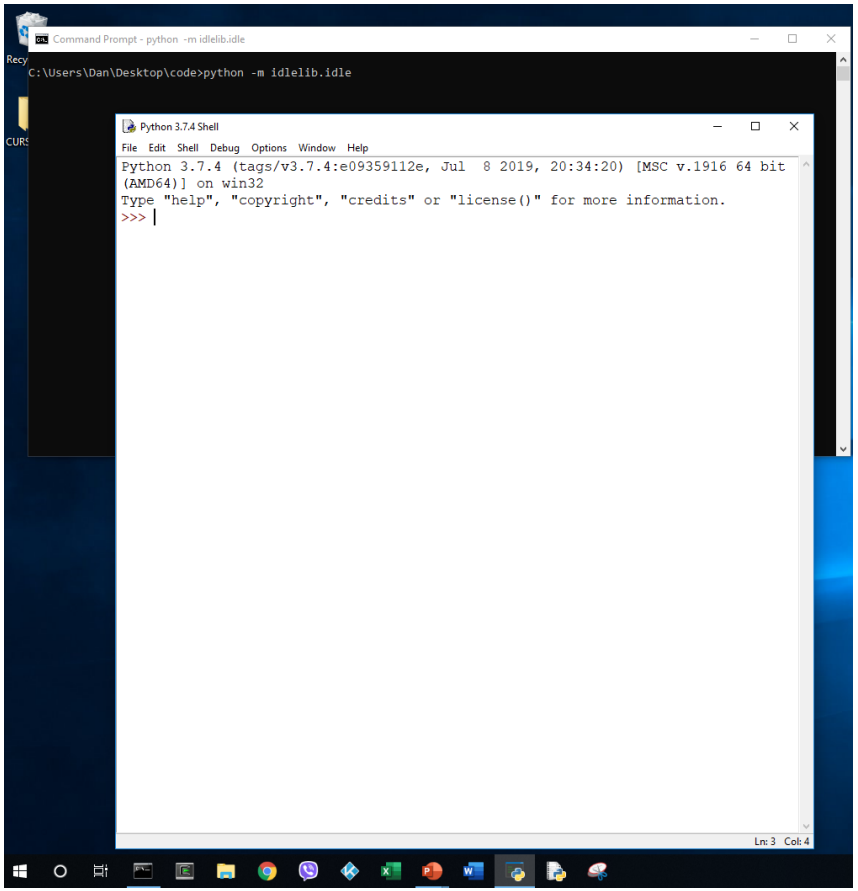
- In **Python 2.X** exista: *execfile('modul.py')* si *exec(open('modul.py'))* cu acelasi effect
- exec(open().read()) este de asemenea corecta.

IDLE

- Este un mediu integrat de dezvoltare (IDE) a programelor in Python.
- Este instalat automat pe toate platformele.
- Se porneste din meniul *Start* sau cu comanda:

C:\>***python -m idlelib.idle***

C:\>*rem optiunea **-m** permite executia unui modul de biblioteca (idle) ca script Python.*



IDLE…

- IDLE permite:
  - executia interactive la promptul >>>
  - editeaza scripte noi cu *File->New*
  - codul este colorat
  - executa un script deschis in editare (*File->Open*) cu *Run->Run Module,* iar output-ul se regaseste in fereastra principala.
  - auto indenteaza, auto completeaza.
  - instructiuni anterioare se reiau prin plasarea cursorului pe respectiva linie si Enter
  - Atributele obiectelor si argumentele unui apel de functie sunt afisate in ferestre *pop-up*, pe parcursul editarii.
  - **Dezavantaje:**
    - programe GUI Python/tkinter cu fire de executie multiple – multithreading – pot bloca IDLE.
    - variabilele sunt importate automat in IDLE, nu in Python
    - calea de cautare a modulelor se actualizeaza automat, nu in Python

Alte IDE

- **Eclipse** cu *plug-in*-ul **PyDev** – initial un Java IDE.
- **Komodo** – ofera fisiere de proiectare (project files); vezi https://www.activestate.com/
- **NetBeans IDE** – pentru Python, suporta CPython si Jython.
- **PythonWin** – extensii ale IDLE, ex. obiecte COM.
- **Wing, Visual Studio**, etc.

Alte metode de rulare Python

- Apeluri de cod Python inclus (*embedded*) in alte limbaje, ex. C:

```
#include <Python.h>


Py_Initialize();
Python_SimpleString("x = 'facultatea ' + 'ISB'" ); //cod Python inclus in C
```

- Executabile binare *frozen*
- Editoare, ex. *emacs,* care lanseaza codul in executie din editor.
- Metode specifice diverselor platforme, ex. drag-and-drop (MAC)
- Din pagini web, script server-side.

Depanarea programelor Python

- Cu ajutorul mesajelor de eroare
- Inserand instructiuni *print()*, temporar.
- Cu *debugger*-ul IDE-ului folosit (meniul *Debug* in IDLE)
- Cu ***pdb***, in linia de comanda.
- Cu optiunea ***-i***, *python -i modul.py*, care trece excutia in modul interactiv la incheierea executiei scriptului.
- Cu ***Winpdb***, in regim GUI si de consola.