Utilizzo di Python in una startup AI

Mi presento

- Ex-ricercatore universitario, ho vissuto per 10 anni nei Paesi Bassi.
- Non ho una formazione informatica ma negli ultimi 10 anni ho utilizzato Python quotidianamente.
- Al momento lavoro in una startup di cui sono cofondatore.

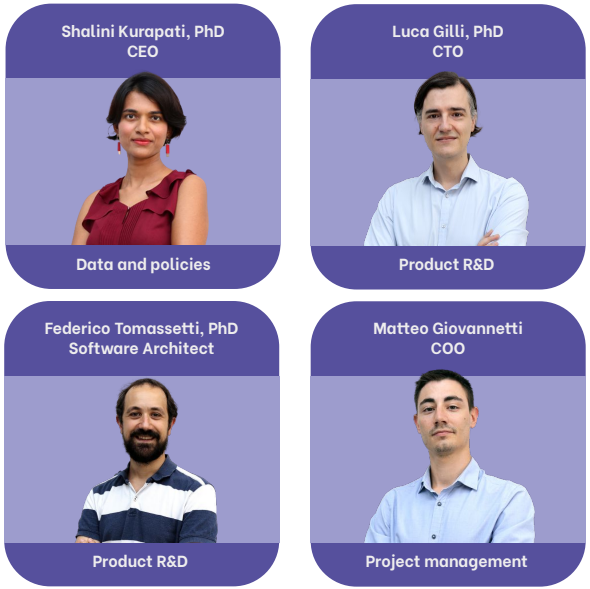
Clearbox AI

Startup ospitata presso l'incubatore di imprese del Politecnico di Torino.

Idea originale → fornire strumenti basati su dati sintetici per analisi di modelli di Machine Learning.

Forte componente R&D.

Founding Team

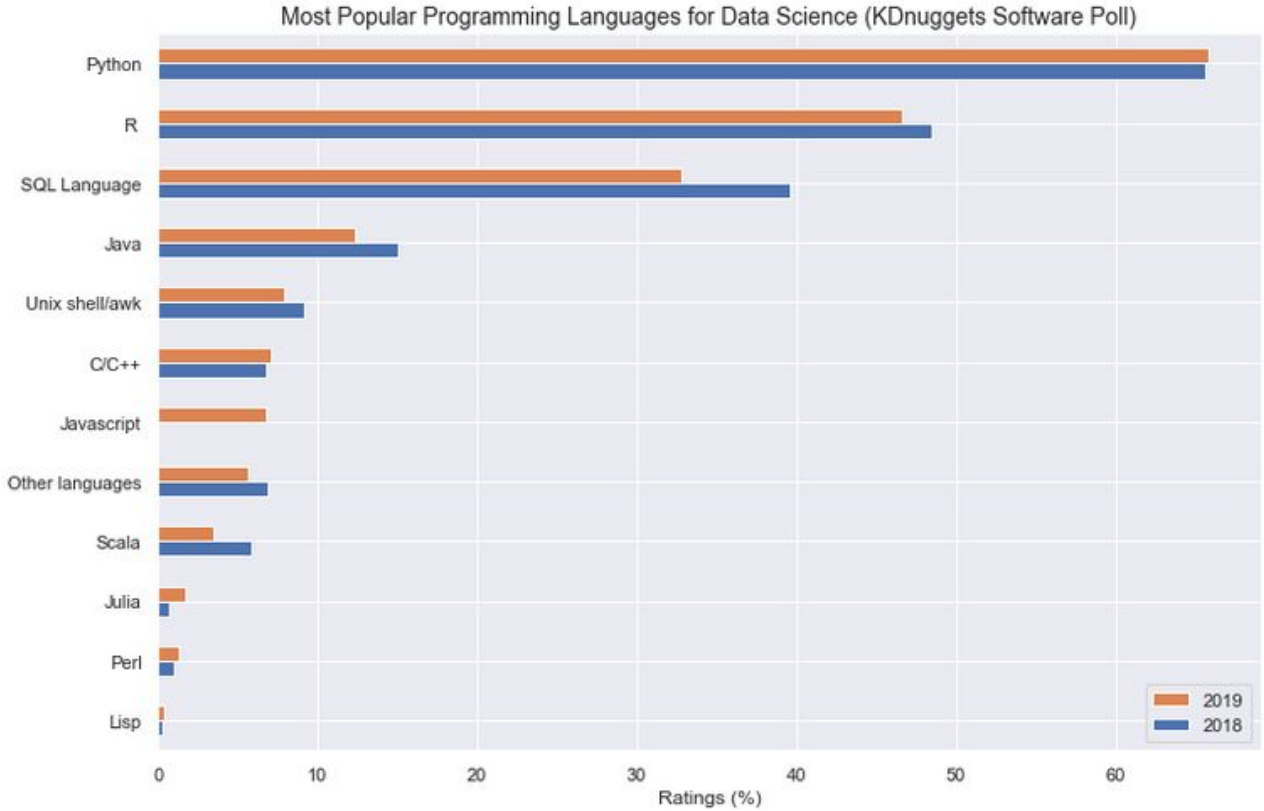


Trusted by

CHU de Liège · ITU · Fraunhofer · TU Delft · Capgemini · Horsa Group · PlugAndPlay

Python in data science

Python in data science



https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

Python in data science

Ragioni per cui Python e' cosi' popolare in ambito data science:

1. Relativamente semplice da imparare.

2. Qualunque sia ambito applicativo, esistono librerie open-source basate su Python.

Python

```python
# Program to check if a number is prime or not

num = int(input("Enter a number: "))
flag = False

if num > 1:
    for i in range(2, num):
        if (num % i) == 0:
            flag = True
            break

if flag:
    print(num, "is not a prime number")
else:
    print(num, "is a prime number")
```

C

```c
// Program to check if a number is prime or not

#include <stdio.h>

int main() {
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 2; i <= n / 2; ++i) {
        if (n % i == 0) {
            flag = 1;
            break;
        }
    }

    if (n == 1) {
        printf("1 is neither prime nor composite.");
    }
    else {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }

    return 0;
}
```

https://medium.com/aiadventures/why-python-af7b663e6e08

Python come strumento per la data science

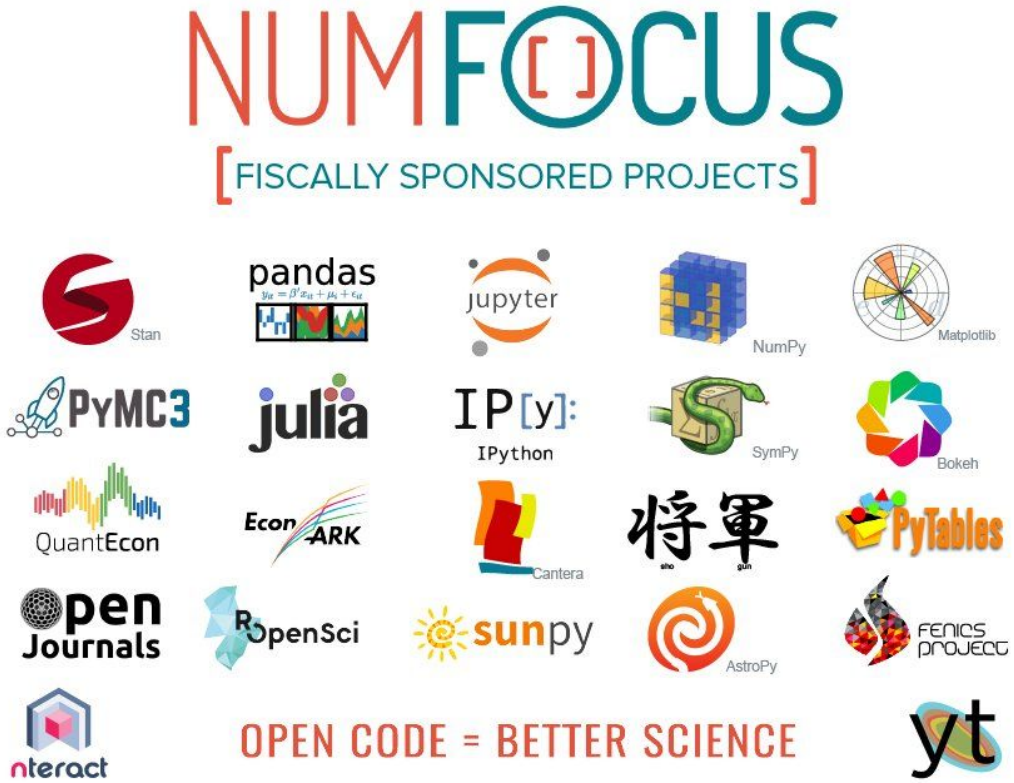
Calcolo vettoriale, matriciale, algebra lineare.

Librerie per il calcolo scientifico in diversi campi.

NumFOCUS

Organizzazione non-profit creata per supportare sviluppo di software scientifico open-source.

Supportata da grandi compagnie (Amazon, Microsoft, etc)



Limiti Python
(non insormontabili)

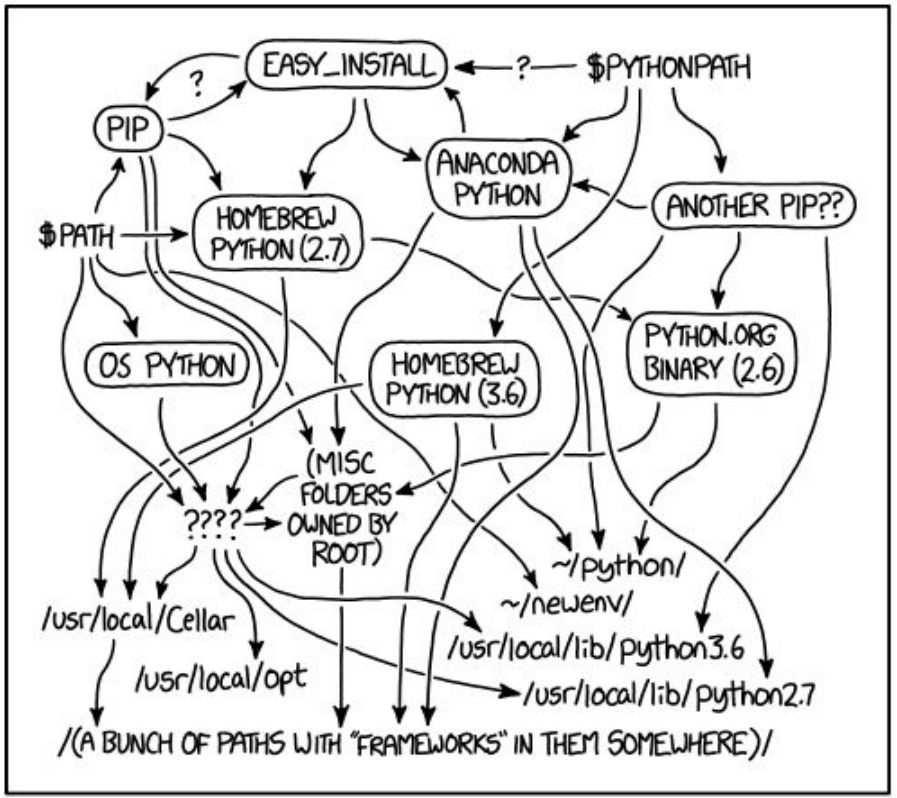
Riproducibilità

La grande abbondanza di librerie e risorse tende a far diventare ambienti di sviluppo caotici e spesso difficili da riprodurre.

Riproducibilità

La grande abbondanza di librerie e risorse tende a far diventare ambienti di sviluppo caotici e spesso difficili da riprodurre.

Sia in ambito software che in ambito machine learning dovremmo essere in grado di ottenere gli stessi risultati in ambienti di produzione diversi, non sempre scontato.



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

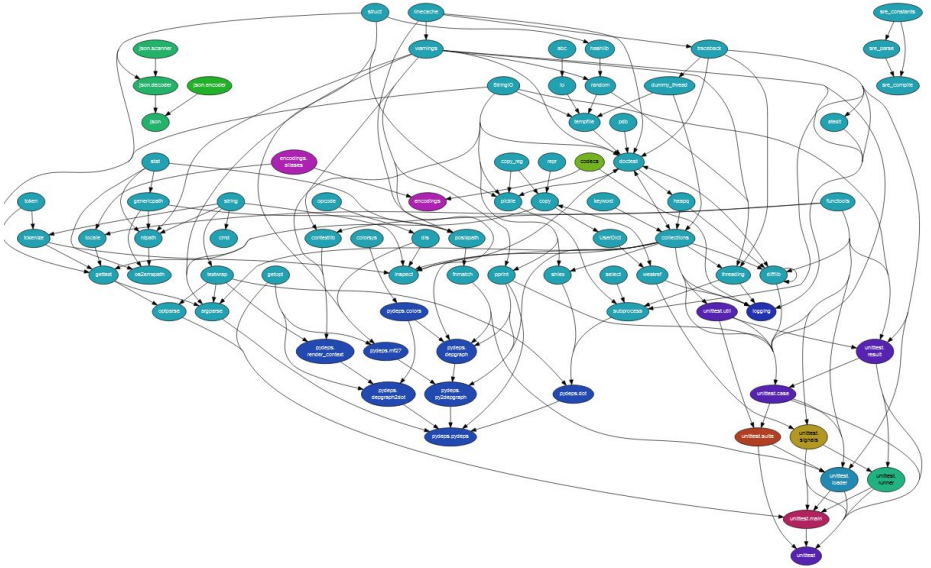
https://xkcd.com/1987/

Gestione dipendenze

Passaggio da sviluppo a messa in produzione e' spesso reso difficile da problemi legati alla gestione delle dipendenze.

Caso estremo: aggiornamento di una piccola libreria secondaria puo' rompere applicazione.

Diversi strumenti per migliorare la situazione: pipenv, poetry.

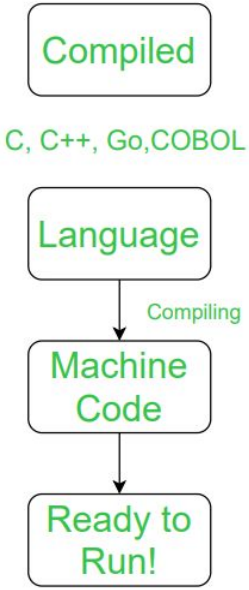


https://github.com/thebjorn/pydeps

Velocità di calcolo

Velocità di calcolo

Linguaggi compilati vs linguaggi interpretati



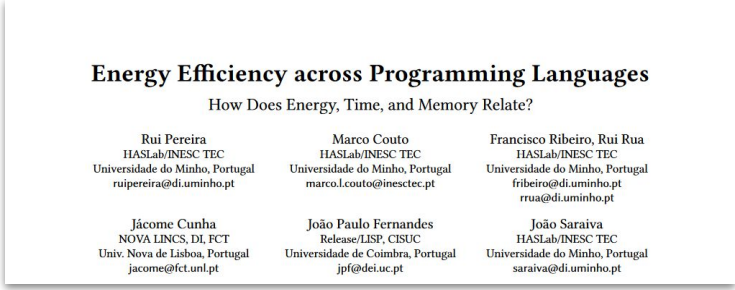
https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/

Velocità di calcolo

Il fatto che Python sia un linguaggio interpretato e dinamicamente tipizzato lo rende molto più lento rispetto a linguaggi come C o Fortran.

Impatto ambientale

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84



Energy Efficiency across Programming Languages

How Does Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

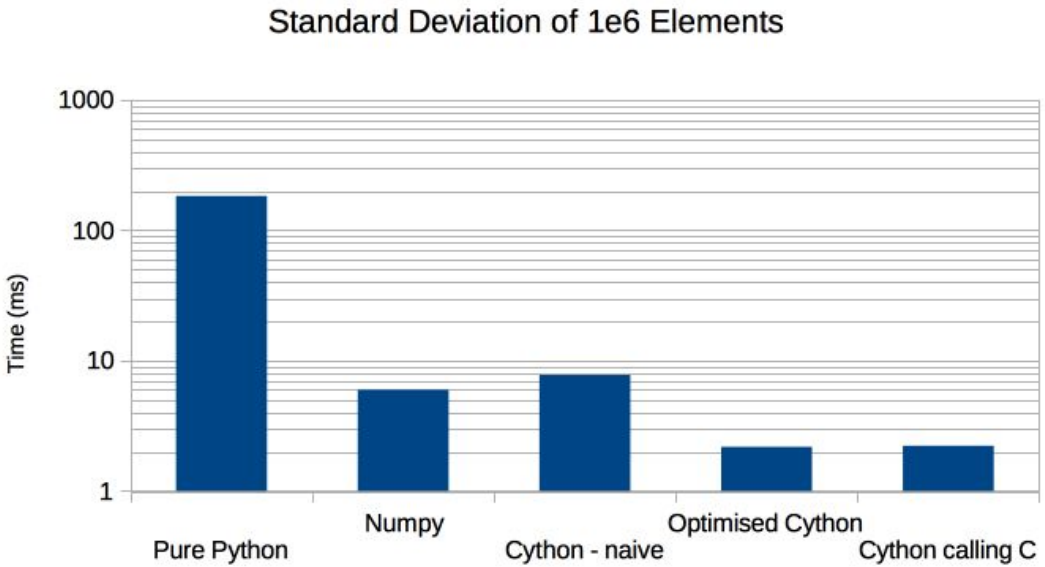
João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSLE.pdf

Cython

Esistono diversi tool creati per rendere Python più performante.

Uno dei più popolari è Cython. Cython permette di compilare funzioni Python usando un linguaggio ibrido tra C e Python.



https://notes-on-cython.readthedocs.io/en/latest/std_dev.html

Cython

Pochi accorgimenti nella scrittura di funzioni Cython permettono di migliorare drasticamente velocita'.

```
def f(x):
    return x**2-x

def integrate_f(a, b, N):
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

```
cdef double f(double x):
    return x**2-x

def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s, x, dx
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

https://www.infoworld.com/article/3250299/what-is-cython-python-at-the-speed-of-c.html

Python in startup AI

- Ormai il linguaggio da usare quando si lavora in ambito Data Science/Machine Learning.
- Comunità open-source senza rivali.
- Limiti legati al linguaggio possono essere superati con adeguati accorgimenti.

clearbox AI
humans && machines

Thanks for Reading

Feel free to contact us:


www.clearbox.ai


luca@clearbox.ai


@ClearboxAI