

Linguaggi di Programmazione



Barriere di Linguaggio

Ci interessano gli elaboratori elettronici **digitali** (computer)

- Informazioni codificate mediante numeri binari (0/1)
- Operazioni elementari piuttosto semplici (specifica binaria)

Il linguaggio nativo di un computer si chiama **linguaggio macchina**

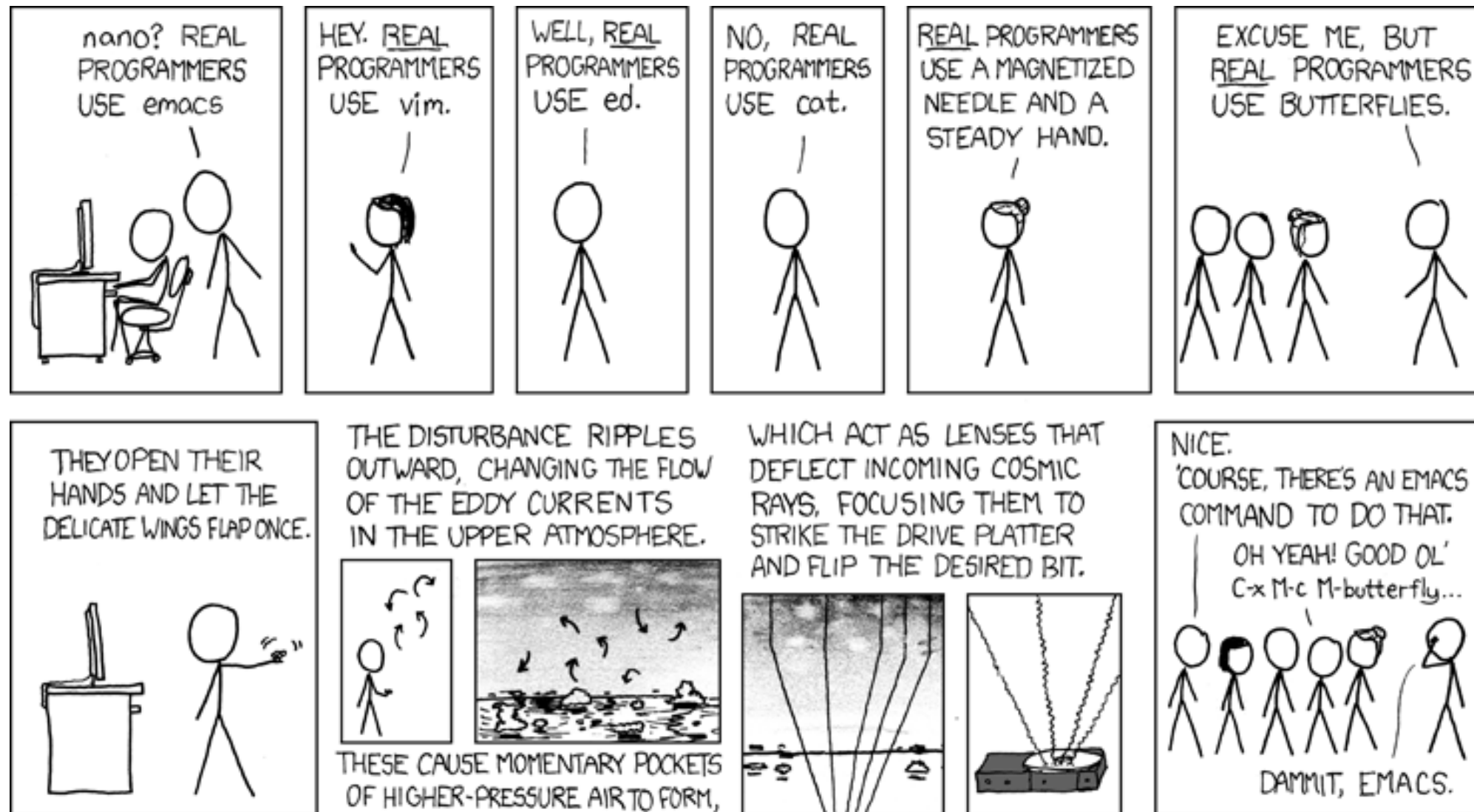
...Consiste in sequenze di 0/1, di solito visualizzate così:

000034f0	54 b5 dd 4d e8 bf 0a 3c c6 5d ee 0b 97 3d b7 11	TpYMèç.<E]î.-=.
00003500	91 72 e9 f5 96 89 80 3c 9a 0b 31 ce a6 fb f4 f9	'réð-¾E<š.1Î ûôù
00003510	1c 04 df 70 96 e6 5d c2 b9 65 25 cb 9f 0e e7 23	..Bp-æ]Â¹ešEÿ.ç#
00003520	ca 6c 62 8c dd a6 1a db dc 19 d3 f1 fd 06 27 33	ÊlbœY .ÔÜ.Óñý.'3
00003530	a8 21 dc ce a3 53 94 6f 0c 05 8e ad 62 f9 3a b5	~!ÜîšS"o..Ž-bù:µ
00003540	d9 df dd e8 a4 b0 49 7d 46 91 85 22 e2 4e 0e 79	ÙBYè«°I}F'..."âN.y
00003550	ed 6b 61 0d 1a e8 f4 da 6e 1e 20 00 8f 0b 5c 05	íka..èôÚn. . .\.
00003560	db 4f c5 45 09 d6 05 f3 ef c6 e8 78 e1 ef d5 e6	ÛOÂE.Ö.óîEèxáíÕæ
00003570	d7 93 a3 19 16 5d b3 d3 ba 46 38 6a c6 31 81 71	*"£..]'°Ó°F8jE1 q
00003580	60 dc bc 56 6a 02 9f 16 34 f7 31 6e 60 02 d2 7a	`Ü»Vj.ÿ.4÷1n`.Òz
00003590	fa ba 59 86 b5 48 bf 18 1b 0a 21 66 e6 48 95 15	ú°Y†µHç...!fæH*.
000035a0	e4 de 1d e7 59 64 59 23 bb cc c5 c0 af 16 4a a4	äP.çYdY#»îÄÄ°.J»
000035b0	f1 a4 79 7e af 45 71 5e 46 04 cf 63 74 c9 6d eb	ñxy~¬Eq^F.İctÉmä
000035c0	d8 11 9d 00 d9 c8 31 e7 13 28 8c 01 2a 83 3e 85	Ø. .ÙÈ1ç.(E.*f>...
000035d0	61 9f 98 a4 40 c3 ab c2 06 04 b5 fd 51 52 20 59	aÿ~»@Ä«Ä..µýQR Y
000035e0	4c 6d 62 f6 6e 1e e6 e8 ba 45 95 cb 89 0c 9a 74	Lmbön.æè°E•Ë%.št
000035f0	19 cf ee 56 be 30 96 13 87 86 30 8a 24 26 e4 60	.İîV%0-.+†0Š\$&ä`



Barriere di Linguaggio

Conseguenza: usare un computer in modo diretto è **difficile**



Linguaggi e Traduttori

Come venirne fuori (mentalmente sani)?

Idea: usare il linguaggio macchina per definire **un programma**:

- ...Che accetti un linguaggio **più ricco ed astratto**
- ...E lo **traduca** in linguaggio macchina

Un programma del genere si chiama **traduttore**:



Linguaggi e Traduttori

Con i traduttori possiamo definire linguaggi sempre più **astratti**

Senza traduttore: **linguaggio macchina**

```
0100 0000 0000 1011
0100 0000 0000 1001
0000 0000 0000 1000
```

- Istruzioni e dati in codifica binaria
- Spesso visualizzato con rapresentazione esadecimale
 - 0100 0000 0000 1011 → 400B
- Gestibile in memoria senza ulteriori codifiche
- Eseguitibile direttamente dal computer



Linguaggi e Traduttori

Con i traduttori possiamo definire linguaggi sempre più astratti

Con un traduttore semplice: linguaggio assembly

```
LOADA H  
LOADB Z  
ADD
```

- Tipi di dato nativi (gestibili direttamente dal computer)
- Associazione 1-1 con istruzioni binarie
- ...Ma rappresentazione simbolica!



Linguaggi e Traduttori

Con i traduttori possiamo definire linguaggi sempre più **astratti**

Con un traduttore avanzato: linguaggi **di alto livello**

```
a = 2
z = a + 2
print(f'{z}')
```

- Tipi di dato nativi e derivati
- Istruzioni complesse
- Relativamente semplici da comprendere

I linguaggi di alto livello sono sufficientemente astratti da essere **indipendenti dall'elaboratore fisico**



Linguaggi di Alto Livello

Raggiunto un livello di astrazione sufficiente:

- Un linguaggio si può basare su una **astrazione** dell'elaboratore
- Programmare diventa molto **più facile**
- Lo stesso programma può essere tradotto su **elaboratori diversi**

Un linguaggio (di programmazione) così si dice **di alto livello**

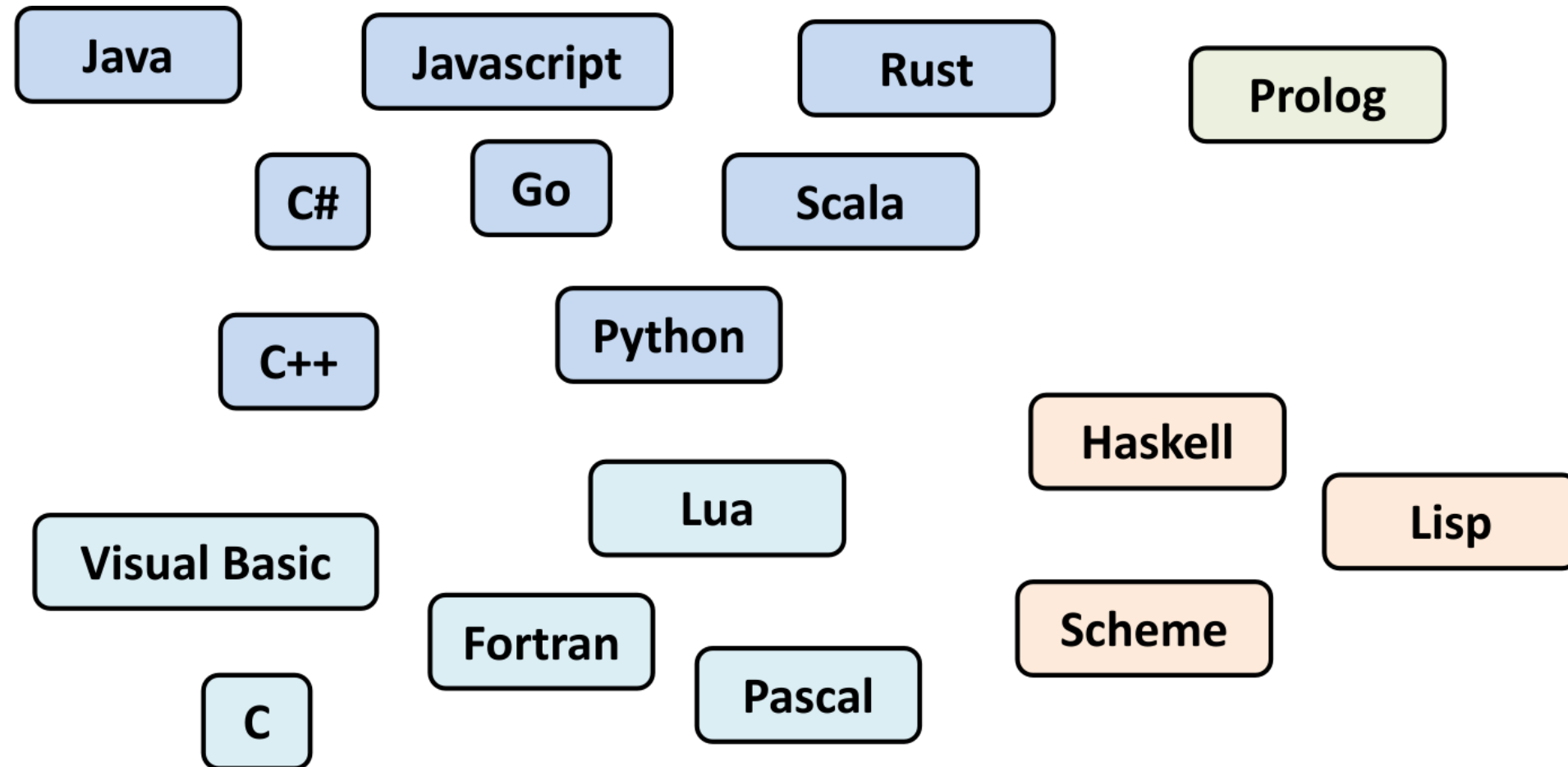
Questo tipo di decomposizione in livelli di astrazione

...È una delle tecniche più utilizzate in informatica per gestire la complessità

- L'abbiamo già vista in azione parlando del Sistema Operativo



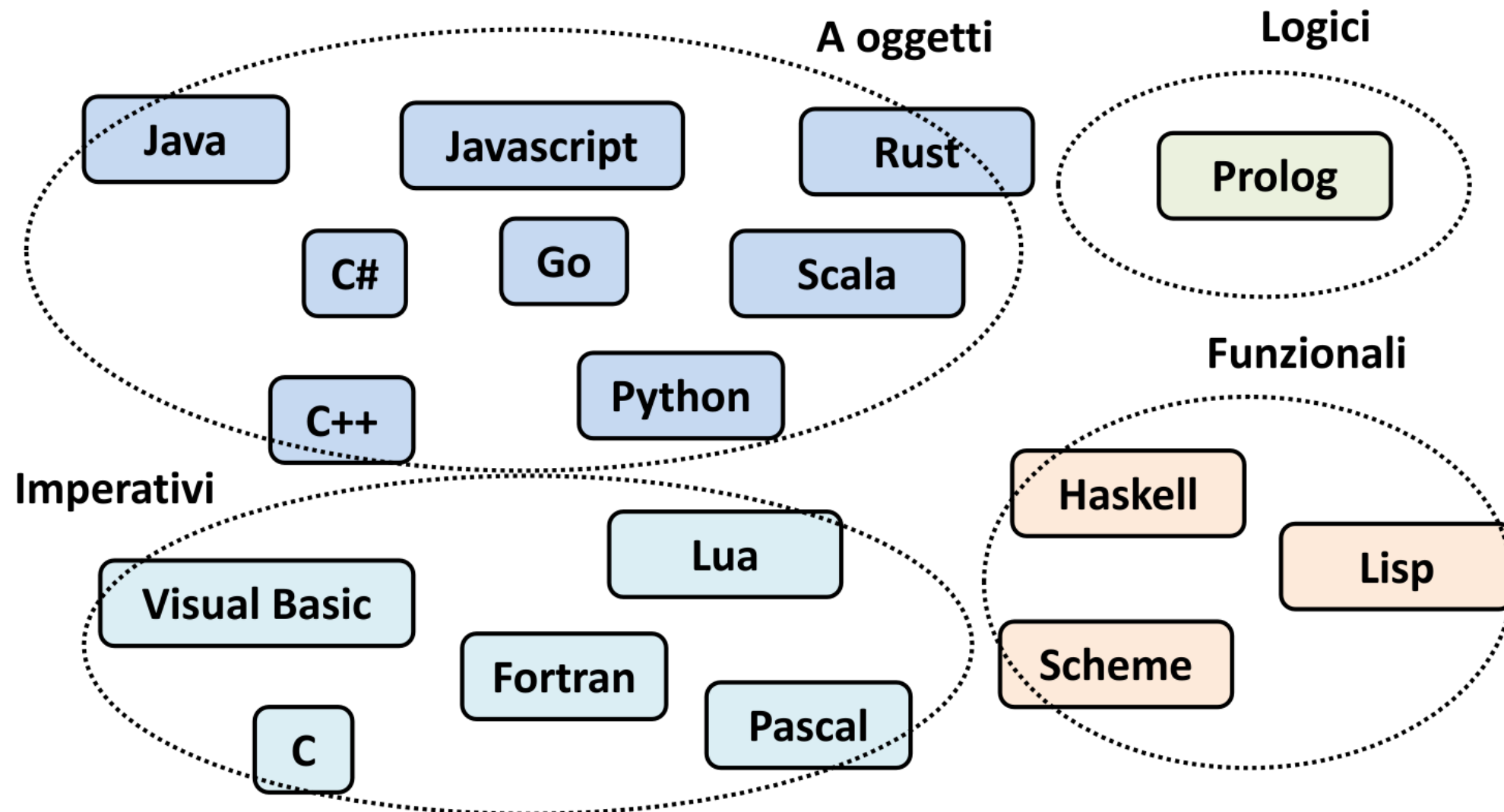
Linguaggi di Alto Livello: Tipologie



Elaboratore:



Linguaggi di Alto Livello: Tipologie



Elaboratore:



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi **imperativi**

Astrazione: macchina in grado di eseguire **sequenze di istruzioni**

```
int main() {  
    printf("Hello, World!");  
    return 0;  
}
```

- Il codice di esempio è in linguaggio C



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi a **oggetti**

Astrazione: tipi di **dati astratti** con **operazioni programmabili**

```
class Complex:
    def __init__(self, rval, ival):
        self.rval = rval
        self.ival = ival

    def __add__(self, c):
        return Complex(self.rval+c.rval, self.ival+c.ival)
```

- Il codice di esempio è in linguaggio Python
- ...Che è in realtà un linguaggio multi-paradigma...



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi **funzionali**

Astrazione: **funzioni** matematiche

```
((lambda (x) (+ x x)) 5)
```

- Il codice di esempio è in linguaggio Lisp

Linguaggi **logici**

Astrazione: **logica** matematica

```
likes(mary, food) .  
likes(john, wine) .  
?- likes(mary, X)
```

- Il codice di esempio è in linguaggio Prolog

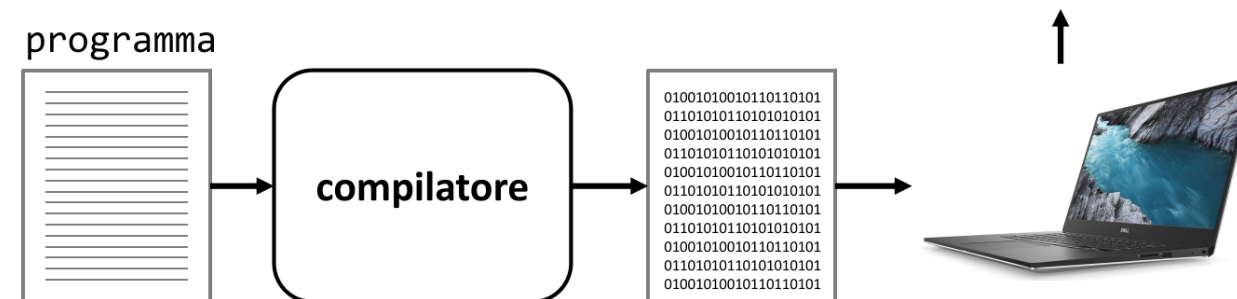


Tipi di Traduttori

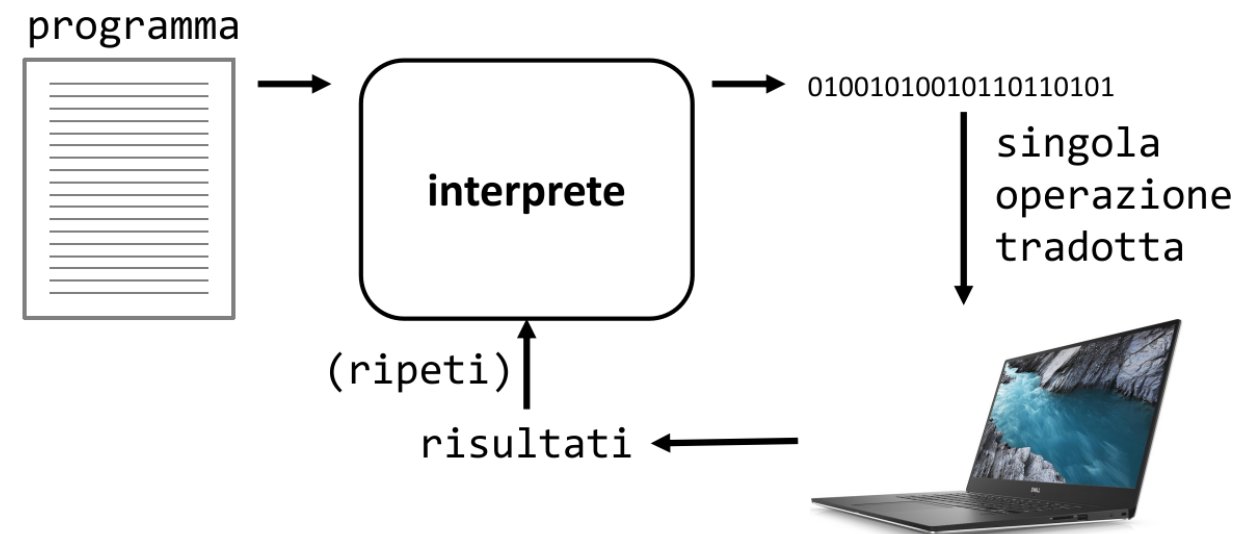
Distinguiamo **due categorie** principali di traduttori

...A seconda di quando avvenga l'esecuzione

■ **Compilatori:** l'intero programma viene tradotto e quindi eseguito



■ **Interpreti:** le operazioni vengono tradotte ed eseguite individualmente



Tipi di Traduttori

Le due tipologie hanno caratteristiche complementari

■ Compilatori:

- Programmare è tipicamente **più lento** (bisogna compilare)
- ...Ma l'esecuzione è rapidissima

■ Interpreti:

- Programmare è tipicamente **molto veloce**
- ...Ma l'esecuzione è più lenta

Esistono anche gradi intermedi

- E.g. compilazione per un interprete specializzato
- E.g. interprete + compilazione per specifiche parti del programma



Che Tipo di Linguaggio?

Ma che cosa intendiamo per linguaggio?

Dal dizionario:

“Un linguaggio è un insieme di parole e di metodi di combinazione delle parole usate e comprese da una comunità di persone.”

È una definizione **poco precisa**:

- Non evita le **ambiguità** dei linguaggi naturali
- Non descrive processi computazionali **automatizzabili**
- Non aiuta a stabilire **proprietà**



Che Tipo di Linguaggio?

In pratica, dobbiamo definire **due aspetti** di un linguaggio:

Si chiama **sintassi**...

- L'insieme di regole formali per la scrittura di programmi in un linguaggio
- ...Che dettano le modalità per costruire frasi corrette

Si chiama **semantica**...

- L'insieme dei **significati** da attribuire alle frasi

Ne conseguono due tipi di errori:

- Errori sintattici: la frase non può essere tradotta
- Errori semantici: la frase non ha il significato corretto



Sintassi e Notazione EBNF

Per specificare la sintassi useremo a volte la **notazione EBNF**

- ...Che sta per Extended Backus-Naur Form
- E.g. sintassi di un numero naturale

```
<naturale> ::= 0 | <cifra-non-nulla>{<cifra>}  
<cifra-non-nulla> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<cifra> ::= 0 | <cifra-non-nulla>
```



Sintassi e Notazione EBNF

Per specificare la sintassi useremo a volte la **notazione EBNF**

- ...Che sta per Extended Backus-Naur Form
- E.g. sintassi di un numero naturale

```
<naturale> ::= 0 | <cifra-non-nulla>{<cifra>}  
<cifra-non-nulla> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<cifra> ::= 0 | <cifra-non-nulla>
```

Come leggerla:

- "<simbolo>" = sostituibili secondo le regole indicate dopo il segno "::="
- "|" sta per "oppure"
- "{}" sta per "ripeti zero o più volte"
- "[]" opzionale



Sintassi e Notazione EBNF

Guardiamo meglio l'esempio

```
<naturale> ::= 0 | <cifra-non-nulla>{<cifra>}
```

- Un numero naturale si può riscrivere come 0
- ...Oppure come una cifra non nulla seguita da zero o più cifre

```
<cifra-non-nulla> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

- Una cifra non nulla è 1 o 2 o 3, etc.

```
<cifra> ::= 0 | <cifra-non-nulla>
```

- Una cifra è 0 oppure una cifra non nulla



Semantica

Per indicare la semantica useremo **le parole**, e.g.

```
time.sleep(secs)
```

Suspend execution of the calling thread for the given number of seconds.

...Oppure **esempi**, e.g.:

```
>>> year = 2016
```

```
>>> f'Results of the {year} referendum'
```

```
'Results of the 2016 Referendum'
```

O ancora meglio provando ad **eseguire** codice!

```
In [3]: year = 2016  
        f'Results of the {year} referendum'
```

```
Out[3]: 'Results of the 2016 referendum'
```



Linguaggio e Programma

Possiamo finalmente dare **due definizioni più precise**

Si dice **linguaggio di programmazione** una notazione formale
con una semantica eseguibile

Si dice **programma** un testo scritto in un linguaggio di
programmazione

- Un programma è spesso la codifica di un algoritmo
- ...Ma non necessariamente



Ambiente di Sviluppo

Per programmare quindi ci occorre:

- Un elaboratore e un linguaggio
- Un editor (di solito di testo) per il linguaggio
 - E.g. blocco note, VS Code...
- Un traduttore (compilatore o interprete)
- ...Ed in pratica un gestore di file (e.g. esplora risorse)

Oppure un programma che svolga più di un compito, i.e. un **IDE**

Integrated Development Environment (ambiente di sviluppo integrato)

- Un buon IDE per Python è per esempio Spyder
- ...Oppure quello che avete davanti in questo momento, i.e. Jupyter



Breve Introduzione al Linguaggio Python

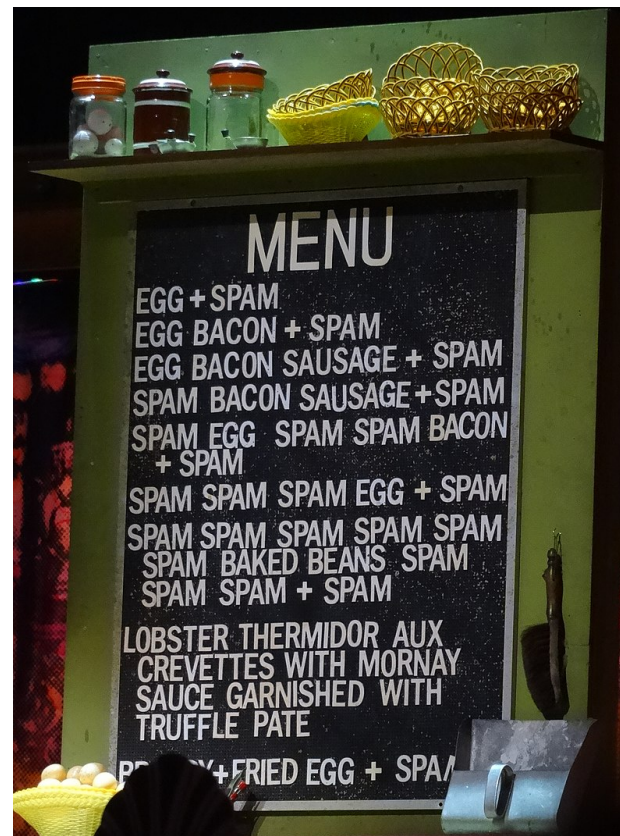


Linguaggio Python

Il linguaggio Python

- È stato progettato nel 1991 dall'Olandese Guido van Rossum
- È arrivato alla versione 3 (per la precisione 3.10)

Il nome "Python" viene dai Monthy Python



Linguaggio Python

Python è un linguaggio:

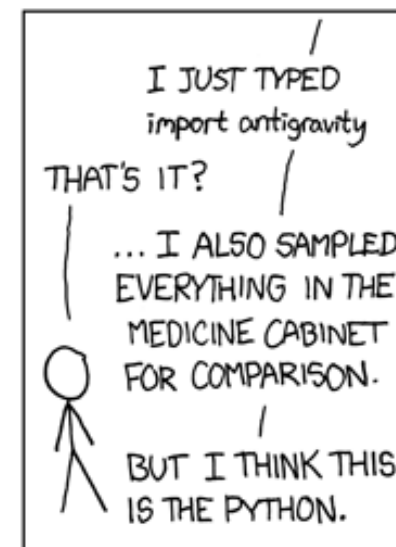
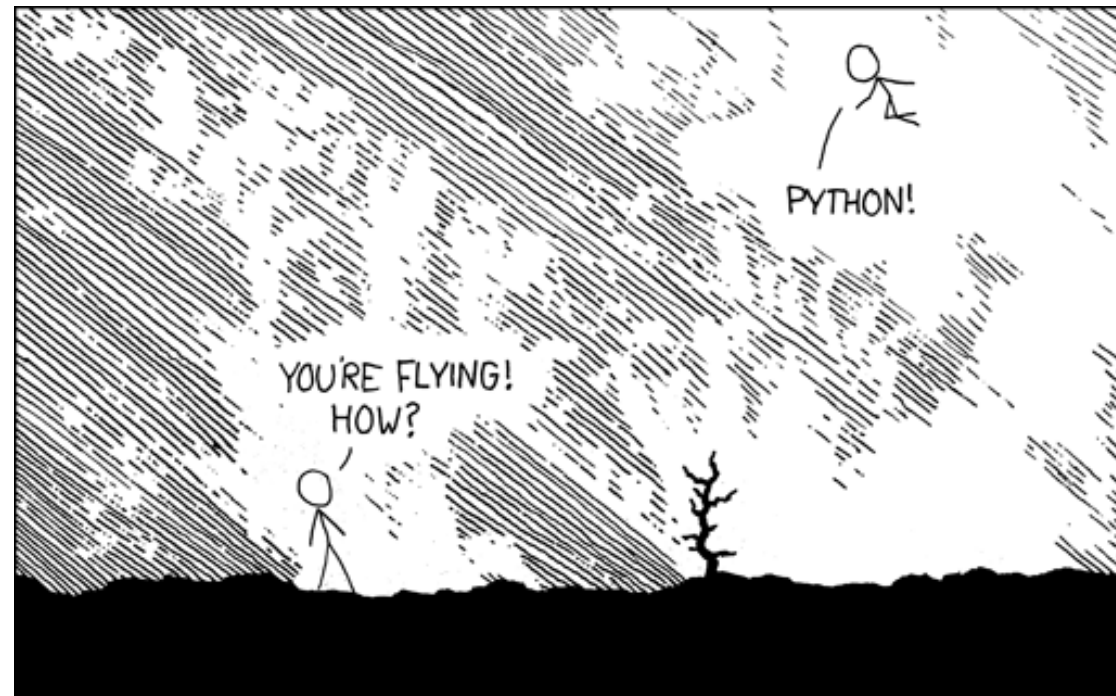
- Multi-paradigma
 - Principalmente a oggetti e imperativo
 - ...Ma con elementi di linguaggi funzionali
- Principalmente interpretato
 - ...Ma con la possibilità di compilare parzialmente moduli
- Corredato di una estensiva libreria di funzioni e moduli

Di solito i linguaggi di questo tipo **non sono indicati per iniziare...**









Linguaggio Python

...Ma Python fa in qualche modo eccezione



Linguaggio Python

Python è considerato al momento il linguaggio più popolare al mondo

Feb 2022	Feb 2021	Change	Programming Language		Ratings	Change
1	3	▲		Python	15.33%	+4.47%
2	1	▼		C	14.08%	-2.26%
3	2	▼		Java	12.13%	+0.84%
4	4			C++	8.01%	+1.13%
5	5			C#	5.37%	+0.93%
6	6			Visual Basic	5.23%	+0.90%
7	7			JavaScript	1.83%	-0.45%



Ambiente di Sviluppo

Per programmare in Python occorre installare un interprete

Noi useremo Anaconda

- ...Ovvero una distribuzione di Python per calcolo scientifico

Potete installare la versione "individual" (~580MB di download)

- La versione "individual" contiene già tutti i pacchetti necessari
- ...Ma all'occorrenza è possibile installarne altri

Impareremo come usare il sistema nella prossima lezione



Ambiente di Sviluppo

Useremo come ambiente di sviluppo il sistema Jupyter

- Jupyter consente l'accesso ad un interprete Python via browser web
- Permette di combinare "celle" di testo in formato markdown
- ...E "celle" di codice che possono essere **eseguite**

È il sistema che utilizzeremo anche per le slide del corso!

Conseguenza: l'intero materiale del corso è **eseguibile**

- Ogni lezione sarà contenuta in un archivio compresso (file `.zip`)
- L'archivio conterrà le slide in PDF
- ...Ma anche il codice della lezione

Impareremo come sfruttarlo nella prossima lezione



Da Fare a Casa

Per la prossima lezione, installate Anaconda sui PC personali

Potete trovare istruzioni dettagliate sul sito del corso

- Installare il sistema è sufficiente
- ...Vederemo come usarlo e come installare pacchetti la prossima lezione

Non dimenticate che:

In questo corso, il lavoro a casa è insostituibile

Detto questo, non vi corre dietro nessuno:

- La frequenza non è obbligatoria e tutto il materiale è disponibile online
- Trovate il vostro ritmo e scegliete un appello di conseguenza



Da Fare a Casa: Installare Pacchetti Anaconda

Anaconda fornisce un gestore di pacchetti accessibile da terminale

...Ossia un gestore di estensioni del linguaggio Python

- È un comando che si chiama "conda"
- Eseguendolo con "conda --help" otterrete qualcosa di questo tipo:

```
usage: conda [-h] [-V] command ...
```

```
conda is a tool for managing and deploying applications, environments and packages.
```

```
Options:
```

```
...
```

```
install      Installs a list of packages into a specified conda
              environment.
```

```
list         List linked packages in a conda environment.
```

```
...
```

```
remove       Remove a list of packages from a specified conda environment.
```



Da Fare a Casa: Installare Pacchetti Anaconda

Anaconda fornisce un gestore di pacchetti accessibile da terminale

- Con `conda list`
 - ...Potete visualizzare i pacchetti già installati
- Con `conda install <nome pacchetto>`
 - ...Potete installare nuovi pacchetti
- Con `conda remove <nome pacchetto>`
 - Potete rimuovere un pacchetto installato

Provate a:

- Visualizzare la lista dei pacchetti installati
- Installare il pacchetto "numpy", con `conda install numpy`
 - Se avete la versione base di Anaconda, numpy sarà già installato



Da Fare a Casa: Installare Pacchetti Anaconda

Assicuriamoci di avere tutti i pacchetti necessari per il corso

Installate tutti i pacchetti indicati nel file `requirements.txt`

■ Questi sono:

```
numpy  
scipy  
matplotlib  
jupyter
```

■ Potete farlo con `"conda install --file requirements.txt"`



Esercizio: Avviare la Lezione

Finalmente possiamo eseguire una lezione per la prima volta

- Scaricate ed estraete la lezione di oggi
- Aprite un terminale nella cartella
- Eseguite il comando `"jupyter notebook"`

Se tutto va bene:

- Si aprirà automaticamente una finestra del vostro browser
- ...E presenterà il contenuto della cartella corrente
- Potete aprire uno qualunque dei notebook cliccando sul file corrispondente

Per terminare:

- Chiudete il browser
- Tornate al terminale e premete `"CTRL+c"`

