Elementi di programmazione per l'analisi dei testi

Rachele Sprugnoli

rachele.sprugnoli@unipr.it

UNIVERSITÀ
DI PARMA

INTRODUZIONE

La programmazione è…

- …dare istruzioni al computer per risolvere problemi
- …un'arte
- …una scienza
- …un divertimento (!)

NON diventerete dei programmatori professionisti MA è un punto di partenza: competenza utile per la **ricerca** e il **lavoro** che permette di **essere autonomi** nella gestione dei dati testuali, **capire** meglio come funzionano i programmi fatti da altri, **capirsi** meglio con gli informatici

Un programma è…

- …una serie di istruzioni che spiegano come effettuare un'operazione…
  - di tipo matematico, come la soluzione di un sistema di equazioni
  - di tipo testuale, come la ricerca e la sostituzione di una parola in un documento
  - di tipo grafico come l'elaborazione di un'immagine o la riproduzione di un filmato

Un algoritmo è…

- …una successione di istruzioni o passi che definisce le operazioni da eseguire sui dati per ottenere i risultati desiderati
- …la parte concettuale alla base del programma

Un linguaggio di programmazione è…

- …un linguaggio artificiale (o formale) eseguibile dal computer
- …un codice fatto di segni con una sintassi e una semantica come una lingua naturale ma
  - non è ambiguo: una dichiarazione = un significato
  - non è ridondante: no prolissità
  - non ammette significati figurati

N.B. Esistono altri linguaggi formali: la notazione della matematica per esprimere le relazioni tra numeri e simboli; la notazione della chimica per descrivere la struttura chimica delle molecole

Tipi di linguaggi di programmazione

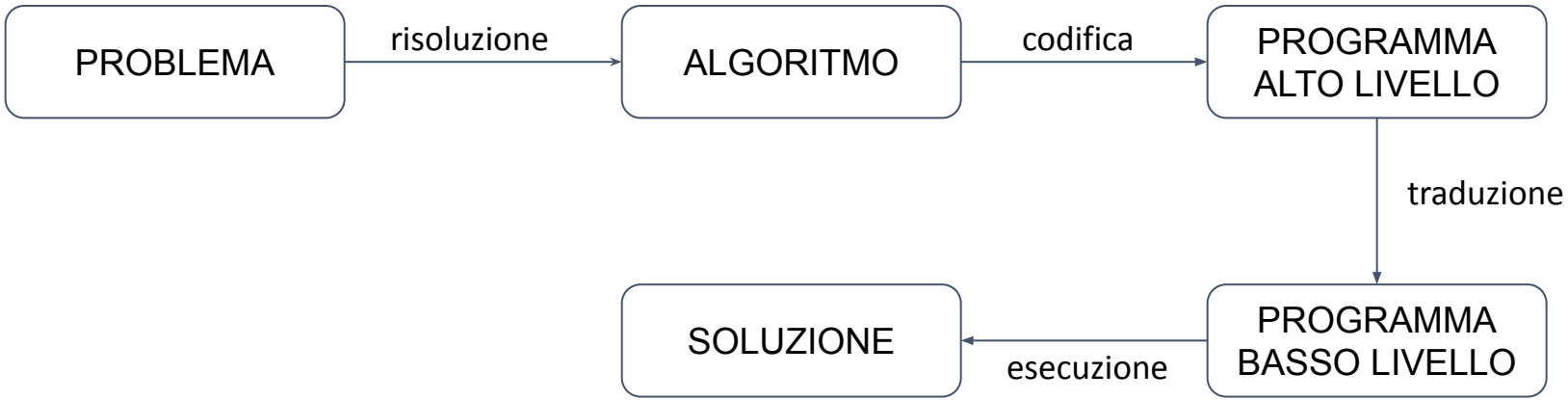
- Due tipi di linguaggi in base al modo in cui le istruzioni sono codificate:
  1) linguaggi di **basso livello** (linguaggi macchina, linguaggi assemblativi), più vicini al funzionamento fisico del computer, specifici per famiglia di computer
  2) linguaggi di **alto livello** (e.g., Java, Python), più vicini al linguaggio naturale, indipendenti dal tipo di computer
- Necessità di tradurre un programma scritto con linguaggio di alto livello in un linguaggio di basso livello per farlo eseguire

“Traduzione”

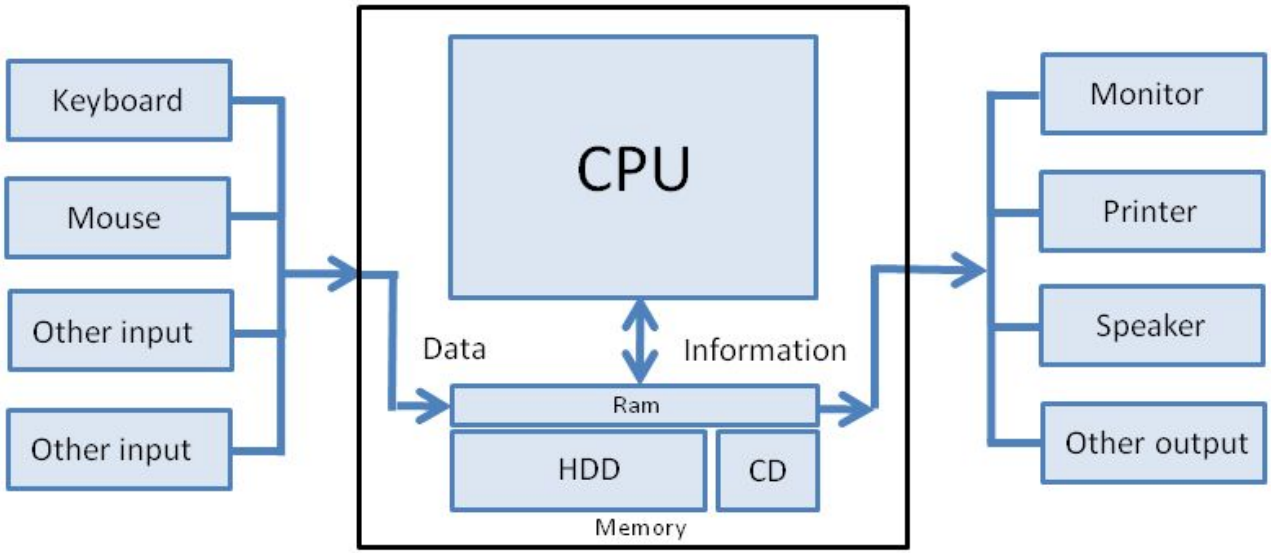
- Modi per tradurre da alto a basso livello:

  → COMPILAZIONE: il traduttore (*compilatore*) traduce tutto il programma e poi esegue la versione tradotta

  → INTERPRETAZIONE: il traduttore (*interprete*) traduce ed esegue il programma un’istruzione alla volta



Cosa c'è sotto…

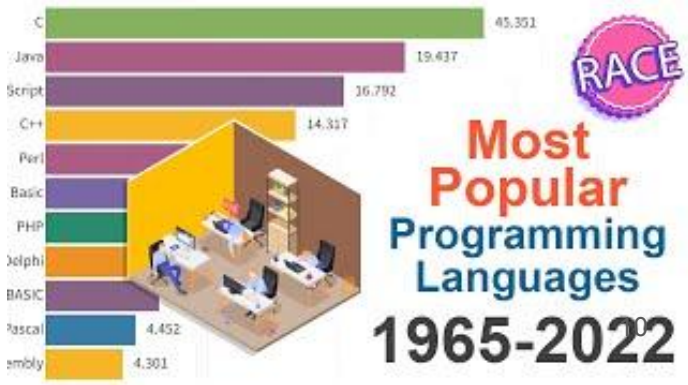


Da: https://commons.wikimedia.org/wiki/File:Computer2.png, autore Hr.hanafi, licenza: https://creativecommons.org/licenses/by-sa/3.0/deed.en

Perché Python

- Gratuito, aperto e disponibile per Windows, Mac e Linux
- Ampiamente utilizzato
- Supportato da una grande comunità
- Costantemente aggiornato: Python 2.x → Python 3.x
- Documentazione: https://docs.python.org/3/

Video:
https://youtu.be/qQXXI5QFUfw?si=FRrz3n0bJudtJefr



Come scrivere/eseguire uno script in Python

- Installazione in locale
  - linea di comando
  - ambiente di sviluppo: Pycharm, Visual Studio Code, PyDev

    → IDE (*Integrated Development Environment*)

- Notebook online (*cloud computing*)
  - Google Colab: è necessario avere un **account Google**

    → tutto si salva in un'apposita cartella sul proprio Drive
  - Informazioni di base: https://mcgrawect.princeton.edu/guides/Google-Colab-Introduction.pdf

Elementi base: un po’ di terminologia

- Enunciato = istruzione di un programma
- Variabile = contenitore, con un nome, in cui memorizzare i dati
- Lista = elenco di dati
- Dizionario = oggetto che contiene coppie “chiave: valore”
- Ciclo = ripetizione di un’istruzione in base a una condizione
- Costruzione IF = fare qualcosa SE alcune condizioni sono soddisfatte
- Funzione = un'unità di codice che svolge un'attività specifica
- Metodo = funzione definita per un oggetto specifico
- Libreria = codice riutilizzabile per eseguire una certa attività
- Modulo = parte di una libreria

Enunciato

- Righe di istruzioni tradotte ed eseguite dall'interprete Python

- Tutto ciò che NON è un **commento** = spiegazione aggiunta dal programmatore per dare informazioni utili e illustrare lo scopo degli enunciati

  → Iniziano con il carattere #

  → All'inizio dello script servono a spiegare lo scopo del programma

  → Possono essere aggiunti anche dopo ogni enunciato

Enunciato

- Righe di istruzioni tradotte ed eseguite dall'interprete Python

- Tutto ciò che NON è un **commento** = spiegazione aggiunta dal programmatore per dare informazioni utili e illustrare lo scopo degli enunciati

```
1 # commento su una riga
2
3 '''
4 commento
5 su più
6 righe
7 '''
8
9 """
10 altro commento
11 su più
12 righe
13 """
```

Variabile

- Nome che si riferisce a valori → serve a memorizzare dei valori da riutilizzare

```
a = 13
b = "Hello, world!"
c = 42.42
```

TIPO INTERO (integer)

TIPO STRINGA (string → sempre tra virgolette)

TIPO DECIMALE (float → si usa “.” e non “,”)

→ Un valore si assegna a una variabile usando il simbolo =

→ Il simbolo di uguale (in senso aritmatico) è ==

Lista

- Una sequenza modificabile di dati, la cui dimensione aumenta o diminuisce automaticamente quando vengono inseriti o rimossi elementi

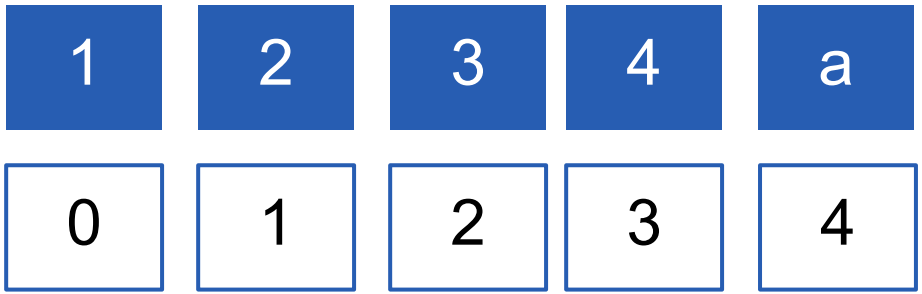
```
list =[1,2,3,4,a]
```

Una lista (chiamata list) con 4 numeri interi e la variabile a

N.B. Come dovrei scrivere l'ultimo dato della lista per renderlo una stringa?

Lista

```
list =[1,2,3,4,a]
```




```
list[0] → 1
list[3] → 4
```

Lista di liste

```
list_a = [1,2,3]
list_b = [4,5,6]

lol = [list_a,list_b] → [[1,2,3],[4,5,6]]

lol[0][1] → 2
```

`[0]` → indice della lista

`[1]` → indice dell'elemento della lista

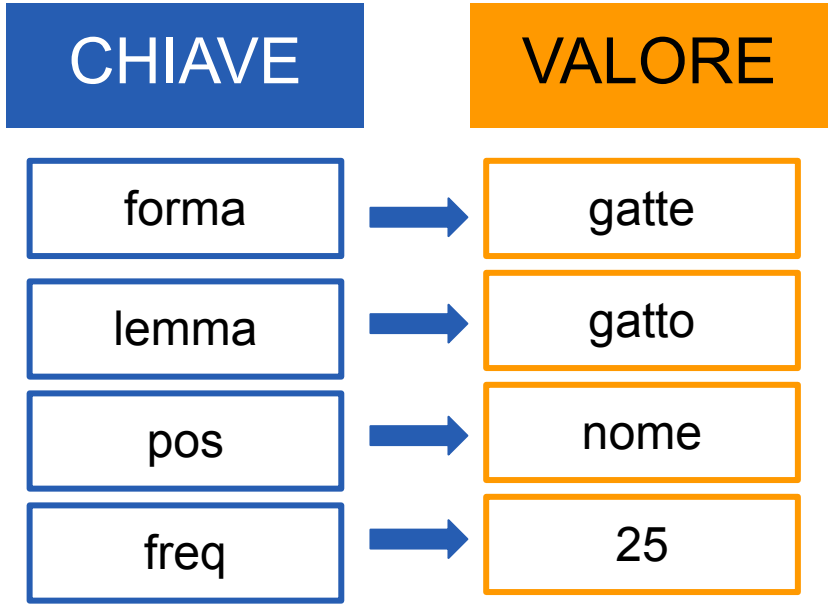
`lol[1][2]` → ?

Dizionario

- Un contenitore che memorizza associazioni tra coppie di oggetti (chiave e rispettivo valore)

```
word1 = {
"forma" :  "gatte",
"lemma" :  "gatto",
"pos" :  "nome",
"freq" : 25
 }
```

word1["pos"] → nome

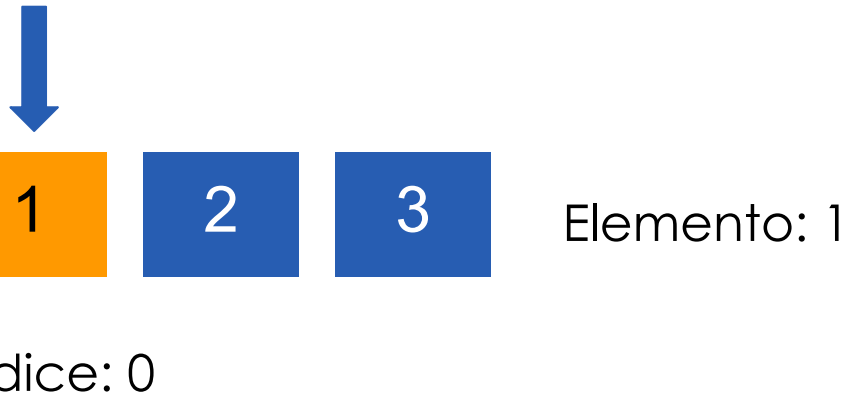


Ciclo

- Sequenza di istruzioni che vengono eseguite ripetutamente
- Ciclo (*loop*) FOR

```
list = [1, 2, 3]
for item in list:
    print(item)
```

```
1
2
3
```



Ciclo

- Sequenze di istruzioni che vengono eseguite ripetutamente
- Ciclo FOR

```
list = [1, 2, 3]
for item in list:
    print(item)
```

```
1
2
3
```

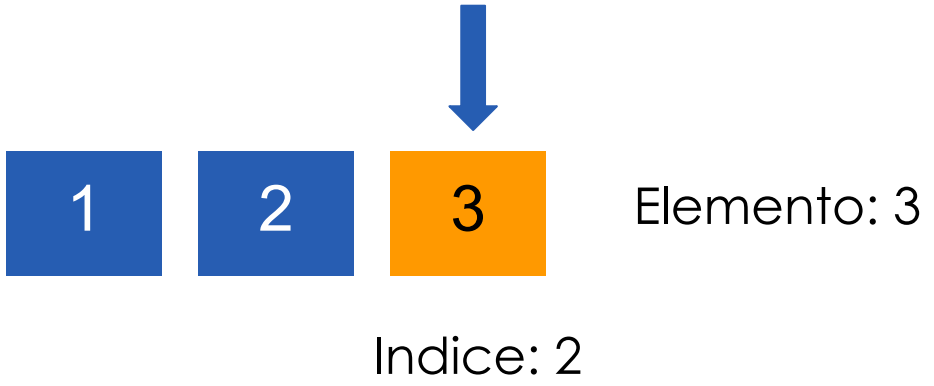
Ciclo

- Sequenze di istruzioni che vengono eseguite ripetutamente
- Ciclo FOR

```
list = [1, 2, 3]
for item in list:
    print(item)
```

```
1
2
3
```

N.B. L'istruzione `print` è indentata!



Costruzione IF

- Istruzione condizionale: controlla se si verificano determinate condizioni e varia di conseguenza il comportamento del programma

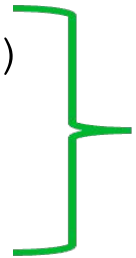
```
a = 10
if a > 15:
    print ("A è maggiore di 15")
else:
    print ("A non è maggiore di 15")

A non è maggiore di 15
```

Funzione

- Blocco di codice riutilizzabile che esegue un insieme specifico di istruzioni quando viene chiamata (o **evocati**)
- Molte funzioni sono già definite e basta invocarle indicandone il nome e gli argomenti (o parametri) posti tra parentesi tonde

```
print("Hello, world!")
print(12)
print(a)
```



stampa il valore posto tra parentesi

- Altre funzioni si possono **implementare** (creare ex novo)

Funzione

- Implementare una funzione

```
def add(a,b):
    result = a + b
    return result

add(5,10) → 15
add(2,2) → 4
```

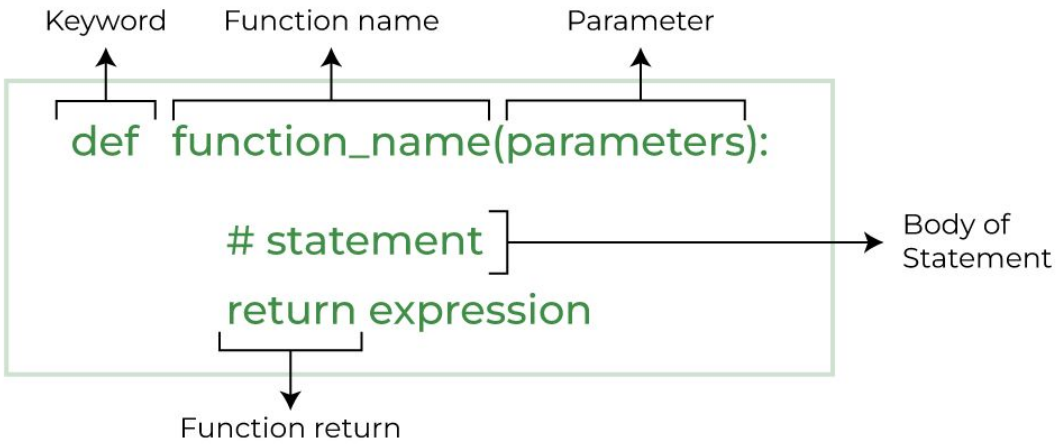


Immagine da: https://www.geeksforgeeks.org/python-functions/

Metodo

- Funzione creata appositamente per gestire oggetti specifici

  nome_dato.nome_metodo(eventuali_parametri)

  - Per esempio, `append` e `sort` sono METODI delle liste

    ```
    list =[1,2,3,4,a]
    ```
    `list.append(15)` → aggiunge 15 in fondo alla lista
    `list.sort()` → mette in ordine gli elementi della lista

Libreria

- Un insieme di codice riutilizzabile per eseguire una certa attività
  - è formata da MODULI
  - i moduli contengono FUNZIONI

- Esempi:
  - NLTK (Natural Language ToolKit)
  - SpaCy
  - Pandas
  - Matplotlib

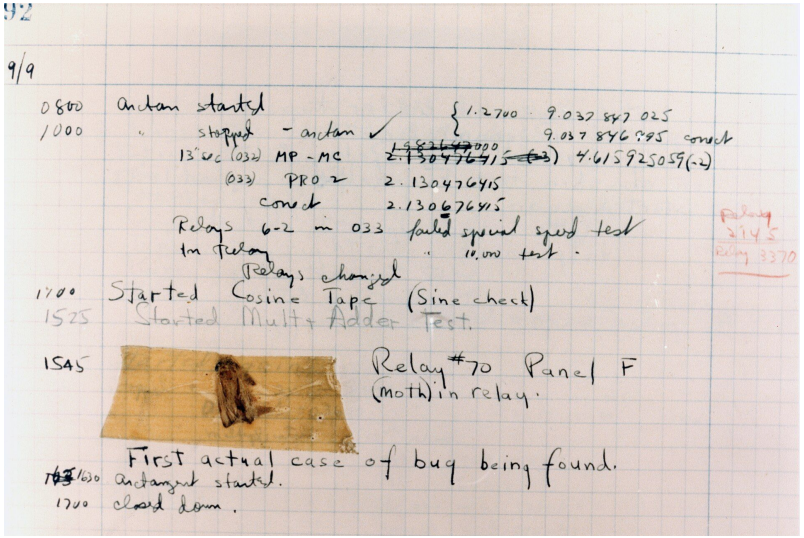
Un po’ di pratica

Lezione1.ipynb:
https://colab.research.google.com/drive/1DFJSuo2mRwXG-TP_c26oQAyk7xn3Gttb?usp=sharing

- File → Salva una copia su Drive → ognuno lavora su un file diverso salvato nel proprio Drive
- Estensione ipynb → formato di Jupyter Notebook (piattaforma informatica interattiva basata sul web)
- File → Scarica → Scarica .py → per lanciare lo script in locale (è sempre consigliabile salvare una copia di backup!)

Errori

- Bug → il procedimento di ricerca e correzione degli errori è chiamato **debug**



9 settembre 1947: falena bloccata in un computer Mark II

«It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise — this thing gives out and [it is] then that "Bugs" — as such little faults and difficulties are called — show themselves and months of intense watching, study and labor are requisite before commercial success or failure is certainly reached.»

Estratto da una lettera di Edison del 1878

Tipi di errori

Errori.ipynb:
https://colab.research.google.com/drive/1ABmpgxc78qhncVUQwhl58OfA1tMP0WxI?usp=sharing

- Di compilazione
- Di esecuzione

Esercizio 1

- Creare un nuovo file Colab
- Stampare
  - Scrivi uno script che stampi il tuo nome
  - Scrivine uno che stampi il tuo nome e il tuo cognome su due righe distinte: prima riga col nome e seconda riga col cognome

Esercizio 2

- Liste
  - Crea una lista con una serie di voti
  - Stampa l'ultimo voto della lista

Esercizio 3

- Cicli
  - Scrivi uno script che stampi tutti i numeri da 0 a 48 uno sotto l'altro
    - Usa la funzione `range(arg1,arg2)` che genera una sequenza di numeri ordinati da `arg1` ad `arg 2`
  - Alla fine stampa un messaggio di chiusura del conteggio, ad esempio "Finito!"

Soluzioni esercizi

- https://colab.research.google.com/drive/1IsYRxIWNXVIDuDHBM4QEZX1nXsof3PNY?usp=sharing