Università Roma Tre – Dipartimento di Matematica e Fisica

Percorso Abilitante Speciale
Classe A048 – Matematica Applicata

Corso di Informatica

Il linguaggio di programmazione Python

Marco Liverani
(liverani@mat.uniroma3.it)

Sommario

- Linguaggi di programmazione: caratteristiche generali
- Paradigmi di programmazione
- Compilatori ed interpreti
- Il linguaggio Python
- Esempi

Linguaggi di programmazione 1

- Sono linguaggi artificiali progettati per la codifica di algoritmi
- Sono un compromesso tra linguaggio naturale e linguaggio macchina
- Sono privi delle ambiguità tipiche di un linguaggio naturale, ma sufficientemente chiari da essere facilmente compresi
- Sono costituiti da
  - un alfabeto di simboli e delle parole chiave
  - una grammatica e una sintassi
  - una semantica

Linguaggi di programmazione 2

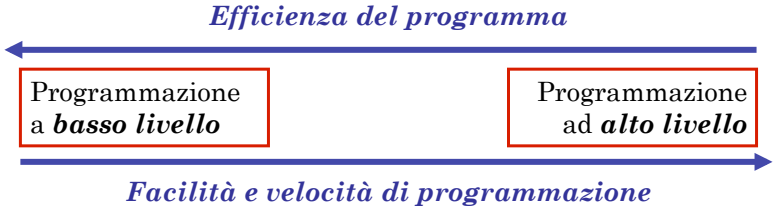
- Codificare un programma utilizzando il *linguaggio macchina* è assai arduo e richiede una conoscenza approfondita del funzionamento di un particolare calcolatore (o meglio: del microprocessore che costituisce la CPU della macchina)
- Per ovviare a questo problema, che ha costituito per molti anni un grosso limite alla diffusione della programmazione e quindi anche dell'uso dei calcolatori, sono stati sviluppati dei **linguaggi di programmazione più evoluti**, che si pongono a metà strada fra il nostro linguaggio naturale ed il linguaggio macchina
- Sono **semplici** e **poveri** (poche parole chiave, poche regole), ma **privi di qualsiasi ambiguità**

Linguaggi di programmazione 3

- In informatica si parla di **programmazione a basso livello** quando si utilizza un linguaggio molto vicino alla macchina, al suo funzionamento interno
- Si parla invece di **programmazione ad alto livello** quando si utilizzano linguaggi più sofisticati ed astratti, slegati dal funzionamento fisico della macchina
- Si viene così a creare una **gerarchia di linguaggi**, dai meno evoluti (il *linguaggio macchina* e l'*assembler*) a quelli più evoluti (*Pascal*, *Fortran*, *Cobol*, *Perl*, *Java, Python, ...*)
- Il **linguaggio Python** è tra i linguaggi moderni di livello più alto, privo di formalismi eccessivi e per questo adatto anche alla didattica

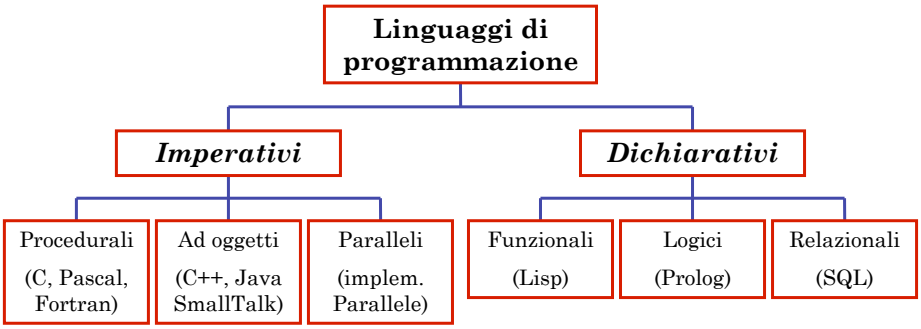
Linguaggi di programmazione 4

- La programmazione a *basso livello* è più ardua e meno intuitiva, ma consente di sviluppare programmi molto efficienti su uno specifico sistema hardware/software, sfruttandone a fondo le caratteristiche.
- Ad *alto livello* la programmazione è più "naturale" e rapida, ma è possibile che non consenta di produrre software particolarmente efficiente.



Paradigmi di programmazione 1

- Possiamo aggregare i numerosi linguaggi di programmazione esistenti sulla base del **modello astratto di programmazione** che sottintendono e che è necessario adottare per utilizzarli.



Paradigmi di programmazione 2

- **Linguaggi imperativi**:
  - Il modello computazionale è basato sui *cambiamenti di stato della memoria* della macchina.
  - È centrale il concetto di *assegnazione di un valore* ad una locazione di memoria (*variabile*).
  - Il compito del programmatore è costruire una sequenza di assegnazioni (spesso reiterandole più volte) che producano lo stato finale (in modo tale che questo rappresenti la soluzione del problema).
- **Linguaggi dichiarativi**:
  - Il modello computazionale è basato sui concetti di *funzione* e *relazione*.
  - Il programmatore non ragiona in termini di assegnazioni di valori, ma di *relazioni fra entità* e di *valori di una funzione*.

Compilatori ed interpreti 1

- Un **programma** è la ***codifica*** di un algoritmo eseguita utilizzando un linguaggio di programmazione.
- L'unico linguaggio che la macchina è in grado di interpretare è il *linguaggio macchina*.
- Per eseguire un programma scritto in un linguaggio di alto livello **è necessario dunque tradurlo in linguaggio macchina**.
- Naturalmente è possibile eseguire la **traduzione in modo automatico** utilizzando un programma "traduttore" (un ***compilatore*** o un ***interprete***).
- Ogni traduttore è in grado di interpretare e tradurre *un solo* linguaggio (compilatore C, interprete Perl, compilatore Pascal, ecc.).

Compilatori ed interpreti 2

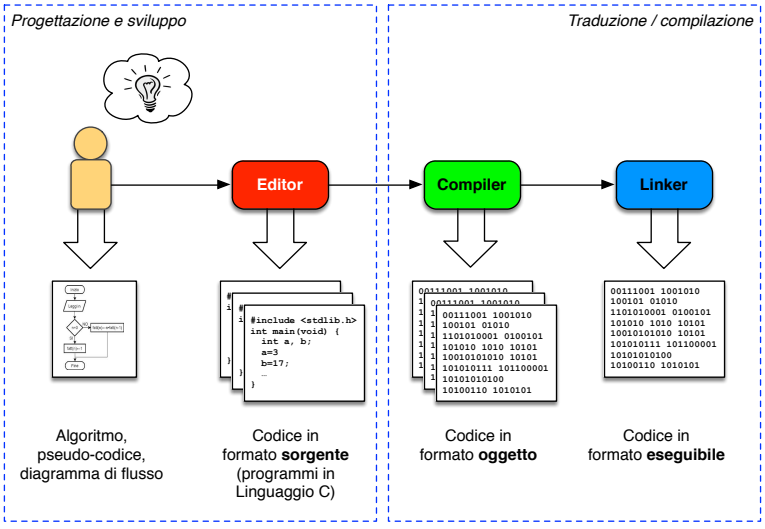
- **Compilatore**: esegue una sola volta il processo
  - *Legge tutte* le istruzioni del *programma "sorgente"*, ne verifica la correttezza sintattica e le traduce in linguaggio macchina.
  - *Memorizza* su disco il *programma "eseguibile"* tradotto in linguaggio macchina.
- Al termine della compilazione avremo un programma "eseguibile" in linguaggio macchina.
- La traduzione di ogni istruzione del programma avviene *una sola volta*, anche se una stessa istruzione viene ripetuta più volte all'interno del programma.
- Non ho bisogno di avere il compilatore ed il "sorgente" per *eseguire* il programma: mi basta il programma "eseguibile".

Compilatori ed interpreti 3

- **Interprete**: itera più volte questo processo
  - *Legge* un'istruzione del programma "sorgente"
  - *Traduce* l'istruzione in linguaggio macchina
  - *Esegue* l'istruzione
  - Passa all'interpretazione dell'*istruzione successiva*
- Al termine di questa operazione, del programma in linguaggio macchina non rimane alcuna traccia (la traduzione *non viene memorizzata*).
- Se il programma torna più volte su una stessa istruzione, questa verrà tradotta (ed eseguita) *ogni* volta.
- È necessario disporre dell'interprete per poter eseguire il programma.

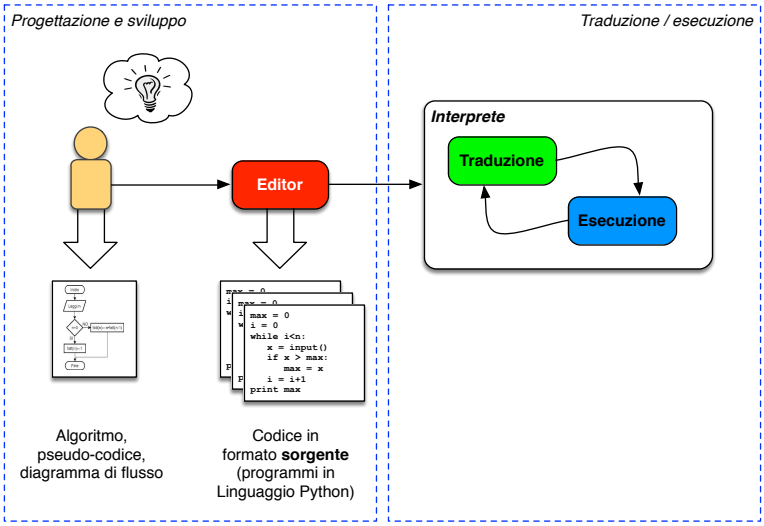
Progettazione, codifica ed esecuzione 1

Il processo di progettazione, codifica, traduzione nel caso di linguaggi *compilati*



Progettazione, codifica ed esecuzione 2

Il processo di progettazione, codifica, traduzione nel caso di linguaggi *interpretati*



Il linguaggio Python

- È un linguaggio interpretato di alto livello
- È un linguaggio "multi-paradigma":
  - supporta la programmazione strutturata
  - supporta la programmazione a oggetti (OOP)
  - è dotato di caratteristiche di programmazione funzionale
- Una caratteristica del Python è il fatto che l'indentazione delle istruzioni ha un valore sintattico e consente di definire blocchi contenuti all'interno di una struttura algoritmica (un ciclo, una condizione, ecc.)
- Il codice scritto in Python è molto leggibile, se paragonato al codice di altri linguaggi di programmazione

Istruzioni del linguaggio 1

- Assegnazione:
  variabile = *valore*
- Leggi:
  variabile = **input**(“*messaggio*”)
- Scrivi:
  print “*messaggio*”, *variabile*
- Se ... allora ... altrimenti:
  if *condizione*:
  istruzioni
  else:
  istruzioni

Istruzioni del linguaggio 2

- Cicli per ripetere più volte delle istruzioni:
  while *condizione*:
  istruzioni

  for *variabile* **in** *lista*:
  istruzioni

Esempio 1

- Letti in input due numeri stampa la somma

codice

```
print "Inserisci due numeri: "
a = input();
b = input();
print "Hai inserito ", a, " e ", b
c = a+b
print "La somma di ", a, " e ", b, " e' ", c
```

esecuzione

```
python esempio1.py
Inserisci due numeri:
4
8
Hai inserito i numeri  4  e  8
La somma di  4  e  8  e'  12
```

Esempio 2

- Letti in input n numeri stampa il massimo

codice

```
n = input("Inserisci il numero di elementi: ");
print "Inserisci ", n, " numeri: "
max = 0
i = 0
while i<n:
   x = input();
   if x > max:
      max = x
   i = i+1
print "Massimo: ", max
```

esecuzione

```
python esempio2.py
Inserisci il numero di elementi:
3
Inserisci  3  numeri:
5
10
7
Massimo:  10
```