Laurea in Ingegneria Gestionale

Corso di Fondamenti di Informatica
A.A. 2017/2018

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Introduzione

Le seguenti slide sono una rielaborazione di slide analoghe preparate per vari corsi di introduzione alla programmazione.

In particolare sono in larga parte tratte dai corsi di Giorgio Fumera (Univ. di Cagliari), nel riadattamento di Raffaele Iacomussi (Sapienza), e, in misura minore, dai corsi di Diego Calvanese (Univ. di Bolzano) e Mario Gianni (Sapienza).

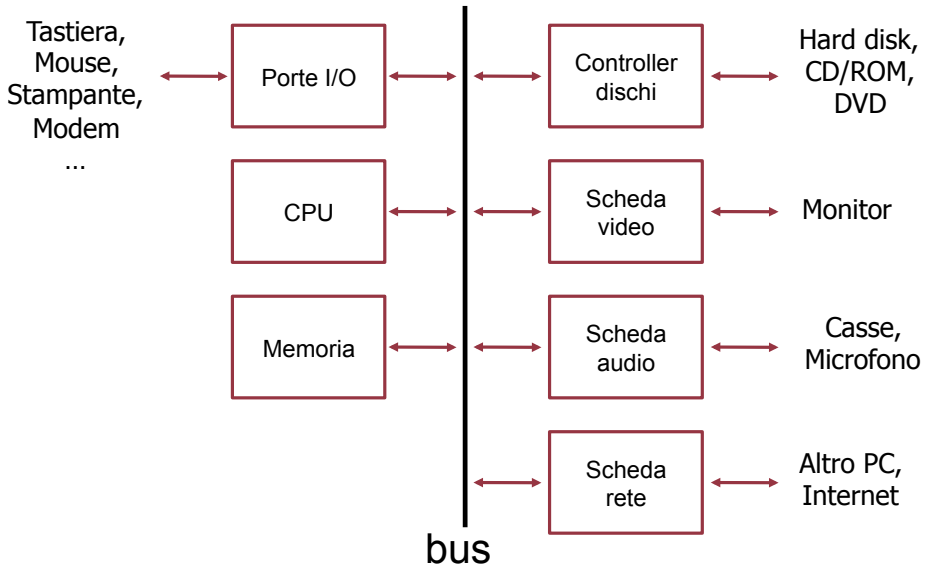
Cosa è un calcolatore

- Un calcolatore è una macchina che esegue gli ordini dati
- Gli ordini sono espressi sotto forma di istruzioni in un linguaggio di programmazione
- Le istruzioni vengono sintetizzate in un programma
- Il programma permette l'elaborazione di dati in ingresso per produrre dei dati in uscita

Come è fatto un calcolatore?

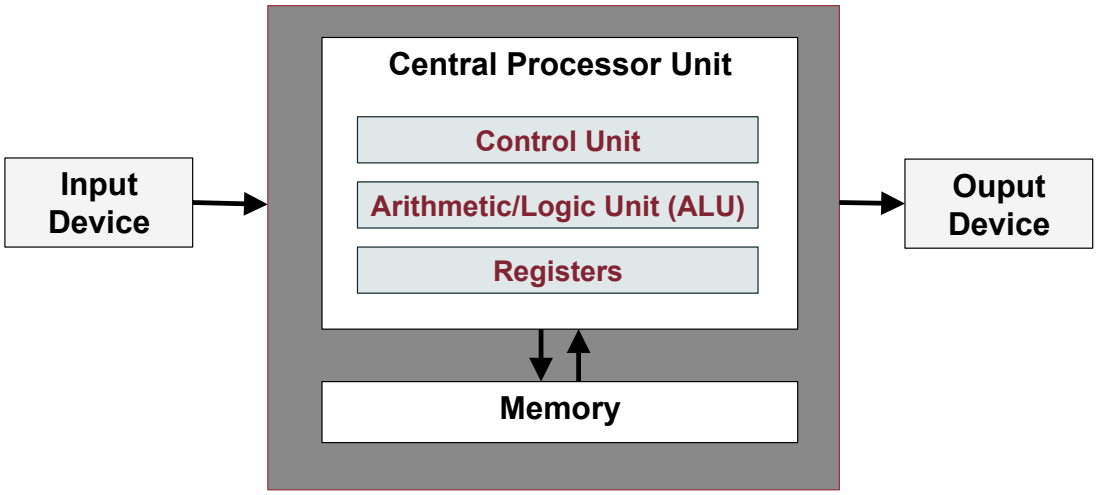
- Hardware
  - Processore
  - Memoria
  - Periferiche
- Funzionamento
  - Esegue istruzioni elementari
  - Esegue istruzioni molto velocemente
  - Deve essere programmato: il software, cioè i programmi, caratterizzano il compito che esso svolge

Schema di un calcolatore



La Macchina di Von Neumann

L'architettura dei moderni computer si basa su quella proposta nel 1945 da Von Neumann, in cui ci sono tre elementi principali: CPU, Memoria, Dispositivi di Input/Output



Linguaggi di Programmazione

- Linguaggio macchina
  ```
  00111011 01001111 10101011…
  ```
- Linguaggio assemblativo (Assembly)
  ```
  iload intRate
  bipush 100
  if icmpgt intError
  ```
- Linguaggi ad alto livello
  ```
  if (intRate > 100) ...
  ```

Nota: “Linguaggio di programmazione” viene spesso usato solo per riferirsi ad un linguaggio ad alto livello

Linguaggio di Programmazione

- Un linguaggio di programmazione contiene un insieme di istruzioni che indicano al calcolatore cosa fare.
- Le istruzioni hanno una sintassi e una semantica ben determinata.
- Esempi di linguaggi di alto livello: C, **Python** (studiato in questo corso), Java, Prolog, Lisp.
- La traduzione verso il basso (il linguaggio macchina) può avvenire tramite un compilatore.

Sintassi e Semantica

- SINTASSI
  - Insieme delle regole che specificano come scrivere un programma formalmente corretto.
  - Come in italiano, è possibile scrivere un testo formalmente corretto, ma senza alcun significato.
- SEMANTICA
  - Regole per attribuire un significato alle istruzioni e alla loro composizione, cioè ai programmi.
  - Analogamente, l'italiano assegna un significato alle parole e una logica ai costrutti.

Programma

- I programmi caratterizzano i compiti che il calcolatore svolge.
- Un programma (indipendentemente dal linguaggio in cui è scritto) è costituito da due aspetti fondamentali:
  - DATI: rappresentazione delle informazioni relative al dominio di interesse.
  - OPERAZIONI: manipolazioni della rappresentazione che realizzano le funzionalità richieste.

Rappresentazione delle informazioni

- Le informazioni si rappresentano mediante tipi di dato.
- Un tipo di dato raccoglie informazioni dello stesso tipo (ad esempio, i numeri interi, i caratteri, le stringhe, ecc.).
- Nei linguaggi di programmazione ad alto livello esistono tipi di dato predefiniti ed è possibile definire tipi di dato per rappresentare informazioni di interesse specifiche per un'applicazione (ad esempio, autovetture, persone, ecc.).

Realizzazione delle operazioni

In genere, si realizza un'operazione quando deve essere risolto un problema.

Per delegare ad un calcolatore la soluzione di un problema è necessario individuare un algoritmo che risolva il problema.

Algoritmi e programmi

Problemi da risolvere usando elaboratori possono essere di natura molto varia.

Esempi

1. Dati due numeri, trovare il maggiore.
2. Dato un elenco di nomi e numeri di telefono (rubrica o elenco telefonico) e un nome, trovare il numero di telefono corrispondente.
3. Data la struttura di una rete stradale e le informazioni sui flussi dei veicoli, determinare il percorso più veloce da A a B.
4. Scrivere tutti i numeri pari che non sono la somma di due numeri primi (Congettura di Goldbach).
5. Decidere per ogni programma Python e per ogni dato in ingresso, se il programma Phyton termina quando viene eseguito su quel dato.

Caratteristica comune ai problemi

Informazioni in ingresso ➔ Informazioni in uscita

Osservazioni sulla formulazione dei problemi:

- descrizione non fornisce un metodo risolutivo (si pensi all'esempio 3)
- descrizione del problema è talvolta **ambigua** o **imprecisa** (ad esempio, 2 con Mario Rossi che compare più volte)
- per alcuni problemi **non è noto un metodo risolutivo** (ad esempio 4)
- esistono problemi per i quali è stato dimostrato che **non può esistere un metodo risolutivo** (ad esempio 5)

Noi consideriamo solo problemi per i quali è noto esistere un metodo risolutivo.

Per delegare ad un calcolatore la soluzione di un problema è necessario:

1. Individuare un **algoritmo** che risolve il problema, ovvero un insieme di passi che, eseguiti in ordine, permettono di calcolare i risultati a partire dalle informazioni a disposizione.

   Proprietà di un algoritmo:

   non-ambiguità: le istruzioni devono essere univocamente interpretabili dall'esecutore

   eseguibilità: ogni istruzione deve poter essere eseguita (in tempo finito) con le risorse a disposizione

   finitezza: l'esecuzione dell'algoritmo deve terminare in tempo finito per ogni insieme di dati in ingresso

2. **Rappresentare in un linguaggio di programmazione (LDP)**

   (a) L'algoritmo
   (b) Le informazioni a disposizione
   (c) Le informazioni utilizzate dall'algoritmo
   (d) Le informazioni fornite al termite

   → (a) Programma
   (b) Dati in ingresso
   (c) Dati ausiliari
   (d) Dati in uscita

Riassumendo:

problema ➔ algoritmo ➔ programma

Individuare un algoritmo per un dato problema può essere molto complesso: Conviene operare per **livelli di astrazione**:

- si parte da una versione molto generale
- si **raffinano** via via le varie parti dell'algoritmo fino ad ottenere una descrizione dettagliata che può essere direttamente codificata in un LDP

→ metodo dei raffinamenti successivi

Pseudocodice per la specifica di algoritmi

- si usano **frasi in linguaggio naturale** che esprimono operazioni o condizioni
- operazioni vengono eseguite in sequenza, tranne che per le . . .
- **strutture di controllo** dell'esecuzione delle operazioni (specificate usando parole chiave)

if condizione
then operazione 1
else operazione 2

for ogni valore compreso tra . . . e . . .
do operazione

while condizione
do operazione

do operazione
while condizione

Lo pseudocodice si presta bene al metodo dei raffinamenti successivi: un **raffinamento** consiste nel sostituire un'operazione con

- una sequenza di operazioni, oppure
- una struttura di controllo

Esempio: Calcolo del massimo comun divisore di due interi positivi m ed n

Algoritmo

1. calcola l'insieme I dei divisori di m
2. calcola l'insieme J dei divisori di n
3. calcola l'insieme K intersezione di I e J (divisori comuni)
4. restituisci il valore massimo tra quelli in K

Raffinamento del passo 1

1.1 inizializza I all'insieme vuoto
1.2 **for** ogni numero i compreso tra 2 ed m
do if i è divisore di m
then aggiungi i ad I

Scrivere un altro algoritmo per il calcolo del MCD che si basi sulla seguente proprietà (Euclide)

$$MCD(m,n) = \begin{cases} m, & \text{se } m = n \\ MCD(m-n,n), & \text{se } m > n \\ MCD(m,n-m), & \text{se } m < n \end{cases}$$

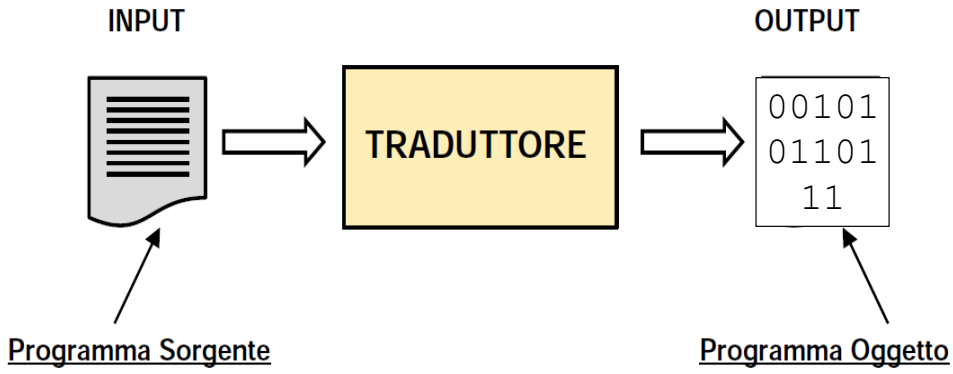
Programmi: rappresentano algoritmi in un linguaggio di programmazione

Esempio: Programma che
1. Legge due numeri da tastiera
2. Ne calcola il MCD e
3. Lo stampa

```
m=int(input("Immetti un intero positivo: "))
n=int(input("Immetti un intero positivo: "))
print("Il massimo comun divisore di "+str(m)+ " e "+str(n)+" è ");
while (m != n):
  if (m > n):
    m = m - n
  else:
    n = n - m
print(m)
```

Compilazione

- Un compilatore è un programma traduttore, impiegato per produrre codice oggetto (in linguaggio macchina) a partire da codice sorgente scritto in un dato linguaggio di programmazione di livello più alto.
- Questo processo si chiama compilazione.



Paradigmi di Programmazione (1/2)

Esistono differenti paradigmi di programmazione che si distinguono per l'enfasi che pongono sui due aspetti fondamentali: **dati** (oggetti) e **operazioni**.

I paradigmi di programmazione principali sono:

- IMPERATIVO: enfasi sulle operazioni intese come **azioni, comandi, istruzioni** che cambiano lo stato dell'elaborazione. Gli oggetti sono funzionali alla elaborazione.
- FUNZIONALE: enfasi sulle operazioni intese come **funzioni** che calcolano risultati. Gli oggetti sono funzionali alla elaborazione.
- ORIENTATO AGLI OGGETTI: enfasi sugli **oggetti** che complessivamente rappresentano il dominio di interesse. Le operazioni sono funzionali alla rappresentazione

Paradigmi di Programmazione (2/2)

- In genere, in un programma sono utilizzati più paradigmi di programmazione.
- Quindi i linguaggi di programmazione forniscono supporto (in misura diversa) per i vari paradigmi.

Laurea Triennale in Ingegneria Gestionale

Corso di Fondamenti di Informatica A.A. 2016/2017

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Introduzione a Python

Python

- Sviluppato all'inizio degli Anni Novanta da Guido van Rossum.
  - Il suo nome fu scelto per via della passione dell'autore per i Monty Python e per la loro serie televisiva Monty Python's Flying Circus (BBC, 1969 - 1974).
- L'intento era quello di progettare un linguaggio che consentisse di operare agevolmente con dati complessi e articolati:
  - Dinamicità
  - Semplicità
  - Flessibilità

Python

- Supporta **diversi paradigmi di programmazione**:
  - Imperativo
  - Funzionale
  - Orientato agli oggetti
- Offre un controllo dei tipi
  - Forte (strong typing): Ogni elemento sintattico che denota un valore (ad es. variabile) è associato ad un determinato tipo durante l'esecuzione
  - Eseguito al run-time (dynamic typing): il controllo del tipo della variabile è effettuato a runtime

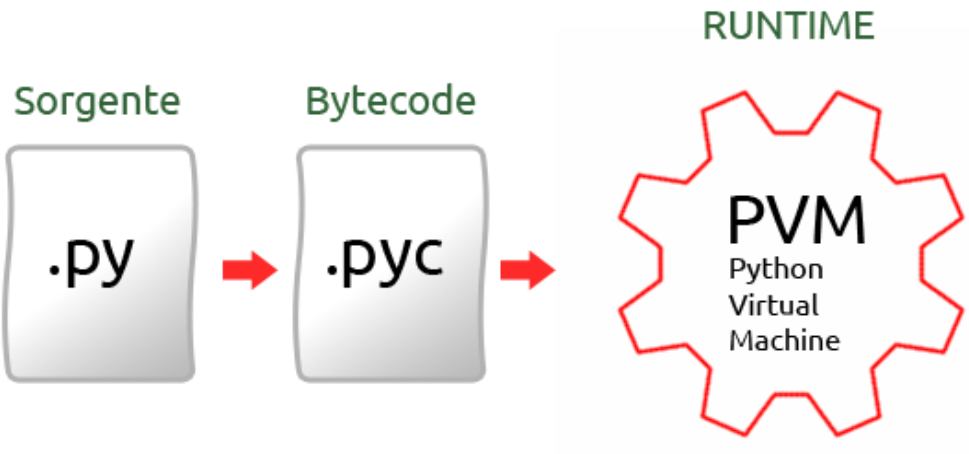
Python

- Offre una gestione automatica della memoria.
  - Garbage collector.
- Fornito di una libreria built-in estremamente ricca.
- Offre costrutti robusti per la gestione delle eccezioni
  - come mezzo per segnalare e controllare eventuali condizioni di errore (incluse le eccezioni generate dagli errori di sintassi).

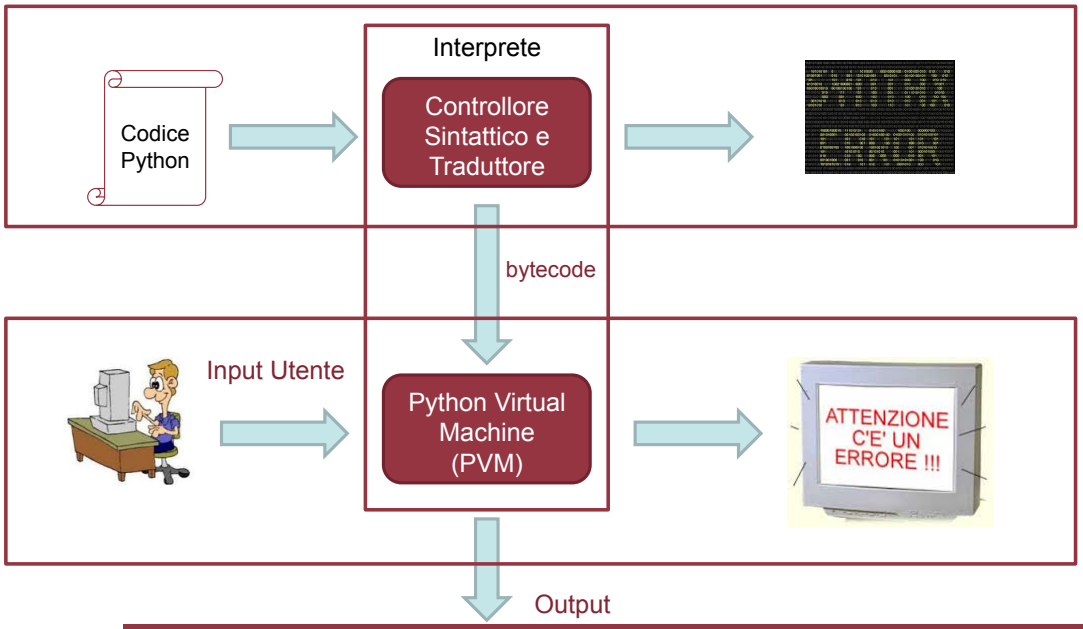
Python

- Linguaggio interpretato (o meglio pseudo-compilato):
  - un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo.
  - il codice sorgente passa prima da una fase di pre-compilazione in bytecode (riduce la dipendenza dall'hardware).
    - viene riutilizzato dopo la prima esecuzione del programma, evitando così di dover ogni volta interpretare il sorgente ed incrementando di conseguenza le prestazioni (in modo trasparente per l'utente).
  - non esiste una fase di compilazione separata che generi un file eseguibile partendo dal sorgente.
    - il codice sorgente non viene convertito direttamente in linguaggio macchina, ma il bytecode prodotto viene eseguito da una **macchina virtuale Python (PVM)**
  - Portabilità.
  - Indipendenza dalla piattaforma.

Interpretazione di un programma Python (1\2)



Interpretazione di un programma Python (2\2)



Esecuzione di script *.py*

Per prima cosa bisogna generare un semplice file di testo e salvarlo con estensione *.py*, per esempio, *helloworld.py*.

A questo punto possiamo aprire il file creato con un qualsiasi editor di testi (non Word, bensì Notepad, Notepad++, Sublime Text o simili) e scrivere:

```
print('Hello World!')
```

Si noti che, su Windows, potrebbe non essere immediato modificare l'estensione del file (che di default è .txt).

In questo caso, occorre cambiare l'impostazione del sistema che nasconde le estensioni (oppure modificare l'estensione dall'editor di testo, se possibile).

Come primo esempio, abbiamo aggiunto al nostro file helloworld.py una sola riga di codice, costituita dalla funzione `print()`, che come risultato stamperà una stringa.

Eseguendo il file da terminale, otterremo quanto segue:

```
$ python helloworld.py
Hello World!
```

Modalità Interattiva

- In ogni istallazione di Python è disponibile anche un interprete interattivo in grado di leggere e valutare man mano le espressioni inserite dall'utente
- In modalità interprete interattivo digitando dei comandi Python si ottiene subito una risposta:

```
>>> 5 * 3
15
>>>
>>> a = 5
>>> b = 6
>>> 2 * (a+b) + 3*a
37
```

Ambiente di sviluppo Python

- Integrato
  - IDLE
- Editor di testo
- Finestra di terminale
- Modalità
  - Scripting.
  - Interactive.

Ambiente di sviluppo Python