# Concetti di base di informatica

## Informatica@SEFA 2018/2019 - Lezione 1

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Lunedì, 24 Settembre 2018

*“In science, if you know what you are doing, you should not be doing it. In engineering, if you do not know what you are doing, you should not be doing it. Of course, you seldom, if ever, see either pure state.”*

*(**trad.** “Nelle scienze se sai cosa stai facendo allora dovresti fare altro. In ingegneria è quando non sai cosa stai facendo che dovresti fare altro. Naturalmente è raro, se non impossibile, essere esattamente in una due delle situazioni.”)*

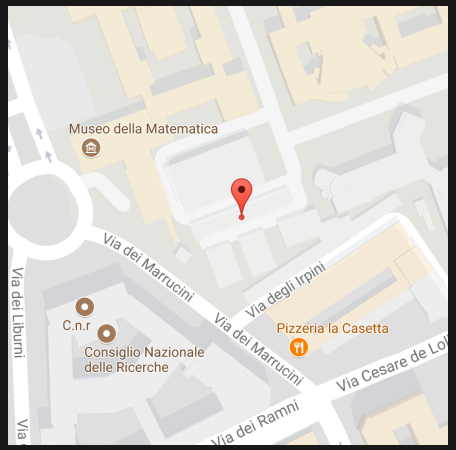
—Richard Hamming, The Art of Doing Science and Engineering

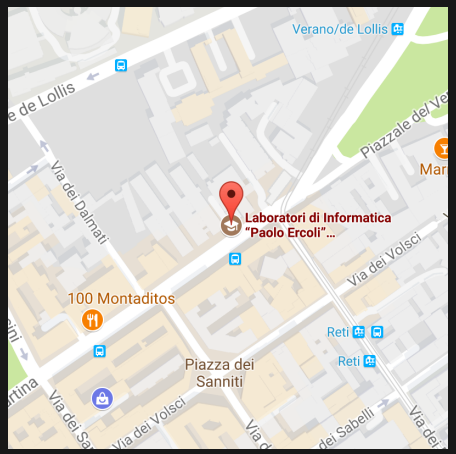
# Benvenuti alla Sapienza

# Informatica@SEFA 2018/2019

- **Docente**: Massimo Lauria
- **Email:** massimo.lauria@uniroma1.it
- **Ufficio:** Stanza n.9 - $4^o$ piano - DSS
- **Ricevimento:** Lunedì 15.00-17.00 (per appuntamento via email)

# Orari e Aule




**Aula 15** - polo didattico CU035

- Mercoledì, ore 15.00-17.00
- Venerdì, ore 12.30-14.30

**Aula XV** - Laboratorio Via Tiburtina, 205

- Lunedì, ore 8.30-10.30

# Libri di testo

- **Fondamenti di Programmazione in Python** di *F. Pellacini*. (link, 10 euro circa)
- **Manuale SQL** di *Proprietà del Dipartimento Tesoro - Ministero del Tesoro, del Bilancio e della Programmazione Economica*. (link, gratis)

# Informazioni aggiornate e notizie sul corso

INFORMATICA@SEFA 2018/2019 Lezioni Esame

## Massimo Lauria

massimo.lauria@uniroma1.it

+39-06-49910496

Stanza n.9 - 4º piano - Dipartimento di Science Statistiche (CU002)

Ricevimento: Lunedì 15.00-17.00

## Avvisi

Nessun avviso.

## Descrizione del corso

Questo corso è stato progettato per dare agli studenti delle nozioni rudimentali di informatica. Ci sarà una minima parte di contenuti teorici e una più sostanziale parte di contenuti tecnici. Gli studenti dovranno imparare a ragionare in maniera logica e non ambigua, in maniera tale da risolvere problemi computazionali scrivendo dei programmi in linguaggio **Python**. Gli studenti inoltre dovranno imparare ad organizzare ed utilizzare *basi di dati relazionali*, interfacciandosi con esse attraverso il linguaggio SQL.

## Testi e risorse consigliate

- (ebook) **Fondamenti di Programmazione in Python** di *F. Pellacini*. [Circa 10 euro]

`http://massimolauria.net/courses/infosefa2018/`

# Diario delle lezioni

Sul sito del corso (cliccare su ’Lezioni’) troverete il diario delle lezioni, che include la descrizione ed il materiale di ogni lezione.

`http://massimolauria.net/courses/infosefa2018/journal.html`

# Attività didattiche

Il corso prevede:

- lezioni frontali
- questionari
- esercitazioni in laboratorio
- partecipazione attiva degli studenti

Nessuna di queste attività influisce sul voto. Pertanto non abbiate paura di **sbagliare** e **partecipare attivamente**.

# Questionario (anonimo)

Indirizzo:
bit.ly/INFO2018-01

# Informatica??

# Introduzione all'informatica

In inglese "computer science"

- È una disciplina principalmente matematica
- Nasce come derivazione della logica
- Ha una forte componente tecnica

# Introduzione all'informatica

In inglese "computer science"

- È una disciplina principalmente matematica
- Nasce come derivazione della logica
- Ha una forte componente tecnica

Non solo programmazione dei computer:

> *"L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi."*
>
> – E. Dijkstra

# Informatica sul dizionario

**informatica** *[in-for-mà-ti-ca] s.f.* Scienza e tecnica che si occupa del trattamento automatico dell'informazione, per mezzo di elaboratori elettronici in grado di raccogliere i dati nella propria memoria, e di riordinarli secondo il programma assegnato.

# Informatica sul dizionario

**informatica** *[in-for-mà-ti-ca] s.f.* Scienza e tecnica che si occupa del trattamento automatico dell'informazione, per mezzo di elaboratori elettronici in grado di raccogliere i dati nella propria memoria, e di riordinarli secondo il programma assegnato.

- (SÌ) ha una componente scientifica ed una tecnica
- (NO) non solo elaboratori elettronici

# Pensiero computazionale (o algoritmico)

Per poter risolvere un problema con il computer

1. formulazione **non ambigua** del problema
2. determinare **i passi logici** che portino alla soluzione
3. codificare questi passi in un programma
4. eseguire su input che codifica quel problema

Gli errori non sono mai “colpa del computer”

# Algoritmo

Figure: Abū Jaʿfar Muḥammad ibn Mūsā al-Khwārizmī (780–850 ca.)

*”una sequenza ordinata e finita di passi (operazioni o istruzioni) elementari che conduce a un ben determinato risultato in un tempo finito”.* — *Wikipedia*

- la lunghezza dell’algoritmo non dipende dall’input
- costituito da passi elementari
- termina sempre
- per ogni input produce un risultato ben definito

# I computer ed il mondo reale

Chi scrive il programma (o chi lo commissiona) **decide** come modellare i problemi **reali** che gli interessano.

Modellare un problema

- attività scientifica
- attività ingegneristica
- a volte addirittura filosofica
- individuare le **caratteristiche essenziali**

Un programma risolve **solo** il modello del problema.

# Machine learning / AI

- mostra all’utente prodotti che comprerà (Amazon)
- mostra un* potenziale fidanzat* (OkCupid)
- mostra le attività più interessanti (Facebook)

# Machine learning / AI

- mostra all'utente prodotti che comprerà (Amazon)
- mostra un* potenziale fidanzat* (OkCupid)
- mostra le attività più interessanti (Facebook)

Come vengono risolti questi problemi?

1. si **decide** come analizzare i tanti dati
2. si **decide** quali sono i parametri significativi
3. si **decide** come usarli per ottenere conclusioni
4. si aggiusta il tiro

# In questo corso impareremo a...

- ragionare in maniera astratta e generale
- riportare le astrazioni ad una dimensione tecnica
- programmare
- programmare in Python
- interrogare archivi di dati in SQL

# Ripasso delle nozioni di base

Per andare avanti con il corso servirà rivedere

- matematica
- logica
- notazioni importanti

# Storia e architettura dei calcolatori

# Charles Babbage e Ada Lovelace (183x)

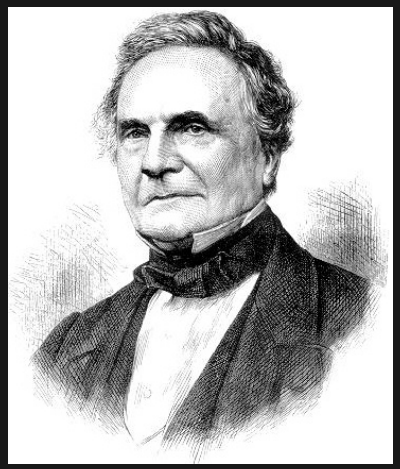


Figure: Chales Babbage (1791–1871)

Figure: Ada Lovelace (1815–1852)

- macchina differenziale (1822)
- macchina analitica (1837) mai realizzata

# Alan Turing (1936)

Figure: Alan Turing (1912–1954)

David Hilbert chiede nel 1928: esiste una **procedura meccanica** in grado di stabilire se un'affermazione matematica è un teorema o meno?

Per risolvere il problema si deve spiegare cosa sia una procedura meccanica: la "spiegazione" di Turing è la **macchina di Turing**. Noi le chiamiamo "computer".

# La macchina universale

L'idea **fondamentale** della macchina Turing (embrionale in Babbage)

- macchina polivalente e **universale**
- una macchina programmabile
- il programma è un input come gli altri

**Tesi di Church-Turing:** *tutto ciò che è calcolabile lo è tramite una macchina di Turing più un programma.*

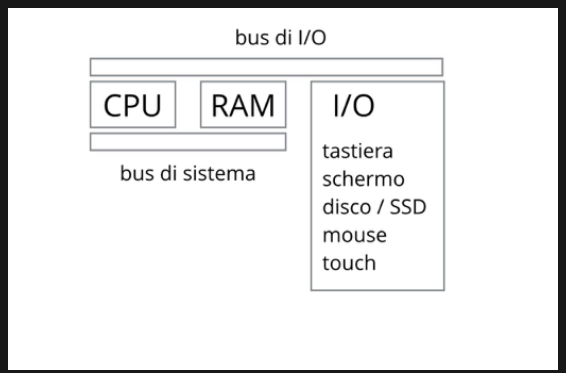
Il risultato è che i computer sono utilizzati per scopi ben oltre quelli per cui sono stati progettati.

# Modello di Von Neumann (1945)

Figure: John von Neumann (1903–1957)

Modello su cui sono basati gli odierni calcolatori, più o meno.

- CPU: il cervello
- RAM: la memoria
- I/O: dispositivi di input and output
- bus: canali di comunicazione



# Memorie

La memoria RAM è volatile

- tutto viene perduto quando si spegne il computer
- costituita da una gerarchia di memorie (cache L1,L2, memoria principale)

Le memorie secondarie sono permanenti

- dischi rigidi, memorie flash su USB, etc…

Si lavora su RAM ma si salva su memoria secondaria

# Tempi di accesso (approssimati, ca. 2010)

Un'istruzione CPU: 1/1.000.000.000 sec = 1 nanosec

| Accesso ai dati | tempo (in ns) |
|---|---|
| memoria cache L1 | 0.5 |
| memoria cache L2 | 7 |
| RAM | 100 |
| 1MB seq. da RAM | 250.000 |
| 4K random disco rigido SSD | 150.000 |
| 1MB seq. da disco SSD | 1.000.000 |
| disco rigido HDD | 8.000.000 |
| 1MB seq. da disco HDD | 20.000.000 |
| pacchetto dati Europa -> US -> Europa | 150.000.000 |

Fonte: Peter Norvig, `http://norvig.com/21-days.html`

Updated by: Jeff Dean `https://ai.google/research/people/jeff`

# Efficienza

Un programma complesso, eseguito su molti dati, può risultare lento.

- la correttezza è la cosa principale
- ma un programma troppo lento è inutilizzabile

Dovete ancora imparare a scrivere programmi corretti ed è presto per parlarvi di efficienza, tuttavia

- i vostri programmi devono girare in tempi ragionevoli,
- se non lo fanno probabilmente ci sono errori logici.