

Introduzione alla Programmazione

Corso di Laurea in
Ingegneria Informatica e Automatica



Aspetti Organizzativi e Introduzione

Professori:

Marco Schaerf (canale 1: A-L)

Giuseppe Santucci (canale 2: M-Z)

Prima lezione

- Aspetti organizzativi
- Descrizione del corso
- Introduzione ai linguaggi di programmazione e a Python
- Prime istruzioni in Python
- L'editor IDLE e l'uso della Shell dei comandi

Informazioni generali sul corso

9 CFU (a partire dall'AA 2017-18, prima era da 12 CFU)

Ha cambiato nome ma corrisponde (con piccolo modifica di programma) al vecchio corso di Fondamenti di Informatica 1

Il corso è erogato nel primo semestre (27 Settembre 2022 – 23 Dicembre 2022).

Due canali:

Canale 1 (A-L) – prof. Marco Schaerf (marco.schaerf@uniroma1.it)

Canale 2 (M-Z) – prof. Giuseppe Santucci (giuseppe.santucci@uniroma1.it)

Le lezioni si tengono in **Aula 204**, presso l'edificio Marco Polo,
viale dello scalo San Lorenzo 82

L'attività di laboratorio si svolge presso il laboratorio Paolo Ercoli
via Tiburtina 205 (aula 15)

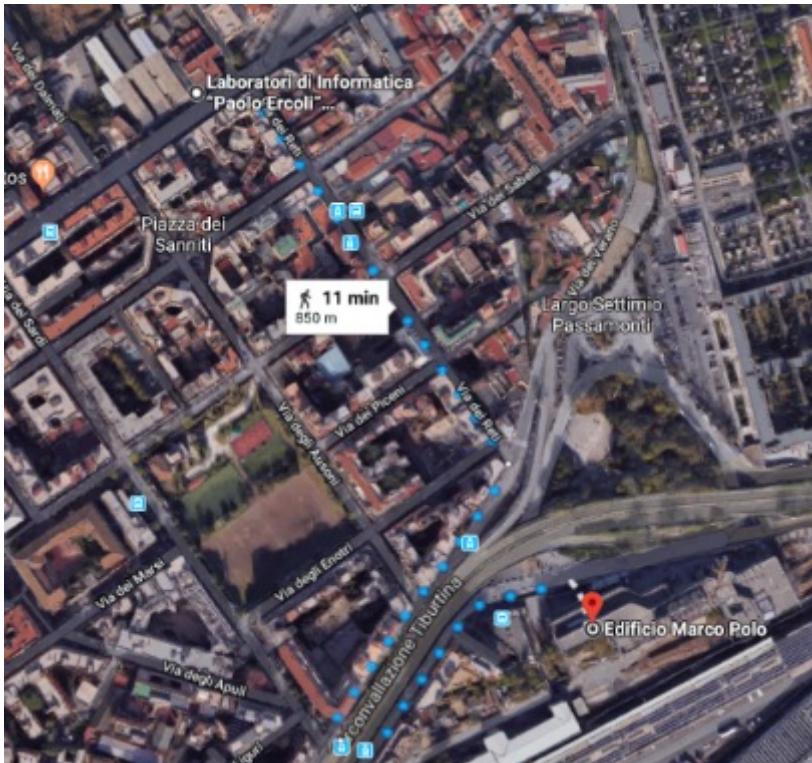
Orari Canale 1

- Martedì Laboratorio 11:00–13:00 – (aule 15, 16, prenotatevi su prodigit SOLO per queste due ore)
- Mercoledì 11:00–13:00
- Giovedì 11:00–13:00
- Venerdì 13:00–15:00

Orari Canale 2

- Martedì Laboratorio 13:00–15:00 – (aule 15, 16, prenotatevi su prodigit SOLO per queste due ore)
- Martedì 15:00–17:00
- Mercoledì 17:00–19:00
- Venerdì 15:00–17:00

Esercitazioni presso il laboratorio Paolo Ercoli



Via Tiburtina, 205

Le esercitazioni in laboratorio si svolgono ogni martedì nella fascia oraria 11:00-15:00.

Sono organizzate in turni in presenza e da remoto (daremo maggiori informazioni su questo nei prossimi giorni).

**Prima Esercitazione:
Martedì 4 Ottobre 2022**

Il corso in estrema sintesi

- Prerequisiti: nessuno (veramente)
- I contenuti del corso sono divisi in due parti:
 - **Programmazione**: semplici programmi scritti in un linguaggio comprensibile da un calcolatore. In questo corso studiamo il linguaggio **Python**
 - **Teoria: “Modelli dell’informatica”** : l’informatica non è solo programmazione

Il corso non è facile (anche per chi sa già programmare):

- **Impegnarsi e studiare fin dall’inizio** (anche se non ci sono interrogazioni in classe)
- *Esercitarsi nella programmazione (Laboratorio)*

Programma: Programmazione in Python

- **Nozioni introduttive:** Il calcolatore. Algoritmi e Programmi. Linguaggi di programmazione. La compilazione
- **Nozioni elementari sulla programmazione in Python:** I/O di base. Uso dell'ambiente di sviluppo
- **Aspetti di base della programmazione in Python:** Espressioni aritmetiche e tipi di dati elementari. Variabili e istruzioni di assegnazione. Il tipo di dato stringa
- **Decisioni:** costrutti if ed else-if (elif)
- **Cicli:** Ciclo while. Ciclo for. Cicli annidati
- **Funzioni e moduli:** Introduzione alla programmazione Python con funzioni. Moduli e loro uso. Esecuzione delle funzioni
- **Liste:** Proprietà di base. Operazioni sulle liste. Algoritmi elementari che fanno uso di liste. Rappresentazione di tabelle e matrici
- **Dizionari:** Proprietà di base. Accesso e manipolazione di dizionari
- **File e file system:** Apertura, chiusura e manipolazione di file di testo. Funzioni di base per l'accesso al file system
- **Librerie avanzate e visualizzazione:** Numpy, MatPlotLib e Pandas

Programma: Modelli dell'Informatica

- **Architettura dei calcolatori:** Architettura di von Neumann. Esempi di Linguaggio Macchina. La legge di Moore
- **Rappresentazione dell'Informazione:** Rappresentazione di caratteri e stringhe, Rappresentazione dei numeri positivi. Rappresentazione in complemento a due. Rappresentazione in virgola mobile. Errore assoluto ed errore relativo

Appelli d'esame

- 2 Appelli fra il 9 gennaio ed il 17 febbraio 2023
- 2 Appelli fra il 1 giugno ed il 21 luglio 2023
- 1 Appello fra il 1 settembre ed il 15 settembre 2023

- Appelli straordinari:

1 Appello fra il 13 marzo ed il 14 aprile 2023, riservato a studenti part-time, fuori corso nell'A.A. 2022/23, studenti genitori, studenti lavoratori, studenti con disabilità o D.S.A

1 Appello fra il 2 ottobre e il 3 novembre 2023, riservato a studenti part-time, fuori corso nell'A.A. 2021/22, studenti genitori, studenti lavoratori, studenti con disabilità o D.S.A. e agli studenti iscritti al terzo anno nell'A.A. 2022-2023

Modalità d'esame

- La prova d'esame consiste di
 - **Una prova pratica al calcolatore**, durante la quale lo studente dovrà realizzare alcune funzioni in **Python** e rispondere a delle **domande di teoria**. Lo durata dell'esame è di circa 1h e 45m
- La prova è identica e si svolge nello stesso giorno per i due canali
- Durante il corso saranno mostrati degli esempi di prove di esame

Materiale Didattico

- Per la parte di Programmazione

- **Slide e dispense del corso**
(distribuite sul sito)
 - **Testi e soluzioni delle esercitazioni**
 - Libri di Riferimento (non obbligatori):

Horstmann, Necaise "Concetti di Informatica e fondamenti di Python",
(Testo più semplice, adatto a chi non ha mai programmato)

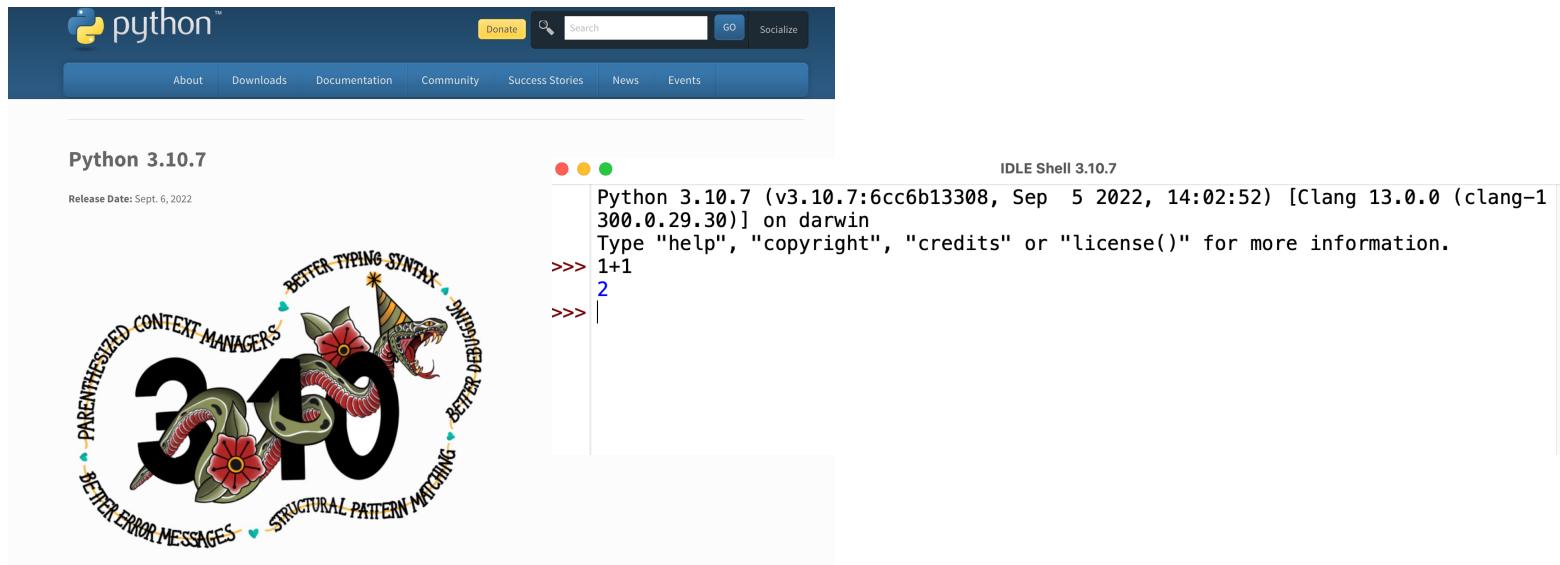
John V. Guttag “Introduzione Alla Programmazione Con Python:
Dal Pensiero Computazionale Al Machine Learning”,
(testo più avanzato, adatto a chi sa già programmare)

- Per la parte di Modelli

- **Slide e dispense** (distribuite sul sito)

Ambiente di Lavoro per la Programmazione

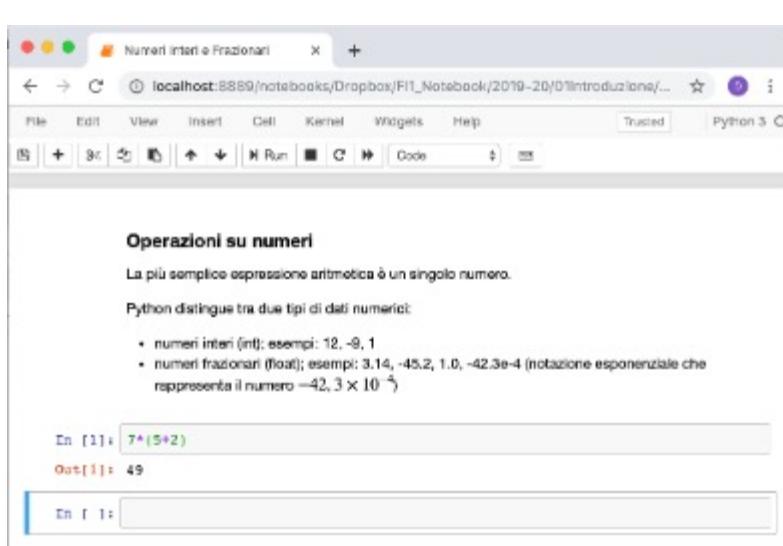
- Python 3.10.7 – disponibile al sito
<https://www.python.org/>



- Editor **IDLE**, incluso nell'installazione di Python

Consultazione file Notebook su Python (distribuiti sul sito del corso)

Le note del corso su Python saranno distribuite nella forma di file Notebook (file .ipynb) che possono essere consultati con Jupyter Notebook (o Jupyter Lab)



Le istruzioni relative all'installazione degli ambienti di lavoro sono disponibili sul sito del corso

Sito Web del Corso

- Sulla piattaforma **Google Classroom**
<https://classroom.google.com/c/NTQ1Njg4NzE1ODA5?cjc=ivqwabh>
- E' necessario registrarsi al corso usando l'account Google Sapienza (username=vostra email istituzionale)
- Alternativamente potete accedere a classroom con l'account Google Sapienza dal link <https://classroom.google.com>
- Una volta effettuato il login su classroom, cliccate sul simbolo + in alto a destra



- Vi verrà chiesto il codice del corso. Inserite **ivqwabh**



Sito Web del Corso



Introd. alla programmazione

Avvisi del corso di laurea



<https://classroom.google.com/c/NDAxOTU1NTQ0Mjc4?cjc=y2ae7sp>

Introduzione:

L'elaborazione delle Informazioni

Elaborazione delle Informazioni

Un sistema per la rappresentazione e l'elaborazione di informazioni può

- raccogliere impressionanti quantità di dati
- elaborare i dati secondo le istruzioni fornite producendo nuovi dati
- rendere disponibili questi dati con prospettive diverse a utenti diversi e in parti diverse del mondo

Esempi: social networks, banca via internet, calcolo scientifico, video giochi

Elaborazione delle Informazioni

- Computer (calcolatore, microprocessore...)
 - Esegue semplici operazioni aritmetiche e logiche ad una velocità molto maggiore degli esseri umani (1 nanosecondo per operazione = 1 milionesimo di un millesimo di secondo, 10^{-9} secondi)
- Programma
 - Insieme di istruzioni eseguibili da un computer
- Hardware
 - Dispositivi fisici di un sistema di elaborazione
- Software
 - Programmi eseguiti su un elaboratore

Cos'è l'informatica?

L'informatica NON è lo studio dei computer come artefatti.
Non studieremo come si costruisce un computer (o come lo si ripara)

Siamo interessati a USARE i computer

- L'informatica ha a che fare con lo studio di risoluzione di problemi (“problem solving”) per la cui soluzione si scrive un programma che viene eseguito da un computer
- L'informatica si basa su una serie di tradizioni intellettuali che comprende aspetti della matematica e dell'ingegneria

Cos'è l'informatica?

L'informatica svolge un ruolo sempre più importante in altre discipline:

- Biologia: analizzare il genoma umano
- Economia: la creazione di migliori modelli economici e finanziari
- Ambiente: i modelli climatici richiedono la moderna tecnologia informatica
- Letteratura: Analisi computerizzata aiuta a risolvere paternità contestata
- Cinema e intrattenimento: utilizzo di informatica grafica nei film e nei videogiochi
- Vita di tutti i giorni: sistemi di guida automatica, navigatori ...

Programma e dati di ingresso

Un **programma** è una sequenza di istruzioni scritte in un linguaggio di programmazione (ed eseguibili da un calcolatore) che realizzano un algoritmo

Dati di ingresso: di solito un problema non deve essere risolto una sola volta ma molte volte con dati (di ingresso) diversi

Esempi:

- Trovare il percorso più breve in una rete stradale:
dati di ingresso: rete stradale in esame, punto di partenza e punto di arrivo
- Eseguire una ricerca nel web:
dati di ingresso: web (in quel momento), parole date nella ricerca

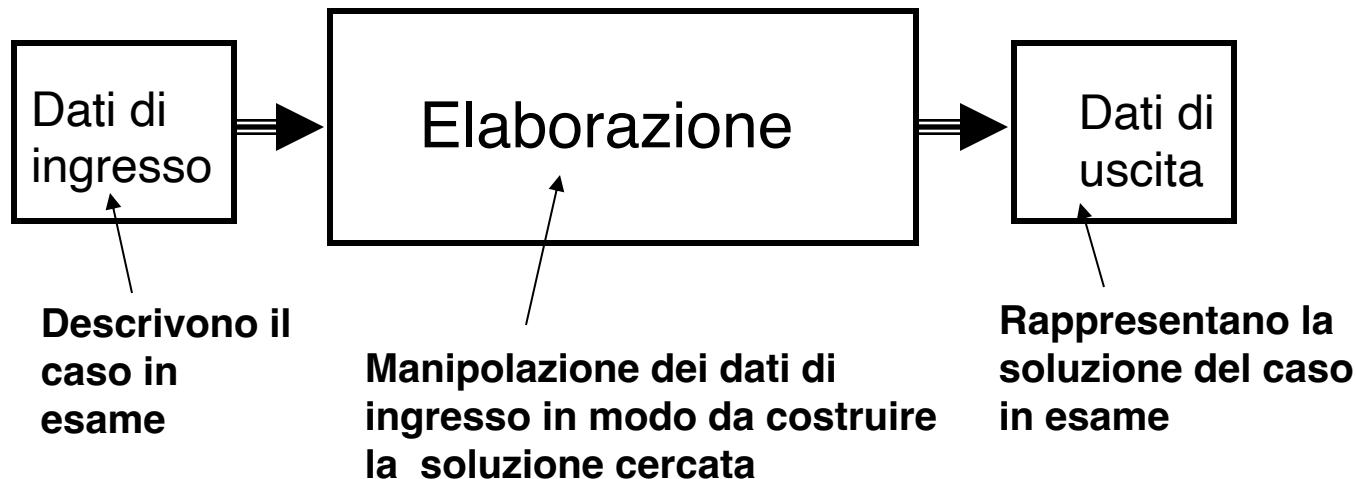
Elaborazione dell'informazione

Risoluzione di un problema

1. Dato un problema **si trova un algoritmo** di soluzione
2. Si scrive l'algoritmo di soluzione in un **programma eseguibile da un calcolatore**
3. Dato un programma, un calcolatore e dei dati di ingresso, **l'esecuzione del programma con i dati di ingresso** sul calcolatore risolve il problema su quei dati

Elaborazione dell'informazione

Come viene risolto un problema :



Algoritmi e Programmi

L'informatica studia la risoluzione di problemi (“problem solving”) realizzando un **programma** di soluzione che viene eseguito da un computer

Un **algoritmo** è come una “ricetta di cucina”, ovvero un procedimento schematico, composto da una sequenza di istruzioni elementari, che consente di risolvere un problema

Esempi di problemi: ordinare alfabeticamente una serie di nomi, trovare le pagine più interessanti in una ricerca su un motore di ricerca, trovare il percorso stradale più breve

Algoritmo

In modo informale, si può pensare a un algoritmo come ad una procedura per risolvere un problema

Una definizione più formale richiede che:

- I passi (le **istruzioni**) dell'algoritmo siano definiti in modo chiaro e **non ambiguo**
- Sia eseguibile, nel senso che i suoi passi siano **eseguibili da una macchina**
- Sia finito, nel senso che **termini dopo un numero finito di passi**

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 linguaggio di programmazione

→ 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 do operazione 1
else do operazione 2

while condizione
do operazione

for ogni valore compreso tra . . . e . . .
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 comune divisore** di due interi positivi m ed n*

Algoritmo

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

Raffinamento del passo 1

- 1.1 inizializza A all'insieme vuoto
- 1.2 **for** ogni numero i compreso tra 1 ed m
 if i è divisore di m
 then aggiungi i ad A

Raffinamento del passo 2

Identico al raffinamento del passo 1

...

Programmi: rappresentano algoritmi in un linguaggio di programmazione

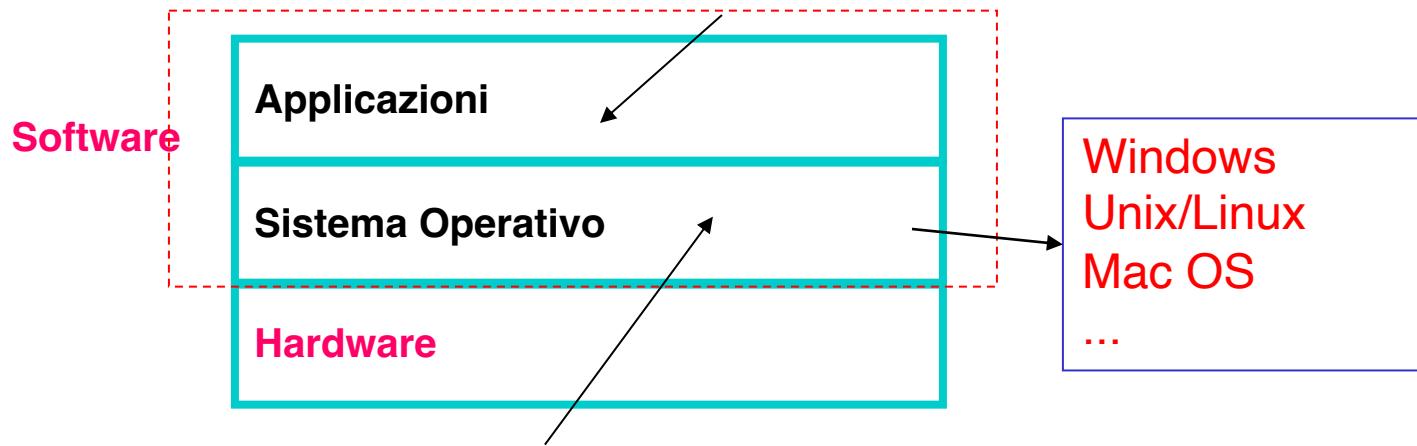
Esempio: Programma che

- a) Legge due numeri da tastiera
- b) Ne calcola il MCD e
- c) 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 ");
A=set()
B=set()
for i in range(1,m+1):
    if m % i == 0:    #il resto della divisione tra m e i vale zero
        A.add(i)
for i in range(1,n+1):
    if n % i == 0:
        B.add(i)
C = A & B           # & calcola l'intersezione
print(max(C))
```

Applicazioni e Sistema Operativo

Applicazioni: È il livello di SW con cui interagisce l'utente e comprende programmi quali: *Word*, *PowerPoint*, *Excel*, *Chrome*,



È il livello di SW che interagisce direttamente con l'HW e che si occupa di un uso corretto ed efficiente delle risorse fisiche

Linguaggi di programmazione

Dato un problema, un algoritmo di soluzione deve essere scritto in un linguaggio comprensibile da una macchina

I calcolatori sono in grado di comprendere solo programmi scritti in ***linguaggio macchina***, che è un linguaggio specifico per il processore (CPU) del calcolatore e complicato per gli uomini

I programmi sono scritti in un linguaggio più facile da utilizzare; ce ne sono tantissimi che sono detti ***linguaggi ad alto livello***; Python è uno di questi, semplice da imparare e quindi adatto per iniziare; ce ne sono ovviamente molti altri..

Linguaggi di programmazione

Ci sono molti linguaggi di programmazione: **Pascal, C, C++, Java, Ada, Perl, Php, Python...**

- Questi linguaggi condividono gli stessi principi
- Ognuno però ha caratteristiche specifiche che lo distinguono dagli altri

Nelle applicazioni di oggi

- Elaborazione complesse dei dati sono spesso fatte usando linguaggi diversi
- Imparando i concetti fondamentali di un linguaggio sarete in grado di apprendere da soli nuovi linguaggi

Linguaggi di programmazione

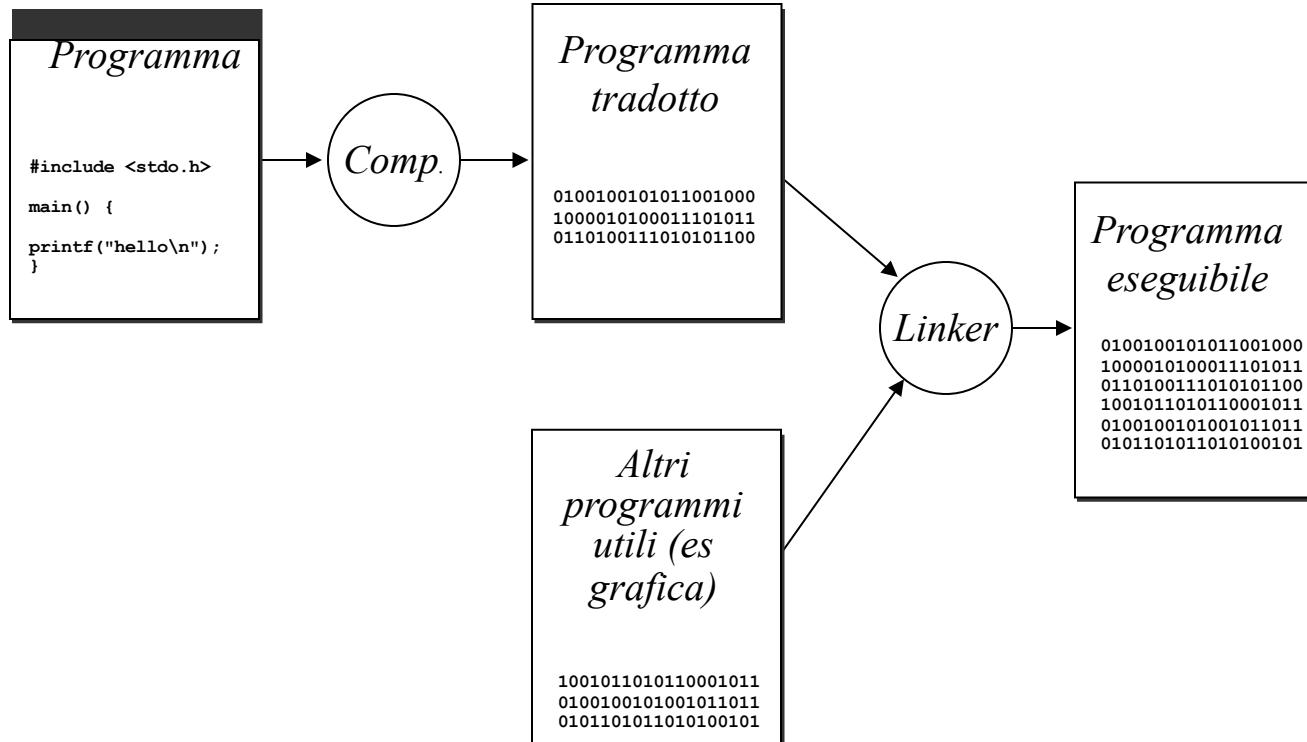
I linguaggi ad alto livello **non sono** direttamente eseguibili da un elaboratore

Per eseguire un programma scritto in un linguaggio ad alto livello dobbiamo **tradurlo** nel linguaggio macchina

Due possibilità principali

- **Compilatore:** programma che traduce un linguaggio ad alto livello in un linguaggio macchina eseguibile dal sistema operativo
- **Interprete:** il programma scritto in linguaggio ad alto livello viene simulato senza tradurlo completamente (una istruzione alla volta)

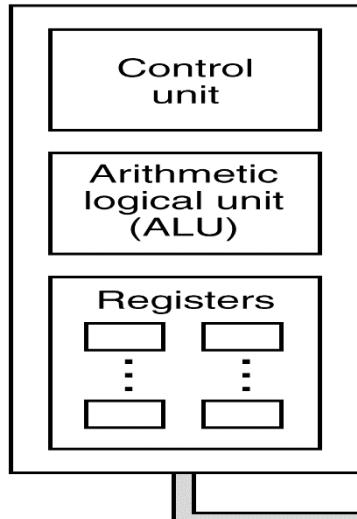
Il processo di compilazione



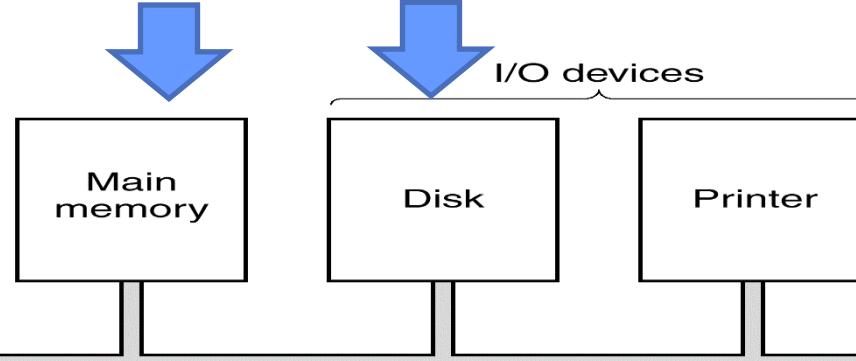
Ambienti di programmazione



Central processing unit (CPU)



Ambiente di programmazione
(Editor, shell, compilatore,
debugger, etc.)



Bus

Introduzione:

Il linguaggio Python

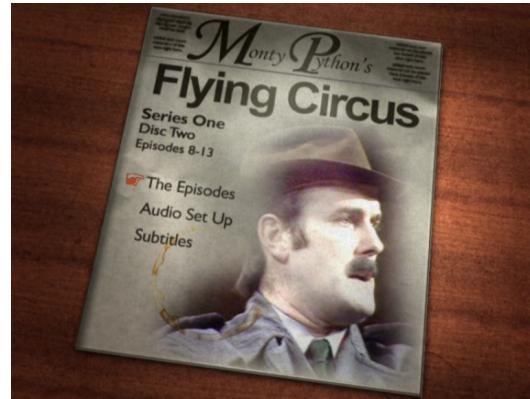
Python

- 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



Monty Python ?

- Comici inglesi
- Monty Python's Flying Circus
- BBC 1969-74
- Comicità grottesca e surreale (inventori della parola Spam)
 - Graham Chapman
 - John Cleese
 - Terry Gilliam
 - **Eric Idle → editor Python**
 - Terry Jones
 - Michael Palin



Python

- Python è **free software**: il download dell'interprete e l'uso di Python è completamente gratuito, ma oltre a questo Python può essere liberamente modificato e ridistribuito, secondo le regole di una licenza pienamente *open-source*
- Lo sviluppo di Python viene coordinato dall'organizzazione no-profit [Python Software Foundation](#)
- Guido van Rossum è rimasto per anni il “*Benevolo dittatore a vita*” di Python, colui che all'interno della comunità di sviluppatori avrebbe comunque avuto il peso maggiore nelle decisioni relative allo sviluppo del linguaggio
- Dal 2018 van Rossum ha però lasciato il suo ruolo, per via delle crescenti difficoltà legate al suo svolgimento. Ha, in particolare, dichiarato:
- *“I don't ever want to have to fight so hard for a PEP (ndr: proposta di modifica al linguaggio) and find that so many people despise (sic) my decisions.”*

Python

- Supporta **diversi paradigmi di programmazione**:
 - **Imperativo** (enfasi sulle operazioni intese come **comandi** che cambiano lo stato dell'elaborazione)
 - **Funzionale** (enfasi sulle operazioni intese come **funzioni** che calcolano risultati)
 - **Orientato agli oggetti** (enfasi sugli **oggetti** che complessivamente rappresentano il dominio di interesse)
- Offre un **controllo dei tipi**
 - **Forte** (strong typing): Ogni elemento sintattico che denota un valore (ad es. variabile) è associato a un determinato tipo durante l'esecuzione
 - **Eseguito a run-time** (dynamic typing): il controllo del tipo della variabile è effettuato a runtime

Python

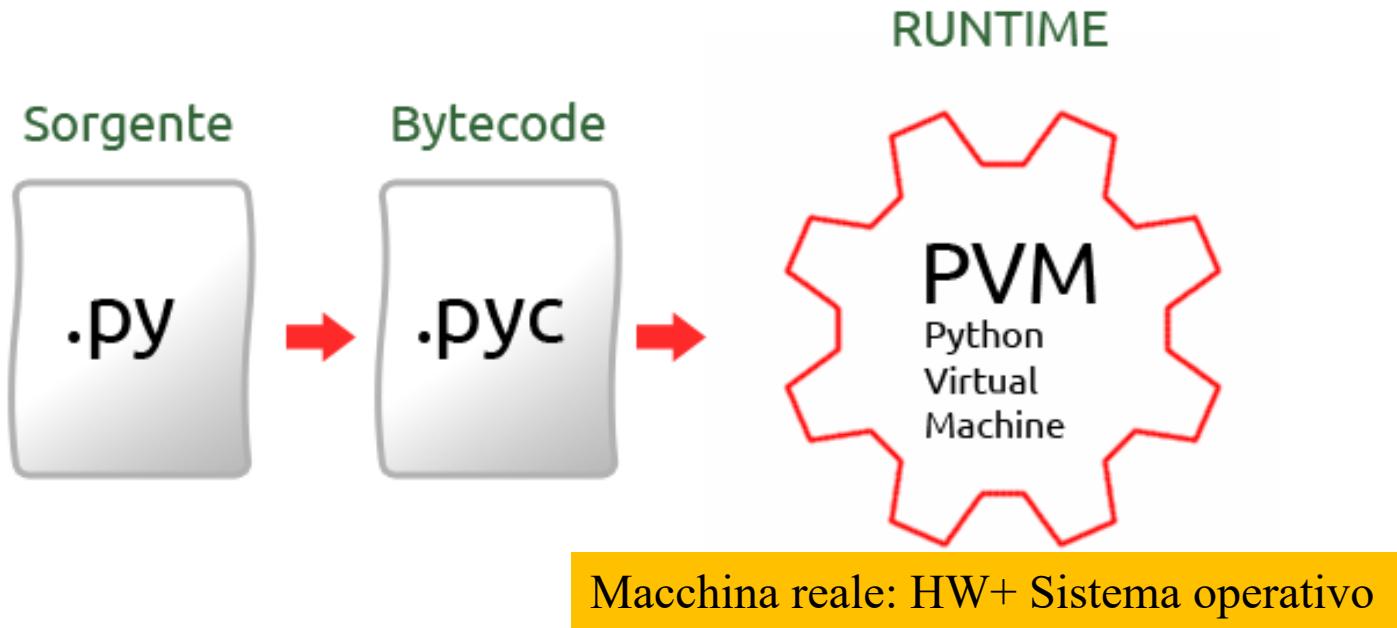
- Offre una gestione automatica della memoria
 - Garbage collector
- Fornito di una libreria built-in estremamente ricca
- Sono disponibili numerosissime altre librerie
- Offre costrutti robusti per la gestione delle eccezioni
 - come mezzo per segnalare e controllare eventuali condizioni di errore (incluse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

Python

Linguaggio **interpretato** (o meglio pseudo-compilato):

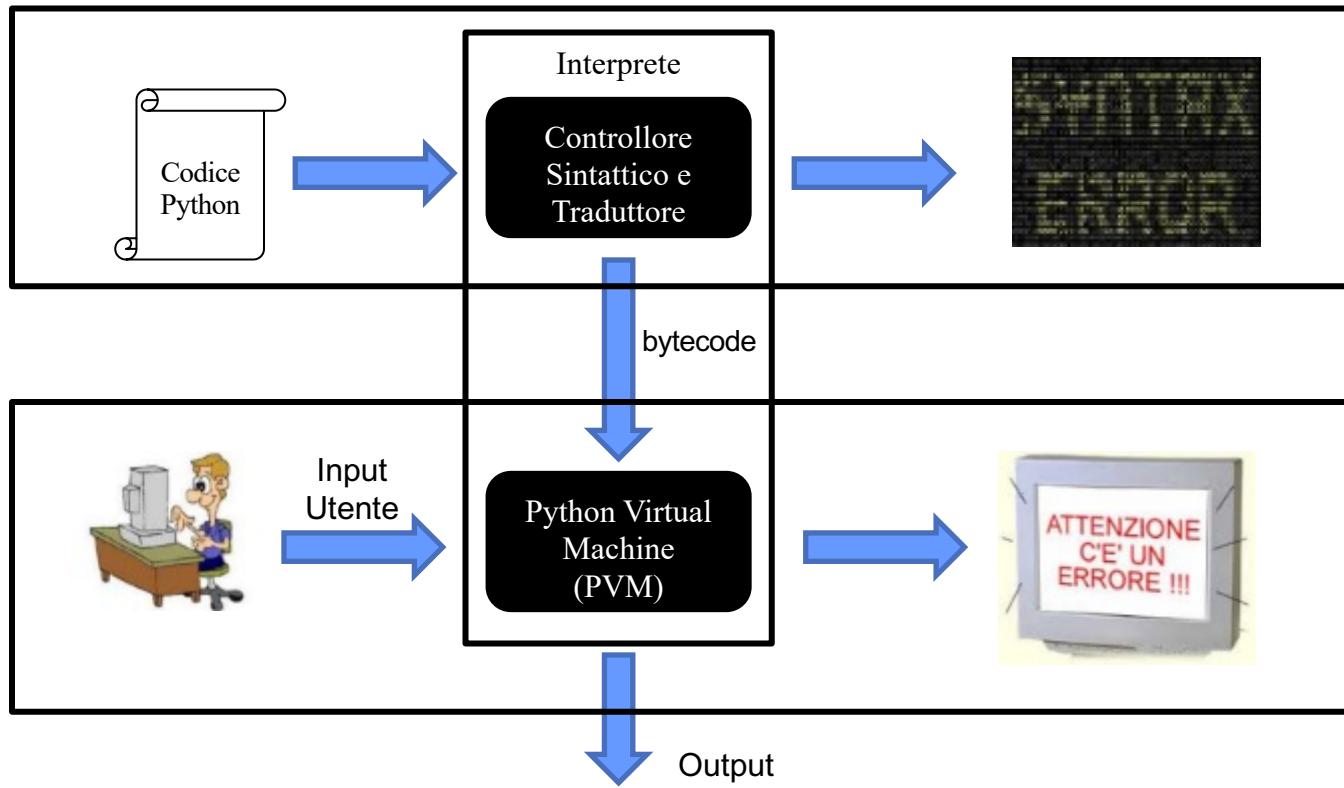
- un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo
- il codice sorgente passa prima per una fase di pre-compilazione in bytecode (riduce la dipendenza dall'hardware)
 - viene riutilizzato dopo la prima esecuzione del programma, evitando così di dover ogni volta interpretare il sorgente e 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
- Java segue grossomodo la stessa filosofia

Interpretazione di un programma Python (1\2)



Una macchina virtuale è un **programma** per una macchina reale (cpu+hw+sistema operativo) che simula il comportamento di una macchina reale

Interpretazione di un programma Python (2\2)



Esecuzione di script/programmi .py

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

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

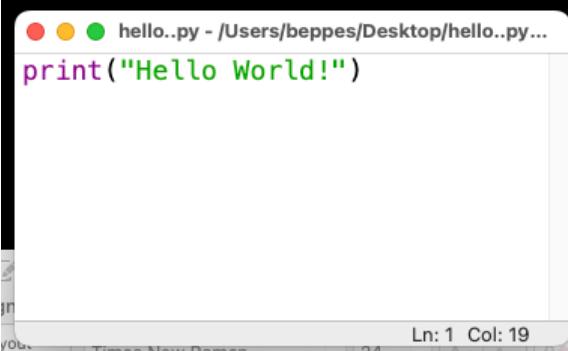
Il modo più agevole per creare uno script python è usare un **Editor per programmi Python** (come ad esempio IDLE – vedi oltre)

Come primo esempio, abbiamo inserito nel file **helloworld.py** una sola riga di codice, costituita dalla funzione **print()**, che come risultato stamperà una stringa
Eseguendo il file dalla console (detta anche terminale), otterremo quanto segue:

```
$ python helloworld.py
```

```
Hello World!
```

Nota: Se usiamo un Editor per programmi Python (come IDLE), è possibile eseguire lo script aprendolo nell'editor e usando i comandi per l'esecuzione messi a disposizione dell'editor. Verrà automaticamente mostrata una finestra di shell con il risultato dell'esecuzione.



```
● ○ ● hello..py - /Users/beppes/Desktop/hello..py...
print("Hello World!")
```

IDLE



```
IDLE Shell 3.10.7
Python 3.10.7 (v3.10.7:6cc6b13308, Sep  5 2022,
14:02:52) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license"
()" for more information.

>>>
=====
RESTART: /Users/beppes/Desktop/hello..py =====
Hello World!
>>>
```

Shell

Modalità Interattiva

- In ogni installazione 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
```

```
>>> a
```

```
5
```

- N.B. In uno script Python per vedere il valore di una espressione o il contenuto di una variabile occorre usare una print:

- `print(5*3)`

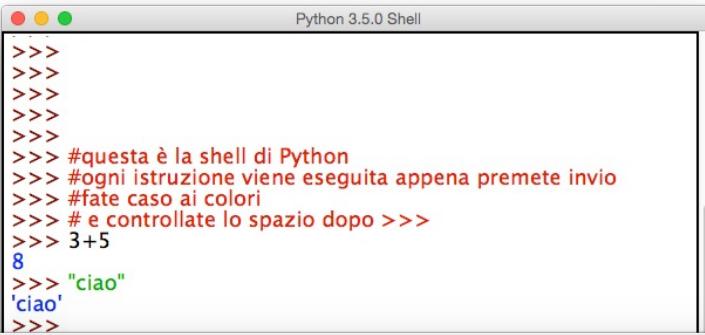
- `print(a)`



Ambiente di sviluppo Python

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

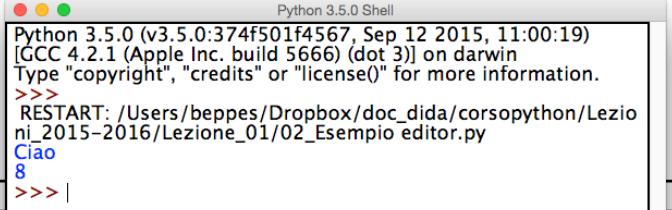
Python IDLE: Shell o Editor?



```
>>>
>>>
>>>
>>>
>>>
>>> #questa è la shell di Python
>>> #ogni istruzione viene eseguita appena premete invio
>>> #fate caso ai colori
>>> # e controllate lo spazio dopo >>>
>>> 3+5
8
>>> "ciao"
'ciao'
>>>
```



IDLE :
Integrated DeveLopment Environment



02_Esempio editor.py - /Users/beppes/Dropbox/doc_dida/corsopython/Lezioni_2015-2016/Lezione_01/02_Esempio editor.py (3.5.0)

```
#questo è l'editor di file di Python
#tutte le istruzioni vengono eseguite insieme (F5 o Run->Run module)
#tutte le righe che cominciano con # vengono ignorate (commenti)
#NON mostra il risultato delle espressioni --> per vedere il risultato DOVETE usare print
#fate caso ai colori

print("Ciao")
print(5+3)
8+8
```

F5 ->

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 12 2015, 11:00:19)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /Users/beppes/Dropbox/doc_dida/corsopython/Lezioni_2015-2016/Lezione_01/02_Esempio editor.py
Ciao
8
>>> |
```

Per la prossima lezione

- Scaricate e installate sul vostro computer:
 - python 3.10.7
 - Jupyter
- Le istruzioni si trovano sul sito