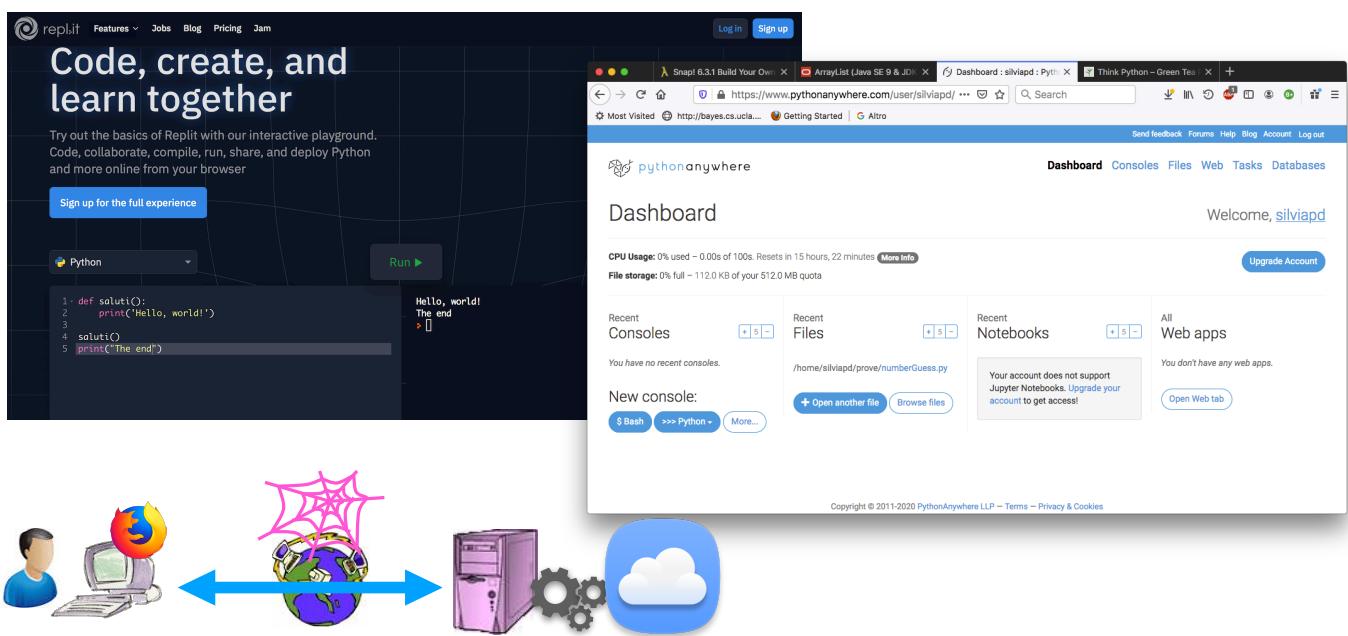


Esercizio

- Scrivere ed seguire il programma dei 3 giochi in Snap!
<https://snap.berkeley.edu>
- Scrivere ed eseguire il programma dei 3 giochi in Python, nell'ambiente di sviluppo prescelto.
- Provare a fare piccole variazioni del programma e vedere che succede.
- **Ambienti di sviluppo per Python:**
 - <https://www.programiz.com/python-programming/online-compiler/>
 - https://www.w3schools.com/python/trypython.asp?filename=demo_compiler
 - (molte funzionalità) <https://www.pythonanywhere.com/>
 - <https://pythononlinecompiler.com/python-online-compiler-310/>
 - https://www.onlinegdb.com/online_python_interpreter



correttezza e qualità del software

- estremamente complesso descrivere in maniera **precisa** ed **esaustiva**
 - **cosa deve fare** un programma (specifica/algoritmo) **esistono tecniche e buone prassi (di ing software)**
 - **cosa fa** un programma e **come lo fa** (implementazione)
- Avere a disposizione **il codice sorgente** offre completa **trasparenza**, ma non completa **intelligibilità**.

cosa fa il programma...

- Avere a disposizione **il codice sorgente** offre completa **trasparenza**, ma non completa **intelligibilità**.

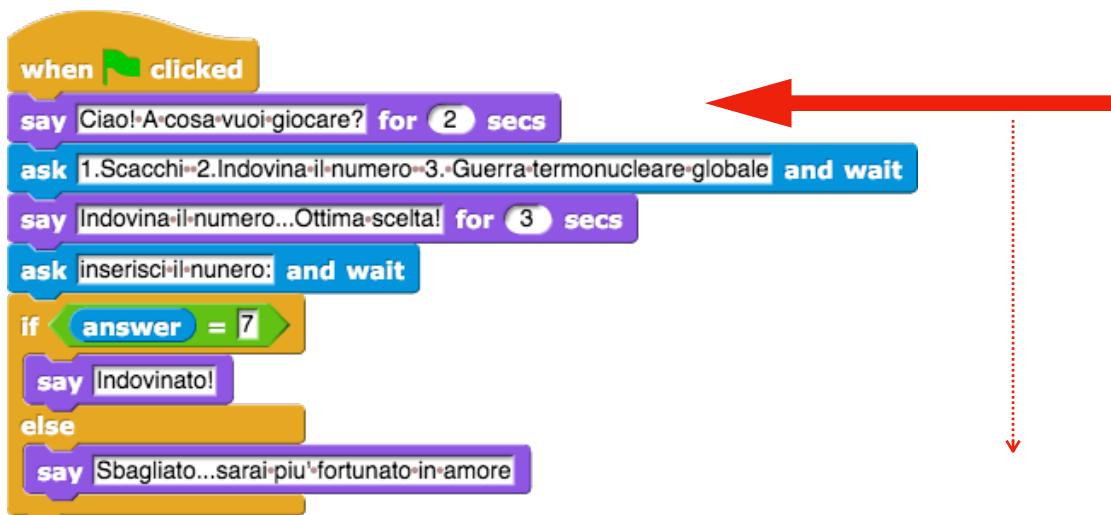
- Il codice **scritto** è una sequenza di istruzioni
- La macchina **esegue** una sequenza di istruzioni
- ma la **sequenza di istruzioni eseguite** non sempre coincide con la **sequenza scritta (flusso di controllo)**

flusso di controllo

- Chiameremo **flusso di controllo** (*flow of control*) l'ordine in cui il computer esegue le istruzioni:
 - La macchina **inizia** eseguendo la prima istruzione, poi la successiva, e così via, **seguendo l'ordine in cui le istruzioni appaiono** nel programma.
 - l'esecuzione di alcune istruzioni (**if**, **while**, **fun()**, **throw...**) provoca il **salto del controllo** ad una specifica istruzione, che non è la successiva
 - l'esecuzione **si ferma** dopo aver eseguito l'ultima istruzione del programma.

Lo strumento online **Python Tutor** permette di eseguire un programma visualizzando come il controllo fluisce da un'istruzione alla successiva.

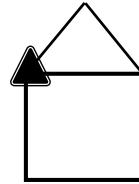
<http://pythontutor.com/visualize.html#mode=edit>



```
print('Ciao! a cosa vuoi giocare?')
input('1.Scacchi 2.Indovina il numero 3.Guerra termonucleare globale')
print('Hai scelto: Indovina il numero Ottima scelta!')
answer = int(input('inserisci il numero: '))
if answer == 7 :
    print('Indovinato!')
else:
    print('Sbagliato...sarai più fortunato in amore')
```

The Python code shown in the image is identical to the Scratch script above. Red arrows point from the "say" and "ask" blocks in the Scratch script to their corresponding lines in the Python code, illustrating the flow of control. The "if" block in the Scratch script corresponds to the conditional block in the Python code, and the "else" block corresponds to the "else:" part of the conditional statement.

in **Snap!** ci sono istruzioni per disegnare una linea e girare il cursore, è un linguaggio pensato anche per questo.



Algoritmo per *disegnare una casa*:

- *disegna un quadrato*
- *disegna un triangolo*

Decomposizione

Algoritmo per *disegnare un quadrato*:

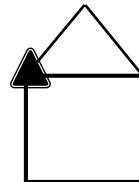
ripeti 4 volte

- *gira il cursore di 90 gradi* ➔
- *disegna un lato*

Algoritmo per *disegnare un triangolo*:

...

in **Snap!** ci sono istruzioni per disegnare una linea e girare il cursore, è un linguaggio pensato anche per questo.

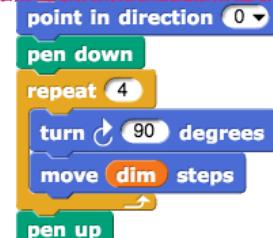


Algoritmo per *disegnare una casa*:

- *disegna un quadrato*
- *disegna un triangolo*

Decomposizione

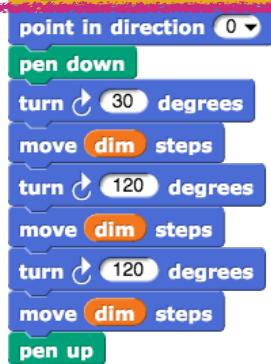
dall'algoritmo
all' **implementazione**



Algoritmo per *disegnare un quadrato*:

ripeti 4 volte

- *gira il cursore di 90 gradi* ➔
- *disegna un lato*

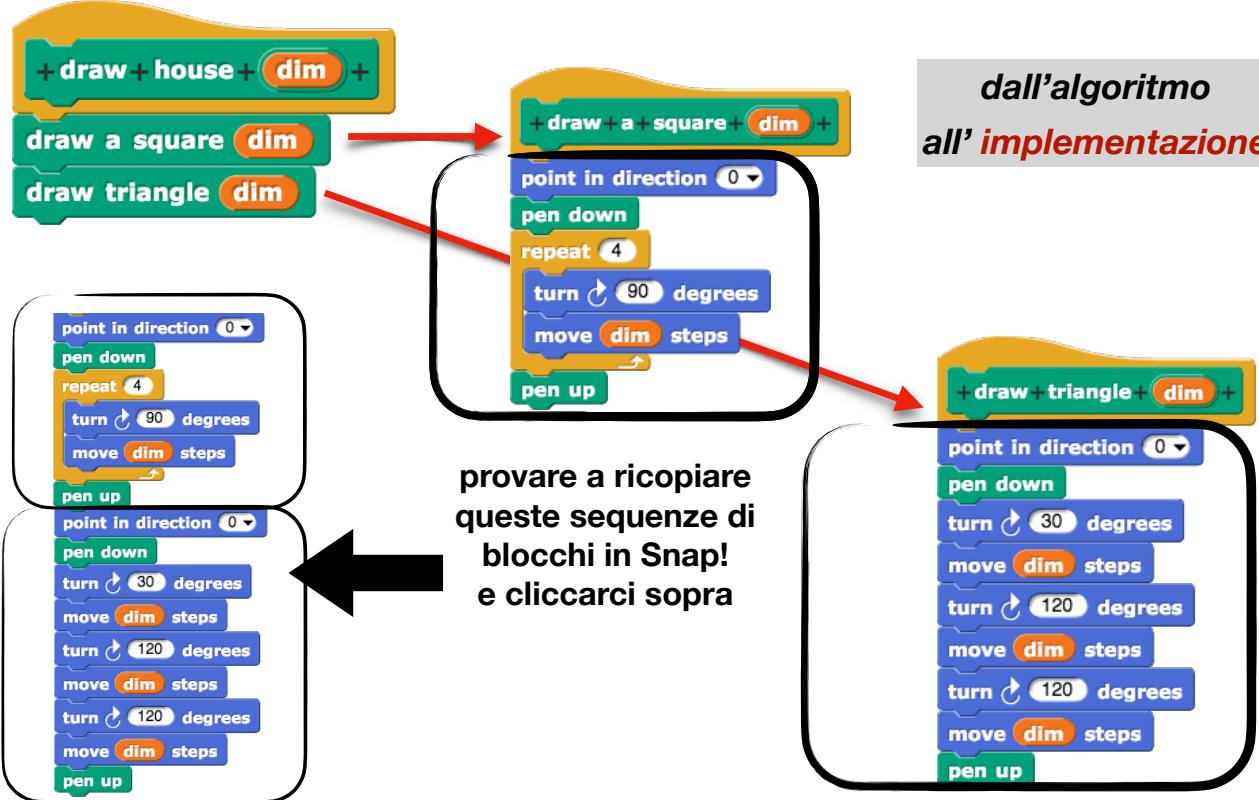
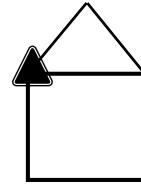


Algoritmo per *disegnare un triangolo*:

...

Algoritmo per disegnare una casa:

- disegna un quadrato
- disegna un triangolo

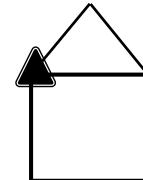


```
# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)
```

```
# procedura per disegnare un quadrato
def draw_square(dim):
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)
```

```
# procedura per disegnare un triangolo
def draw_triangle(dim):
    print('gira di 30 e disegna lato lungo',dim)
    print('gira di 120 e disegna lato lungo',dim)
    print('gira di 120 e disegna lato lungo',dim)
```

```
# programma: disegna una casa usando la procedura
draw_house(50)
```



python 3

in Python non ci sono istruzioni per disegnare, quindi scriviamo le parole (opp. usare la libreria grafica:

`import turtle`)

```

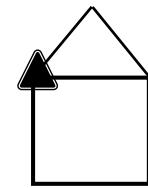
# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)

# procedura per disegnare un quadrato
def draw_square(dim):
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)
    print('gira di 90 e disegna lato lungo',dim)

# procedura per disegnare un triangolo
def draw_triangle(dim):
    print('gira di 30 e disegna lato lungo',dim)
    print('gira di 120 e disegna lato lungo',dim)
    print('gira di 120 e disegna lato lungo',dim)

# programma che disegna una casa:
draw_house(50)

```



FLUSSO DI CONTROLLO

la sequenza di istruzioni
eseguite non coincide
con la sequenza scritta

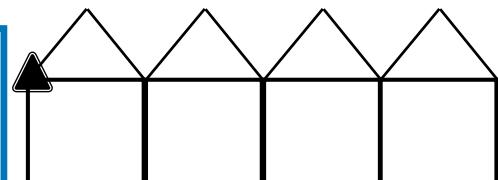
è la prima istruzione eseguita

qual è l'ultima istruzione ad essere eseguita?

Algoritmo per disegnare una fila di case:

ripeti finche' c'è spazio:

- disegna una casa
- sposta il cursore



Decomposizione

Algoritmo per disegnare una casa:

- disegna un quadrato
- disegna un triangolo

Algoritmo per disegnare un quadrato:

ripeti 4 volte

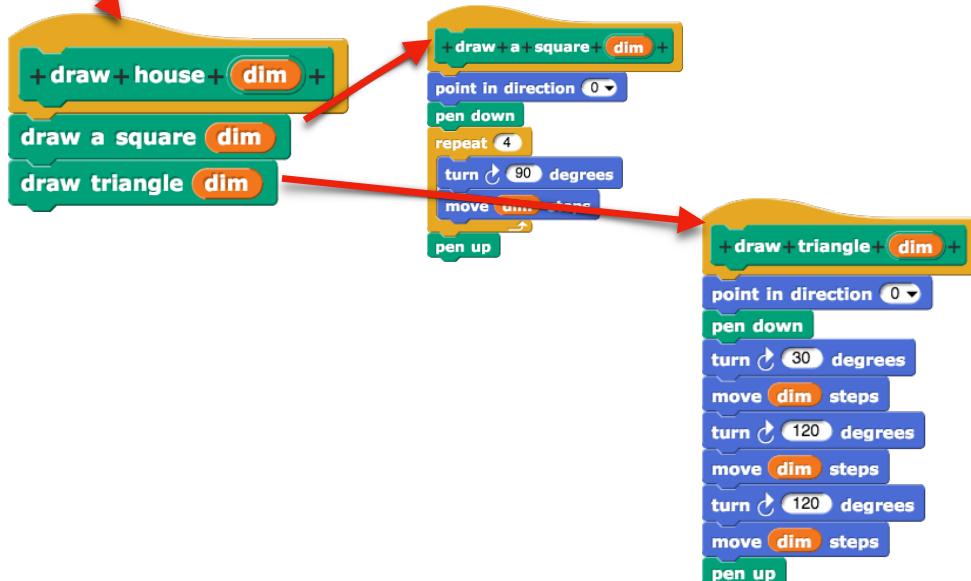
- gira il cursore di 90 gradi
- disegna un lato

Algoritmo per disegnare un triangolo:

...



dall'algoritmo
all' implementazione



```

# procedura per disegnare una fila di case:
def draw_a_row_of_houses():
    max_area=240
    position=0
    # 50 is the dimension of a house
    while position+50 < max_area:
        draw_house(50)
        print('sposta la penna')
        position=position+50

```

```

# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)

```

```

# procedura per disegnare un quadrato
def draw_square(dim):
    print('disegna un quadrato')

```

```

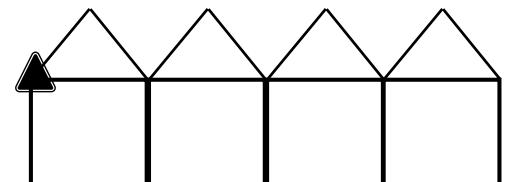
# procedura per disegnare un triangolo
def draw_triangle(dim):
    print('disegna un triangolo')

```

```

# programma che disegna una fila di case:
# invoca l'algoritmo definito sopra
draw_a_row_of_houses()

```



Decomposizione



qual è l'ultima istruzione ad essere eseguita?

è la prima istruzione eseguita

Esercizio

1. tracciare il flusso di controllo del programma che, dopo le definizioni delle procedure, invoca `draw_a_row_of_houses()`
2. tracciare il flusso di controllo del programma che, dopo le definizioni delle procedure, invoca la seguente sequenza di istruzioni:
`draw_house(10)`
`draw_house(15)`

(opzionale: controllare se le soluzioni sono corrette usando Python Tutor)

```
# procedura per disegnare una fila di case:
def draw_a_row_of_houses():
    max_area=240
    position=0
    # 50 is the dimension of a house
    while position+50 < max_area:
        draw_house(50)
        print('sposta la penna')
        position=position+50

# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)

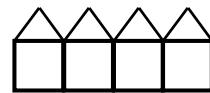
# procedura per "disegnare" un quadrato
def draw_square(dim):
    print('disegna un quadrato')

# procedura per "disegnare" un triangolo
def draw_triangle(dim):
    print('disegna un triangolo')

# programma che disegna una fila di case:
# invoca l'algoritmo definito sopra
draw_a_row_of_houses()
```

**1 processore esegue
1 istruzione per volta**

istruzione 1
istruzione 2
istruzione 3
...



istruzione 1
istruzione 2
istruzione 3
...

istruzione 1
istruzione 2
istruzione 3
...

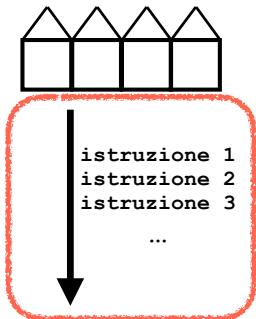
istruzione 1
istruzione 2
istruzione 3
...

**come è possibile
eseguire più istruzioni
in parallelo?**



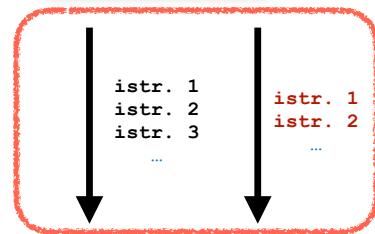
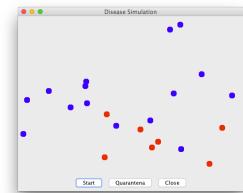
istruzione 1
istruzione 2
istruzione 1
istruzione 1
istruzione 2
istruzione 3
...

interleaving



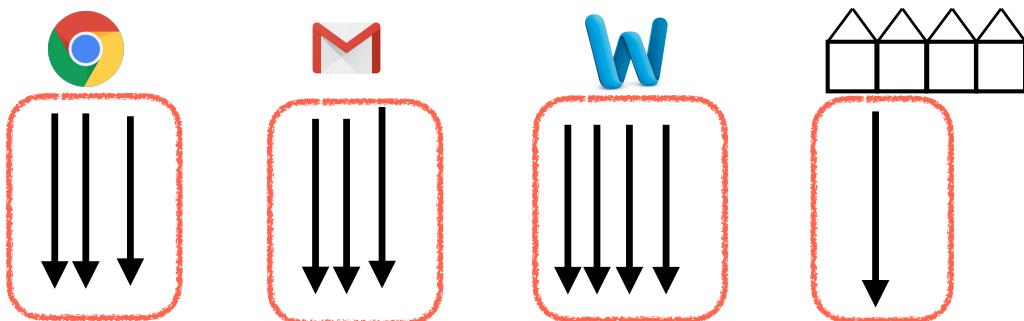
Programma Sequenziale:

1 flusso di controllo



Programma Concorrente:

più flussi di controllo
vengono eseguiti "in parallelo"
(tramite *interleaving*)



istruzione 1
istruzione 2
istruzione 1
istruzione 1
istruzione 2
istruzione 3



istruzione 1
istruzione 2
istruzione 1
istruzione 1
istruzione 2
istruzione 3

Normalmente ci sono **tanti programmi attivi da eseguire** in parallelo, ciascuno con 1 o più flussi di controllo concorrenti.

È il **sistema operativo** che si occupa di fare lo *scheduling*, i.e. scegliere l'interleaving opportuno perché tutti i programmi attivi avanzino nell'esecuzione.

Morale

“Un programma **esegue una sequenza di istruzioni**”...è vero ma più in generale:

- un programma può avviare molti flussi di controllo paralleli, che eseguono ciascuno una sequenza di istruzioni
- dato il codice di un programma, **individuare la sequenza di istruzioni** che portano da input ad output è **molto difficile**
 - ci sono *tool* di *debugging/monitoring/profiling* che aiutano, ma
 - pensare che avendo il codice sorgente, si capisca cosa fa il programma è un'illusione



non c'è il linguaggio migliore, ma c'è il linguaggio **più adatto** ad una data situazione

come **valutare** un linguaggio di programmazione?

come valutare un linguaggio di programmazione?

Diversi aspetti che dipendono dal **contesto d'uso** e dal tipo di **problema da risolvere**

- **efficienza** es. software del sistema operativo, software *embedded* che controlla ad es. la lavatrice o il servosterzo di auto.
- **sicurezza** es. software di centrale elettrica, database della sanità regionale
- **retro-compatibilità** es. software gestionale della banca
- **semplicità di scrittura** es. studenti, professionisti non informatici

TIOBE Index						
Nov 2021	Nov 2020	Change	Programming Language	Ratings	Change	
1	2	▲	Python	11.77%	-0.35%	
2	1	▼	C	10.72%	-5.49%	
3	3		Java	10.72%	-0.96%	
4	4		C++	8.28%	+0.69%	
5	5		C#	6.06%	+1.39%	
6	6		Visual Basic	5.72%	+1.72%	
7	7		JavaScript	2.66%	+0.63%	
8	16	▲	ASM	2.52%	+1.35%	
9	10	▲	SQL	2.11%	+0.58%	
10	8	▼	PHP	1.81%	+0.02%	
11	21	▲	Classic Visual Basic	1.56%	+0.83%	
12	11	▼	Groovy	1.51%	-0.00%	
13	15	▲	Ruby	1.43%	+0.22%	
14	14		Swift			
15	9	▼	R			
16	12	▼	Perl			
17	18	▲	Delphi/C++Builder			
18	13	▼	Go			
19	34	▲	Fortran	1.19%	+0.79%	
20	17	▼	MATLAB	1.17%	+0.07%	
linguaggio "mantenuto" adatto al calcolo scientifico, ottimizzato per massive parallel computing						
21			(Visual) FoxPro			
22			SAS			
23			Prolog			
24			Scratch			
25			COBOL			
26			Lua			
27			PL/SQL			
28			Objective-C			
29			Rust			
30			Lisp			
31			Dart			
32			Ada			
33			Kotlin			
34			D			
35			Scala			
36			Julia			
37			ABAP			
38			PowerShell			
39			Clojure			
40			Haskell			
41			Ladder Logic			
			VBScript			
			VHDL			
			LabVIEW			
			Scheme			
			TypeScript			
47			Apex			
48			Transact-SQL			
49			Logo			
50			Erlang			

Cos'è un Linguaggio di Programmazione?

GOALS

- **comunicare** istruzioni a una macchina
- **esprimere algoritmi** ai programmatorei

C

```
#include <stdio.h>
int main()
{
printf("Hello World");
return 0;
}
```

C++

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Java

```
class HelloWorld
{
    public static void main(String[] a)
    {
        System.out.println("Hello world");
    }
}
```

PHP

```
! Hello World Program
program hello
implicit none
write (*, '(a)') "Hello, World!"
end program hello
```

X10

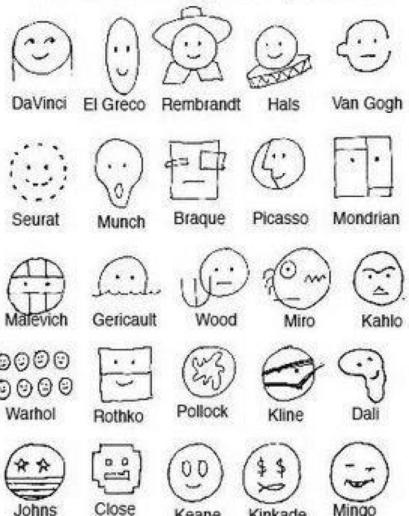
```
import x10.io.Console;
class HelloWorld {
    public static def main(Array[String](1)) {
        finish for (p in Place.places()) {
            sync at (p) {
                Console.OUT.println(
                    "Hello, World: place " + p.id);
            }
        }
    }
}
```

Scrivere in un linguaggio
significa anche pensare in quel linguaggio

esistono diversi **stili** di programmazione

*What is being said is shaped and influenced
by how is being said*

Art History, Simplified



Paradigmi di Programmazione

- Un **paradigma di programmazione** caratterizza la struttura dei programmi e il modo in cui si risolvono i problemi
 - *imperativo*
 - *funzionale*
 - *object-oriented*
 - *dichiarativo*
 - *logico*
 - *event-driven*
 - *concorrente*
 - ...
 - *multi-paradigm languages*

Java From **How to do** to **What to do**

Find the age of the oldest male in the list

```
Person[] people = ... //initialization
int maxAge=-1;
for(int i=0; i < people.length; i++)
    if( people[i].getGender()==MALE && people[i].getAge() > maxAge )
        maxAge = people[i].getAge();

System.out.println("The oldest male is "+ maxAge +" years old");
```

Imperative

```
List<Person> people = ... //initialization
final int maxAge= people.stream()
    .filter(p -> p.getGender()==MALE)
    .mapToInt(p -> p.getAge())
    .max();

System.out.println("The oldest male is "+ maxAge +" years old");
```

Functional

imparare Python

Forum nella pagina Moodle
per aiutarsi e confrontarsi

- **Ambiente di sviluppo:**

- <https://www.programiz.com/python-programming/online-compiler/>
- <https://www.pythontutorial.net/>
- they provide a cloud coding environment to write code stored and run on remote computer systems

- **Libro di riferimento:**

- **Pensare da informatico - Versione Python 3**
<https://www.python.it/doc/Howtothink/Howtothink-html-it/index.htm>
- Think Python. How to Think Like a Computer Scientist. Allen Downey.
<http://greenteapress.com/thinkpython2/thinkpython2.pdf>
The interactive version: <https://runestone.academy/runestone/static/thinkcspy/index.html>

Il primo programma Python 3

Scrivere un programma che stampa a video la stringa `Hello, World!`

1. aprire un **file** di nome `first.py`

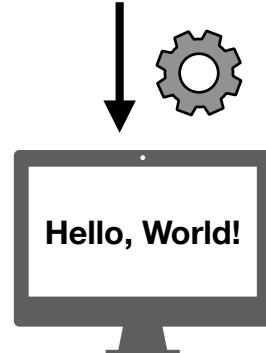
2. nel file va scritto (e **salvato**) il **codice sorgente**, cioè le istruzioni per stampare la stringa, i.e.
`print("Hello, World!")`

3. **eseguire** il programma con il tasto **Run** oppure dalla *console (bash)* con il comando

`python3 first.py`

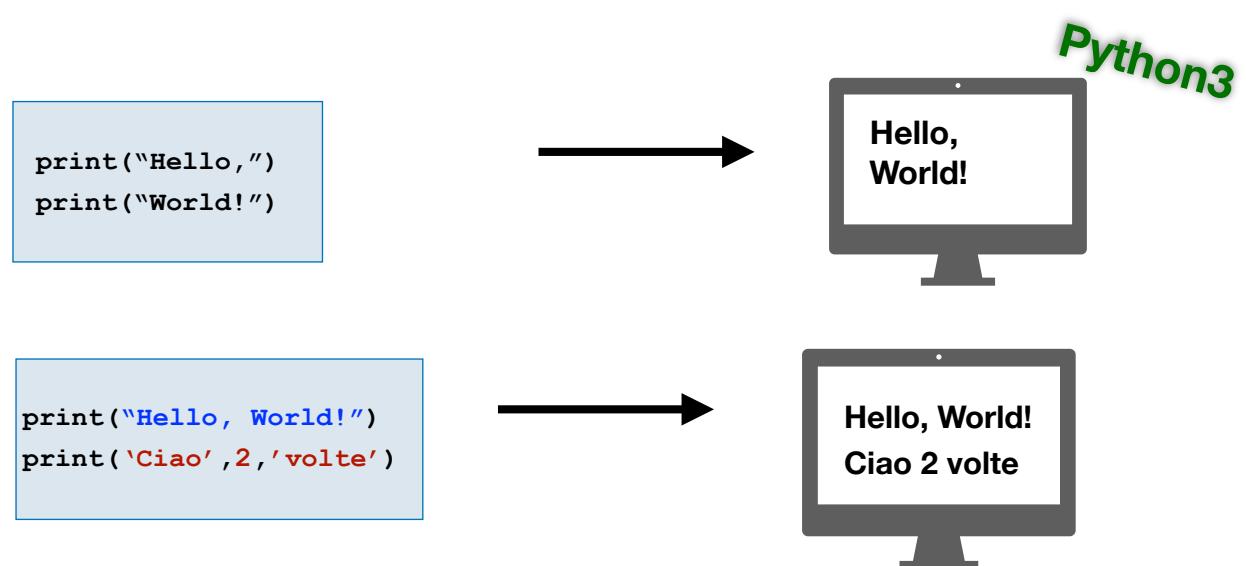
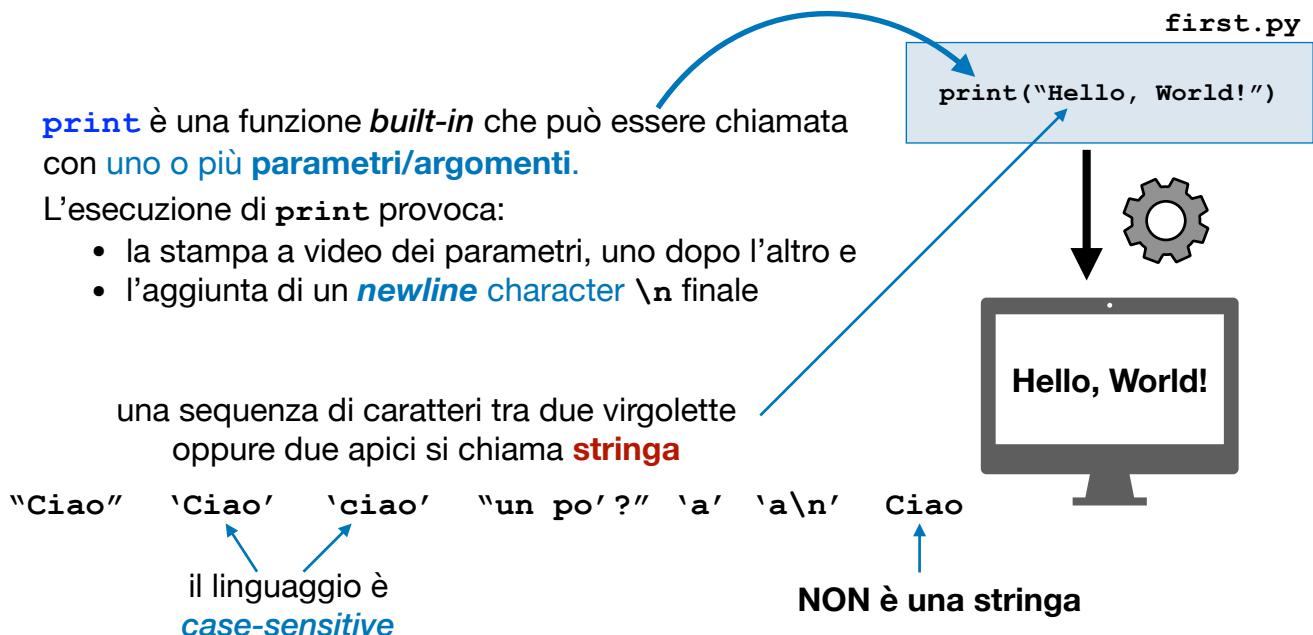
che lancia l'**interprete python**, cioè il programma che traduce il codice sorgente in istruzioni macchina e le esegue una ad una

`first.py`
`print("Hello, World!")`



Il primo programma

Scrivere un programma che stampa a video la stringa `Hello, World!`



```
print ("I'm a computer ... \n") # This is a comment. Note that we added a
                                # further newline at the end of the string

print ("Hello!", " ", end='') # Note end='' that removes the
                            # default suffix character '\n'

print ('How are you?')
```

I'm a computer ...
Hello! How are you?

i **commenti** (dal simbolo `#` fino a fine linea) sono ignorati dall'interprete python