

# Linguaggi di programmazione

Informatica di base – a.a. 2018/2019

Silvio Peroni

[0000-0003-0530-4305](https://orcid.org/0000-0003-0530-4305)

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia  
[silvio.peroni@unibo.it](mailto:silvio.peroni@unibo.it) – [@essepuntato](https://www.essepuntato.it) – <https://www.unibo.it/sitoweb/silvio.peroni/>



Quest'opera è distribuita con [Licenza Creative Commons Attribuzione 4.0 Internazionale](https://creativecommons.org/licenses/by/4.0/)



DIPARTIMENTO DI FILOLOGIA CLASSICA E ITALIANISTICA

# Linguaggio “naturale”

È linguaggio comune, come l'italiano

Può essere scritto o orale

Si è evoluto in maniera naturale all'interno di una comunità

Vantaggio: molto espressivi

Svantaggi: alcune frasi sono intrinsecamente ambigue se non accompagnate da un opportuno contesto

# Interpretazione ambigua

*La vecchia porta la sbarra*



*La vecchia porta la sbarra*

*La vecchia porta la sbarra*



Il significato di una frase può essere disambiguato mediante l'uso di convenzioni sociali o analizzando il contesto in cui l'azione avviene, in modo da restringere il possibile significato di un pezzo di informazione

# Formalizzazione di un linguaggio

I linguaggi naturali non sono formali per definizione

Tuttavia molti studi in linguistica cercano di fornirne una formalizzazione mediante l'uso di **strumenti matematici**

Chomsky è uno dei padri della linguistica moderna – le cui ipotesi di ricerca principali sul linguaggio umano sono:

1. la sua struttura sintattica di base è **rappresentabile mediante una teoria matematica**
2. tale struttura è determinata **biologicamente in tutti gli umani**



# Grammatica formale

Un insieme di regole di produzione di forma  $\text{premessa} ::= \text{espressione}$ , dove la  $\text{premessa}$  e l' $\text{espressione}$  possono contenere uno o più:

- simboli **terminali** (specificati tra virgolette), che identificano tutti i simboli elementari del linguaggio in considerazione (come nomi, verbi, etc.)
- simboli **non terminali** (specificati tra parentesi angolari), che identificano tutti i simboli di una grammatica formale che possono essere sostituiti da una combinazione di simboli terminali e non terminali

Esempio:

$\langle \text{frase} \rangle ::= \text{"Io"} \langle \text{verbo} \rangle$

$\langle \text{verbo} \rangle ::= \text{"scrivo"} \mid \text{"leggo"}$

Il simbolo non terminale  $\langle \text{frase} \rangle$  è il simbolo iniziale.

Frasi che possiamo produrre da questa grammatica:

- Io scrivo
- Io leggo

# Gerarchia di Chomsky

Chomsky ha proposto una gerarchia per descrivere formalmente le relazioni che possono esistere tra diverse grammatiche in termini delle loro possibili strutture sintattiche che sono in grado di generare

Queste tipologie sono caratterizzate dal tipo di simboli che possono essere usati nella premessa e nell'espressione delle regole di produzione (lettere greche usate per indicare una combinazione di simboli terminali e/o non terminali):

- grammatiche **regolari** –  $\langle nt \rangle ::= "t" \mid "t" \langle nt \rangle$
- grammatiche **libere dal contesto** –  $\langle nt \rangle ::= \gamma$
- grammatiche **dipendenti dal contesto** –  $\alpha \langle nt \rangle \beta ::= \alpha \gamma \beta$
- grammatiche **ricorsivamente enumerabili** –  $\alpha ::= \beta$

# Cosa sono i linguaggi di programmazione

Un linguaggio di programmazione è un **linguaggio formale** che obbliga l'uso di specifiche regole sintattiche sviluppate in modo tale da evitare possibili istruzioni ambigue

Solitamente l'**espressività del linguaggio è ridotta** ma tutte le “frasi” componibili trasmettono un **solo possibile significato**

I linguaggi di programmazione sono solitamente basati su **grammatiche libere dal contesto** e possono distinguersi per un basso o elevato livello di astrazione dal linguaggio propriamente in uso da un elaboratore elettronico per eseguire le operazioni

# Linguaggio macchina

Un insieme di istruzioni che possono essere eseguite direttamente dalla CPU (*central processing unit*, o processore) di un computer elettronico

Basato sul **codice binario** – una sequenza di 0 e 1 –  
rivisitato da Leibniz alla fine del diciassettesimo secolo

11010011101110110011011011111110010110110111000011110100110100111000111100001

11010011101110110011011011111110010110110111000011110100110100111000111100001

i n f o r m a t i c a



Gottfried Leibniz



# Linguaggi di programmazione a basso livello

Forniscono **un livello di astrazione** sopra il linguaggio macchina

Permettono di scrivere programmi in modo che siano un pochino più intellegibili dagli umani

Il più famoso linguaggio di questo tipo è l'Assembly – anche se introduce simboli più comprensibili, di solito una linea di codice in Assembly rappresenta una specifica istruzione in linguaggio macchina

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

@@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

@@:
    push ebx
    mov ebx, 1
    mov ecx, 1
```

...

# Linguaggi di programmazione ad alto livello

Caratterizzati da un **forte livello di astrazione** dal linguaggio macchina – esempio: Python

Possono usare parole proprie del linguaggio naturale per definire costrutti specifici, così da essere di più facile comprensione per un umano

Più astrazione da un linguaggio di programmazione a basso livello è fornita, più comprensibile è il linguaggio

```
def fib(n):  
    if n <= 0:  
        return 0  
    elif n <= 2:  
        return 1  
    else:  
        a = 1  
        b = 1  
        while True:  
            c = a + b  
            if n <= 3:  
                return c  
            a = b  
            b = c  
            n = n - 1
```

# I primi linguaggi

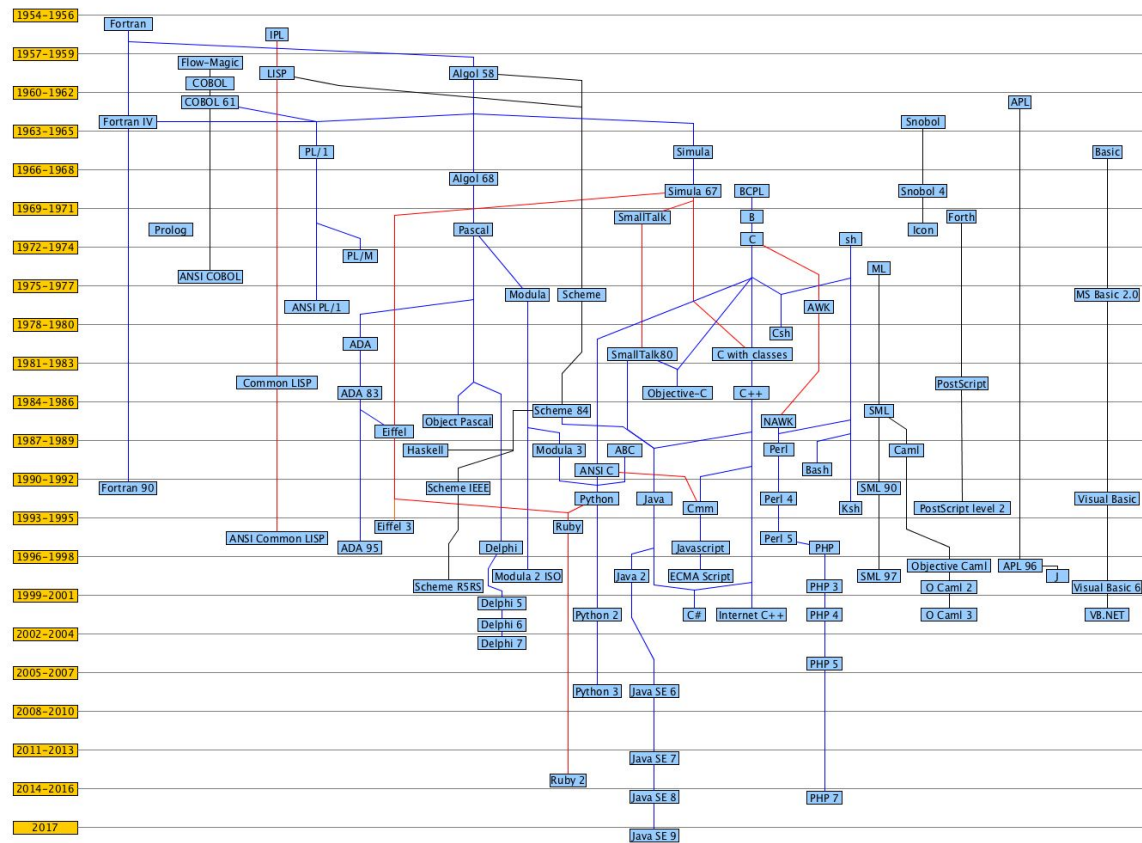
Grace Brewster Murray Hopper è tra i più grandi pionieri dei linguaggi di programmazione

Il primo programmatore dell'Harvard Mark I, un computer elettromeccanico general-purpose usato durante la seconda guerra mondiale

Hopper era fermamente convinta della necessità di avere linguaggi di programmazione che fossero **indipendenti dalle macchine su cui erano utilizzati**, che l'ha portata allo sviluppo del COBOL (linguaggio ad alto livello)



# Evoluzione dei linguaggi di programmazione



# Linguaggi visuali

In modo da facilitare l'avvicinamento all'uso dei linguaggi di programmazione tradizionali, negli ultimi anni sono stati sviluppati diversi linguaggi di programmazione visuali che permettono lo sviluppo di piccoli programmi per risolvere problemi computazionali specifici

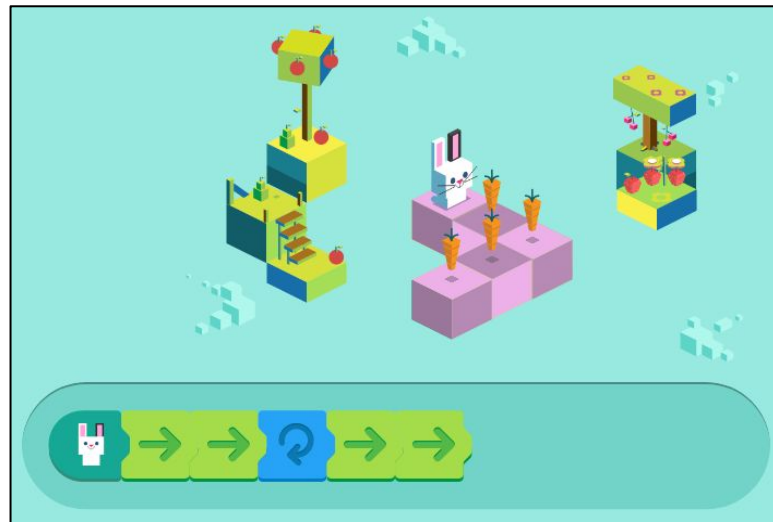
Alcuni sono proposti sotto forma di gioco, e permettono l'introduzione dei costrutti principali propri dei linguaggi di programmazione mediante l'utilizzo di oggetti grafici

# Un esempio

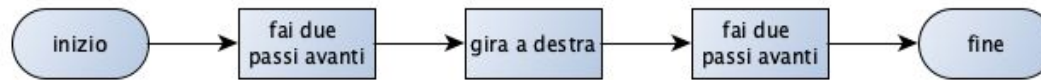
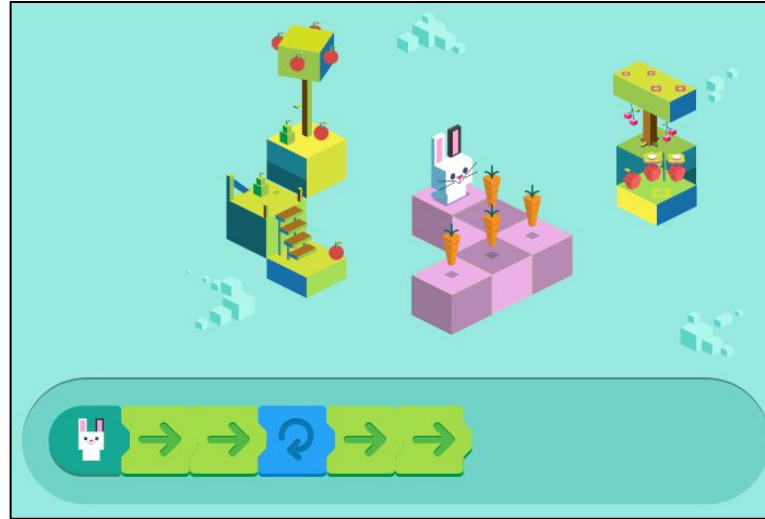
Nel 2017, Google ha messo a disposizione un doodle in cui si deve istruire un coniglio in modo che riesca a mangiare tutte le carote posizionate su un determinato percorso

Le azioni che si possono far compiere al coniglio riguardano attività di movimento, ad esempio “vai avanti” o “gira a destra”, e devono essere disposte in una specifica sequenza

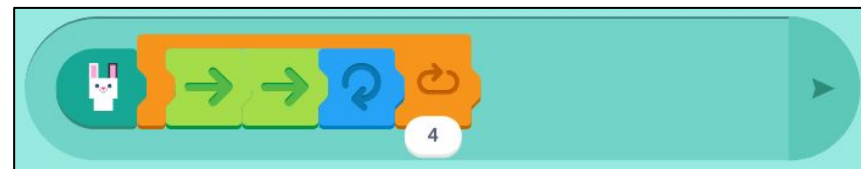
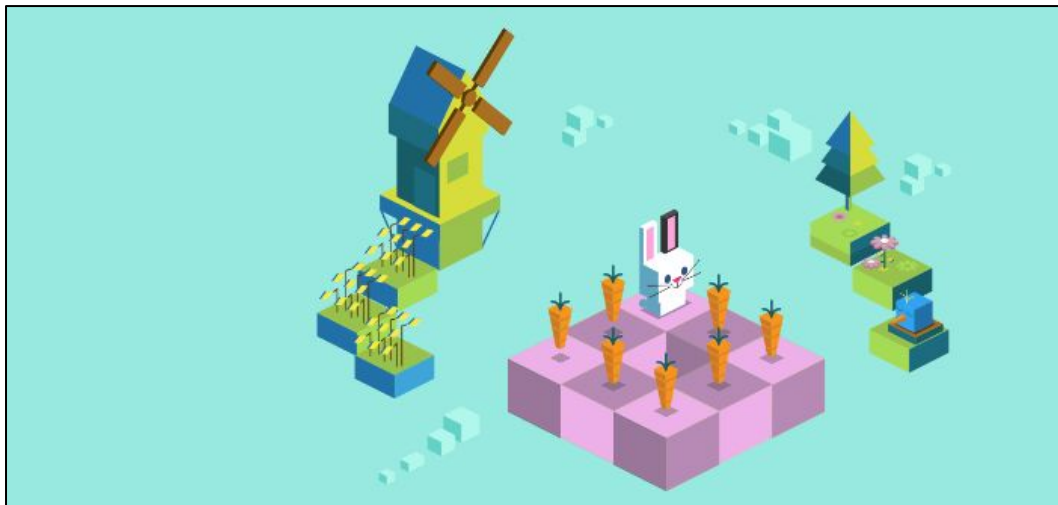
Problema computazionale: mangiare tutte le carote presenti sul percorso



# L'esempio descritto con un diagramma di flusso

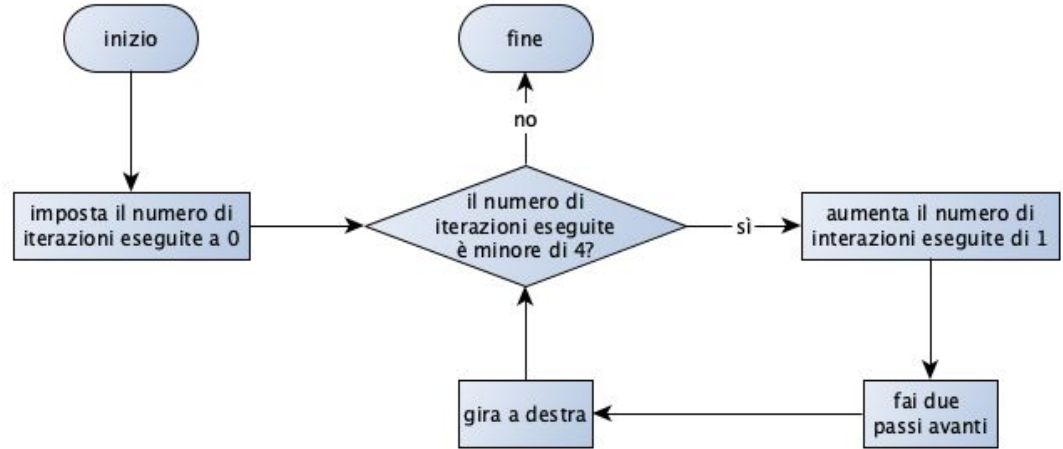
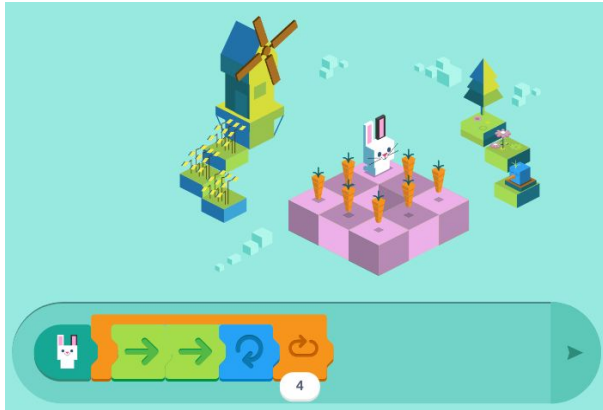


# Un esempio dal vivo





# Iterazioni con i diagrammi di flusso

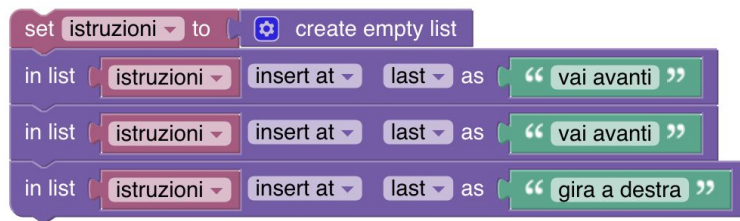


# Un altro linguaggio di programmazione visuale

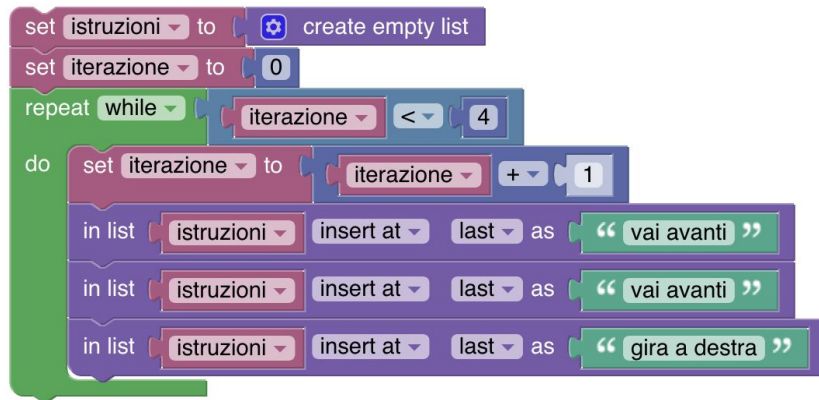
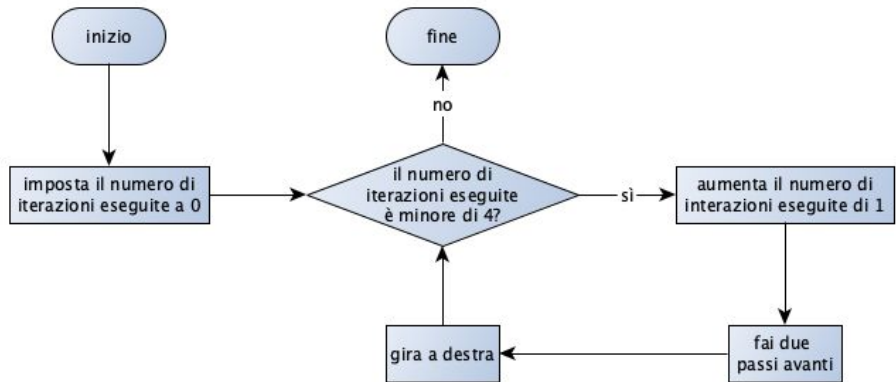
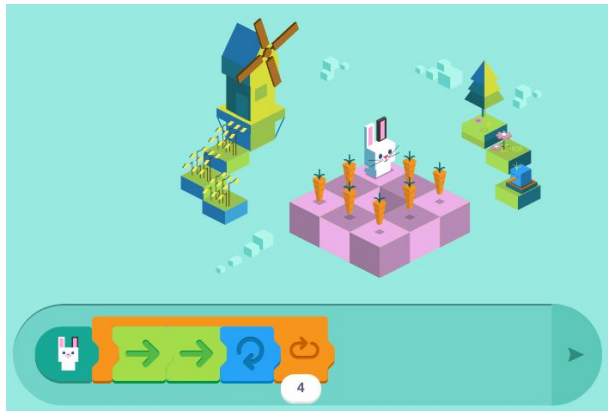
Il linguaggio di programmazione visuale proposto finora è stato sviluppato specificamente per risolvere un particolare problema computazionale

Esistono anche altri linguaggi di programmazione visuale che sono più *general-purpose*, ad esempio Blockly di Google

Questo linguaggio mette a disposizione diversi costrutti propri dei linguaggi di programmazione tradizionali, ma li propone sotto forma di blocchetti che si possono incastrare uno sull'altro o uno dentro l'altro per definire le sequenze di operazioni di un algoritmo



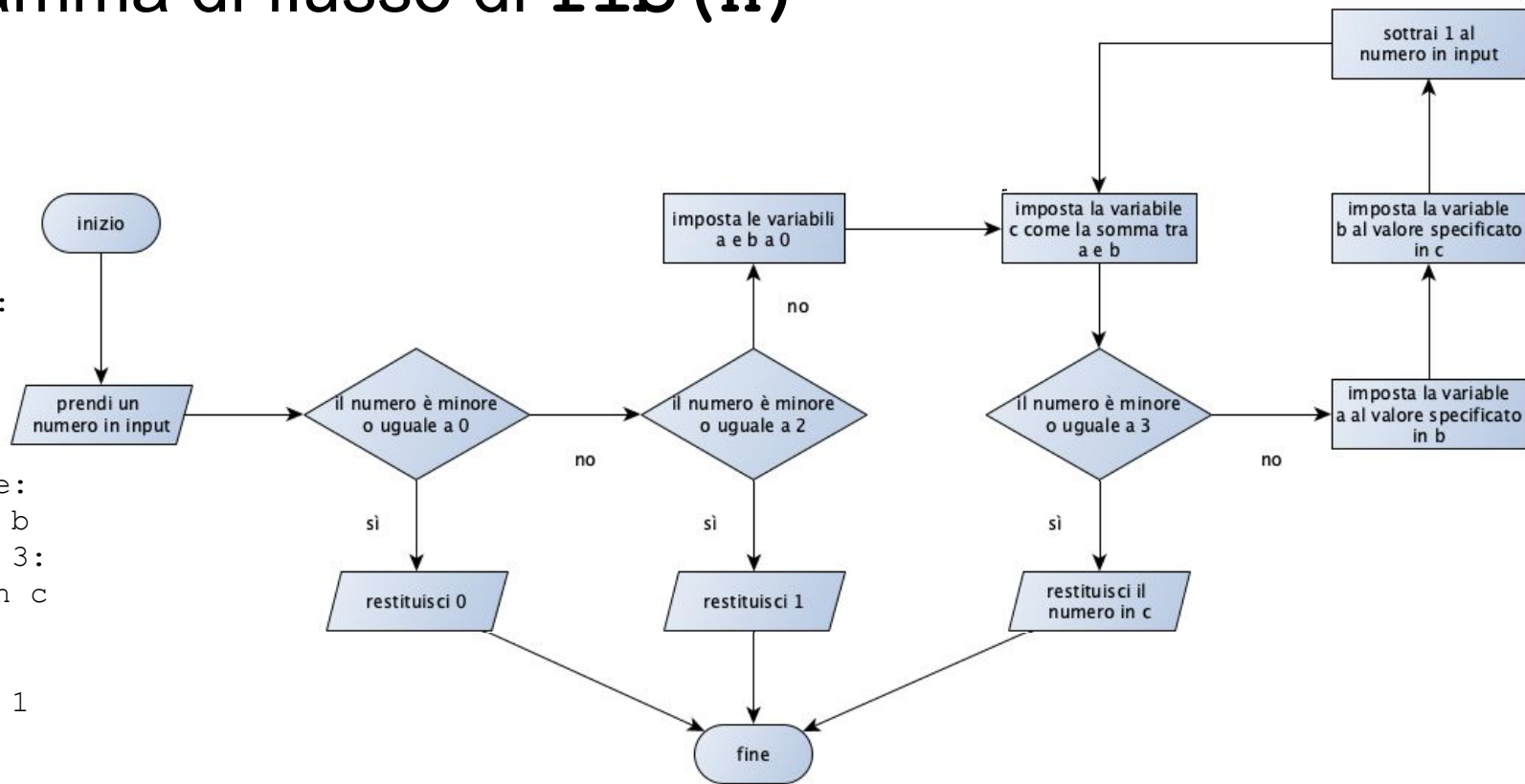
# Iterazioni con Blockly e Python



```
istruzioni = list()
iterazione = 0
while iterazione < 4:
    iterazione = iterazione + 1
    istruzioni.append("avanti")
    istruzioni.append("avanti")
    istruzioni.append("gira a destra")
```

# Diagramma di flusso di `fib(n)`

```
def fib(n):  
    if n <= 0:  
        return 0  
    elif n <= 2:  
        return 1  
    else:  
        a = 1  
        b = 1  
        while True:  
            c = a + b  
            if n <= 3:  
                return c  
            a = b  
            b = c  
            n = n - 1
```



# Fine

## Linguaggi di programmazione

Informatica di base – a.a. 2018/2019

Silvio Peroni

[0000-0003-0530-4305](https://www.unibo.it/sitoweb/silvio.peroni/0000-0003-0530-4305)

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

[silvio.peroni@unibo.it](mailto:silvio.peroni@unibo.it) – [@essepuntato](https://www.unibo.it/sitoweb/silvio.peroni) – <https://www.unibo.it/sitoweb/silvio.peroni/>



Quest'opera è distribuita con [Licenza Creative Commons Attribuzione 4.0 Internazionale](https://creativecommons.org/licenses/by/4.0/)

