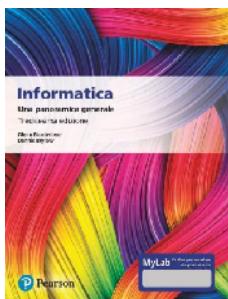


Let's go UP

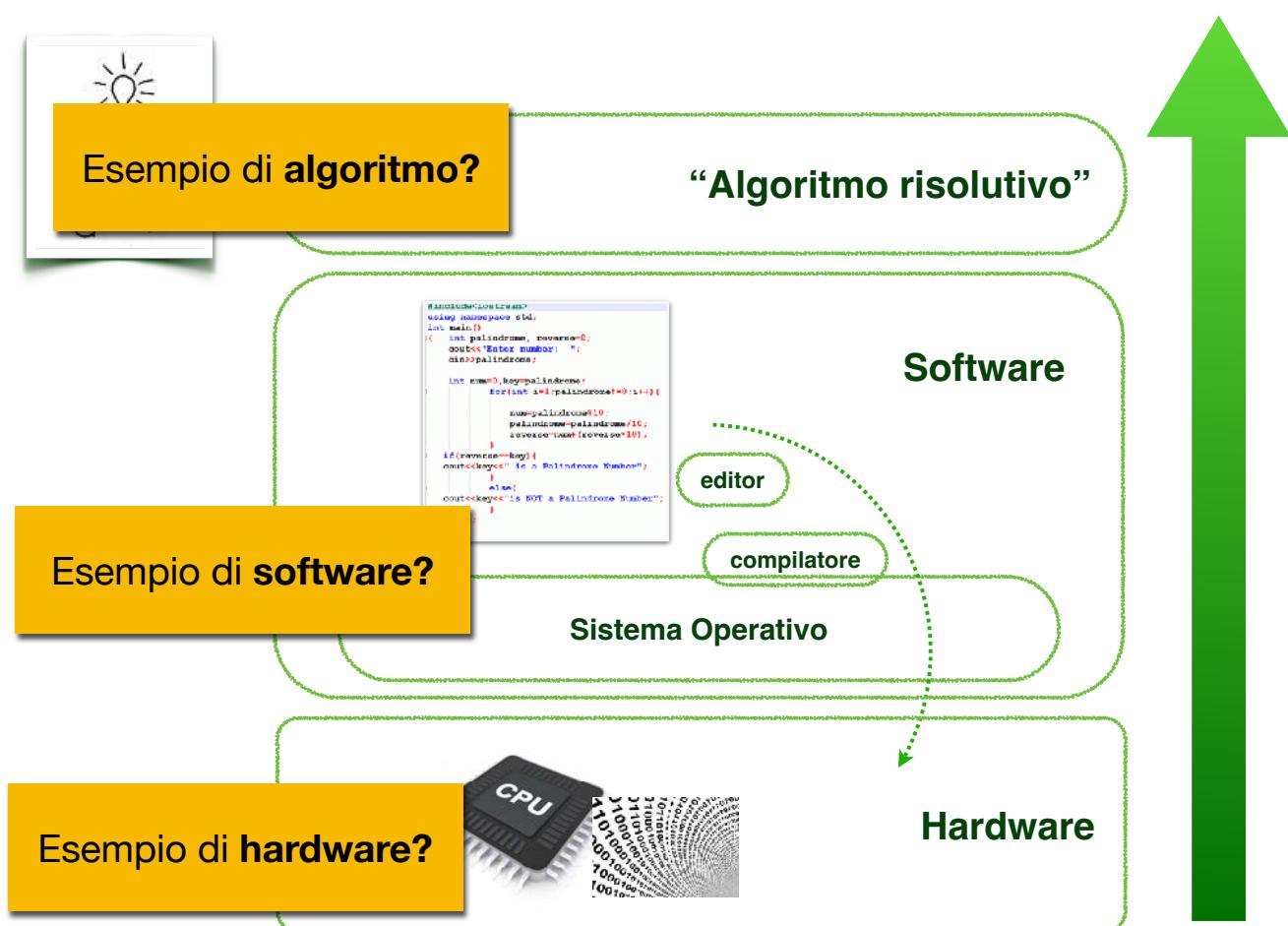


*Computer Science is no more about computers
than astronomy is about telescopes.*

-- Edsger Dijkstra



S. Crafa A.A. 23/24





scopo della II parte del corso



Diventare **consapevoli** del grado di **complessità** e della **non neutralità** della **produzione del software**

1. conoscere e comprendere i **diversi livelli** di astrazione coinvolti nello **sviluppo** e nell'**esecuzione** dei programmi
2. capire **cosa significa** e **che limiti ha** la valutazione della **qualità** del software
3. riflettere e capire **cosa significa** e quanto sia **difficile comunicare** (le caratteristiche) del software



Di cosa parliamo quando parliamo di ‘programmi’

Violetta Lonati, Claudio Mirolo, Mattia Monga

Mondo Digitale Novembre 2022

"La pratica della programmazione come *strumento di consapevolezza* e *cittadinanza attiva*"

*La confusione assai diffusa fra **uso** delle applicazioni informatiche e l'**impresa concettuale** di*

- **immaginarne l’utilità,**
- **progettarle,**
- **realizzarle,**
- **convalidarne il funzionamento** e
- **comprenderne l’impatto**

S. Crafa A.A. 23/24

Di cosa parliamo quando parliamo di ‘programmi’

Violetta Lonati, Claudio Mirolo, Mattia Monga

Mondo Digitale Novembre 2022

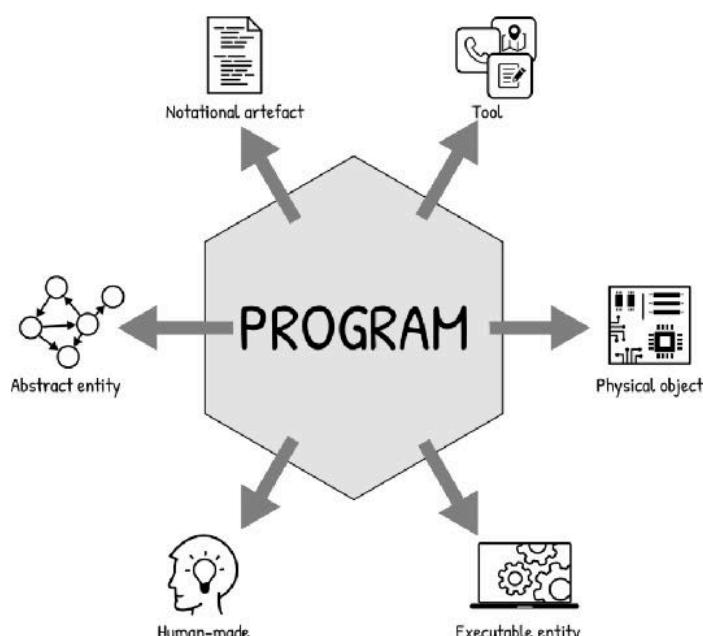
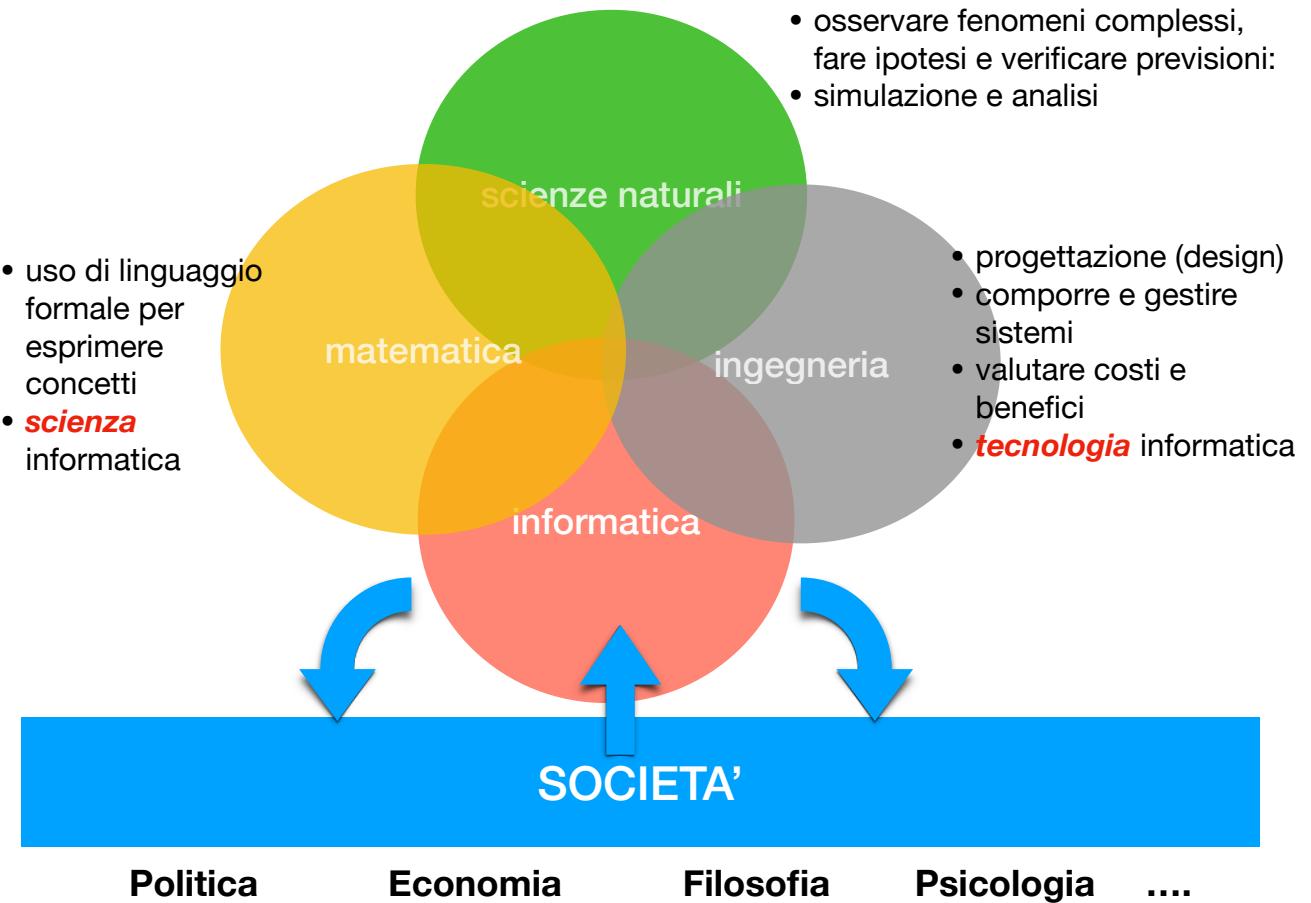


Figura 1

I sei aspetti chiave della natura di un programma (figura tratta dal documento originale [24])



<https://www.cohubicol.com/>

The screenshot shows a modern website for 'COHUBICOL'. The header features a navigation bar with icons for back, forward, search, and user account. The URL 'https://www.cohubicol.com' is displayed in the address bar. The main content area has a dark background with a geometric, wavy pattern. On the left, the word 'COHUBICOL' is written in large, bold, white capital letters. Below it is a subtitle in a smaller, white sans-serif font: 'Counting as a Human Being in the Era of Computational Law'. Underneath that, a tagline reads 'Say cubicle • Think Wittgenstein's cube'. A 'Learn more' button is visible at the bottom left. The top navigation bar includes links for 'About the project', 'News', 'Subscribe', 'Research blog', 'Team', and a magnifying glass icon for search.

gestione delle informazioni

Informazioni



Dati

devono essere in un formato
"machine-consumable"

S. Crafa A.A. 23/24

Informazioni



Dati

devono essere in un formato
"machine-consumable"

CV di Mario Rossi, studente di Diritto e
Tecnologia, atleta, ...



in software Uniweb

```
nome = "Mario Rossi",  
nascita = 11/2/2001,  
matricola = 2194804,  
corso = "Diritto e Tecnologia",  
programma_atleta = Sì,  
media = 27,4
```

→ 0110010100...

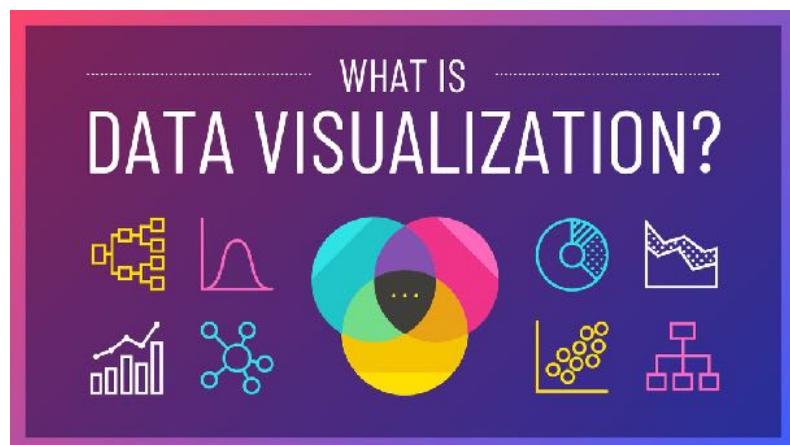
in software CUS

```
nome = "Mario Rossi",  
età = 22,  
società = "Fiamme Oro",  
specialità = "maratona",  
altezza = 178,  
peso = 72
```

→ 110110100...

S. Crafa A.A. 23/24

gestione delle informazioni e visualizzazione



S. Crafa A.A. 23/24



Cruscotto (*dashboard*) di visualizzazione di dati

diagramma di visualizzazione grafica di dati

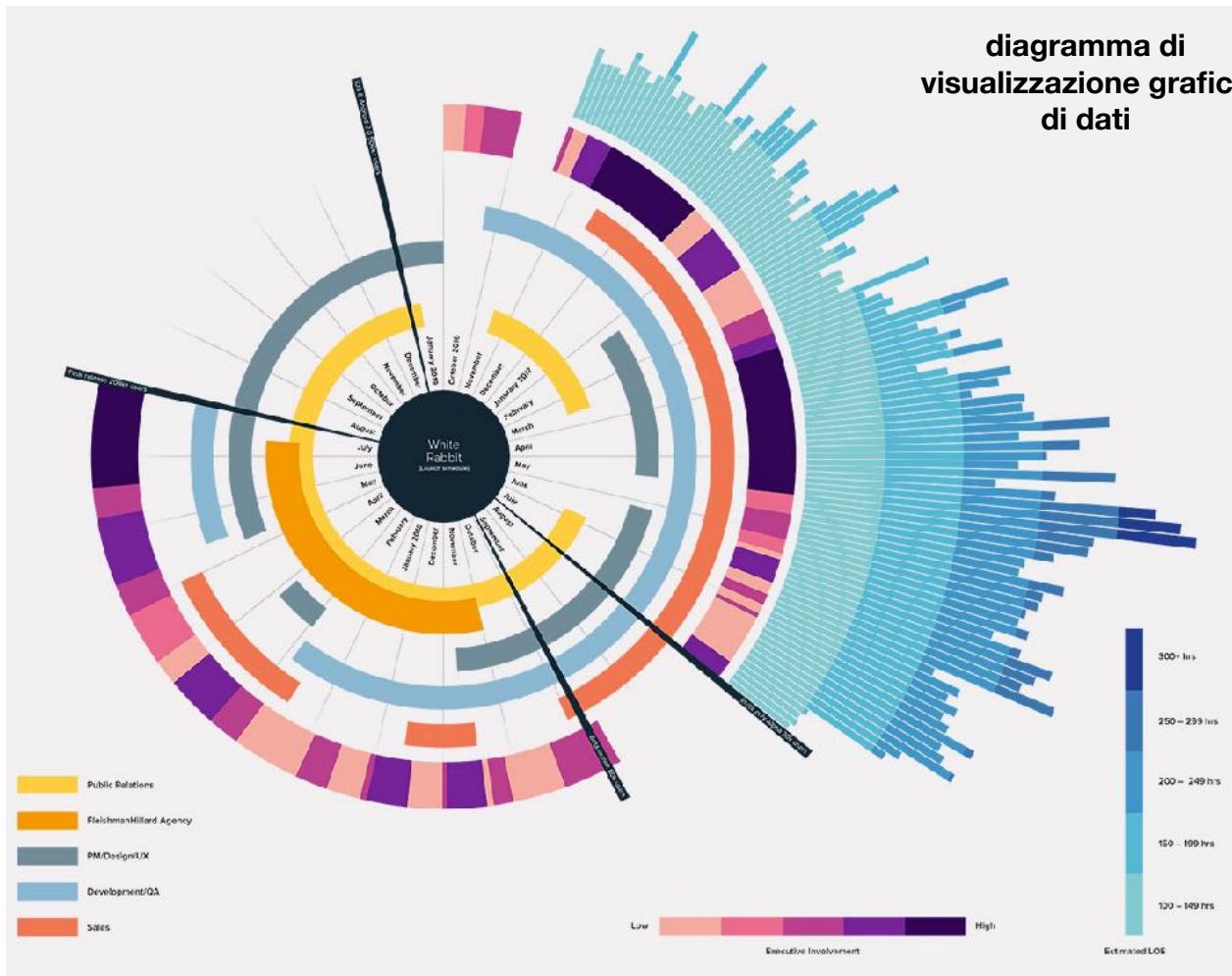
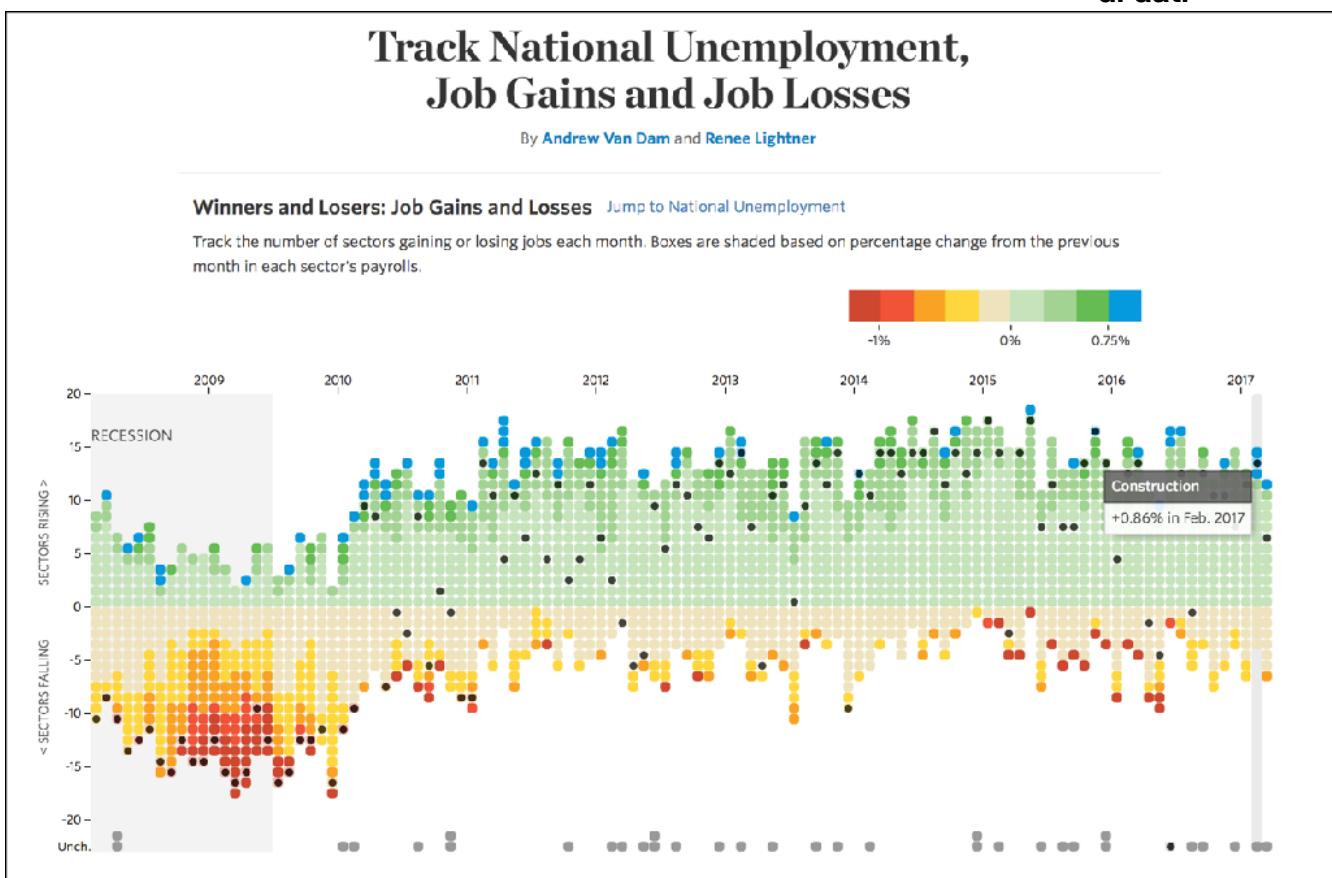


diagramma di visualizzazione grafica di dati





Infografica che visualizza un processo (una sequenza di passi)

- Anche un testo:**
- chiaro, sintetico
 - **ben formattato**
 - con icone grafiche

Austria = Aspetta la vita

Vacanze estive in Austria

Mese valido dal 10 giugno

REGOLE DI INGRESSO (TEST ALL'ENTRATA)

L'ingresso in Austria è consentito per mezzo di un test Covid negativo, una vaccinazione o una prova di guarigione.

- TEST: Test PCR valido per 72 h, test antigenico valido per 48 h. Gli autotest non sono validi ai fini dell'ingresso.
- VACCINAZIONE: la validità inizia 22 giorni dopo la prima dose per un massimo di 3 mesi, la dose successiva estende la validità per altri 6 mesi.
- GUARIGIONE: validità fino a 6 mesi dopo la malattia.

GASTRONOMIA

- Obbligo di registrazione dei propri dati di contatto (dell'ospite)
 - Test all'entrata (test Covid negativo, vaccinazione valida o prova di guarigione)
 - Obbligo mascherina FFP2 (solo all'esterno) tranne quando si è seduti al tavolo
 - Massimo 6 adulti (+ bambini) al coperto e massimo 16 adulti (+ bambini) all'esterno
 - L'orario di chiusura è alle ore 24:00

STRUTTURE RISTETIVE

- Obbligo di registrazione dei propri dati di contatto
 - Test all'entrata (test Covid negativo, vaccinazione valida o prova di guarigione)
 - Obbligo mascherina FFP2 nelle aree comuni
 - All'attività di ristorazione presso le strutture ristetiche si applicano le stesse regole valide per la gastronomia in generale
 - È richiesto un test all'entrata per l'utilizzo del servizio in hotel

MEZZI PUBBLICI E FUNIVIE

- Obbligo mascherina FFP2 in bus, treno e funivia così come in stazione (inclusa la banchina)
- All'interno di cabine della funivia: limitazione della capacità al 75%

TEMPO LIBERO E SPORT

- Obbligo di registrazione dei propri dati di contatto all'interno (ad esempio spa, centri termali, palestre,...)
- Test all'entrata (test Covid negativo, vaccinazione valida o prova di guarigione)
- Obbligo mascherina FFP2 all'interno anche quando si è seduti
- Durante l'attività sportiva, non è obbligatorio l'uso della mascherina
- L'orario di chiusura è alle ore 24:00

CULTURA ED EVENTI

- Obbligo di registrazione dei propri dati di contatto (dell'ospite)
- Test all'entrata (test Covid negativo, vaccinazione valida o prova di guarigione)
- Nessun obbligo di mascherina all'esterno
- Nei musei: nessun obbligo di registrazione dei propri dati di contatto (dell'ospite) e nessun obbligo di presentazione di un test all'entrata
- Eventi con più di 500 persone all'aperto fino ad un massimo di 3.000 persone e al coperto fino a 1.500 persone
- Agli eventi senza posti a sedere assegnati possono partecipare fino ad un massimo di 50 persone. In questo caso non sono possibili servizi di ristorazione
- L'orario di chiusura è alle ore 24:00

Aggiornato al: 9 giugno 2021

TRAVELLING TO ITALY

Italy applies health-related restriction measures to incoming travellers, which may vary depending on their country of origin.

Before entering Italy, from any country of origin and for any travel reason, the digital European [Passenger Locator Form \(PLF\)](#) must be completed.

Travellers from EU countries, the Schengen area, Great Britain and Israel can enter Italy without any quarantine obligation, provided that:

- they fill in the [Passenger Locator Form](#) before entering Italy
- on arrival, present a negative result for a molecular or antigenic swab taken no more than 48 hours prior to entry into Italy
- notify arrival to the Prevention Department of the Local Health Authority ([full-free numbers and regional information](#))

Entry and transit in Italy are forbidden for persons:

- who, in the previous fourteen days, have stayed or transited in Brazil ([see exceptions](#))
- who arrive from, or have stayed in the previous fourteen days in India, Bangladesh and Sri Lanka ([see exceptions](#))

Travellers from all other countries may be required to:

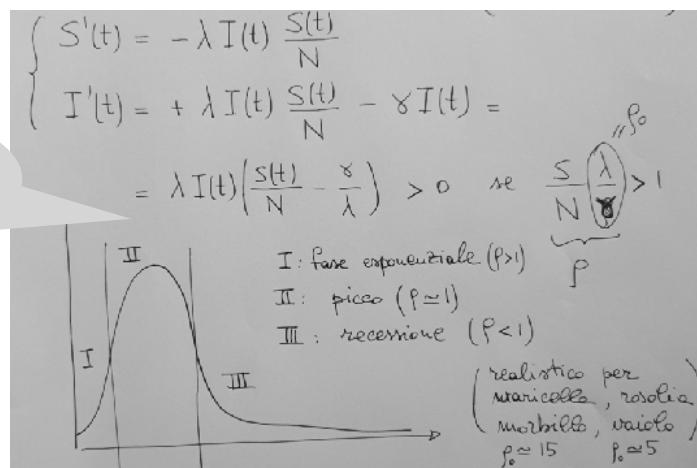
- fill in the [Passenger Locator Form](#) before entering Italy
- present a negative result for a molecular or antigenic swab taken no more than 72 hours prior to entry into Italy
- notify arrival to the Prevention Department of the Local Health Authority ([full-free numbers and regional information](#))
- reach the final destination in Italy only by private means of transport
- undergo mandatory isolation and health surveillance for 10 days
- at the end of the 10 day isolation, take an additional molecular or antigenic swab

Travellers arriving from some non-EU countries can enter Italy without undergoing mandatory isolation and health surveillance provided they use COVID-tested flights (refer to the specific

informatica e analisi di fenomeni complessi

modello epidemiologico per studiare andamento di una pandemia

modello matematico:
equazioni che descrivono il
comportamento del sistema



<https://www.math.unipd.it/news/la-matematica-del-contagio/>

- una **simulazione** permette di modellare una situazione ed **esplorare le sue possibili evoluzioni**. È uno strumento molto potente per migliorare la comprensione di un fenomeno
- La **computazione** è unica nella sua **capacità di rendere concreto il modello astratto definito dalla simulazione**

informatica e analisi di fenomeni complessi

- capacità di analisi, simulazione e previsione tramite **modelli molto sofisticati**
- ma per poter applicare le tecniche informatiche servono **assunzioni** e **metodologie**, che **non sono neutrali**:
 - producono **solo certi tipi di risultati** e
 - i risultati hanno solo un determinato e **preciso senso**, anche se tendiamo ad ampliare il senso e il valore di questi risultati

es. **modello di produttività di un dipendente**:

un algoritmo che, dato in input il nome di un dipendente, dà in output un numero in [0,10]

Mario ---> 8 Luisa ---> 9 Carlo ---> 4

....Luisa è la dipendente "migliore" !?

...posso fare a meno di Carlo!?

- è **comodo** produrre come risultato un **numero**. Ma ad es. questo significa che **stiamo producendo una gerarchia**

S. Crafa A.A. 23/24

informatica e analisi di fenomeni complessi

- capacità di analisi, simulazione e previsione tramite **modelli molto sofisticati**
- ma per poter applicare le tecniche informatiche servono **assunzioni** e **metodologie**, che **non sono neutrali**:

- producono **solo certi tipi di risultati** e
- i risultati hanno solo un determinato e **preciso senso**, anche se tendiamo ad ampliare il senso e il valore di questi risultati

Non Neutrale!

INPUT

...COVID e "la verità dei dati":

- è stato misurato tutto il misurabile, **tutto tradotto in un numero**,
 - ma... n. malati, n.posti in terapia, Rt settimanale era difficile sapere se era numero corretto (caso Lombardia rossa),
 - e il livello disagio psicologico? e il danno didattico-educativo?
- **i dati non sono il fenomeno "datificato"**, e nello scarto si perde qualcosa, a volte essenziale (The MacNamara fallacy)

S. Crafa A.A. 23/24

The MacNamara Fallacy



1. The first step is **to measure whatever can be easily measured.**
This is OK as far as it goes.
2. The second step is **to disregard that which can't be easily measured** or to give it an arbitrary quantitative value.
This is artificial and misleading.
3. The third step is **to presume that what can't be measured easily really isn't important.**
This is blindness.
4. The fourth step is **to say that what can't be easily measured really doesn't exist.**
This is suicide

il Pensiero Computazionale

come pensa un informatico quando affronta un problema?

capacità di **formulare problemi**
in modo che

ammettano **soluzioni** rappresentabili
in un **formato eseguibile**

da un **uomo o da una macchina** o una
combinazione dei due
(*an information processing agent*)

Non Neutrale!

orienta la **formulazione** di un problema non verso la sua analisi o
comprensione, ma **verso una soluzione operativa, effettiva**



cioè scritta in un formato eseguibile in
modo **automatizzabile**

S. Crafa A.A. 23/24

il Pensiero Computazionale

come pensa un informatico quando affronta un problema?

capacità di **formulare problemi**
in modo che

ammettano **soluzioni**
in un **formato eseguibile**

La **programmazione** è un modo di
esercitare ed allenare questa capacità
di **formulare e risolvere i problemi**
(*come la scrittura è un modo per
esprimere i propri pensieri*)

una **serie di strumenti mentali** che ci consentono di

passare da un'idea alla sua realizzazione

- pensare in **maniera creativa** alle possibili soluzioni
- approccio **trial-and-error** (*esplorare e sperimentare*)
- iniziare **programmando** un prototipo

S. Crafa A.A. 23/24

soluzioni in formato eseguibile

- Un **programma** consiste di **una serie di istruzioni** che spiegano alla macchina come effettuare una computazione
 - es. un calcolo matematico, una elaborazione di un testo, un'operazione grafica, la riproduzione di un filmato...
- il **linguaggio di programmazione** è progettato per essere **conciso e non ambiguo**:
 - poche istruzioni con significato univoco e **indipendente dal contesto**
 - esclusa molta dell'espressività del linguaggio naturale, considerando *preciso solo ciò che è formalizzabile* in uno specifico linguaggio di programmazione.

S. Crafa A.A. 23/24

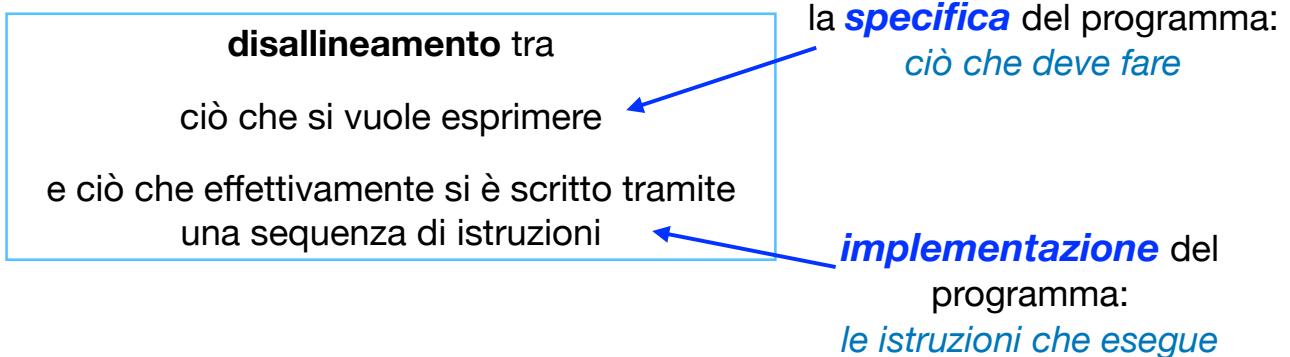
soluzioni in formato eseguibile

- Un **programma** consiste di **una serie di istruzioni** che spiegano come effettuare una computazione
- Un programma può contenere 2 tipi di **errori**:
 - **errori di sintassi**: istruzioni sintatticamente errate o struttura delle "frasi" non corretta (analogo ad errore di ortografia).
 - sono **individuati dalla macchina**, che non sa come procedere
 - **errori di semantica (o di logica)**: istruzioni sintatticamente corrette, quindi **eseguite dalla macchia**, ma **portano ad un risultato scorretto**.
 - il programmatore ha ragionato in modo errato, **sbagliando l'idea risolutiva** (e.g. errato modo di calcolare la media esami)
 - il ragionamento del programmatore è corretto, ma **nella fase di "codifica del ragionamento"**, ha inserito nel software delle istruzioni che non rappresentano correttamente il suo ragionamento.

S. Crafa A.A. 23/24

errori di semantica (o di logica)

- istruzioni sintatticamente corrette, quindi eseguite dalla macchia, ma che portano ad un risultato scorretto.



S. Crafa A.A. 23/24

**consideriamo un vero
programma**

...in che linguaggio?

Welcome to Snap!

Snap! is a broadly inviting programming language for kids and adults that's also a platform for serious study of computer science.

[Run Snap! Now](#) [Example Projects](#) [Reference Manual](#)

<https://snap.berkeley.edu>

Welcome to the Beauty and Joy of Computing—BJC

In this course, you will create programs using the Snap! programming language, you will learn some of the most powerful ideas of computer science, and you will discuss the social implications of computing, thinking deeply about how you can be personally active in promoting the benefits and reducing the possible harms.

For the best experience with Snap!, make sure your browser is up to date.

UNIT 1: INTRODUCTION TO PROGRAMMING

- Practice AP Create Task

UNIT 2: ABSTRACTION

UNIT 3: DATA STRUCTURES

- Practice AP Create Task

UNIT 4: HOW THE INTERNET WORKS

UNIT 5: ALGORITHMS AND SIMULATIONS

- Practice AP Create Task

UNIT 6: HOW COMPUTERS WORK

UNIT 7: FUNCTIONS AND RECURSION

UNIT 8: RECURSIVE FUNCTIONS

AP CSP ENDORSED

The Beauty and Joy of Computing is endorsed by the College Board as an endorsed provider of curriculum and professional development for AP Computer Science Principles (AP CSP). This endorsement affirms that all components of Beauty and Joy of Computing's offerings align to the AP Computer Science Principles and the AP CSP assessment. Using an endorsed provider affords schools access to resources including an AP CSP syllabus pre-approved by the College Board's AP Course Audit, and officially recognized professional development that prepares teachers to teach AP CSP.

bjc Beauty and Joy of Computing

<https://bjc.edc.org/bjc-r/course/bjc4nyc.html>

```

when green flag clicked
say [Ciao! A cosa vuoi giocare? for 2 secs
ask [1.Scacchi||2.Indovina il numero||3.Guerra termonucleare globale] and wait
say [Indovina il numero...Ottima scelta!] for 3 secs
ask [inserisci il numero:] and wait
if [answer = 7]
  say [Indovinato!]
else
  say [Sbagliato...sarai piu fortunato in amore]
  
```

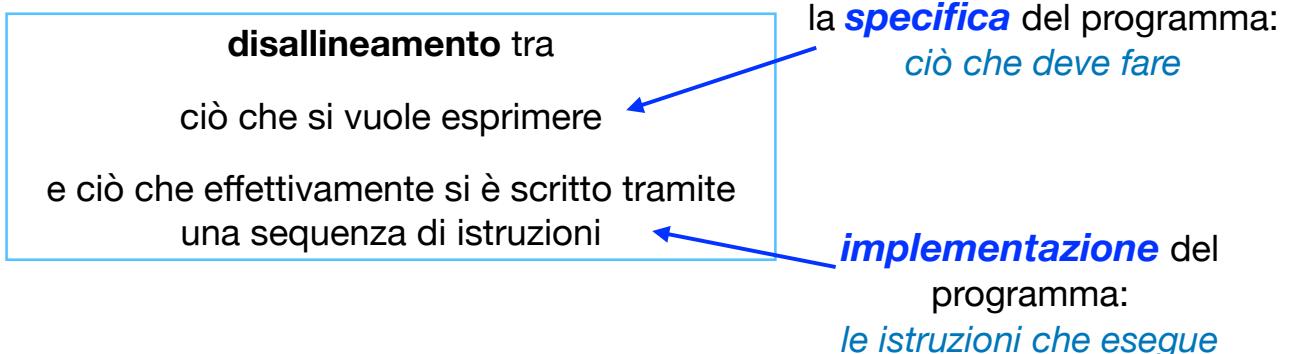
È un programma sbagliato?

```

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 piu fortunato in amore')
  
```

errori di semantica (o di logica)

- istruzioni sintatticamente corrette, quindi **eseguite dalla macchia**, ma che portano ad un **risultato scorretto**.



Come si individuano gli errori semanticici ?



- spesso non è facile accorgersi che il risultato finale non è corretto!
- possono rivelarsi solo nel caso di particolari input,
es. software di calcolo tasse può dare risultato errato solo quando inserisco mese 04 senza 0.

S. Crafa A.A. 23/24

l'hacking di The DAO



- Giugno 2016, **The DAO** è un fondo di investimento completamente automatizzato su Ethereum di 150 m\$ e 11.000 persone: un partecipante sfrutta una vulnerabilità del codice per trasferire 50m \$ al fondo Dark DAO.
- **Secondo il contratto legale della DAO tutte le regole sono definite dal codice stesso**
- quindi c'è **incertezza legale** se questo era un **furto** oppure una **feature** del codice.

S. Crafa A.A. 23/24

init

```

note: Simple Bet v2
note: Mitch bets Jack that BTC will be over $2000 by Dec 5, 2014
note: This "init" code runs one time only when the contract is first created
note: These addresses are fake but you can see the format

In save slot Mitch put 0xf4b7cc7fae866a2275972317598e7d036cfca0dc
in save slot Jack put 0x52c5535efae90c06e04c627aa5c716a092050c5e

```

body

```

note: Grabbing outside data like the BTC price will be available eventually...
note: ...by asking a "data feed" contract once the ecosystem evolves
note: Until then, we can take the data supplied to the contract as the BTC price.
note: We use a temp slot here since we don't need to store it between runs.

in temp slot BTC put 1st Input

note: We can use an 'OR' block here to avoid duplicating the code for each person

when
  contract caller = data at save slot Mitch
  OR
  contract caller = data at save slot Jack

then
  note: block.timestamp will be seconds elapsed since 1970 GMT.
  note: So for 12/04/2015 that's 1449107200

  when
    block timestamp > 1449107200
    then
      if
        data at temp slot BTC > 2000
        then
          note: If BTC is over 2000 pay off Mitch
          spend contract balance to data at save slot Mitch
        else
          note: Otherwise it must not be over 2000 so pay off Jack
          spend contract balance to data at save slot Jack

  note: We don't need explicit 'stop' blocks since contract will stop when it reaches the end.

```

CONTRACT

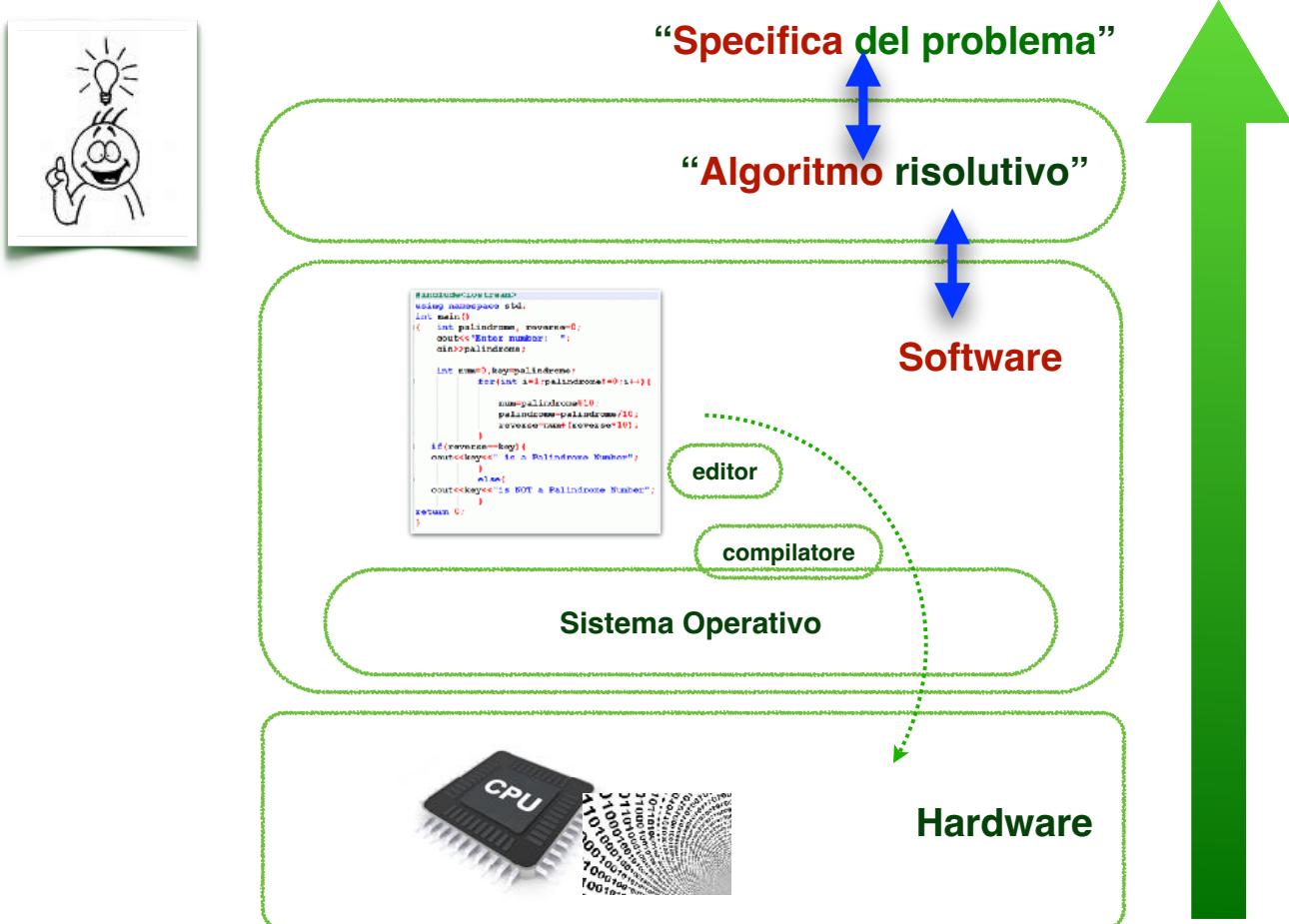
```

<contract>
  ...
</contract>

```

future su bitcoin (BTC)

linguaggio Marlowe
per scrivere smart contracts finanziari
sulla piattaforma Cardano



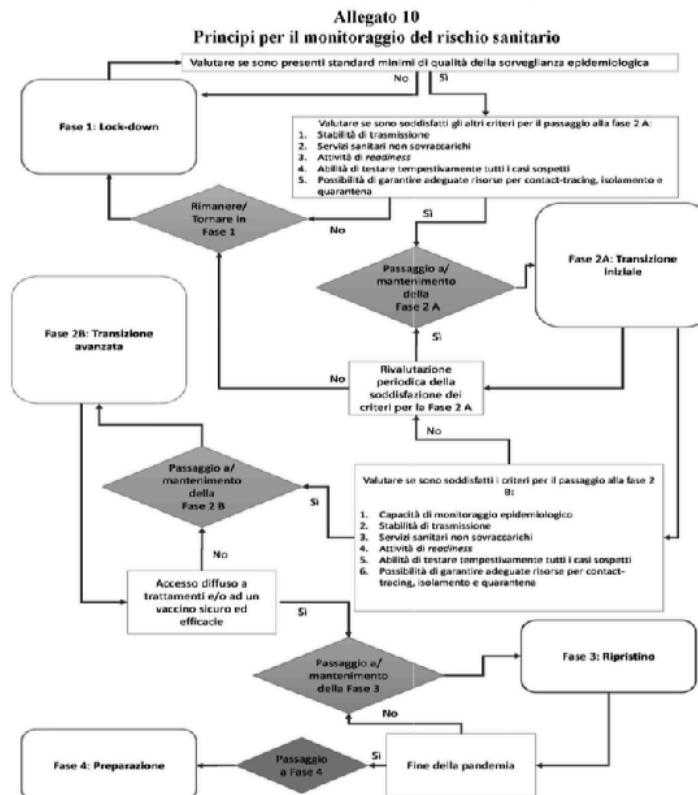
Algoritmo vs Software

- rappresenta la logica di funzionamento di un programma
- è scritto in pseudo-codice, anche a parole o con diagrammi
(es. algoritmo che assegnava i colori-*Covid alle regioni in autunno 2020*)

- è il codice di un programma, cioè una sequenza di istruzioni scritte in un preciso linguaggio di programmazione
- è eseguibile da un computer.

S. Crafa A.A. 23/24

27-4-2020 GAZZETTA UFFICIALE DELLA REPUBBLICA ITALIANA Serie generale n. 108



Algoritmo vs Software

- Un **algoritmo** è "*una sequenza non ambigua di passi che determina la procedura di soluzione di un problema*".
 - Ma questa sequenza di passi può essere descritta in qualsiasi modo (es. una ricetta per la pasta alla carbonara scritta a parole)
- L'algoritmo va **implementato**, cioè tradotto in un **software**: una sequenza di **istruzioni** scritte in un *linguaggio di programmazione*:
 - solo questa traduzione è *pienamente non ambigua ed effettivamente eseguibile*.
 - lo stesso algoritmo può essere implementato in modi diversi e in linguaggi diversi

S. Crafa A.A. 23/24

Algoritmo

- Un algoritmo è scritto con **qualche linguaggio adatto** ad esprimere "seguenze di passi che portano alla soluzione".
 - Ci sono **tanti linguaggi possibili**, come i diagrammi di flusso, o i sofisticati diagrammi UML, o varie forme di **pseudo-codice**
- Anche il linguaggio naturale si presta a scrivere un algoritmo. Usando però
 - un elenco di frasi brevi e chiare, e
 - i costrutti "se ... allora ... altrimenti ..." e "ripeti ..."

istruzione A
istruzione B
se **allora** istruzione C1
 altrimenti istruzione C2
istruzione D

istruzione A
ripeti
 istruzione B
 istruzione C

ripeti ..2.. **volte**
ripeti **finché** ..elenco non è vuoto..
ripeti **per ogni** giorno di settimana

S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un elenco di prezzi di articoli, dice il totale della spesa

Algoritmo:



IDEA :

*considero un prezzo per volta e tengo traccia del **totale parziale**, che è la somma dei prezzi già considerati.*

S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un elenco di prezzi di articoli, dice il totale della spesa

Algoritmo:

INPUT : elenco di prezzi in euro **OUTPUT** : totale della spesa in euro

- tieni traccia del totale, usando una variabile di nome *tot* che all'inizio ha valore 0
- **ripeti** **finché** l'elenco non è esaurito:
 - prendi il valore attuale di *tot*
 - sommaci il prossimo numero nell'elenco
 - aggiorna il valore di *tot* con la somma appena calcolata
- restituisci il valore della variabile *tot*

S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un elenco di prezzi di articoli, dice il totale della spesa

Algoritmo:

Algoritmo tipico per sommare una serie di numeri

INPUT : elenco di prezzi in euro **OUTPUT** : totale della spesa in euro

- tieni traccia del totale, usando una variabile di nome *tot* che all'inizio ha valore 0
- **ripeti finché** l'elenco non è esaurito:
 - prendi il valore attuale di *tot*
 - sommaci il prossimo numero nell'elenco
 - aggiorna il valore di *tot* con la somma appena calcolata
- restituisci il valore della variabile *tot*

esempio di **esecuzione** di algoritmo:

```
INPUT: [10, 3.50, 11, 17.99, 22]
tot = 0
tot = 10 (0+10)
tot = 13.50 (10+3.50)
tot = 24.50 (13.50 + 11)
tot = 42.49 (24.50 + 17.99)
tot = 64.49 (42.49+22)
OUTPUT : 64.49
```

Esempio

Specifiche:

- un programma che, dato un elenco di prezzi di articoli, **stampa lo scontrino della spesa**

Algoritmo:

INPUT : elenco di prezzi in euro **OUTPUT** : **scontrino** della spesa in euro

- tieni traccia del totale, usando una variabile di nome *tot* che all'inizio ha valore 0
- **ripeti finché** l'elenco non è esaurito:
 - prendi il prossimo numero dell'elenco e memorizzalo in una variabile di nome *x*
 - stampa sullo scontrino il valore di *x* e vai a capo
 - prendi il valore attuale di *tot*
 - sommaci il valore **di** *x*
 - aggiorna il valore di *tot* con la somma appena calcolata
- stampa sullo scontrino la scritta "totale spesa="
- stampa sullo scontrino il valore della variabile *tot*

Esemp

Specifiche:

- un programma che, dato un elenco di prezzi, stampa il **scontrino della spesa**

Algoritmo:

INPUT : elenco di prezzi in euro **OUTPUT : scor**

- tieni traccia del totale, usando una variabile di nome *tot* che all'inizio ha valore 0
- **ripeti** finché l'elenco non è esaurito:
 - prendi il prossimo numero dell'elenco e memorizzalo in una variabile di nome *x*
 - stampa sullo scontrino il valore di *x* e vai a capo
 - prendi il valore attuale di *tot*
 - sommaci il valore di *x*
 - aggiorna il valore di *tot* con la somma appena calcolata
- stampa sullo scontrino la scritta "totale spesa="
- stampa sullo scontrino il valore della variabile *tot*

esempio di **esecuzione** di algoritmo:

INPUT: [10, 3.50, 11, 17.99, 22]
tot = 0
tot = 10 (0+10)
tot = 13.50 (10+13.50)
tot = 24.50 (13.50 + 11)
tot = 42.49 (14.50 + 17.99)
tot = 64.49 (32.49+22)

OUTPUT :

10
3.50
11
17.99
22
totale spesa = 64.49

S. Crafa A.A. 23/24

Esercizio IMPORTANTE su Algoritmi

Scrivere i seguenti algoritmi, trovando un linguaggio efficace.

Inserire le soluzioni, eventualmente incomplete, sul forum Moodle e discuterne

ATTENZIONE: questo è un esercizio difficile perché non è facile trovare il modo più adatto di scrivere questi algoritmi. Lo scopo vero dell'esercizio, non è arrivare alla soluzione ma **esplorare quali difficoltà si incontrano provandoci**.

- Scrivere l'algoritmo per preparare la carbonara. Elencare all'inizio gli ingredienti (cioè i dati di input) e poi descrivere l'algoritmo per realizzare il piatto.
- Scrivere l'algoritmo per calcolare la media dei voti degli esami sostenuti.
- Il gioco "pari o dispari" ammette 2 giocatori: il primo giocatore scommette sempre su "pari", mentre il secondo scommette sempre su "dispari". Ogni giocatore sceglie un numero, e se la somma dei due numeri è pari vince il primo giocatore, altrimenti vince il secondo. Scrivere l'algoritmo di un programma che gioca a "pari o dispari"
- Scrivere algoritmo di un torneo sportivo ad eliminazione diretta a partire dai quarti di finale. Cioè ci sono 4 coppie di squadre che giocano, le vincitrici vanno alle semifinali e le vincitrici alla finale.
- Scrivere un algoritmo per l'elezione del capoclassa.

S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un numero n , dice se è **primo** oppure no

*un numero è primo
se è divisibile solo per 1 e per se stesso
cioè
se non è multiplo di nessun numero*

es. 11 è primo, 7 è primo
12 non è primo infatti $12:3=4$ con resto 0
130 non è primo infatti $130:10=13$ con resto 0

S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un numero n , dice se è **primo** oppure no

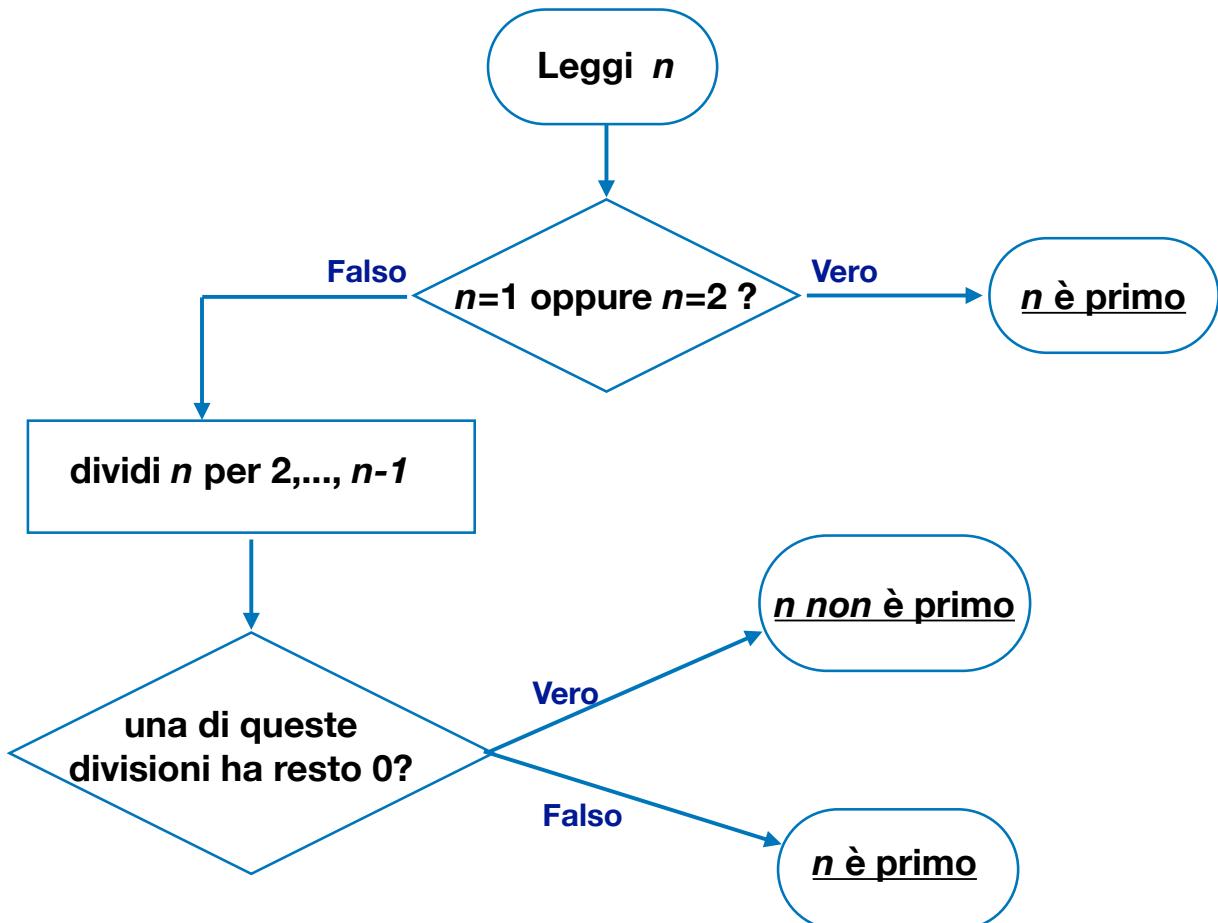
Algoritmo:

INPUT : un numero n

OUTPUT : " n è primo" opp " n non è primo"

- **se** n è il numero 1 oppure 2, **allora** n è primo
- **altrimenti**:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n-1$
 - **se** una di queste divisioni ha resto 0 allora dici " n non è primo"
 - **altrimenti** (cioè tutte le divisioni hanno resto non 0) dici " n è primo"

S. Crafa A.A. 23/24



S. Crafa A.A. 23/24

Esempio

Specifiche:

- un programma che, dato un numero n , dice se è primo oppure no

Algoritmo:

INPUT : un numero n **OUTPUT :** "n è primo" opp "n non primo"

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n-1$
 - se una di queste divisioni ha resto 0 allora

è ok dividere fino a $n-1$? $n/2$?
è ok separare i casi $n=1,2$?

dato l'algoritmo posso ragionare sulla **correttezza**
della procedura risolutiva

quando mi sembra di avere un algoritmo corretto
devo **scrivere un software che lo esegue**

S. Crafa A.A. 23/24

Esempio: Algoritmo che, dato un numero n dice se è primo oppure no

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n-1$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo

che istruzioni software gli faccio corrispondere?

- faccio **tutte** le divisioni e poi guardo **tutti** i risultati?
- ma se **$n = 100$** , divido per 2,3,4,...,99 ma potrei accorgermi subito che già la divisione per 2 ha resto 0, evitando altre **98 divisioni inutili** !
- scelgo l'istruzione:

"dividi n per div e controlla il resto, se il resto è 0 smetti, altrimenti ripeti dividendo per $div+1$. Inizia con $div=2$ e smetti di ripetere se $div = n-1$ "

S. Crafa A.A. 23/24

Esempio: Algoritmo che, dato un numero n dice se è primo oppure no

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n-1$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo

```
n = int(input())
if n==1 or n==2:
    print("n primo")
else:
    div = 2
    resto = n%div
    while (resto != 0 and div < n-1):
        div = div+1
        resto = n%div
    if resto == 0:
        print("n non primo")
    else:
        print("n primo")
```

"dividi n per div e controlla il resto, se il resto è 0 smetti, altrimenti ripeti dividendo per $div+1$. Smetti di ripetere se $div=n-1$ "

S. Crafa A.A. 23/24

Esempio: Algoritmo che, dato un numero n dice se è primo oppure no

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n-1$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo

**L'algoritmo è corretto
ma**

**I'implementazione ha
un errore di semantica**

es. 8--> "n non primo"
7--> "n non primo"

```
n = int(input())
if n==1 or n==2:
    print("n primo")
else:
    div = 1
    resto = n%div
    while (resto != 0 and div < n-1):
        div = div+1
        resto = n%div
    if resto == 0:
        print("n non primo")
    else:
        print("n primo")
```

errore!

S. Crafa A.A. 23/24

Esempio: Algoritmo che, dato un numero n dice se è primo oppure no

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n/2$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo



- L'implementazione richiede di definire (*disambiguare*) tanti dettagli dell'algoritmo
- Più difficile se chi ha scritto l'algoritmo è diverso da chi scrive il software!

```
n = int(input())
if n==1 or n==2:
    print("n primo")
else:
    div = 1
    resto = 1
    while (resto != 0 and div <= n/2):
        div=div+1
        resto = n%div
    if resto == 0:
        print("n non primo")
    else:
        print("n primo")
```

corretto ?!

S. Crafa A.A. 23/24

diversi livelli di astrazione



un errore può stare
su ciascuno dei livelli !!

1. **Specifiche** del programma: "Il programma prende in input un numero e ritorna in output se è primo oppure no"

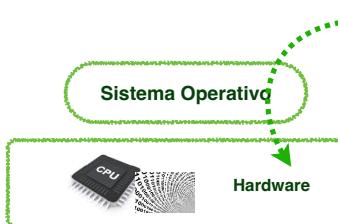


2. **Algoritmo** risolutivo:

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n/2$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo



3. **Implementazione**



```
n = int(input())
if n==1 or n==2:
    print("n primo")
else:
    div = 2
    resto = n%div
    while (resto != 0 and div < n-1):
        div=div+1
        resto = n%div
    if resto == 0:
        print("n non primo")
    else: print("n primo")
```

S. Crafa A.A. 23/24

app di tracciamento contatti in pandemia



1. **Specifiche** del programma: "Il programma prende in input ... calcola ... manda il risultato a ..."



2. **Algoritmo** risolutivo:

- attiva il bluetooth low energy,
- genera stringa casuale ogni 5 sec e inviala
- controlla se arriva una stringa e memorizza
- ...



3. **Implementazione**



S. Crafa A.A. 23/24

diversi livelli di astrazione



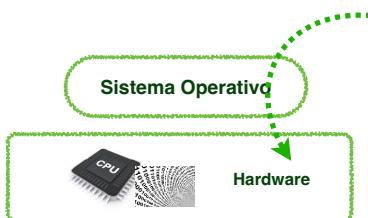
1. **Specifica** del programma:



2. **Algoritmo** risolutivo:



3. **Implementazione**



Twitter ----> X ?

UniWeb?

cercare esempi di applicazioni in cui queste distinzioni vi sembrano interessanti
=> Forum

S. Crafa A.A. 23/24

App Immuni

<https://www.immuni.italia.it/>

- High level description

<https://github.com/immuni-app/immuni-documentation>

<https://github.com/immuni-app/immuni-dashboard-data/blob/master/dati/andamento-settimanale-dati-regionali-latest.csv>

- documentazione tecnica

<https://github.com/immuni-app/immuni-app-android>

<https://github.com/immuni-app/immuni-documentation/blob/master/Technology.md>

- esempi di codice sorgente (Kotlin e Go)

<https://github.com/immuni-app/immuni-app-android/blob/development/app/src/main/java/it/ministerodellasalute/immuni/logic/exposure/repositories/ExposureAnalyticsStoreRepository.kt>

<https://github.com/immuni-app/immuni-app-android/blob/development/app/src/main/java/it/ministerodellasalute/immuni/logic/notifications/AppNotificationManager.kt>

<https://github.com/google/exposure-notifications-server/blob/main/internal/integration/client.go>

diversi livelli di astrazione

1. **Specific**a del programma
2. **Algoritmo** risolutivo
3. **Implementazione**
4. **Infrastruttura di esecuzione**



- molto spesso i 3 livelli sono **confusi e mescolati**
- es. *inizio prototipo* di implementazione con *idea grezza di cosa fa*
 - se l'**algoritmo resta implicito**, rischio ci siano errori di correttezza
 - se la specifica (chiara) manca (es. solo commenti al codice), è difficile comunicare cosa fa il software, e mantenerlo



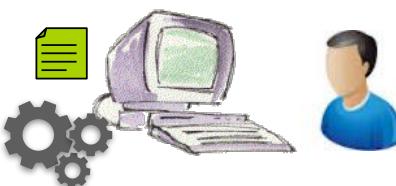
S. Crafa A.A. 23/24

diversi livelli di astrazione

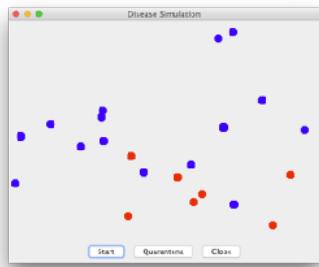
1. **Specific**a del programma
2. **Algoritmo** risolutivo
3. **Implementazione**
4. **Infrastruttura di esecuzione**

*spesso l'infrastruttura di esecuzione
ha importanti conseguenze
sui livelli superiori !*

dove avviene la computazione?



applicazione locale



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



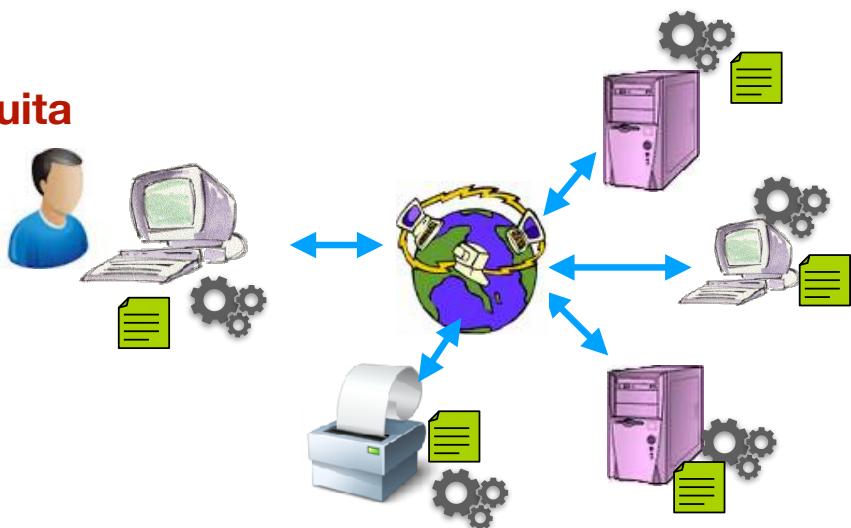
- **dove si esegue il programma?**
- se salvo/apro un progetto, dove risiede?

S. Crafa A.A. 23/24

dove avviene la computazione?

applicazione distribuita

es. mail, stampare

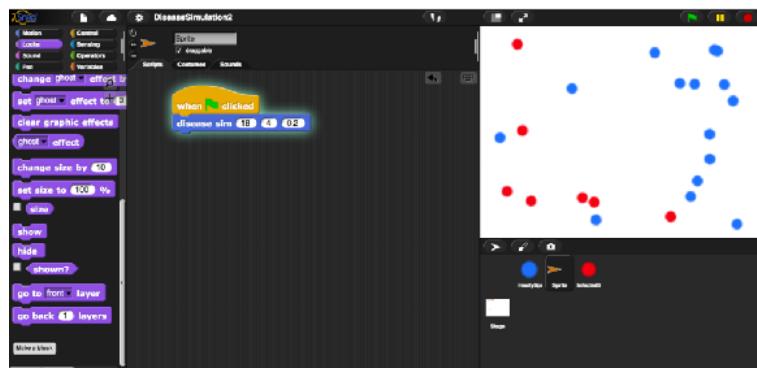
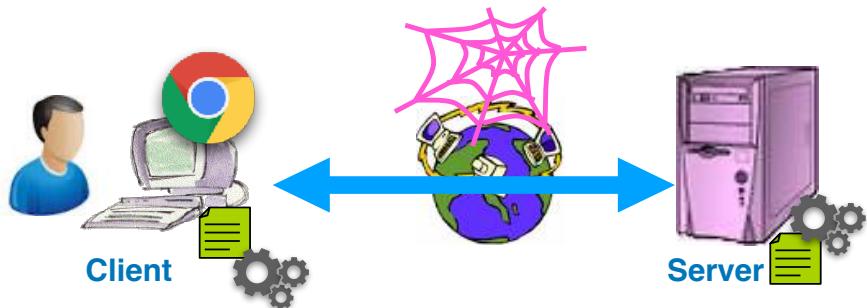


S. Crafa A.A. 23/24

dove avviene la computazione?

applicazione Web

es. web mail, banking online,
twitter



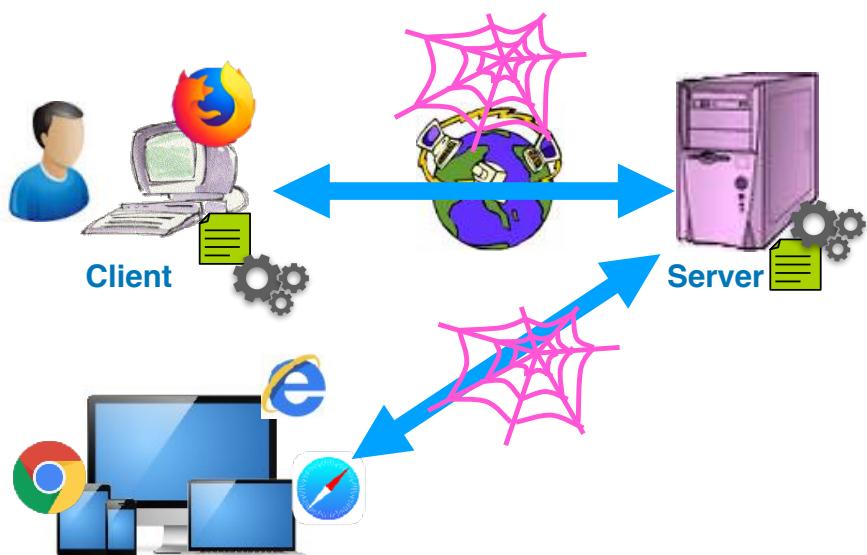
- dove esegue il programma?
- se salvo/apro un progetto, dove risiede?

S. Crafa A.A. 23/24

dove avviene la computazione?

applicazione Web

es. web mail, banking online,
Il Corriere



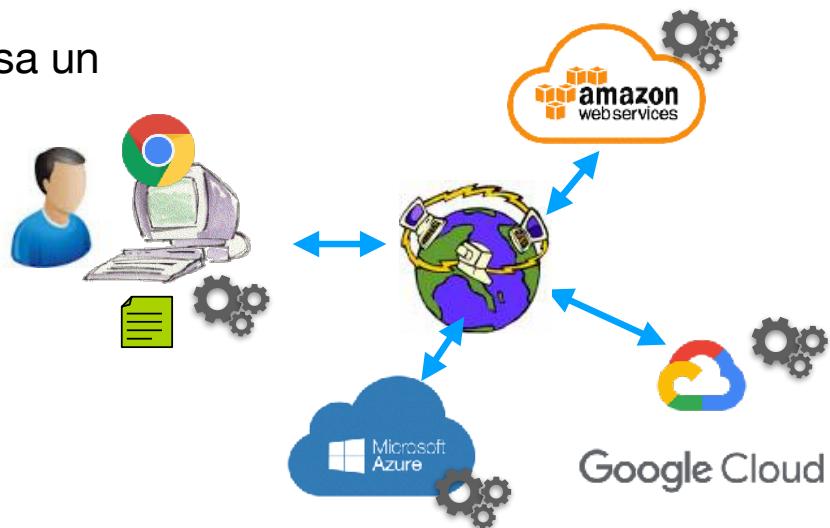
se un'applicazione funziona
tramite il browser web, la si
può usare da desktop, tablet,
smartphone

S. Crafa A.A. 23/24

dove avviene la computazione?

applicazione che usa un servizio cloud

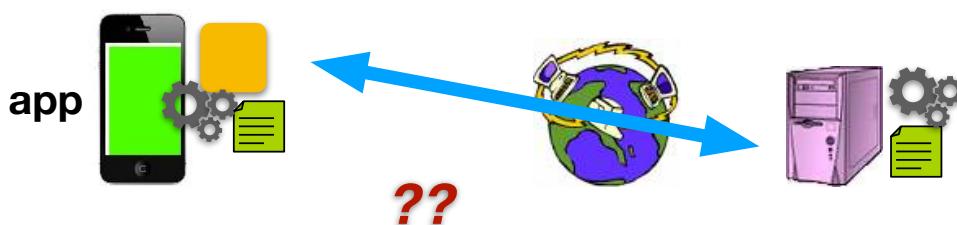
es. Moodle,
firma digitale,
archivio foto



cloud = un sistema distribuito privato e opaco

S. Crafa A.A. 23/24

dove avviene la computazione?



La app puo' causare computazione sia locale che remota

- se si connette a server remoto, **che dati gli manda?**
- **non è sempre facile capire** se c'è computazione remota (opp. disattivare rete)

c'è tanta complessità nascosta (**astrazione**) tra le istruzioni di un programma e l'esecuzione della macchina!
Bisogna esserne Consapevoli

S. Crafa A.A. 23/24

diversi livelli di astrazione

1. **Specific**a del programma
2. **Algoritmo** risolutivo
3. **Implementazione**
4. **Infrastruttura di esecuzione**



*spesso l'infrastruttura di esecuzione
ha importanti conseguenze
sui livelli superiori !*

- un **computer** (esecuzione *locale*)
- uno **smartphone** (esecuzione *parzialmente remota*)
- un **sistema distribuito** (esecuzione *su server tramite interfaccia web*)
- un **cloud** = un **sistema distribuito privato e opaco**

S. Crafa A.A. 23/24

diversi livelli di astrazione

1. **Specific**a del programma
2. **Algoritmo** risolutivo
3. **Implementazione**
4. **Infrastruttura di esecuzione**

- **Un errore può stare su ciascun livello:**

- tenere presente che esistono tutti
 - *saper assegnare ad ogni errore il livello a cui appartiene*

- **Persone diverse si occupano di ciascun livello:**

- ogni livello richiede competenze specifiche, dunque *responsabilità specifiche*
 - molto difficile comunicare informazioni (corrette) tra livelli, per molti motivi

S. Crafa A.A. 23/24

Algoritmo vs software

Esempio:

un programma che calcola la graduatoria degli insegnanti di scuola

1. L'algoritmo sottostante potrebbe utilizzare delle **regole di posizionamento discriminatorie**, fedelmente tradotte nel software.
 - In questo caso il **software** è corretto, mentre il problema sta nell'**algoritmo**.
 - **di chi è la responsabilità?**
 - la regola discriminatoria stava nella **specifica** del programma?
 - chi ha commissionato il software, **che informazioni/vincoli ha comunicato?** **In che formato** li ha comunicati?
 - se come **documentazione** abbiamo solo il **codice** del software, è difficile attribuire la responsabilità

S. Crafa A.A. 23/24

Algoritmo vs software

Esempio:

un programma che calcola la graduatoria degli insegnanti di scuola

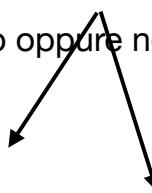
2. Consideriamo inoltre il caso in cui **l'algoritmo/specifica non dia alcuna indicazione su come trattare i casi di pari merito**
 - cioè la **specifica** del programma (i.e. cosa deve fare) non è **completa**.
 - L'**algoritmo/specifica** quindi è **ambiguo**, ma il **software** che lo implementa è per definizione **non ambiguo**:
 - le istruzioni dei linguaggi di programmazione sono fatte in modo che **la macchina sappia come comportarsi in ogni situazione**,
 - **incluse le situazioni a cui il programmatore non ha esplicitamente pensato!**

S. Crafa A.A. 23/24

sono (semanticamente) equivalenti o no a seconda di come è inizializzata la variabile **ammesso**, cioè a seconda di come è scritto il resto del programma

Porzione di codice che decide chi è ammesso oppure no ad una speciale posizione in graduatoria:

- `if (x > y) then ammesso = true`
- `if (x > y) then ammesso = true else ammesso = false`
- `if (x >= y) then ammesso = true else ammesso = false`



istruzioni molto simili ma che
possono produrre **conseguenze molto diverse!**

- Piccole differenze nelle istruzioni, come queste, possono finire per gestire implicitamente i casi di pari merito,
- anche senza l'esplicita intenzione del programmatore, che si è limitato ad implementare l'algoritmo senza accorgersi che la specifica non era completa.

S. Crafa A.A. 23/24

Pubblicato il 08/04/2019

N. 02270/2019REG.PROV.COLL.
N. 04477/2017 REG.RIC.



R E P U B B L I C A I T A L I A N A

IN NOME DEL POPOLO ITALIANO

Il Consiglio di Stato

in sede giurisdizionale (Sezione Sesta)

ha pronunciato la presente

SENTENZA

sul ricorso numero di registro generale 4477 del 2017, proposto da [...] contro

Ministero dell'Istruzione dell'Università e della Ricerca, Ufficio Scolastico Regione Puglia [...] per la riforma

della sentenza del T.A.R. per il Lazio, Sede di Roma, n. 12026 del 2016

Secondo gli appellanti, tale algoritmo avrebbe disposto i trasferimenti in una provincia piuttosto che in un'altra, in un posto di sostegno piuttosto che in un posto comune, senza tener conto delle preferenze indicate nelle rispettive domande di trasferimento, senza alcuna motivazione e in difetto della benché minima trasparenza

[...]

L'utilizzo di una procedura informatica che conduca direttamente alla decisione finale non deve essere stigmatizzata, ma anzi, in linea di massima, incoraggiata: essa comporta infatti numerosi vantaggi quali, ad esempio, la notevole riduzione della tempistica procedimentale per operazioni meramente ripetitive e prive di discrezionalità, l'esclusione di interferenze dovute a negligenza (o peggio dolo) del funzionario (essere umano) e la conseguente maggior garanzia di imparzialità della decisione automatizzata.

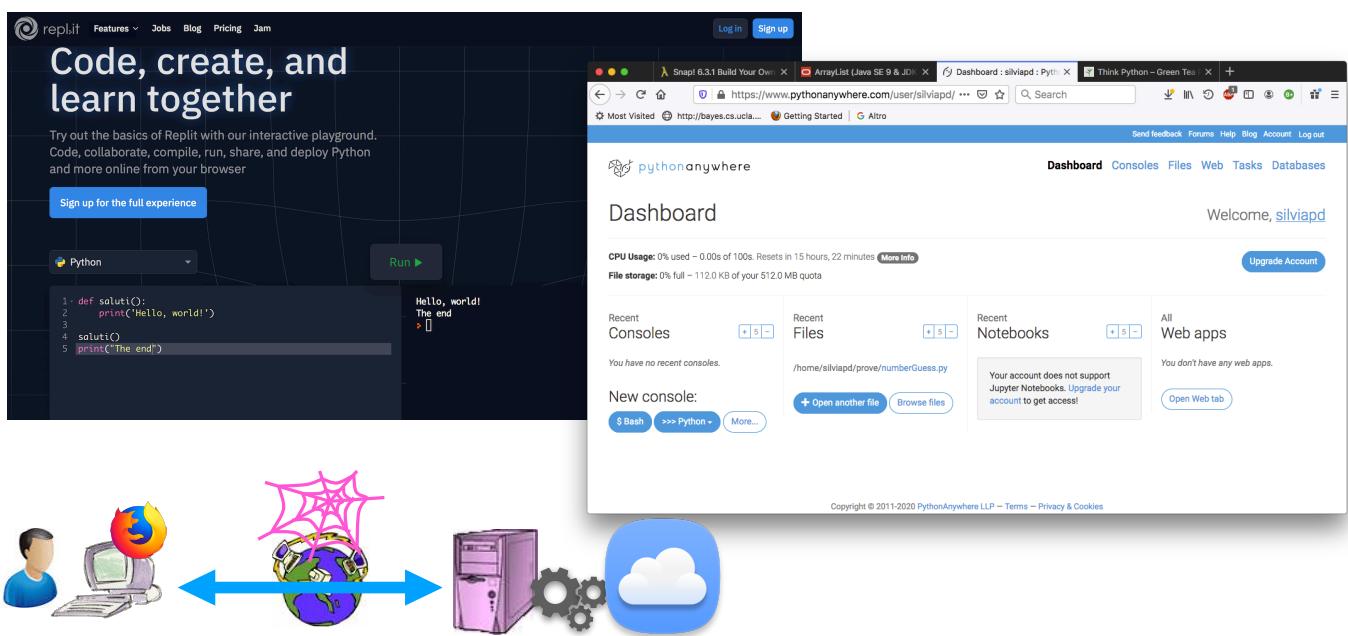
9 – Alla luce delle riflessioni che precedono, l'appello deve trovare accoglimento, sussistendo nel caso di specie la violazione dei principi di imparzialità, pubblicità e trasparenza, poiché non è dato comprendere per quale ragione le legittime aspettative di soggetti collocati in una determinata posizione in graduatoria siano andate deluse.

Infatti, l'impossibilità di comprendere le modalità con le quali, attraverso il citato algoritmo, siano stati assegnati i posti disponibili, costituisce di per sé un vizio tale da inficiare la procedura.

Non solo, gli esiti della stessa paiono effettivamente connotati dall'illogicità ed irrazionalità denunciate dalle appellanti, essendosi verificate situazioni paradossali per cui docenti con svariati anni di servizio si sono visti assegnare degli ambiti territoriali mai richiesti e situati a centinaia di chilometri di distanza dalla propria città di residenza

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**.

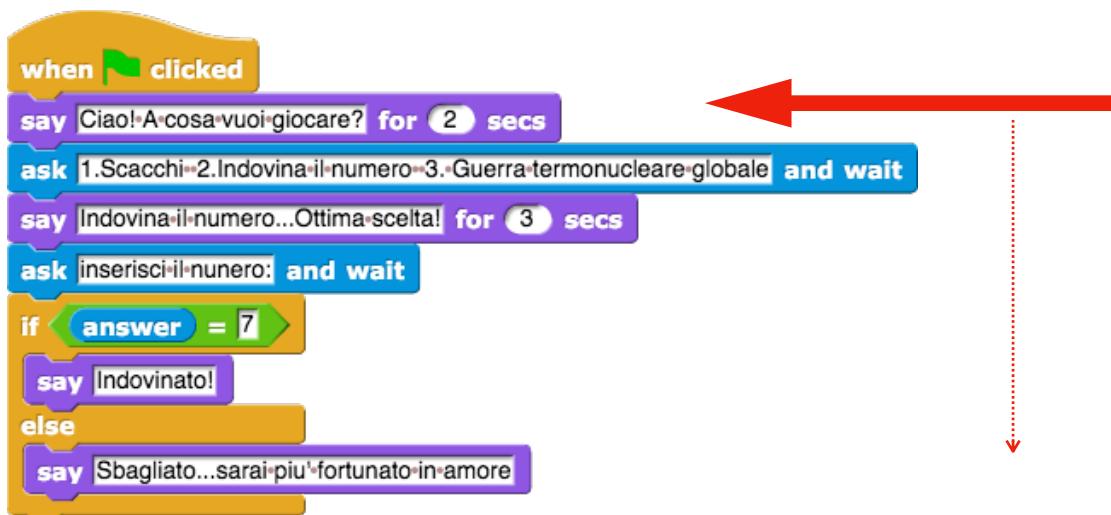
- Il codice **scritto** è una sequenza di istruzioni
- La macchina **esegue** una sequenza di istruzioni
- ma la **sequenza di istruzioni eseguite** non sempre coincide con la **sequenza scritta (flusso di controllo)**

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.

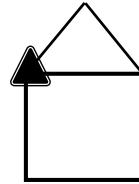
Lo strumento online **Python Tutor** permette di eseguire un programma visualizzando come il controllo fluisce da un'istruzione alla successiva.

<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 piu fortunato in amore')
```

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*:

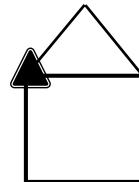
ripeti 4 volte

- *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*

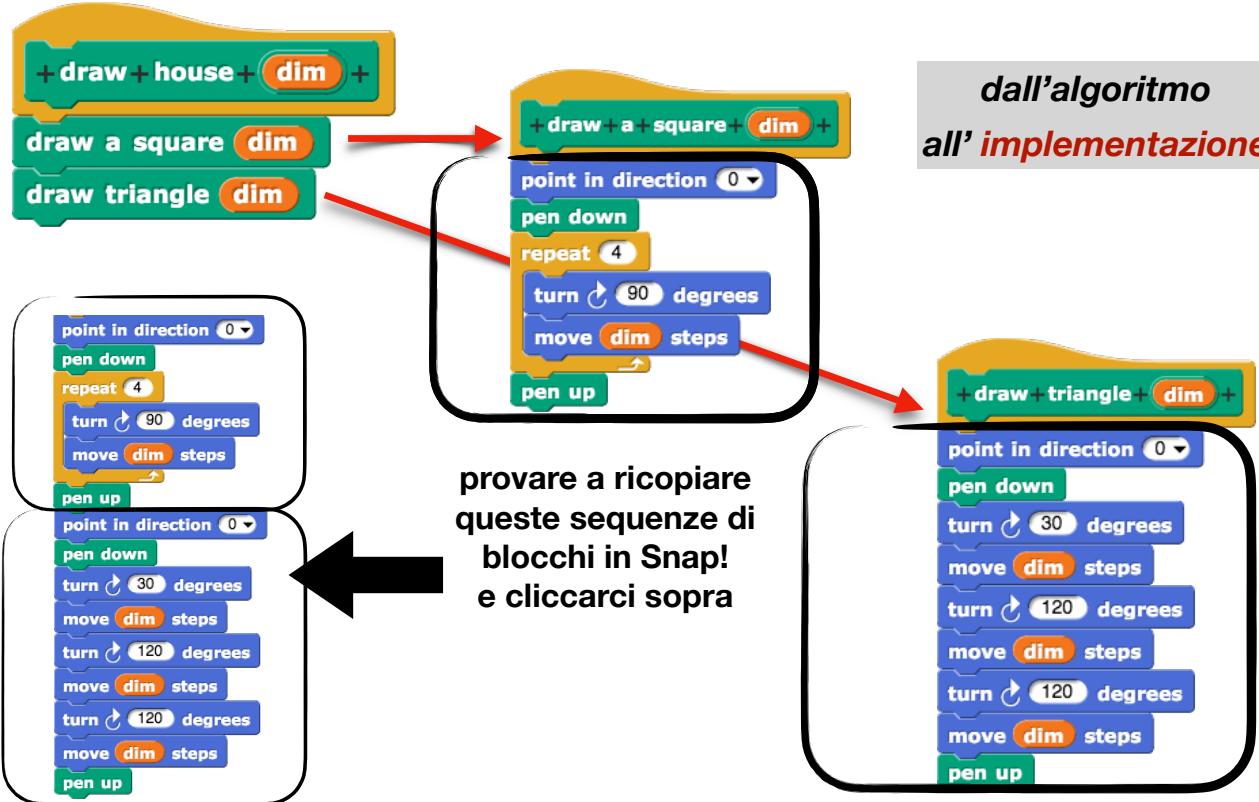
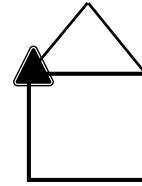


Algoritmo per *disegnare un triangolo*:

...

Algoritmo per disegnare una casa:

- disegna un quadrato
- disegna un triangolo

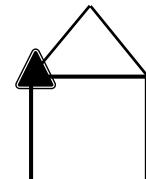


```
# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)
```

```
# 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`)

```

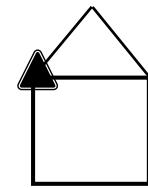
# procedura per disegnare una casa
def draw_house(dim):
    draw_square(dim)
    draw_triangle(dim)

# 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

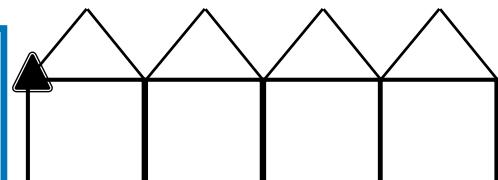
è la prima istruzione eseguita

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

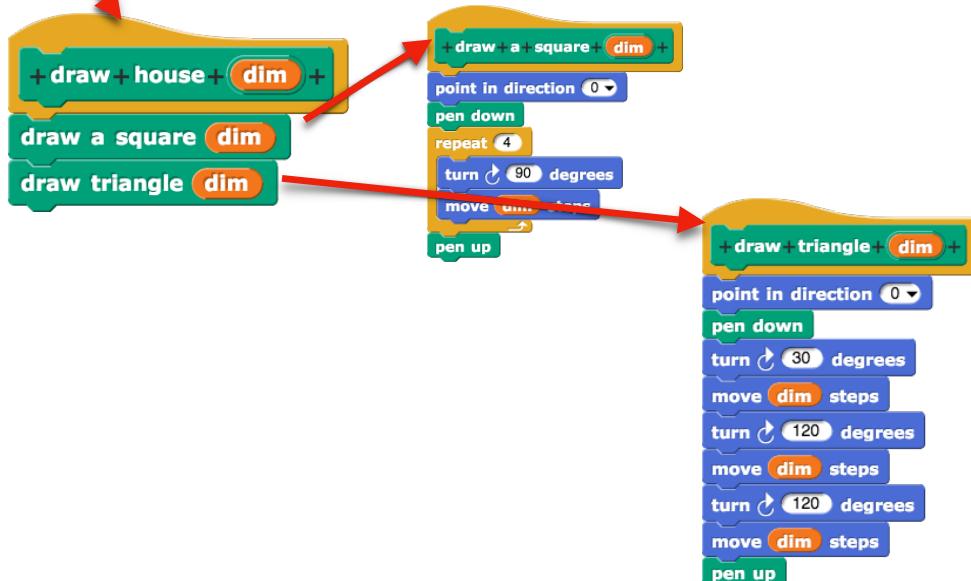
- gira il cursore di 90 gradi
- disegna un lato

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')

```

```

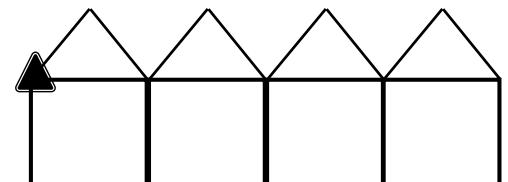
# procedura per disegnare un triangolo
def draw_triangle(dim):
    print('disegna un triangolo')

```

```

# 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)

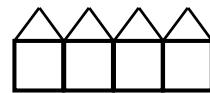
# procedura per "disegnare" un quadrato
def draw_square(dim):
    print('disegna un quadrato')

# 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
...

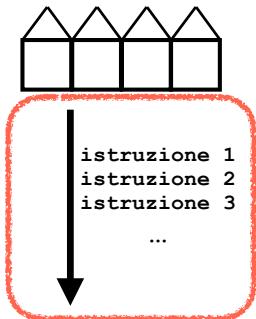
istruzione 1
istruzione 2
istruzione 3
...

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



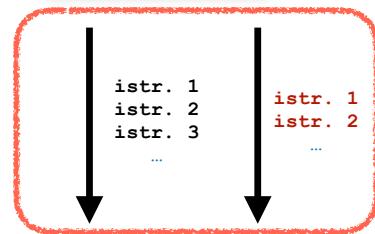
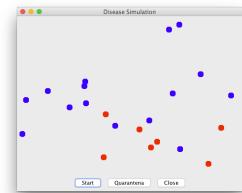
istruzione 1
istruzione 2
istruzione 1
istruzione 1
istruzione 2
istruzione 3
...

interleaving



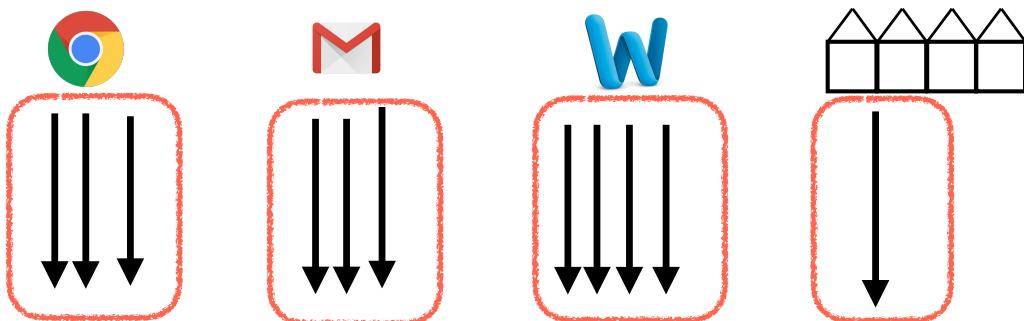
Programma Sequenziale:

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!")`

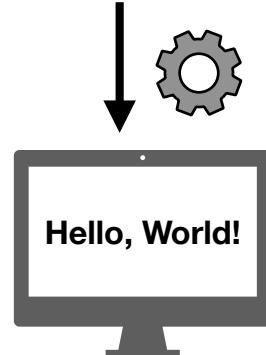
3. **eseguire** il programma con il tasto **Run** oppure dalla *console (bash)* con il comando

`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

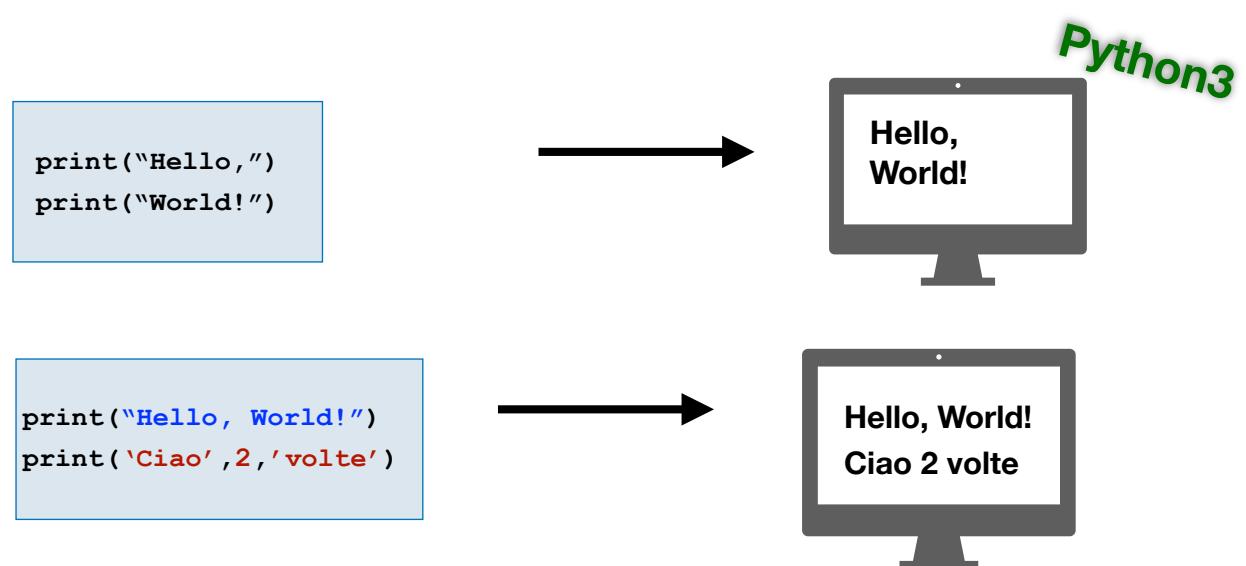
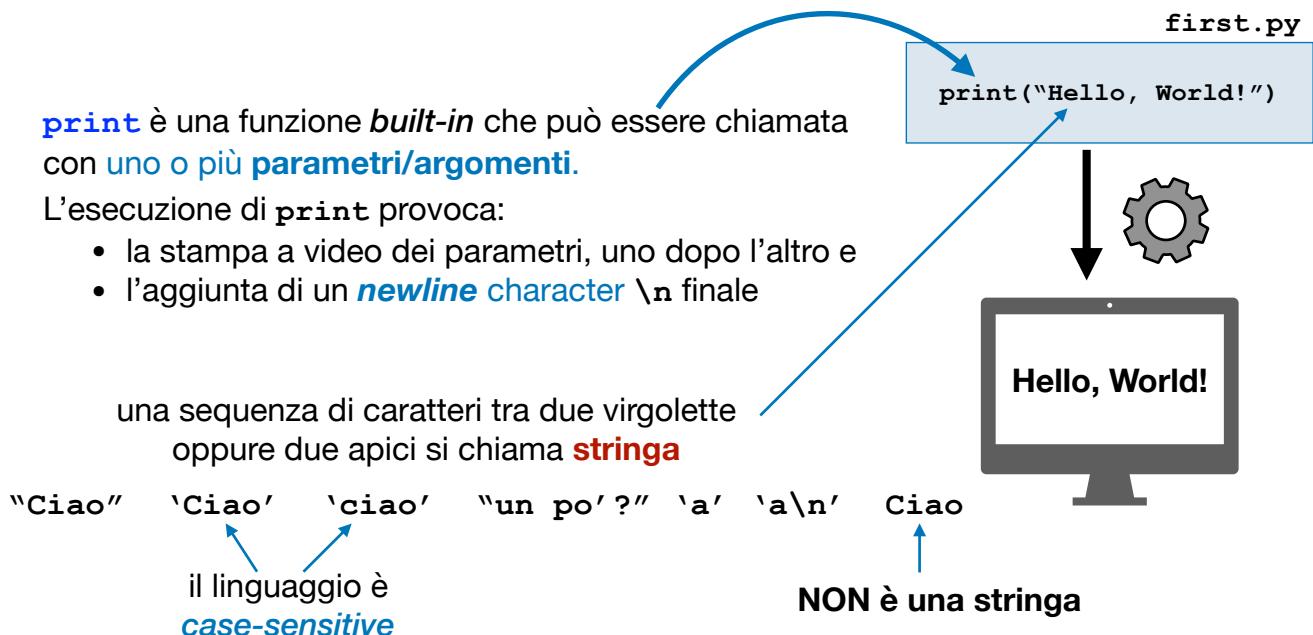
`first.py`

```
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

Valori e Tipi

- Un **valore** è un elemento base, es. una **stringa**, un **numero**, un **carattere**...
- Ogni valore ha un corrispondente **tipo**, che indica quali operazioni si possono fare con quel valore
 - 'Ciao Mondo' e "un po'" sono **stringhe**
`type('Ciao Mondo')` restituisce <class 'str'>
 - 2 è un valore di tipo **intero** (la funzione `type(2)` restituisce il tipo del valore passato come parametro, in questo caso <class 'int'>)
 - 42.3 è un valore di tipo decimale, detto **floating-point**, 'a virgola mobile' (`type(42.3)` restituisce <class 'float'>)
- `type('2')` restituisce <class 'str'>
- `type(2.0)` restituisce??

Esercizio

- Che differenza c'è tra questi due programmi ?

Programma A

```
type(2)  
type('2')
```

Programma B

```
print(type(2))  
print(type('2'))  
print("type(2)")
```

- Eseguirli e darsi una spiegazione del loro comportamento

Valori ed Espressioni

- i valori si possono combinare con degli **operatori** ed ottenere delle **espressioni**.
- espressioni aritmetiche:
 - $3+5$, $3-5$,
 - $7/3$ (divisione), $7//3$ (divisione intera), $7\%3$ (modulo, i.e. resto della divisione intera)
 - $4*5$, $8**3$ (potenza)
- operatori sulle stringhe:
 - `'arco'+'baleno'` (concatenamento)
 - `'bello'*3` (ripetizione)

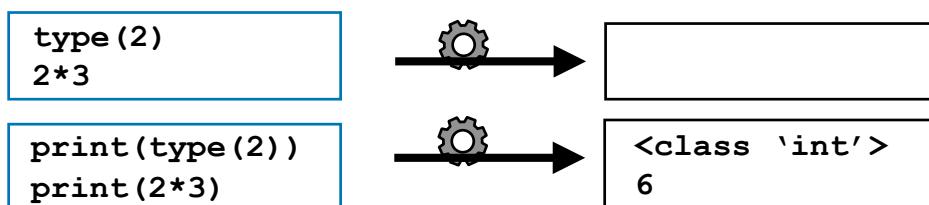
consuete regole di precedenza

Espressioni e Comandi

- Un **comando** è una porzione di codice che la macchina **esegue** e che ha qualche **effetto**. es. `print('pippo')` provoca una stampa a video
- Un'**espressione** è una porzione di codice che la macchina **valuta**, cioè trova e restituisce il **valore** dell'espressione. es. $(3+5)/3-4$ ha valore -8

`print("The result of 4+8=", 4+8)`  The result of 4+8= 12

L'interprete **valuta** i valori delle due **espressioni** passate come parametro, ed esegue la funzione `print` con gli **argomenti** "The result of 4+8=" e 12



Esercizio

- espressioni aritmetiche:
 - $3+5$, $3-5$,
 - $7/3$ (divisione), $7//3$ (divisione intera), $7\%3$ (modulo, i.e. resto della divisione intera)
 - $4*5$, $8**3$ (potenza)
- Scrivere un programma che visualizza il risultato delle precedenti espressioni:

```
print(7/3)
print(7%3)
```

```
print("Divisione: 7/3 = ", 7/3)
print("Modulo: 7%3 = ", 7%3)
```

Esercizio

- Che output produce il seguente programma?

| | |
|---|--------------------------------|
| print("4-8=", 4-8) | 4-8= -4 |
| print("4*8 = ", 4*8) | 4*8 = 32 |
| print("8/3=", 8/3) | 8/3= 2.6666666666666665 |
| print("8 raised to the power of 3=", 8**3) | 8 raised to the power of 3=512 |
| print("7//3 = ", 7//3) # integer division | 7//3 = 2 |
| print("7%3=", 7%3) # rest of the integer division | 7%3=1 |
| print() | |
| print("2*4+4*2 = ", 2*4+4*2) | 2*4+4*2 = 16 |
| print("2*(4+4)*2 = ", 2*(4+4)*2) | 2*(4+4)*2 = 32 |
| print('arco', 'baleno') | arco baleno |
| print('arco'+'baleno') | arcobaleno |
| print(('arco'+'baleno')*3) | arcobalenoarcobalenoarcobaleno |

- Che output produce il seguente programma?

```
print(2+4, 3*4)           6 12
```

```
print("2+4=", 2+2)        2+4=4      errore logico! (bug)  
                                l'esecuzione continua!
```

```
print(2+4=, 2+4)          SyntaxError    l'esecuzione si  
                                interrompe
```

conversioni di tipo

```
print('USD$', 100)         stampa USD$ 100  
  
print('USD$' + 100)       Type error: can only concatenate str  
                           (not "int") to str  
  
print('USD$' + str(100))  stampa USD$100  str(100) converte un int in str
```

`str(4)` è un' espressione di valore `'4'` converte l' `int` `4` nella stringa `'4'`
`str(4.56)` è un' espressione di valore `'4.56'` converte il `float` `4.56` nella stringa `'4.56'`

`int(4.65)` è un' espressione di valore `4` converte il `float` `4.65` nell' `int` `4`
`int('4')` è un' espressione di valore `4` converte la stringa `'4'` nell' `int` `4`
`int('4.65')` è espressione la cui valutazione produce `ValueError: invalid literal
for int() with base 10: '4.65'`

Valori e Variabili

- una **variabile** è un **nome** che fa riferimento ad un **valore**
- l'**istruzione di assegnamento** crea una nuova variabile, specificandone il nome, e le assegna un valore

```
messaggio = "cerca la X e scava"  
n = 24  
pi = 3.141592653589793
```

ATTENZIONE:
= non indica
l'uguaglianza!!

```
print(messaggio)      cerca la X e scava  
print(pi, type(pi)) 3.141592653589793 <class'float'>  
print(n+2)            26
```

valuta il **valore della variabile** e
lo usa nell'operazione `_+2`

- i nomi delle variabili possono contenere numeri e lettere,
- ma non possono iniziare con un numero
- per convenzione sono minuscoli e si usa _ es. `person_name` e `data_di_nascita`

Assegnamento

Un successivo assegnamento **modifica** il valore della variabile

```
messaggio = "cerca la X e scava"  
messaggio = "il tesoro non si trova mai sotto la X"  
print(messaggio)      # stampa il tesoro non ...
```

```
x = 10  
x = x + 1  
y = x
```

l'espressione a destra ha valore $10+1$ cioè 11
l'espressione a destra ha valore 11

Istruzione di assegnamento

- è sempre della forma **variable = espressione**
- viene eseguita nel modo seguente:
 1. si **valuta l'espressione a destra** di = e si ottiene un **valore**
 2. questo valore viene **assegnato alla variabile a sinistra** di =

Esercizio

```
# Programma che scambia il valore di due variabili

x = 5
y = 10

# crea una variabile in più di supporto
temp = x
x = y
y = temp

print('valore di x dopo lo scambio:',x)
print('valore di y dopo lo scambio:',y)
```

Modificare il programma in modo tale che **alla fine stampi anche il valore che avevano x e di y prima dello scambio**. Queste istruzioni aggiuntive di stampa non possono avere come argomento i numeri 5 e 10.

input da tastiera

Analogamente a `print`, in python c'è la *built-in* function `input`:

- ferma l'esecuzione **in attesa** che l'utente digitи qualcosa;
 - alla pressione del tasto return/enter, il programma riprende
 - e la funzione **input restituisce** ciò che è stato inserito sotto forma di **stringa**, che può essere memorizzata in una variabile.
 - Si può anche specificare un *prompt* come testo che chiede l'input

```
testo1 = input("Inserisci testo ")
print("Hai inserito: ", testo1)
testo_2 = input("Come ti chiami? ")
print("Hai inserito: ", testo_2)
input("Oggi piove?")
print("Bye bye")
```

il dato inserito dall'utente è restituito dalla funzione input, ma non è memorizzato quindi va perduto

Come si comporta il seguente programma?

```
text_anni = input("quanti anni hai?")
eta = int(text_anni)
print(text_anni * 2)
print(eta * 2)
print(type(text_anni), type(eta))
input()
print('e ora?')
```

Come si comportano i seguenti programmi?

```
anni = input("quanti anni hai?")
y = anni + 10
print(y)
```

```
text_anni = input("quanti anni hai?")
eta = int(text_anni)
y = eta + 10
print(y)
```

Esercizi

```
name = input ("your name: ")  
age = input ("your age: ")  
  
year = input("your birth year: ")  
  
future_age = int(year)+10  
  
print("The age of ", name, "in 2032 will be ", future_age)
```

dov'è l'errore?

```
n = input()    # inserisce 3  
  
x = n+'4'  
  
y = int(n)+3  
  
print(x, y)
```

cosa stampa?

Valori e Tipi

- Un **valore** è un elemento base, es. una **lettera**, un **numero**, una **parola**...
- Ogni valore ha un corrispondente **tipo**, che indica quali operazioni si possono fare con quel valore
 - **2** è un valore di tipo **intero** (la funzione `type(2)` restituisce il tipo del valore passato come parametro, in questo caso `<class 'int'>`)
 - **42.3** è un valore di tipo decimale, detto **floating-point**, ‘a virgola mobile’ (`type(42.3)` restituisce `<class 'float'>`)
 - **'Ciao Mondo'** e **"un po'"** sono **stringhe** (`type('Ciao Mondo')` restituisce `<class 'str'>`)
 - **True** e **False** sono i (sol) due valori di tipo **booleano** (`type(True)` restituisce `<class 'bool'>`)

espressioni booleane

- le **espressioni booleane** (o *predicati*) sono espressioni la cui valutazione produce vero o falso,
- ad es. sono expr booleane quelle che si ottengono con i seguenti operatori:
 - **operatori di confronto**

5 == 6
x != 5
x > y
x < y
5 >= 3
3 <= 3

= è per assegnamento
== è uguaglianza



Esempi:

5 == 6 ha valore **False**
2+3 == 5 ha valore **True**
dati gli assegnamenti
 $x = 4$ $y = -4$ $z = x+y$
 $x > y$ ha valore **True**
 $x < y$ ha valore **False**
 $x >= z$ ha valore **True**
 $z <= 0$ ha valore **True**

espressioni booleane

- le **espressioni booleane** (o *predicati*) sono espressioni la cui valutazione produce vero o falso,
- ad es. sono expr booleane quelle che si ottengono con i seguenti operatori:
 - **operatori di confronto**

5 == 6
x != 5
x > y
x < y
5 >= 3
3 <= 3



• operatori logici

($x > 0$) **and** ($x < 10$)
($x > 0$) **or** ($y > 0$)
not ($x == y$)



Esempi:

dati gli assegnamenti $n=4$ $m=3$ $a=7$
(($n+m$) / 2 > 0) **and** ($a == 7$) ha valore **True**
not ($m == 3$ **or** $n > 5$) ha valore **False**

Esercizi

Le seguenti tre espressioni booleane sono **equivalenti**, cioè per qualsiasi valore delle variabili x e y, sono tutte e tre vere oppure tutte e tre false

$x \leq y$

$(x < y) \text{ or } (x == y)$

$\text{not } (x > y)$

1. Scrivere due espressioni booleane equivalenti all'espressione $x \geq y$
2. Scrivere due espressione booleane equivalenti a $x \neq y$
3. Scrivere un'espressione booleana che ha valore **True se e solo se** la variabile x ha un valore che **sta** nell'intervallo di numeri [0,...,10]
4. Scrivere un'espressione booleana che ha valore **True se e solo se** x **non sta** nell'intervallo di numeri [0,...,10]
5. Scrivere un'espressione booleana che coinvolge 3 variabili x,y,z e ha valore **True se e solo se** x **sta** nell'intervallo di numeri [y,...,z]

Esercizi

Le seguenti tre espressioni booleane sono **equivalenti**, cioè per qualsiasi valore delle variabili x e y, sono tutte e tre vere oppure tutte e tre false

$x \leq y$

$(x < y) \text{ or } (x == y)$

$\text{not } (x > y)$

1. Scrivere due espressioni booleane equivalenti a $x \leq y$
2. Scrivere due espressione booleane equivalenti a $x \neq y$
3. Scrivere un'espressione booleana che ha valore **True se e solo se** la variabile x ha un valore che **sta** nell'intervallo di numeri [0,...,10]
4. Scrivere un'espressione booleana che ha valore **True se e solo se** x **non sta** nell'intervallo di numeri [0,...,10]
5. Scrivere un'espressione booleana che coinvolge 3 variabili x,y,z e ha valore **True se e solo se** x **sta** nell'intervallo di numeri [y,...,z]

come si può usare il
computer per controllare
la soluzione?

Esercizio

Quali delle espressioni seguenti è equivalente all'espressione `num <= 23` ?
(Sugg. la lista contiene 2 espressioni equivalenti)

- `num < 23 and num == 23`
- `num < 23 or num == 23`
- `num < 23 or num = 23`
- `not num > 23`
- `not num > 22`

come si può usare il
computer per controllare
la soluzione?

controllare la
soluzione usando
python

Esercizio

Quali delle seguenti espressioni booleane ha valore True?

A. (True and False) and (not (True and False))

B. (not (True or False)) or (True or False)

- solo A
- solo B
- A e B
- né A né B

controllare la
soluzione usando
python

Esercizio

- Scrivere un programma che calcola e stampa quanti secondi ci sono in 3 ore e quanti in 12 ore.
- Scrivere un programma che calcola e stampa quanti giorni sono trascorsi dal 1 gennaio 2023 ad oggi.

**ci sono tante diverse soluzioni,
provate a confrontarle**

Come si comporta il seguente programma?

```
text_anni = input("quanti anni hai?")
eta = int(text_anni)
print(text_anni * 2)
print(eta * 2)
print(type(text_anni), type(eta))
input()           inserisco 20
print('e ora?')  quindi text_anni ha valore '20'
y = 'FINE'       mentre eta ha valore 20
print(y)          2020
                  40
                  <class 'str'>  <class 'int'>
                  si ferma in attesa della pressione di
                  Enter/Return
                  e ora?
                  FINE
```

Esercizi

```
name = input ("your name: ")
age = input ("your age: ")

year = input("your birth year: ")

future_age = int(year)+10

print("The age of ", name, "in 2033 will be ", future_age)
```

dov'è l'errore?

```
n = input()    # inserisce 3
x = n+'4'
y = int(n)+3
print(x, y)
```

cosa stampa?

```

name = input ("your name: ")
age = input ("your age: ")
year = input("your birth year: ")
future_age = int(year)+10
print("The age of ", name, "in 2033 will be ", future_age)

```

dov'è l'errore?

age non year !

```

n = input()    # inserisce 3
x = n+'4'
y = int(n)+3
print(x, y)    # stampa 34 e 6

```

Errori

- **syntax errors**: le regole sintattiche sono rigide, l'interprete **non riconosce le parole** del linguaggio.
 - `print("ciao) pront('ciao') a= 1 " 2`
 - **prima di iniziare ad eseguire** il programma, l'interprete controlla se ci sono errori di sintassi, e se ne trova non inizia nemmeno.
- **runtime errors** (*errore in esecuzione*): l'esecuzione è iniziata ma l'interprete trova un'**istruzione che non riesce ad eseguire correttamente**
 - `print(new_var)` **NameError: name 'new_var' is not defined**
 - `'cielo'+ 2` **TypeError: can only concatenate str (not "int") to str**
 - **l'esecuzione si interrompe** in questo punto
- **errori di semantica** (*o di logica*): hanno a che fare con il **senso** del programma. Il programma viene eseguito completamente ma **non fa la cosa giusta**.
 - Il programma esegue le istruzioni indicate, ma l'algoritmo/ la logica risolutiva è errata, oppure è disallineato rispetto alla specifica

Debugging: come trovare gli errori?

- leggere ed interpretare i messaggi d'errore dell'interprete.
Attenzione: a volte il problema non sta nella riga indicata nel messaggio, ma un po' prima
- riguardare linea per linea, immaginando l'effetto sul programma
- inserire in punti critici delle istruzioni `print`
 - per controllare il valore delle variabili in quel punto,
 - o per controllare che un certo punto del codice sorgente sia effettivamente raggiunto durante l'esecuzione
- debugger tools (es. Python Tutor)

Espressioni ed Istruzioni

- Un'**istruzione o comando** è una porzione di codice che la macchina **esegue** e che ha qualche **effetto**
- Un'**espressione** è una porzione di codice che la macchina **valuta**, cioè trova e restituisce il **valore** della stessa.

```
print('Calcolo:', 4+8)
print(4 >= 8)
print( not (4 >= 8) )
```

non è un errore, è
un'espressione di valore False



```
Calcolo: 12
False
True
```

l'interprete

1. **valuta i valori delle espressioni** passate come parametro alla `print`,
2. esegue la funzione `print` usando come **argomenti i valori calcolati**

Variabili: Valori e Assegnamento

Cosa stampa?

```
x = 10  
y = x + 1  
z = (x==8)  
s = 'ciao'  
print (z, s*2)
```

| | |
|---|--------|
| x | 10 |
| y | 11 |
| z | False |
| s | 'ciao' |

False ciaociao

l'istruzione di assegnamento

- è sempre della forma **variable = espressione**
- viene eseguita nel modo seguente:
 1. si **valuta l'espressione a destra** di = e si ottiene un **valore**
 2. questo valore viene **assegnato alla variabile a sinistra** di =

Esercizi - Soluzioni

3. Scrivere un'espressione booleana che ha valore **True** se e solo se la variabile x ha un valore che **sta** nell'intervallo di numeri [0,...,10]

(x >= 0) and (x<=10)

4. Scrivere un'espressione booleana che ha valore **True** se e solo se x **non sta** nell'intervallo di numeri [0,...,10]

(x<0) or (x>10) **not((x>=0) and (x<=10))**

5. Scrivere un'espressione booleana che coinvolge 3 variabili x,y,z e ha valore **True** se e solo se x **sta** nell'intervallo di numeri [y,...,z]

(x>=y) and (x<=z)

controllare la soluzione usando python

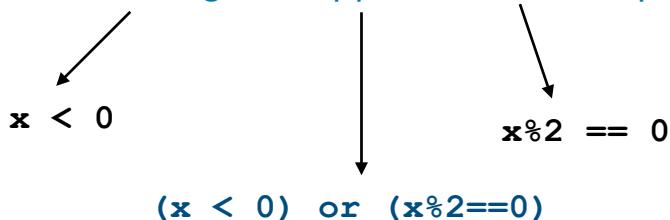
Esercizio

Scrivere un'espressione booleana che è **Falsa** se e solo se x è una variabile di valore un numero negativo oppure un numero pari

not (**(x<0) or (x%2==0)**)

(not(x<0)) and (not(x%2==0))

Scrivere un'espressione booleana che è **Vera** se e solo se x è una variabile di valore un numero negativo oppure un numero pari



controllare la soluzione usando python

Esercizio

Scrivere un'espressione booleana che è **Falsa** se e solo se x è una variabile di valore un numero negativo oppure un numero pari

not (**(x<0) or (x%2==0)**)

(x >= 0) and (x%2 != 0)

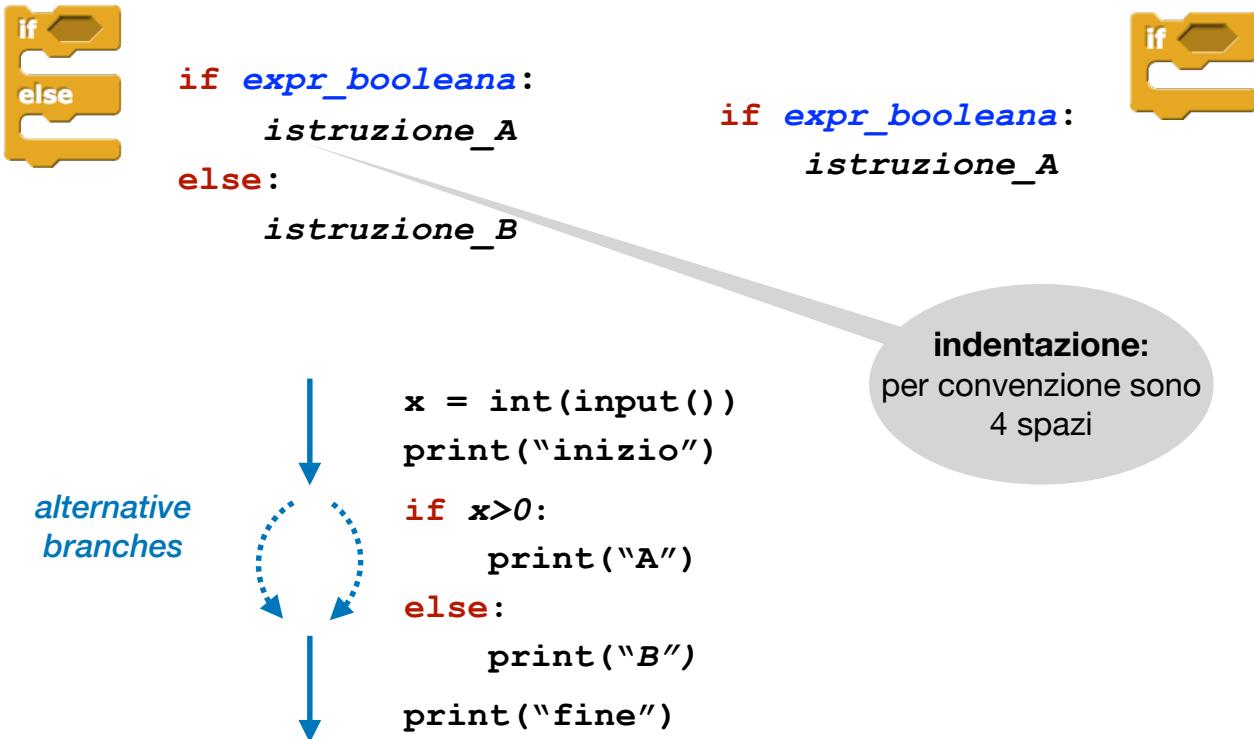
Esempio di programma per controllare la soluzione:

```
print('Due espressioni False se e solo se inserisco un  
numero negativo oppure pari')  
  
x = int(input('inserisci un numero: '))  
  
expr1 = not((x<0) or (x%2==0))  
expr2 = (x >=0) and (x%2 != 0)  
  
print('La prima expr booleana ha valore ', expr1)  
print('La seconda expr booleana ha valore ', expr2)
```

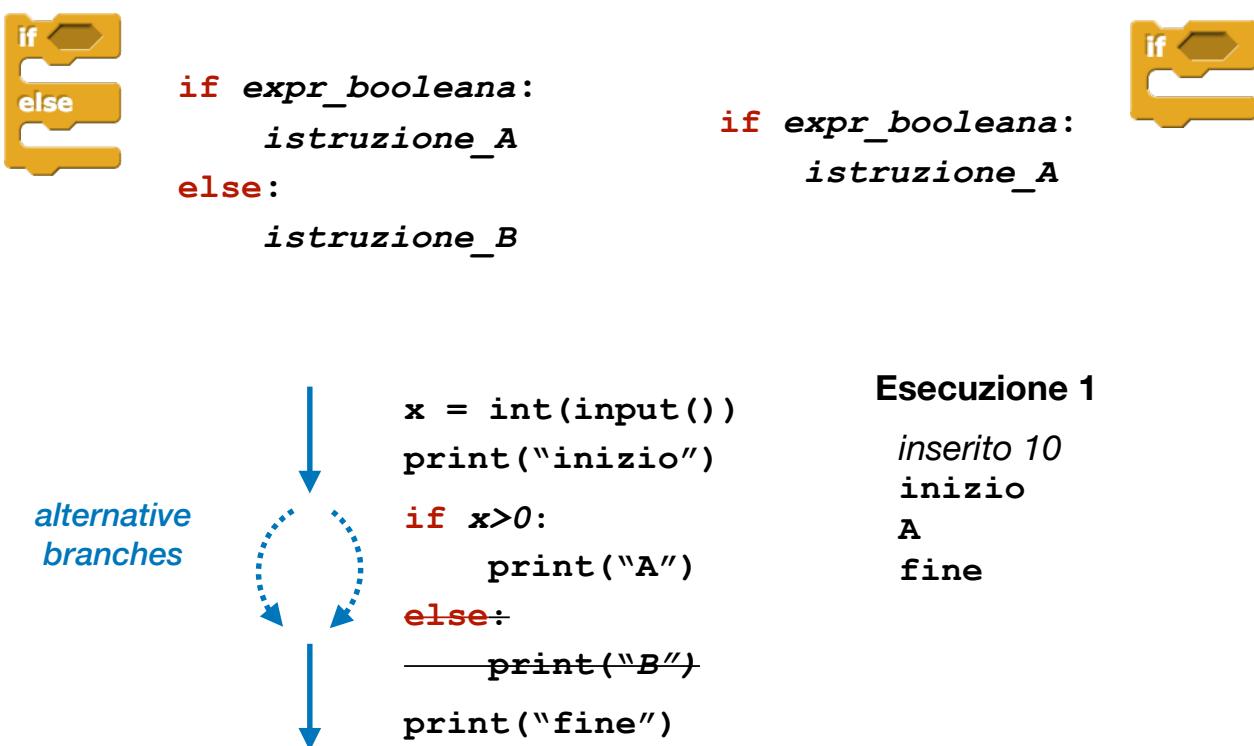
test con

- n. negativo e pari
- n. negativo e dispari
- n. positivo e pari
- n.positivo e dispari
- zero

Istruzioni condizionali



Istruzioni condizionali



Istruzioni condizionali



```
if expr booleana:  
    istruzione_A  
else:  
    istruzione_B
```



if expr booleana:

questa riga si esegue
sempre:

la condizione (expr booleana)
va valutata

alternative
branches

```
x = int(input())  
print("inizio")  
if x>0:  
    print("A")  
else:  
    print("B")  
print("fine")
```

Esecuzione 2

inserito -4
inizio
B
fine

che output produce questo programma?

```
1 a = 3  
2 b = 4  
3 if a+1==b:  
4     print("yes")  
5  
6 if b%2 == 0:  
7     print("b pari")  
8 else:  
9     print("b dispari")  
10  
11 print("buona giornata")
```

istruzione 1

istruzione 2

istruzione 3

yes

b pari

buona giornata

istruzione 4

istruzione 5

indentazione:
per convenzione sono
4 spazi

attenzione ai :
altrimenti è
errore di sintassi

che output produce questo programma?

```
1  a = 3                      istruzione 1
2  b = 4                      istruzione 2
3  if a+5==b:                 ]
4      print("yes")            istruzione 3
5
6  if  b%2 == 0:               ]
7      print("b pari")        b pari
8 else:                       buona giornata
9     print("b dispari")      ]
10
11 print("buona giornata")   istruzione 5
```

che output produce questo programma?

```
a = 3
b = a
if (a==b or a > b):
    print("pippo")
    print("hello")
else:
    print("pluto")
```

blocco di istruzioni:
identificato da
allineamento **indentato**

pippo
hello



```
if expr_booleana:
    blocco_istruzioni_A
else:
    blocco_istruzioni_B
```



```
if expr_booleana:
    blocco_istruzioni
```

Esempio

- Scrivere un programma che chiede di inserire due numeri e stampa il più grande dei due.

```
x = int(input("primo int:"))
y = int(input("secondo int:"))

if x > y:
    print(x)
else:
    print(y)
```

Se sono inseriti due numeri uguali
stampa uno dei due... quale?

Esempio

- **Specifiche:** scrivere un programma che chiede di inserire due numeri e stampa il più grande dei due. Se sono uguali il programma deve stampare la stringa uguali

```
x = int(input("primo int:"))
y = int(input("secondo int:"))
```

```
if x==y:
    print('uguali')

if x > y:
    print(x)
else:
    print(y)
```

se inserisco due numeri uguali,
stampa la stringa uguali ma
ANCHE il numero

```
x = int(input("primo int:"))
y = int(input("secondo int:"))
```

```
if x==y:
    print('uguali')
else:
    if x > y:
        print(x)
    else:
        print(y)
```

se inserisco due numeri uguali,
stampa SOLO la stringa uguali

Esempio

- **Specifiche:** scrivere un programma che chiede di inserire due numeri e stampa il più grande dei due. Se sono uguali il programma deve stampare la stringa uguali

```
x = int(input("primo int:"))
y = int(input("secondo int:"))
```

```
if x > y:
    print(x)
else:
    if x == y:
        print('uguali')
    else:
        print(y)
```

come si comporta?
che istruzioni sono?

tracciare il flusso di controllo di
qualche esecuzione

Esempio

- **Specifiche:** scrivere un programma che chiede di inserire due numeri e stampa il più grande dei due. Se sono uguali il programma deve stampare la stringa uguali

```
x = int(input("primo int:"))
y = int(input("secondo int:"))
```

```
if x > y:
    print(x)
```

```
if y > x:
    print(y)
```

```
if x==y:
    print('uguali')
```

```
x = int(input("primo int:"))
y = int(input("secondo int:"))
```

```
if x > y:
    print(x)
```

```
if y > x:
    print(y)
```

```
else:
    print('uguali')
```

come si comportano ?
sono programmi corretti ? come lo verifico?

Esempio

Che differenza c'è tra questi due programmi?

- confrontare il comportamento quando si inseriscono due numeri uguali
- confrontare il comportamento quando si inseriscono due numeri diversi

```
x = int(input("primo int:"))
y = int(input("secondo int:"))

if x==y:
    print('uguali')

if x > y:
    print(x)
else:
    print(y)
```

Fare attenzione
all'indentazione

```
x = int(input("primo int:"))
y = int(input("secondo int:"))

if x==y:
    print('uguali')

    if x > y:
        print(x)
    else:
        print(y)
```

se inseriti due numeri
diversi non fa nulla

che output produce questo programma?

```
a = 3
b = a
if (a==b or a > b):
    print("pippo")
    print("hello")
    if b%2==0:
        print("pari")
    else:
        print("dispari")
else:
    print("pluto")
```

pippo
hello
dispari

in un blocco di istruzioni si può
inserire qualsiasi istruzione valida,
anche un'altra istruzione if-else

condizioni innestate

```
if x == y:  
    print("x e y sono uguali")  
else:  
    if x < y:  
        print("x minore di y")  
    else:  
        print("x maggiore di y")
```

equivalente a questa istruzione

```
if x < y:  
    print("x minore di y")  
elif x > y:  
    print("x maggiore di y")  
else:  
    print("x e y sono uguali")
```

più di due rami (chained conditionals)

```
scelta = input('scegli a,b o c')  
if scelta == 'a':  
    print('prima scelta')  
elif scelta=='b':  
    print('seconda scelta')  
elif scelta=='c':  
    print('terza scelta')  
else:  
    print('scelta errata')
```

le condizioni vengono **controllate in ordine**:

- se la prima è falsa, viene controllata la seconda e così via
- appena una condizione è vera, si esegue il ramo corrispondente e l'istruzione termina
- ci possono essere tanti rami **elif** (else if) e al più un ramo **else** alla fine
- se ci sono più condizioni vere, viene eseguita sempre solo la prima

Esercizi

1. Scrivere un programma che legge da input 3 interi e scrive in output se sono tutti e tre uguali, oppure se sono ordinati in ordine crescente, oppure se sono ordinati in ordine decrescente, oppure se non è vero nessuno dei casi precedenti.
2. Scrivere un programma che legge da input 4 interi, li memorizza in 4 variabili, e stampa la loro somma, la media, il valore minimo e il massimo.
3. Dati tre bastoncini, non sempre è possibile riuscire a sistemarli in modo da formare un triangolo. Ad esempio se uno è lungo 12cm e gli altri due sono lunghi 1cm non si riesce a disporli a triangolo. Esiste la seguente regola:
date 3 lunghezze, se una qualsiasi è maggiore della somma delle altre due, allora non è possibile formare un triangolo che abbia per lati quelle lunghezze
Scrivere un programma che dati in input tre interi, controlla se possono essere le lunghezze dei lati di un triangolo.

Imparare a leggere il codice

- **Descrivere a parole** *come si comporta* il seguente codice

```
x = int(input('Inserisci un numero: '))
y = int(input('Inserisci un numero: '))
z = int(input('Inserisci un numero: '))

if x<y and y<z:
    print('GIALLO')
else:
    if x>y and y>z:
        print('BLU')
    else:
        print('ROSSO')
```

- **Descrivere a parole** *la specifica* del seguente codice

Il programma prende in input 3 numeri, e

- se *sono 3 numeri crescenti* stampa GIALLO
- se *sono 3 numeri decrescenti* stampa BLU
- in *ogni altro caso* stampa ROSSO

Esercizio

Imparare a leggere il codice

- Descrivere a parole **la specifica** del seguente codice

```
x = int(input('Inserisci un numero: '))
y = int(input('Inserisci un numero: '))
z = int(input('Inserisci un numero: '))

if x%2!=0:
    if y%2!=0 and z%2!=0:
        print('GIALLO')
    else:
        print('ROSSO')
else:
    if y%2==0 and z%2==0:
        print('BLU')
    else:
        print('ROSSO')
```

Esercizio

- Descrivere a parole **la specifica** del seguente codice, indicando in quali casi stampa Rosso, in quali casi stampa Blu, e in quali casi stampa Giallo

```
n = int(input('Inserisci un numero '))
if n > 5:
    print("Rosso")
else:
    if n > 10:
        print("Blu")
    else:
        print("Giallo")
```

- Descrivere a parole **la specifica** del seguente codice, indicando in quali casi stampa Rosso, in quali casi stampa Blu, e in quali casi stampa Giallo

```
n = int(input('Inserisci un numero '))
if n > 5:
    print("Rosso")
else:
    if n < 5:
        print("Blu")
    else:
        print("Giallo")
```

Esercizio: Anni bisestili

Un anno è **bisestile** (*leap* in inglese) se:

- è divisibile per 4, ma non anche divisibile per 100
(es. 1996 e 2020 non 1900)
- con un'eccezione: se è divisibile per 400, allora è bisestile
(es. anno 2000)

Si considerino i seguenti programmi Python, che stampano **anno bisestile** se la variabile **year** ha un valore che corrisponde ad un anno bisestile, altrimenti stampano **anno normale**.

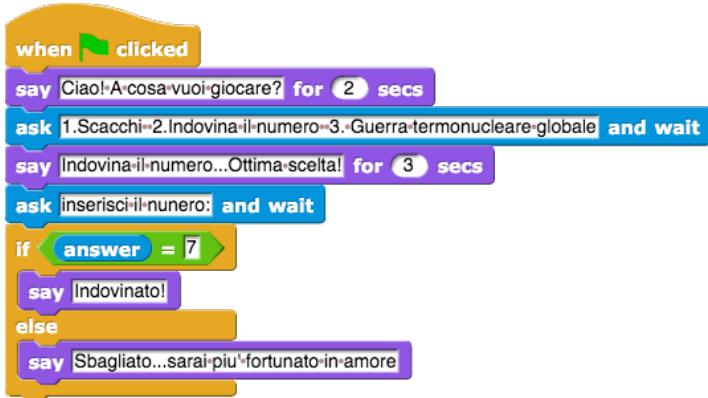
Sono tutti programmi **corretti**, che hanno la **stessa specifica**.

```
if year%400 ==0 or (year%4==0 and (not year%100==0)):  
    print('anno bisestile')  
else:  
    print('anno normale')
```

```
if year%400 ==0:  
    print('anno bisestile')  
else:  
    if year%100==0:  
        print('anno normale')  
    else:  
        if year%4==0:  
            print('anno bisestile')  
        else:  
            print('anno normale')
```

```
if year%4 ==0:  
    if not year%100==0:  
        print('anno bisestile')  
    else:  
        if year%400==0:  
            print('anno bisestile')  
        else:  
            print('anno normale')  
else:  
    print('anno normale')
```

Discutere **quale versione** è la più **leggibile**, quella più **elegante**, quella più **facile da ricordare**, e quella in cui è più **facile individuare errori**.



Esercizio:

modificare il programma in modo che tenga
conto di quale gioco ha scelto l'utente
....con creatività

```
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 piu fortunato in amore')
```

Esercizio

- Scrivere un programma che calcola e stampa quanti secondi ci sono in 3 ore e quanti in 12 ore.
- Scrivere un programma che calcola e stampa quanti giorni sono trascorsi dal 1 gennaio 2023 ad oggi 30 novembre 2023.

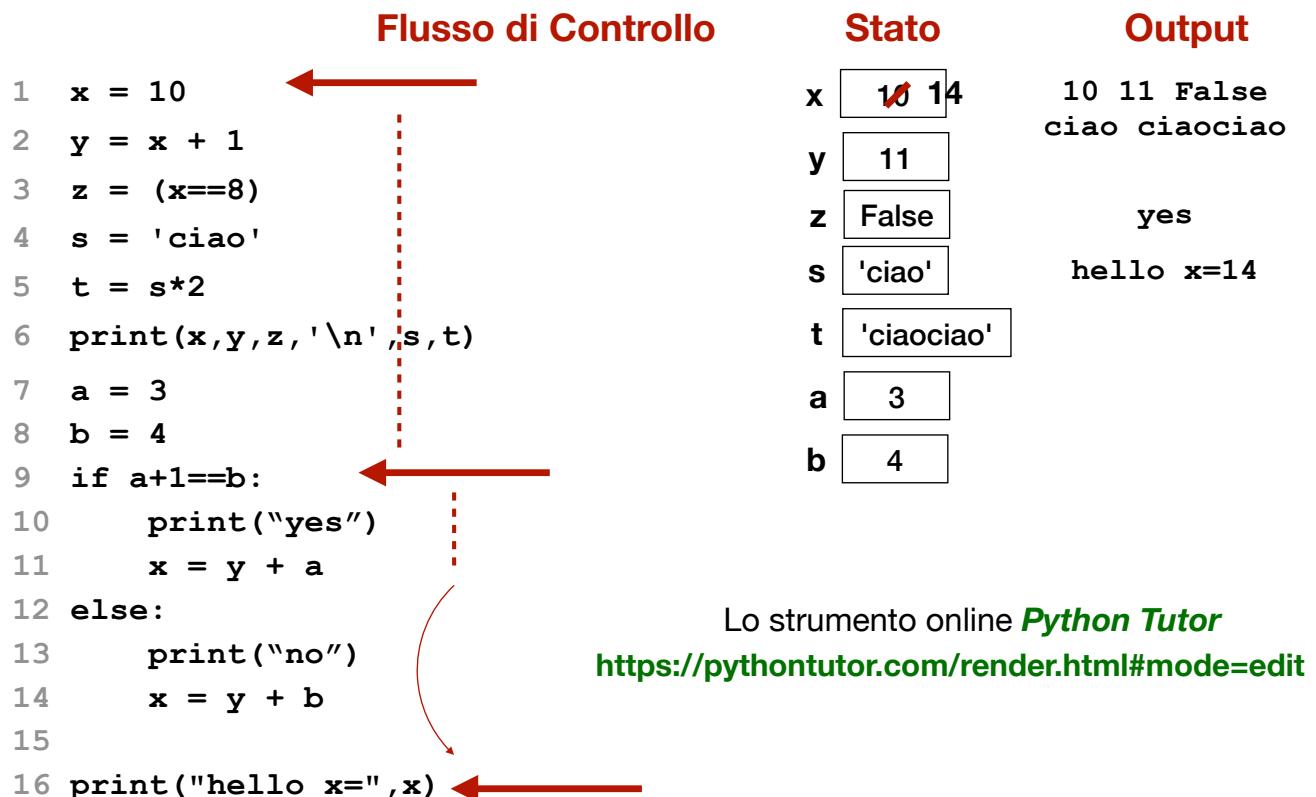
ci sono tante diverse soluzioni,
provate a confrontarle

Esecuzione di un programma

Per tracciare con precisione **cosa accade** durante l'esecuzione di un programma, serve tenere traccia di 3 cose:



Esecuzione di un programma



cicli e iterazione

- **ripetere** un gruppo di istruzioni un certo numero di volte:

```
for variabile in elenco:  
    blocco_istruzioni # body
```

la variabile **itera** sui valori nell'elenco e
per ogni valore esegue il corpo del for



```
for i in [1,2,3,4,5]:  
    print("Stay hungry,")  
    print(" stay foolish!")
```

5 ripetizioni

| | |
|-------------------------|-------------------------------|
| body con <i>i</i> =1 | Stay hungry, stay foolish! |
| body con <i>i</i> =2 | Stay hungry, stay foolish! |
| body con <i>i</i> =3 | Stay hungry, stay foolish! |
| body con <i>i</i> =4 | Stay hungry, stay foolish! |
| body con <i>i</i> =5 | Stay hungry, stay foolish! |

cicli e iterazione

- **ripetere** un gruppo di istruzioni un certo numero di volte:

```
for variabile in elenco:  
    blocco_istruzioni # body
```

la variabile **itera** sui valori nell'elenco e
per ogni valore esegue il corpo del for



```
for i in range(5):  
    print("Stay hungry,")  
    print(" stay foolish!")
```

[0,5) =[0,1,2,3,4]

5 ripetizioni

| | |
|-------------------------|-------------------------------|
| body con <i>i</i> =0 | Stay hungry, stay foolish! |
| body con <i>i</i> =1 | Stay hungry, stay foolish! |
| body con <i>i</i> =2 | Stay hungry, stay foolish! |
| body con <i>i</i> =3 | Stay hungry, stay foolish! |
| body con <i>i</i> =4 | Stay hungry, stay foolish! |

cicli e iterazione

- **ripetere** un gruppo di istruzioni un certo numero di volte:

```
for variabile in elenco:  
    blocco_istruzioni # body
```

la variabile **itera** sui valori nell'elenco e
per ogni valore esegue il corpo del for

[0,5) =[0,1,2,3,4]

```
for i in range(5):  
    print("iteration number:", i)  
    print("Hello!")  
print("fine")
```

5 ripetizioni

```
body  
con i=0 [iteration number:0  
Hello!  
body  
con i=1 [iteration number:1  
Hello!  
body  
con i=2 [iteration number:2  
Hello!  
body  
con i=3 [iteration number:3  
Hello!  
body  
con i=4 [iteration number:4  
Hello!  
fine
```

Esercizio

- Come si comportano i seguenti programmi?

```
for i in range(5):  
    print("iteration:",i)  
    print("Hello!")  
print("fine")
```

```
for i in range(5):  
    print("iteration:",i)  
    print("Hello!")  
    print("fine")
```

fare molta attenzione
all'indentazione.

Questi errori (logici) sono
difficili da trovare

cicli e iterazioni

In generale la funzione `range` ha 3 argomenti:

`range(begin,end,step)` = *numeri da begin a end escluso, di step in step*

```
for i in range(1,8,2):  
    print("hello iteration:" i)
```

numeri da 1 a 8 escluso
di 2 in 2:
[1,3,5,7]

```
for i in range(6,2,-1):  
    print("hello iteration:" i)
```

numeri da 6 a 2 escluso
di -1 in -1
[6,5,4,3]

`range(5)` sta per `range(begin=0,end=5,step=1)` = [0,1,2,3,4]

Esempio

Sapendo che `range(10,0,-1)` corrisponde all'elenco [10,9,8,7,...,2,1]
stampare un conto alla rovescia

```
print("Countdown:")  
for i in range(10,0,-1):  
    print(i)  
print("Ignition!")
```

Countdown:
10
9
8
7
6
5
4
3
2
1
Ignition!



cicli e iterazione

- **ripetere** un gruppo di istruzioni un certo numero di volte:

```
for variabile in elenco:  
    blocco_istruzioni # body
```

qualsiasi elenco, non solo elenco di numeri

```
for lettera in "banana":  
    print(lettera)
```

una stringa è un elenco

di caratteri

```
b ← body con lettera='b'  
a ← body con lettera='a'  
n .....  
a  
n  
a
```

```
frutto = input("scegli un frutto")  
for lettera in frutto:  
    print(lettera*2)
```

in Python ci sono tanti tipi di elenchi:
range(...), stringhe, liste, dizionari, tuple...

cicli e iterazione

- **ripetere** un gruppo di istruzioni un certo numero di volte:

```
for variabile in elenco:  
    blocco_istruzioni # body
```

qualsiasi istruzione:
es. assegnamenti, condizionali,
anche altri for

```
for i in range(10):  
    if i%2==0:  
        print(i)
```

stampa le cifre pari 0 2 4 6 8

Esercizi:

1. Scrivi un programma che usa il comando for per stampare le cifre dispari
2. completa il codice seguente in modo che stampi i primi 10 numeri pari

```
for i in range(???) :  
    print(i)
```

cicli e iterazione

```
i="cane"  
for i in ["cane","gatto","oca"]:  
    for j in range(1,4):  
        print(i, j, end='')  
    print()  
  
j=1 stampa i,j cane 1  
j=2 stampa i,j cane 2  
j=3 stampa i,j cane 3  
print() va a capo  
  
i ="gatto"  
  
j=1 stampa i,j gatto 1  
j=2 stampa i,j gatto 2  
j=3 stampa i,j gatto 3  
print() va a capo  
  
i ="oca"  
  
j=1 stampa i,j oca 1  
j=2 stampa i,j oca 2  
j=3 stampa i,j oca 3  
print() va a capo
```

cane 1 cane 2 cane 3
gatto 1 gatto 2 gatto 3
oca 1 oca 2 oca 3

cicli e iterazione

```
for i in ["Mary's", "Bob's","Joe's"]:  
    for j in ["cat","dog","duck"]:  
        print(i, j, end='')  
    print()
```

Cosa stampa ???

ESERCIZIO: modificare il programma in modo da rendere più leggibile l'output stampando delle virgolette tra gli animali dello stesso padrone. Fare altre modifiche in modo da rendere più simpatico l'output

cicli e iterazione

```
for i in range(1,4):
    for j in range(1,4):
        print(i*j, end=' ')
print()
```

Cosa stampa?

....sono le tabelline!

```
i=1
j=1 stampa 1*1
j=2 stampa 1*2
j=3 stampa 1*3
print()

i=2
j=1 stampa 2*1
j=2 stampa 2*2
j=3 stampa 2*3
print()

i=3
j=1 stampa 3*1
j=2 stampa 3*2
j=3 stampa 3*3
```

Esempio

```
elenco = [True, False] #lista di 2 valori
for a in elenco:
    print("a=", a , " not a=", not a)
```

stampe
a= True not a= False
a= False not a= True

```
elenco = [True, False]
for a in elenco:
    for b in elenco:
        print("a=", a, "b=", b)
```

| iterazioni | stampe |
|------------|-----------------|
| a=True | |
| b=True | a=True b=True |
| b=False | a=True b=False |
| a=False | |
| b=True | a=False b=True |
| b=False | a=False b=False |

```
elenco = [True, False]
for a in elenco:
    for b in elenco:
        print(a,"or", b,"=", a or b)
```

stampe
True or True = True
True or False = True
False or True = True
False or False = False

cicli e iterazione

Descrivere cosa fa questo programma

```
num_input = input("Quanti numeri vuoi inserire?")
n = int(num_input)
sum = 0
for i in range(n):
    msg = "inserisci il prossimo numero "
    x = int(input(msg))
    sum = sum + x
print("somma: ",sum)
```

cosa fa:

specifica del problema → • calcola la somma dei numeri inseriti dall'utente

come lo fa:

- chiede all'utente quanti numeri vuole inserire, poi ripetutamente chiede di inserire il prossimo numero e mantiene la somma dei numeri inseriti finora. Termina quando sono stati inseriti tutti i numeri previsti e ne stampa la somma totale

**descrizione della
soluzione/implementazione**

cicli e iterazione

Descrivere cosa fa questo programma

```
import random

n = random.randint(1,9) # restituisce un numero a caso in [1,9]
sum = 0
for i in range(n):
    msg = "inserisci il prossimo numero "
    x = int(input(msg))
    sum = sum + x
print("somma: ",sum)
```

Esercizio

Descrivere cosa fa questo programma

```
num_input = input("Quanti numeri vuoi inserire?")
n = int(num_input)
sum = 0
for i in range(n):
    msg = "inserisci il numero " + str(i+1) + " di " + str(n)
    x = int(input(msg))
    sum = sum + x
print("somma: ",sum)
```

cicli e iterazione

Descrivere cosa fa questo programma

```
import random
n = random.randint(1,100)
sum=0
for i in range(n):
    msg = "inserisci il prossimo numero oppure -1 per smettere "
    x = int(input(msg))
    if x == -1:
        break # interrompe l'iterazione e il flusso di controllo
              # salta all'istruzione istruzione che segue il for
    else:
        sum = sum + x
print("somma: ",sum)
```

Esercizio

Scrivere un programma che prende in input un numero n, legge n numeri e calcola e restituisce la somma e la media dei numeri, **tra quelli inseriti, che sono pari e multipli di 3**

```
n = int(input("Quanti numeri vuoi inserire?"))
sum = 0
n_da_considerare = 0

#leggi n numeri
for i in range(n):
    new_num=int(input("inserisci il prossimo:"))
    # controlla se pari e multiplo di 3
    if new_num%2==0 and new_num%3==0:
        sum = sum + new_num
        n_da_considerare = n_da_considerare +1

print("somma:", sum)
print("media:", sum/n_da_considerare)
```

Esercizio:

implementare gli algoritmi scritti in pseudocodice in un software python

- Algoritmo per un programma che verifica se un numero è pari o dispari
- Algoritmo per un programma che verifica se un utente è maggiorenne
- Algoritmo per un programma che verifica se una parola è **palindroma**
- Scrivere un programma che, dato un elenco di prezzi di articoli, dice il totale della spesa
- Scrivere un programma che calcola la media degli esami

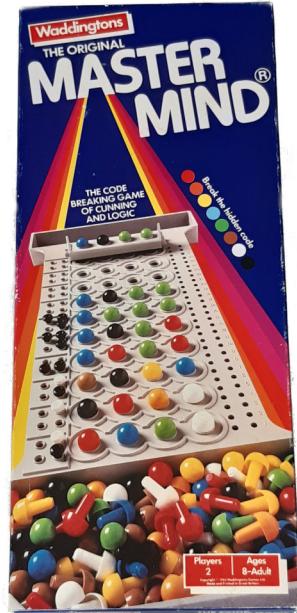
cosa riusciamo ad implementare e cosa non sappiamo fare?

Esercizio del calendario

- **Generalizzare il programma** chiedendo in input una data di quest'anno (giorno e mese) e producendo in output il numero di giorni dal 1 gennaio 2023 a quella data (non facile)

Il gioco Mastermind prevede 2 giocatori:

- il giocatore 1 sceglie una sequenza nascosta di **4** pioli colorati, scegliendo tra **6 colori disponibili** (arancio, blu, nero, verde, giallo, rosso);
- il giocatore 2 deve indovinare la sequenza nascosta. Ad ogni tentativo, propone una sequenza di 4 pioli colorati e la sottopone al controllo;
- controllare una sequenza significa dare due informazioni:
 - quanti pioli ci sono del colore giusto al posto giusto
 - quanti pioli ci sono del colore giusto al posto sbagliato
- in ogni momento il giocatore 2 può vedere tutti i suoi tentativi precedenti con le relative informazioni di controllo
- il gioco può prevedere un numero massimo di tentativi



Esercizio (difficile ma da fare pian piano prima dell'esame)

Scrivere un programma che permette a due giocatori di giocare a Mastermind

N.B. **non significa** che il programma indovina automaticamente la soluzione, ma solo che permette di giocare

- Pensare al funzionamento logico del programma: cosa deve fare? come lo fa?
- Provare a scrivere in qualche modo questo funzionamento

cicli e iterazione: while

- non sempre si sa quante volte ripetere il ciclo. Ad esempio si deve ripetere **finché è vera/falsa una certa condizione**
- Il costrutto più generale per l'iterazione è **while**

```
while condizione:  
    blocco_istruzioni #body
```

1. si valuta la condizione, che deve essere un'espressione booleana
2. **se la condizione è vera**,
 - si esegue il corpo del ciclo e **poi si torna al passo 1.**
3. **se la condizione è falsa**
 - si **esce dal ciclo** e si esegue l'istruzione che segue il while.

- in generale, le istruzioni del corpo devono cambiare il valore di verità della condizione, altrimenti il ciclo si ripete all'infinito (**loop forever**)

while e for

```
# Countdown  
for n in range(10,0,-1):  
    print(n)  
print("Ignition!")
```

```
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
Ignition!
```

```
# Countdown  
n = 10  
while n > 0:  
    print(n)  
    n = n-1  
print("Ignition!")
```

se il corpo di
while non modifica n,
la condizione sarà
sempre vera

cicli e iterazione: while

- non sempre si sa quante volte ripetere il ciclo. Ad esempio si deve ripetere finché è vera/falsa una certa condizione
- Il costrutto più generale per l'iterazione è **while**

```
while condizione:  
    blocco_istruzioni #body
```

```
line = input()  
while line != "stop":  
    print("hai inserito", line)  
    line = input()  
print("finito")
```

Ripete quello che inserisce l'utente finché l'utente non inserisce "stop"

Esempio: indovina il numero

Scrivere un programma che chiede all'utente di indovinare un numero, scelto a caso tra 1 a 9. L'utente è invitato a ritentare finché non indovina il numero

```
import random  
  
segreto = random.randint(1,9) # un numero a caso in [1,9]  
tenta_ancora = True  
  
while tenta_ancora :  
    n = int(input("Inserisci un numero tra 1 e 9: "))  
    if n == segreto:  
        tenta_ancora = False  
    else:  
        tenta_ancora = True  
  
print("Indovinato: ",secreto)
```

Esercizio: indovina il numero

Osserva le differenze tra il codice seguente e quello della slide precedente, e rendersi conto che sono due programmi equivalenti (il ramo else era inutile).

```
import random
segreto = random.randint(1,9) # un numero a caso in [1,9]
tenta = True
while tenta:
    n = int(input("Inserisci un numero tra 1 e 9: "))
    if n == segreto:
        tenta = False
print("Indovinato: ",secreto)
```

Esercizio: Arricchire il comportamento del programma nei modi seguenti

1. Se il tentativo è errato, avvisa l'utente dell'errore prima di chiedere il nuovo numero.
2. Se il tentativo è errato, avvisa se il numero segreto è maggiore o minore di quello inserito.
3. Invece di scegliere un numero a caso in [1,9], il programma inizia chiedendo all'utente un numero, da memorizzare nella variabile `maximum`, che poi usa per scegliere un numero a caso in [1,`maximum`]
4. Tracciare il numero di tentativi errati effettuati dall'utente prima di indovinare il numero, e stampare il numero di tentativi quando alla fine l'utente indovina.

Esercizio

Scrivere un programma che **ripetutamente stampa il doppio del numero inserito** dall'utente. Il programma deve **terminare quando viene inserito il numero 0**, e produrre una stampa finale che indica il numero di numeri inseriti in totale dall'utente.

Esercizio

Scrivere un programma che usa il costrutto while per scorrere i numeri e stampare i primi 5 numeri pari.

```
n_pari = 0
i = 0
while n_pari < 5:
    if i%2==0:
        print(i)
        n_pari = n_pari +1
    i = i+1 # che succede se questa riga viene cancellata?
```

Completare il programma seguente in modo che stampi solo i primi 5 numeri pari.

```
n_pari = 0
for i in range(30):
    ???
```

while

```
line = input()
while line != "end":
    print(line)
    line = input()
print("finito")
```

Ripete quello che inserisce l'utente finché l'utente non inserisce "end"

Le seguenti versioni sono **equivalenti**:

```
ancora = True
while ancora:
    line = input()
    if line != "end":
        print(line)
    else:
        ancora = False
print("finito")
```

```
while True:
    line = input()
    if line != "end":
        print(line)
    else:
        break
print("finito")
```

```
while True:
    line = input()
    if line == "end":
        break
    print(line)
print("finito")
```

Esercizio

```
n = int(input("insert number"))
if n < 5:
    while not (n==5):
        print(n)
        n = n+1
print("Finished")
```

Per quali valori di input produce la stampa Finished?

- solo gli interi minori di 5
- solo 5
- solo gli interi maggiori o uguali a 5
- qualsiasi intero

Cosa stampa se l'utente inserisce 1?

- 1 2 3 4 Finished
- 4
- 1 2 3 4 5 Finished
- 5 Finished

Cosa accade se l'utente inserisce 9?

- stampa 9 10 11 12 13... loop infinito di numeri
- stampa 9 8 7 6 Finished
- stampa Finished
- stampa 9 Finished

Esercizio

1. Scrivere un programma che saluta l'utente stampando una citazione presa a caso tra un gruppetto predefinito di citazioni (sugg. rappresentare una citazione come una stringa)
2. Modificare il programma precedente facendo in modo che venga scritta un'altra citazione ogni volta che l'utente preme il tasto Enter/Return. Il programma termina quando l'utente inserisce la parola 'exit'

Esercizio

- Tracciare il flusso di controllo del seguente programma:

```
1 n = 5
2 while n > 0:
3     print(n)
4     n = n-1
5 print("Ignition!")
```

- Tracciare il flusso di controllo dei seguenti due programmi:

```
1 n_pari = 0
2 i = 0
3 while n_pari < 5:
4     if i%2==0:
5         print(i)
6         n_pari = n_pari +1
7     i = i+1
```

```
1 n_pari = 0
2 i = 1
3 while n_pari < 5:
4     if i%2==0:
5         print(i)
6         n_pari = n_pari +1
```

quiz

```
x = True
print(x or not x)

stampa True o False?
```

```
name = 'John'
age = 24
print(name == 'John' or age==30)

stampa True o False?
```

```
x = True
print(x and not x)

stampa True o False?
```

```
is_hungry = True
is_happy = False
print (is_hungry or (not is_hungry and is_happy))

stampa True o False?
```

quiz

```
in_california = True
is_raining = False
print((in_california and is_raining) or
      (in_california or  ((is_raining and in_california) and
                           (not is_raining))))
```

stampa True o False?

```
x=4
y=4
z=5
print(x == y and z==y)
```

stampa True o False?

```
x=4
y=4
z=5
print(x == y or x==z)
```

stampa True o False?

esempi

I seguenti programmi **terminano?** ...sempre, mai o qualche volta?

```
n = int(input())
while n > 0:
    print(n)
    n = n - 2
```

ok

```
n = int(input())
while n != 0:
    print(n)
    n = n - 2
```

se n è **negativo** loop
se n è pari OK
se n è **dispari** loop

```
n = int(input())
while n != 1:
    print(n)
    if n%2 == 0: # n e' pari
        n = n/2
```

n=8 ok 8 4 2 1
n=7 **loop** 7 7 7 ...
n=24 **loop** 24 12 6 3 3...

esercizi

I seguenti programmi **terminano?** ...sempre, mai o qualche volta?

```
n = int(input())
while n != 1:
    print(n)
    if n%2 == 0: # n e' pari
        n = n/2
    else: # n e' dispari
        n = (n+1)/2
```

si può dimostrare che
 $\forall n > 0$ termina

```
n = int(input())
while n != 1:
    print(n)
    if n%2 == 0: # n e' pari
        n = n/2
    else: # n e' dispari
        n = n*3+1
```

$\forall n > 0$ termina ?

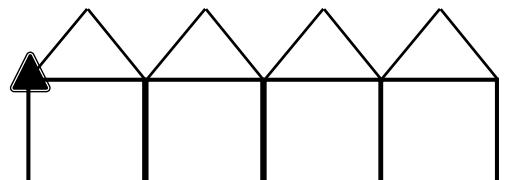
I matematici credono di sì, ma nessuno
è mai riuscito a **dimostrarlo!**

Congettura di Collatz

- La congettura è stata verificata mediante computer (giorni e giorni, rete di computer) per tutti i valori **fino a 2^{68}** circa 10^{20} . (<http://www.ericr.nl/wondrous/>)
- Questi test non potranno mai dimostrare la correttezza della congettura, ma solo l'**eventuale falsità**.

Funzioni

```
# 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

dall'algoritmo
all'implementazione



è ben diverso leggere un programma
dall'alto in basso oppure seguendo il
flusso di esecuzione!

Funzioni

intestazione: **nome** e **lista parametri**

```
1 def incipit():
2     print('Tanto tempo fa')
3     print('in una galassia')
4     print('lontana, lontana ...')
5
6 incipit()           ← chiamata di funzione
7 print('Non ho capito')
8 incipit()
```

corpo: blocco di istruzioni

definizione di funzione

- Una **funzione** (*procedura*) è una serie di istruzioni a cui viene assegnato un nome.
- Il corpo della funzione contiene delle istruzioni che **vengono eseguite solo quando** (e se) la funzione viene **chiamata/invocata** mediante il suo nome

Una funzione deve essere definita prima di essere chiamata (*Esercizio: provare e leggere il messaggio di errore*)

Funzioni e flusso di controllo

```
1 def incipit():
2     print('Tanto tempo fa')
3     print('in una galassia')
4     print('lontana, lontana ...')
5
6 incipit()
7 print('--Non ho capito')
8 incipit()
```

istruzioni eseguite:
6, 2,3,4, 7,8 ,2,3,4

Tanto tempo fa
in una galassia
lontana, lontana ...
--Non ho capito
Tanto tempo fa
in una galassia
lontana, lontana ...

- L'esecuzione inizia sempre dalla prima istruzione/comando, procedendo un'istruzione per volta dall'alto verso il basso.
- Quando viene **chiamata una funzione**, il flusso di controllo anziché proseguire con l'istruzione successiva, **salta nel corpo della funzione chiamata**, ne esegue le istruzioni, e infine **riprende il percorso dal punto che aveva lasciato**

Funzioni e flusso di controllo

```
1 def incipit():
2     print('Tanto tempo fa')
3     print('in una galassia')
4     print('lontana, lontana ...')
5
6 def duplica():
7     incipit()
8     incipit()
9
10 duplica()
11 print('Bye Bye')
```

istruzioni eseguite:

Tanto tempo fa
in una galassia
lontana, lontana ...
Tanto tempo fa
in una galassia
lontana, lontana ...
Bye Bye

- L'esecuzione inizia sempre dalla prima istruzione/comando, procedendo un'istruzione per volta dall'alto verso il basso.
 - Quando viene **chiamata una funzione**, il flusso di controllo anziché proseguire con l'istruzione successiva, **salta nel corpo della funzione chiamata**, ne esegue le istruzioni, e infine **riprende il percorso dal punto che aveva lasciato**

Funzioni

- La funzione può avere dei **parametri**, allora quando la si invoca bisogna passarle degli **argomenti**, uno per ogni parametro.
 - La funzione, come ultima cosa, può restituire un **valore di ritorno**.

`print('Hello', 3)` **chiama** la funzione con **2 argomenti**,
nessun valore di ritorno

`msg = input()` chiama la funzione senza argomenti, ritorna una stringa, usata per inizializzare la variabile `msg`

```
# procedura per "disegnare" un quadrato
def draw_triangle(dim):
    print('disegna triangolo di lato',dim)
```

definisce la funzione di nome `draw_triangle` con 1 parametro e nessun valore di ritorno
(quindi va chiamata con 1 argomento)

Funzioni e parametri

```
1 def stampa2volte(s):
2     print(s)
3     print(s)
```

parametro (è una variabile che si usa nel corpo della funzione)

```
4
5
6 stampa2volte('Tony')
7 stampa2volte('Bruce')
```

argomento

2 parametri -- 2 argomenti

```
1 def stampa_coppia(s1,s2):
2     print(s1,'loves', s2 )
3
4 stampa_coppia('Tony','Pepper')
5 stampa_coppia('Bruce','Betty')
```

Tony loves Pepper
Bruce loves Betty

Funzioni e parametri

```
def stampa2volte(s):
    print(s)
    print(s)
```

parametro

```
stampa2volte('pippo')
```

pippo
pippo

```
stampa2volte('bello'*3)
```

bellobellobello
bellobellobello

```
msg = 'Tony'
stampa2volte(msg)
```

gli argomenti sono **espressioni**:

- se ne **valuta** il valore prima della chiamata
- i parametri sono **inizializzati** con questo valore
- analogo se **si passa una variabile come argomento**
- la variabile passata `msg` "non esiste" nel corpo della funzione, che usa invece solo `s`

```

# 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

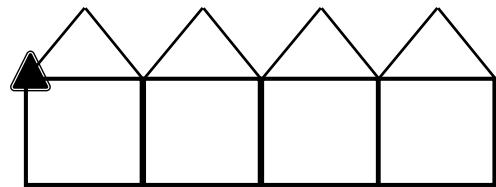
# procedura per disegnare una casa
def draw_house(dim):
    ← draw_square(dim)
    draw_triangle(dim)

# 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()

```



invoca la funzione, quindi ne esegue il corpo passando come argomento il valore 50

esegue le istruzioni
draw_square(50)
draw_triangle(50)
che sono altre due chiamate di funzioni

invoca la funzione, quindi ne esegue il corpo

valori di ritorno

```

def calcola_area(raggio):
    a = 3.14 * (raggio**2)
    return a
    valore:
    1256
x = calcola_area(20)
print("area cerchio di raggio 20:", x)

```

la funzione **termina** e restituisce un **valore**

```

r = int(input("Che raggio vuoi?"))
y = calcola_area(r)
print("L'area del cerchio di raggio", r, "è", y)

r = int(input("Che raggio vuoi?"))
print("L'area del cerchio di raggio", r, "è", calcola_area(r))

```

una chiamata di funzione è un'**espressione** che ha come **valore** il valore ritornato dalla funzione

valori di ritorno

```
def calcola_tassa( reddito, aliquota ):  
    # la tassa è una percentuale del reddito  
    tassa = reddito * (aliquota/100)  
    return tassa
```

la funzione prende
2 parametri e
restituisce un **valore**

```
stipendio = 3500  
trading = 1000  
print("tassa su reddito:", calcola_tassa(stipendio,15))  
print("tassa su reddito da capitale:", calcola_tassa(trading,26))
```

valore: 525

valore: 260

valori di ritorno

```
def scelta_aliquota(totale):  
    if totale < 200000 :  
        return 15  
    else:  
        return 25  
  
stipendio = 350000  
a = scelta_aliquota(stipendio)  
t = calcola_tassa(stipendio, a )
```

- se il flusso di controllo si ramifica, **tutti i rami** devono restituire lo stesso numero di valori

le variabili **a** e **t** sono inizializzate con il valore restituito dalla chiamata delle funzioni

```
def divisibile(x,y):  
    if x%y ==0:  
        return True  
    else:  
        return False  
  
print("Posso dividere 15 per 4? Risposta: ", divisibile(15,4))  
print("Posso dividere 328 per 14? Risposta: ", divisibile(328,14))  
print("Posso dividere 777 per 11? Risposta: ", divisibile(777,11))
```

valori di ritorno

```
def stampa2volte(s):  
    print(s)  
    print(s)  
    return
```

quando non ha nessun valore da ritornare,
la keyword **return** si puo' omettere
(sta per return None)

Esempio

- Funzione che conta **quante volte** una data lettera è presente in una data parola:

```
def conta(lettera,parola):  
    n=0  
    for i in parola:  
        if i == lettera:  
            n = n+1  
    return n  
  
x = conta('p','appalto')  
y = conta('t','attrattore')
```

```
n=0  
for i in 'appalto':  
    if i == 'p':  
  
n=0  
for i in 'attrattore':  
    if i == 't':  
        n=n+1  
return n
```

```
temp = input('inserisci una parola')  
y = conta('p',temp)  
print('la p compare',y,'volte in', temp)
```

Esercizio

- Funzione che cerca **se** una data lettera è presente in una data parola:

```
def cerca(lettera,parola):  
    trovato = False  
    for i in parola:  
        if i == lettera:  
            trovato = True  
    return trovato  
  
x = cerca('p','appalto')
```

- confronta la funzione precedente con questa:

```
def cerca(lettera,parola):  
    trovato = False  
    for i in parola:  
        if i == lettera:  
            trovato = True  
        else:  
            trovato = False  
    return trovato
```

Esercizio

- Funzione che cerca **se** una data lettera è presente in una data parola:

```
def cerca(lettera,parola):  
    trovato = False  
    for i in parola:  
        if i == lettera:  
            trovato = True  
    return trovato  
  
x = cerca('p','appalto')
```

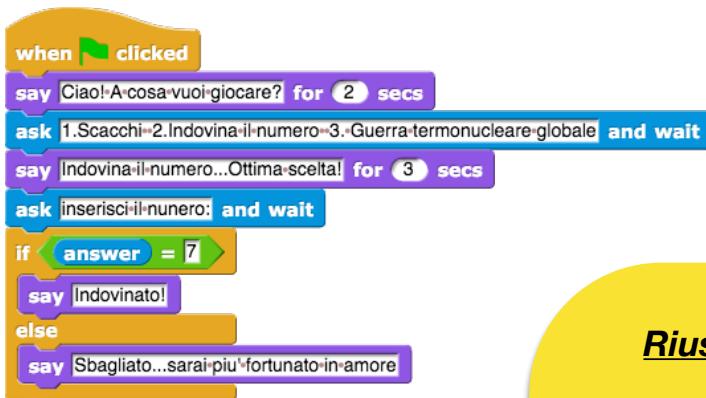
- confronta la funzione precedente con questa, anche tracciando il flusso di controllo generato dalla chiamata delle due diverse funzioni:

```
def cerca(lettera,parola):  
    trovato = False  
    for i in parola:  
        if i == lettera:  
            return True ←  
    return False
```

termina subito la funzione
senza continuare con le
istruzioni successive, quindi
senza completare il ciclo for

perché usare le funzioni

- dare un nome ad un gruppo di istruzioni (*functional or procedural abstraction*) rende il programma **più facile da leggere e correggere** (*code readability*)
 - le funzioni rendono il programma più breve, eliminando il codice ripetitivo (*code reuse*).
 - Se in un secondo momento bisogna fare una **modifica** a quel gruppo di istruzioni, basta farla in un posto solo (*manutenibilità e generalizzazione*, es. aggiungo un parametro)
- dividere un programma lungo in funzioni vi permette di correggere le parti una per una, per poi assemblarle in un complesso funzionante (*decomposition, separation of concerns, unit testing*)
- Funzioni ben fatte sono spesso utili per più programmi. Una volta scritta si puo' riusare (code reuse e **library**)



Riusa il codice del programma “*indovina il numero*”:

- “impacchetta” il codice in una funzione
- invoca la funzione in questo programma, se l’utente ha scelto il gioco 2.

```
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 piu fortunato in amore')
```

strutture dati e data abstraction

I dati in uso

“Il programma chiede di inserire una **data** e fa...”

“Il programma saluta l’utente stampando una **citazione** ...”

“Il programma chiede di inserire un **numero di telefono** e fa...”

“Il programma gestisce una **rubrica telefonica** ...”

“Il programma tiene traccia di una **rete di amicizie** ...”

i dati del problema

serve trovare un modo per **rappresentarli nel programma**
usando i costrutti del linguaggio di programmazione

i dati del programma

come rappresento una *data/citazione/tel...* in un programma python?

I dati in uso

giorno=5
mese=11
anno=2020

“Il programma chiede di inserire una **data** e fa...”

“Nel mezzo del...”

“Il programma saluta l’utente stampando una **citazione** ...”

“Il programma chiede di inserire un **numero di telefono** e fa...”

“Il programma gestisce una **rubrica telefonica** ...”

tel='04982714'

“Il programma tiene traccia di una **rete di amicizie** ...”

i dati del problema

????

serve trovare un modo per **rappresentare**
usando i costrutti del linguaggio di pro-

```
nome1 = 'Mario'  
tel1 = '003933915'  
nome2 = 'Barbara'  
tel2 = '003947195'  
...
```

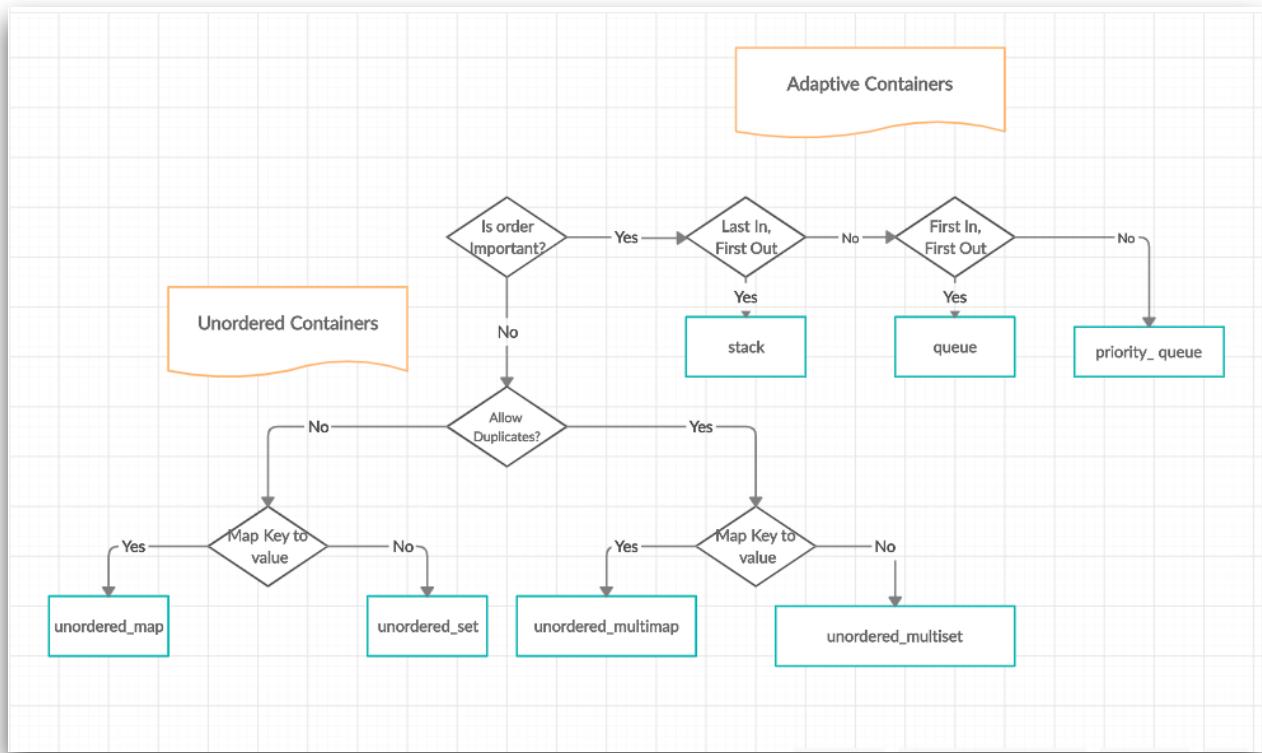
i dati del programma

come rappresento una *data/citazione/tel...* in un programma python?

strutture dati

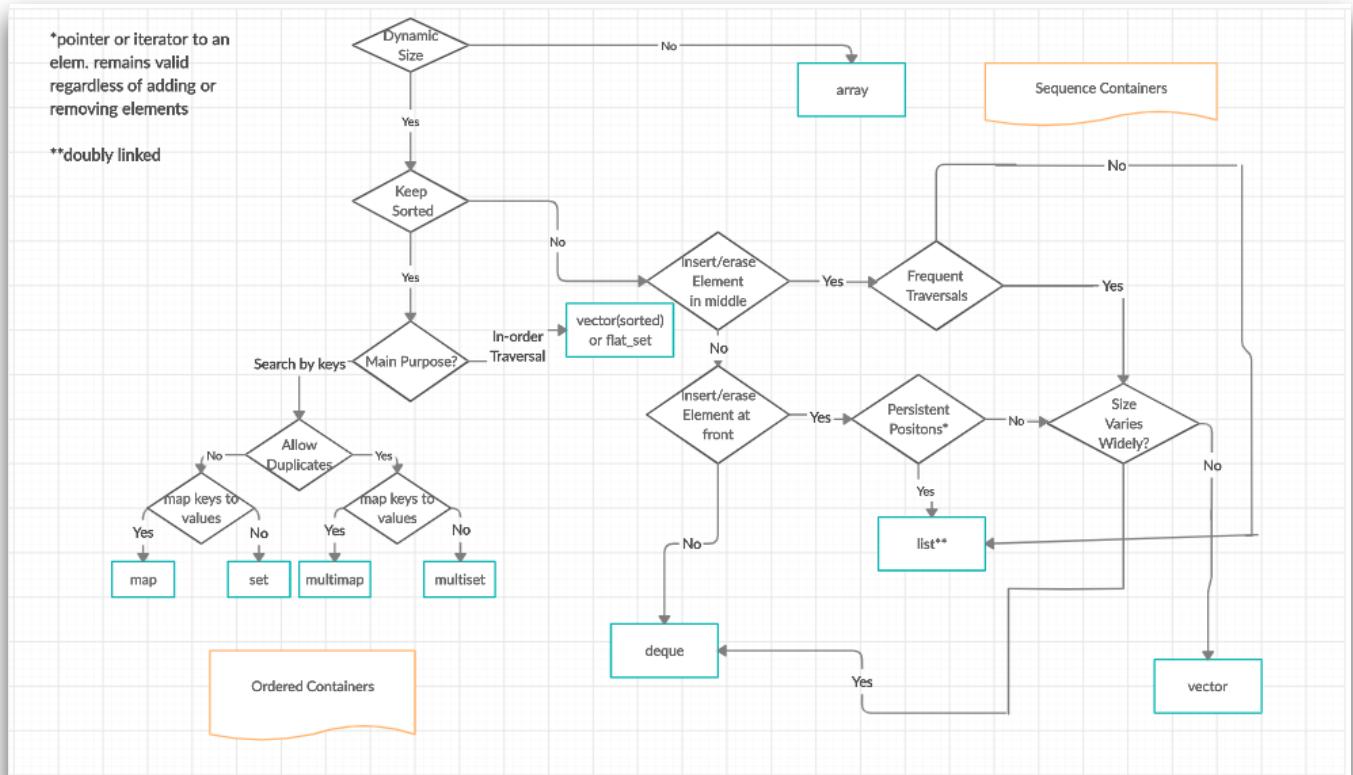
- i dati più semplici sono rappresentati da **valori primitivi** (*interi, decimali, booleani, caratteri, stringhe*), memorizzati in una variabile, o più di una
- i programmi lavorano spesso con dati di natura più complessa, es. *elenchi di persone, tabelle, anagrafe dei clienti, rete di amici, ...*
 - i **dati** che si usano vanno *rappresentati* attraverso i valori e le **strutture dati** offerte dal linguaggio di programmazione.
 - *liste, array, coppie, tuple, dizionari, insiemi, stack, alberi...*
- **Ogni linguaggio** di programmazione offre una **serie di strutture dati**, ciascuna con caratteristiche diverse
 - es. in alcuni casi i **dizionari python** più efficienti dei **vettori C++**
 - es. **Java8** introduce gli **Stream** per **big data applications**

The C++ Standard Template Library (STL)



<https://www.geeksforgeeks.org/the-c-standard-template-library-stl/>

The C++ Standard Template Library (STL)



<https://www.geeksforgeeks.org/the-c-standard-template-library-stl/>

algoritmo e strutture dati

Data la specifica di un problema, **il programmatore deve:**

- individuare i passi che lo risolvono: **algoritmo** risolutivo, **e la sua implementazione** nel linguaggio di programmazione
- scegliere **come rappresentare i dati del problema** usando le **strutture dati più opportune**

le due cose sono legate:

a seconda della struttura dati che sceglie, l'implementazione del programma può risultare più o meno **efficiente** (anche drasticamente) ma anche più **leggibile**, e più **manutenibile**

liste

- Una **lista** è una sequenza di valori, di tipo qualsiasi, detti **elementi** della lista

```
[10,20,30,40] # una lista di interi
['antipasto','primo','secondo','dessert'] # lista di stringhe
[10,'pipper',3.14] # lista di valori diversi
[[1,2],[3,4],[5,6,7]] # lista di liste di interi
[] # lista vuota
```

```
numeri = [10,200]
menu =['antipasto','primo']
vuota =[]
print(numeri,menu,vuota) #stampa [10,200] ['antipasto','primo'] []
```

liste

- Data una lista, si può **accedere a ciascun elemento**, per leggerlo oppure modificarlo, tramite il suo **indice**
- Gli elementi hanno indice **da 0** fino alla **lunghezza della lista-1**

```
menu = ['antipasto','primo','secondo','dessert']  
print(menu[0])          # stampa antipasto  
menu[0] = 'aperitivo'    # sovrascrive l'elemento di indice 0  
menu[-1] = 'gelato'      # sovrascrive l'elemento di indice -1  
print(menu)              #stampa ['aperitivo','primo','secondo','gelato']  
  
for piatto in menu: # variabile piatto itera nell'elenco menu  
    print(piatto)  
  
x = [4,55,20,12]  
quanti_sono = len(x) # lunghezza lista: 4 (len funzione built-in)  
  
# raddoppio i valori nella lista  
for i in range(len(x)): # i in 0,1,2,3 cioe' itera sugli indici  
    x[i] = x[i]*2  
print(x)   # stampa [8,110,40,24]
```

operazioni sulle liste

```
a = [1,2,3]  
b = [4,5,6]  
c = a + b # operazione che concatena le liste  
print(c) # [1,2,3,4,5,6]  
print(a) # [1,2,3]  
  
t = ['d','o','f']  
t.append('q') # operazione che aggiunge un elemento in coda  
print(t) # ['d','o','f','q']  
  
t1 = ['b','l']  
t.extend(t1) # aggiunge in coda tutti gli elementi di t1  
print(t) # ['d','o','f','q','b','l']  
  
t.sort() # ordina gli elementi di t  
print(t) # ['b','d','f','l','o','q']  
  
x = t.pop(1) # elimina e restituisce elemento di indice 1  
t.remove('q') # elimina elemento  
print(t,x) # ['b','f','l','o'] 'd'
```

le liste possono essere **passate come parametri alle funzioni**

Esercizi

- Scrivere una funzione che prende due parametri: un intero **n** e una lista di interi **list**, e **cerca se** (cioè restituisce **True** se) **l'intero n è presente nella lista list**

```
# fun sarà chiamata con 2 argomenti:  
# un intero e una lista di interi  
def fun(n,list):  
    ...
```

- Scrivere una funzione che prende due parametri: un intero **n** e una lista **ordinata** di interi **list**, e **cerca se** (cioè restituisce **True** se) **l'intero n è presente nella lista list**

```
# fun_ord sarà chiamata con 2 argomenti:  
# un intero e una lista già ORDINATA di interi  
def fun_ord(n,list):  
    ...
```

Esercizi

1. Scrivere un **ALGORITMO** che data una lista di numeri produce in output la lista con quei numeri ordinati in senso crescente
2. Implementare l'algoritmo in un **SOFTWARE** python

NOTA: questo esercizio è difficile da svolgere correttamente. Ma è utile provare ad abbozzare un algoritmo, rendersi conto se è corretto almeno in qualche caso, e provare ad implementarlo.

Esercizio

```
a=int(input('primo numero: '))
b=int(input('secondo numero: '))
c=int(input('terzo numero: '))
d=int(input('quarto numero: '))

lista=[a,b,c,d]
max=a
for i in lista:
    if a>i:
        max=a
    if b>i:
        max=b
    if c>i:
        max=c
    if d>i:
        max=d
```

```
a=int(input('primo numero: '))
b=int(input('secondo numero: '))
c=int(input('terzo numero: '))
d=int(input('quarto numero: '))

lista=[a,b,c,d]
max=a
for i in lista:
    if i > max:
        max=i
```

- Confrontare il flusso di controllo di questi due programmi
- Calcolano entrambi correttamente il massimo tra i 4 numeri inseriti?

il sito **The Oracle of Bacon**

- **mantiene un grafo** con attori legati da un arco se hanno recitato in uno stesso film
- dato un attore ne **calcola il numero di Bacon**, i.e. **lunghezza del cammino minimo** che lo separa da Kevin Bacon (**breadth-first search algorithm** starting from Bacon)

- **un grafo**
- **un algoritmo** che risolve un problema:
dati due nodi, calcola il cammino minimo

i dati del problema

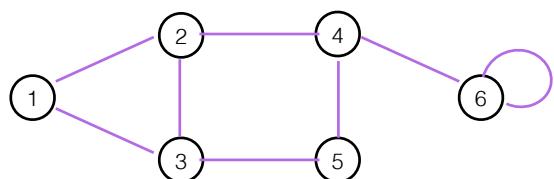


i dati del programma

- una **struttura dati** per rappresentare un grafo
- **implementazione di un algoritmo** su un grafo

rappresentazione di un grafo

Un **grafo** è un insieme di vertici/nodi e un insieme di **archi** tra coppie di nodi:



$$G = (V, E) \text{ con}$$

$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{(1,2), (1,3), (2,3), (2,4), (4,6), (3,5), (4,5), (6,6)\}$$

Come rappresentiamo un grafo in un programma ?

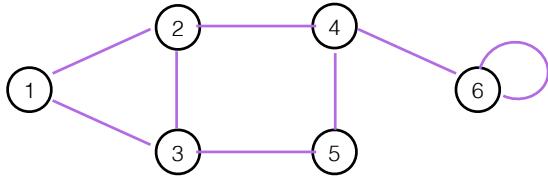
- si potrebbe implementare un grafo come **due liste**:

```
vertici = [1,2,3,4,5,6]
archi = [[1,2],[1,3],[2,3],[2,4],[4,6],[3,5],[4,5],[6,6]]
```

- basta **un intero** e una **lista**:

```
n_vertici = 6
archi = [[1,2],[1,3],[2,3],[2,4],[4,6],[3,5],[4,5],[6,6]]
```

rappresentazione di un grafo



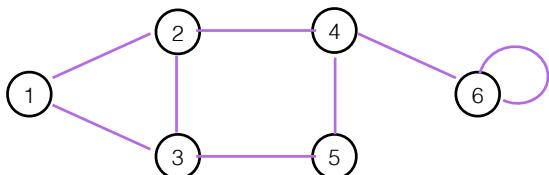
```
n_vertici = 6  
archi = [[1,2],[1,3],[2,3],[2,4],[4,6],[3,5],[4,5],[6,6]]
```

- come calcolare quanti archi escono da nodo 3?
- come calcolare se c'è un arco che collega 1 e 6?
devo cercare in tutto la lista archi !

```
quanti_da_3 = 0  
for x in archi:  
    if x[0]==3 or x[1]==3:  
        quanti_da_3 = quanti_da_3+1
```

```
arco_1_6 = False  
for x in archi:  
    if (x[0]==1 and x[1]==6) or  
        (x[0]==6 and x[1]==1):  
        arco_1_6 = True  
        break
```

rappresentazione di un grafo



$$V=\{1,2,3,4,5,6\}$$

$$E=\{(1,2), (1,3), (2,3), (2,4), (4,6), (3,5), (4,5), (6,6)\}$$

Come rappresentiamo un grafo in un programma?

matrice di adiacenza

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 1 |

in posizione i,j c'è 1 se esiste l'arco (i,j) altrimenti c'è 0

lista di adiacenza

| | |
|-----|---------|
| 1 : | 2, 3 |
| 2 : | 1, 3, 4 |
| 3 : | 1, 2, 5 |
| 4 : | 2, 5, 6 |
| 5 : | 3, 4 |
| 6 : | 4, 6 |

- come calcolare quanti archi escono da nodo 3?
- come calcolare se c'e' un arco che collega 1 e 6?
è immediato!

in posizione i c'è l'elenco di nodi collegati ad i con un arco

matrice di adiacenza

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 1 |

lista di adiacenza

| | |
|-----|---------|
| 1 : | 2, 3 |
| 2 : | 1, 3, 4 |
| 3 : | 1, 2, 5 |
| 4 : | 2, 5, 6 |
| 5 : | 3, 4 |
| 6 : | 4, 6 |

- come calcolare quanti archi escono da nodo 3?
- come calcolare se c'e' un arco che collega 1 e 6? è immediato!

```
matrice = [[0,1,1,0,0,0],
           [1,0,1,1,0,0],
           [1,1,0,0,1,0],
           [0,1,0,0,1,1],
           [0,0,1,1,0,0],
           [0,0,0,1,0,1]]
```

```
lista = [[2,3],
         [1,3,4],
         [1,2,5],
         [2,5,6],
         [3,4],
         [4,6]]
```

```
# la riga del nodo 3 è matrice[2]
quanti_da_3 = 0
for i in matrice[2]: #somma
    quanti_da_3 = quanti_da_3 + i

arco_1_6 = (matrice[0][5] == 1)
```

```
quanti_da_3 = 0
for i in lista[2]: #lunghezza
    quanti_da_3=quanti_da_3 + 1

arco_1_6 = False
for n in lista[0]:
    if n==6:
        arco_1_6 = True
        break
```

Esercizio

Si consideri quanto visto a lezione:

```
matrice = [[0,1,1,0,0,0],
           [1,0,1,1,0,0],
           [1,1,0,0,1,0],
           [0,1,0,0,1,1],
           [0,0,1,1,0,0],
           [0,0,0,1,0,1]]
```

```
lista = [[2,3],
         [1,3,4],
         [1,2,5],
         [2,5,6],
         [3,4],
         [4,6]]
```

```
# la riga del nodo 3 è matrice[2]
quanti_da_3 = 0
for i in matrice[2]: #somma
    quanti_da_3 = quanti_da_3 + i

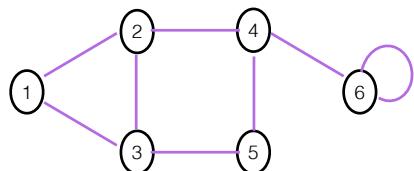
arco_1_6 = (matrice[0][5] == 1)
```

```
quanti_da_3 = 0
for i in lista[2]: #lunghezza
    quanti_da_3=quanti_da_3 + 1

arco_1_6 = False
for n in lista[0]:
    if n==6:
        arco_1_6 = True
        break
```

1. Tracciare il flusso di controllo dei due programmi.
2. Contare quante istruzioni vengono eseguite in totale dai due programmi. Qual è quello che *termina prima*, i.e. esegue meno istruzioni?

Esercizio



Si consideri il programma che usa la seguente rappresentazione dello stesso grafo:

```

n_vertici = 6
archi = [[1,2],[1,3],[2,3],[2,4],[4,6],[3,5],[4,5],[6,6]]

1 quanti_da_3 = 0
2 for x in archi:
3     if x[0]==3 or x[1]==3:
4         quanti_da_3 = quanti_da_3+1
5
6 arco_1_6 = False
7 for x in archi:
8     if (x[0]==1 and x[1]==6) or
9         (x[0]==6 and x[1]==1):
10        arco_1_6 = True
11        break
  
```

Tracciare il flusso di controllo del programma e contare quante istruzioni vengono eseguite in totale.

```

matrice = [[0,1,1,0,0,0],
           [1,0,1,1,0,0],
           [1,1,0,0,1,0],
           [0,1,0,0,1,1],
           [0,0,1,1,0,0],
           [0,0,0,1,0,1]]
  
```

14 passi

```

riga_nodo_3 = matrice[2] #indici da 0
quanti_da_3 = 0
for i in riga_nodo_3: #somma elementi
    quanti_da_3 = quanti_da_3 + i

arco_1_6 = (matrice[0][5] == 1)
  
```

12 passi

```

lista = [[2,3],
          [1,3,4],
          [1,2,5],
          [2,5,6],
          [3,4],
          [4,6]]

quanti_da_3 = 0
for i in lista[2]: #lunghezza
    quanti_da_3=quanti_da_3 + 1

arco_1_6 = False
for n in lista[0]:
    if n==6:
        arco_1_6 = True
        break
  
```

- calcolare quanti archi escono da nodo 3
- calcolare se c'e' un arco che collega 1 e 6

```

n_vertici = 6
archi = [[1,2],[1,3],[2,3],[2,4],[4,6],[3,5],[4,5],[6,6]]

quanti_da_3 = 0
for x in archi:
    if x[0]==3 or x[1]==3:
        quanti_da_3 = quanti_da_3+1

arco_1_6 = False
for x in archi:
    if (x[0]==1 and x[1]==6) or
        (x[0]==6 and x[1]==1):
        arco_1_6 = True
        break
  
```

37 passi

Io **stesso problema**, sullo **stesso grafo** viene risolto con **efficienza molto diversa** a seconda del modo che scelgo per **rappresentare i dati del problema** (il grafo) in **dati del programma**

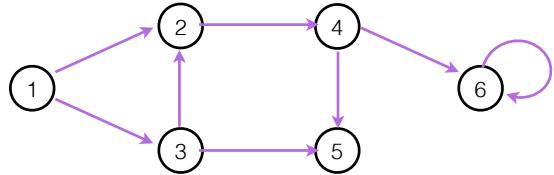
Esercizio

1. Comparare il numero di passi svolti dai tre programmi su un grafo iniziale con **più nodi e più archi**
2. Rifare l'esercizio precedente su un grafo che ha **molti più nodi che archi**
3. Rifare l'esercizio precedente su un grafo che ha **molti più archi che nodi**
4. osservare se ci sono differenze tra qual è la rappresentazione del grafo più efficiente nel caso 2 e 3.

Esercizio

- Considerare il grafo G composto dai vertici= $\{1,2,3,4,5,6\}$ e dagli archi $\{(1,2),(2,4),(1,4),(3,4),(5,3),(4,5),(5,6),(6,1)\}$
- Disegnare il grafo
- Scrivere la corrispondente matrice di adiacenza e la corrispondente lista di adiacenza
- Scrivere un programma che **calcola se escono più archi dal nodo 6 oppure dal nodo 1.**
 - confronta il numero di istruzioni eseguite dal programma che usa la matrice di adiacenza rispetto a quello che usa la lista di adiacenza
- Scrivere un programma che trova **qual è il nodo da cui escono più archi**
 - confronta il numero di istruzioni eseguite dal programma che usa la matrice di adiacenza rispetto a quello che usa la lista di adiacenza

rappresentazione: grafo orientato



matrice di adiacenza $n \times n$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 |

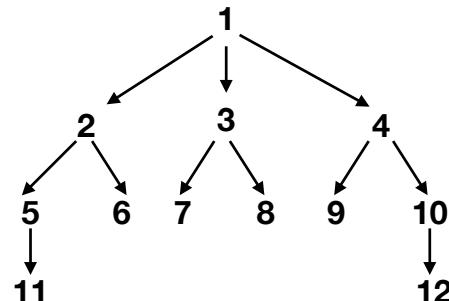
quali archi
escono da nodo 5

lista di adiacenza m

| | |
|-----|---------|
| 1 : | 2, 3 |
| 2 : | 1, 3, 4 |
| 3 : | 1, 2, 5 |
| 4 : | 2, 5, 6 |
| 5 : | 3, 4 |
| 6 : | 4, 6 |

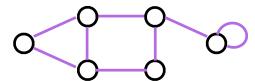
quali archi **entrano**
in nodo 5

Esercizio



- Considerare il grafo **orientato** in figura
- Scrivere la corrispondente matrice di adiacenza e la corrispondente lista di adiacenza
- Scrivere un programma che calcola quanti archi **escono** dal nodo 2 e quanti archi **entrano** nel nodo 2
 - confronta il numero di istruzioni eseguite dal programma che usa la matrice di adiacenza rispetto a quello che usa la lista di adiacenza
- Scrivere un programma che calcola **quanti nodi sono raggiungibili a partire dal nodo 4**
 - è più chiaro il codice che risolve l'esercizio usando la matrice di adiacenza oppure quello che usa la lista di adiacenza?

rappresentazione di un grafo



la **scelta è fortemente dipendente** da:

- che caratteristiche voglio evidenziare,
- **che uso devo farne**,
- quanto è grande il grafo da memorizzare,
- qual è la sua densità

grafi sociali **non**
stanno su memoria di una
singola macchina!

matrice di adiacenza

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 1 |

lista di adiacenza

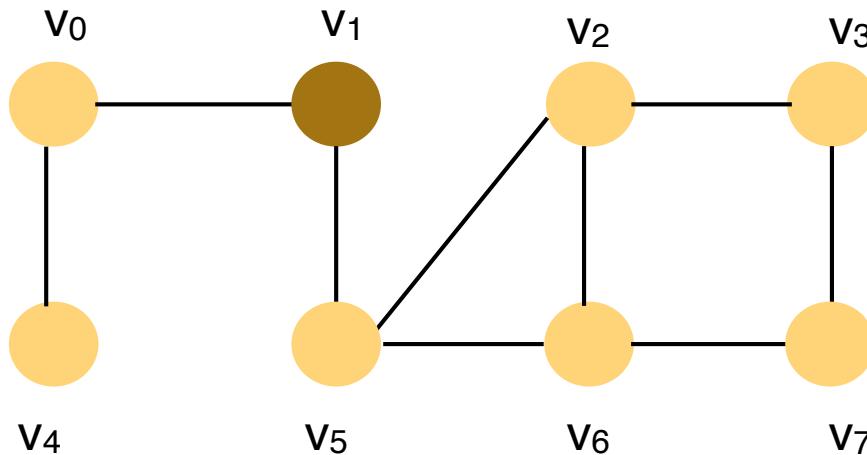
| | |
|-----|---------|
| 1 : | 2, 3 |
| 2 : | 1, 3, 4 |
| 3 : | 1, 2, 5 |
| 4 : | 2, 5, 6 |
| 5 : | 3, 4 |
| 6 : | 4, 6 |

algoritmi di visita di un grafo

visita di un grafo

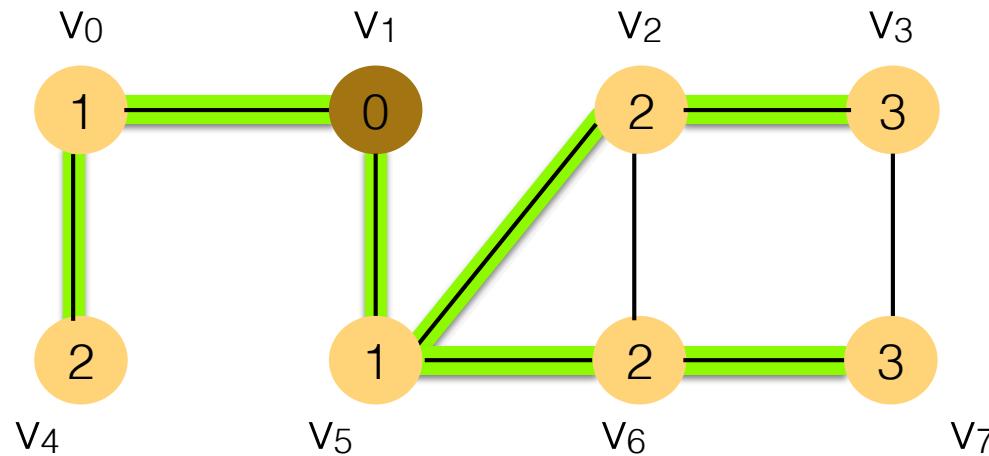
PROBLEMA: dato un grafo G e un suo nodo s , scoprire tutti i nodi di G raggiungibili da s , e trovare la loro distanza minima da s

- Dato un grafo $G=(V,E)$ ed un **nodo sorgente s** in V , definiamo un **algoritmo** che **visita sistematicamente il grafo G** per scoprire tutti i *nodi raggiungibili da s*



algoritmo Breadth-First Search

la visita è detta in **ampiezza** perché l'algoritmo scopre tutti i vertici a distanza **1**, poi scopre tutti quelli a distanza **2**, ...

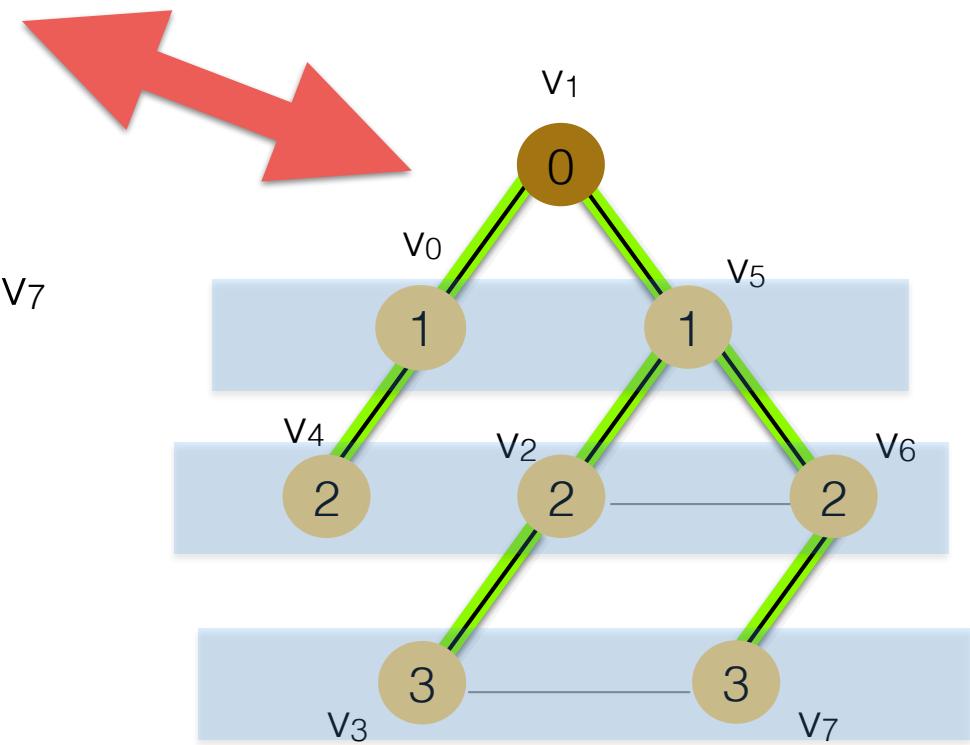
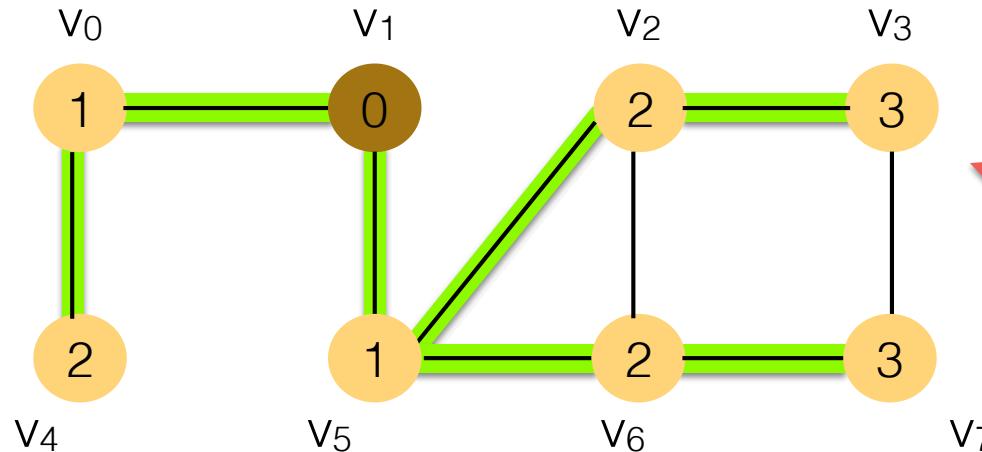


nel contempo

- **calcola la distanza minima**, cioè la lunghezza del cammino minimo, **tra ogni nodo e la sorgente s** (i numeri inseriti nei vertici)
- e produce l'**albero della visita, i cui rami sono cammini minimi** (archi colorati di verde)

algoritmo Breadth-First Search

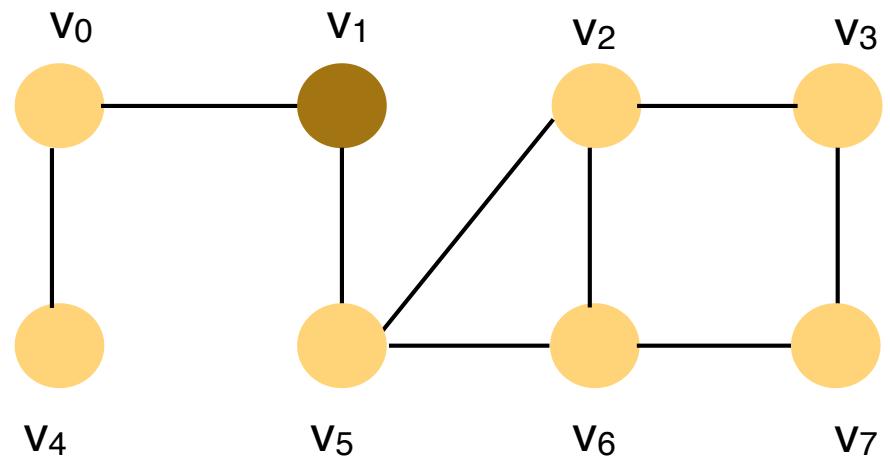
la visita è detta in **ampiezza** perché l'algoritmo scopre tutti i vertici a distanza k prima di scoprire quelli a distanza $k+1$



usato anche da **The Oracle of Bacon** website.
A breadth-first search from the vertex for
Kevin Bacon finds the shortest chain to all
other actors and actresses.

definizione dell'algoritmo

- Dato un grafo G ed un *nodo sorgente* s, **definiamo una sequenza di passi** che permettono di visitare tutti i nodi raggiungibili da s
- L'algoritmo si definisce in **pseudo-codice**
- L'algoritmo andrà poi **implementato** in un programma/software **eseguibile**, scritto in uno specifico linguaggio di programmazione



```
lista = [[1,4],  
[0,5],  
[3,5,6],  
[2,7],  
[0],  
[1,2,6],  
[2,5,7],  
[3,6]]
```

algoritmo Breadth-First Search



Idea

- per mantenere traccia del punto in cui si è arrivati nella visita **coloriamo i nodi**:

○ mai visitato

● visitato e sulla frontiera

● visitato e non più su frontiera

- memorizziamo la **Frontiera** come una **coda** (elenco FIFO) di nodi



- ogni nodo **v** memorizza 3 informazioni, utili per costruire l'albero di visita con i cammini minimi e relative distanze
 - **v.color** il colore di **v**
 - **v.dist** la distanza di **v** dal nodo sorgente **s**
 - **v.padre** il nodo che precede **v** nel cammino minimo che separa **v** da **s**

Passi di Inizializzazione:

1. tutti i nodi: assegna colore bianco, distanza ∞ e padre - (indefinito)
2. nodo s: assegna colore grigio e distanza 0 (padre -)
3. inserisce s nella Frontiera

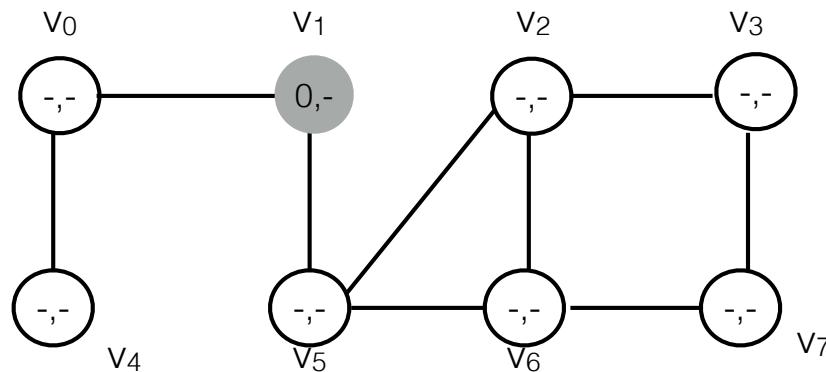
pseudo-codice

`:=` indica l'assegnamento,
`forEach` sta per una iterazione
`v.color`, bianco, `Front.add(s)`, ecc..
sono **scritture intuitive**, non
costrutti di programmazione

```
foreach v in V
    v.color := bianco
    v.dist  :=  $\infty$ 
    v.padre := -
    s.color := grigio
    s.dist  := 0
    Front.add(s)
```

Front

V1



mai visitato

visitato e **sulla frontiera**

visitato e non più su
frontiera

Corpo:

finché la frontiera non è vuota

prendo il primo nodo in coda, ***u***

per ogni nodo ***v*** adiacente a ***u***

se ***v*** è bianco **allora**

aggiungo *v* alla frontiera
colorandolo di grigio e
aggiorno *v.dist* e *v.padre*

coloro ***u*** di nero

```
while Front.isEmpty == false
    u := Front.pop
    foreach v in AdjList(u)
        if v.color == bianco
            v.color := grigio
            v.dist := u.dist + 1
            v.padre := u
            Front.add(v)
    u.color := nero
```



Idea



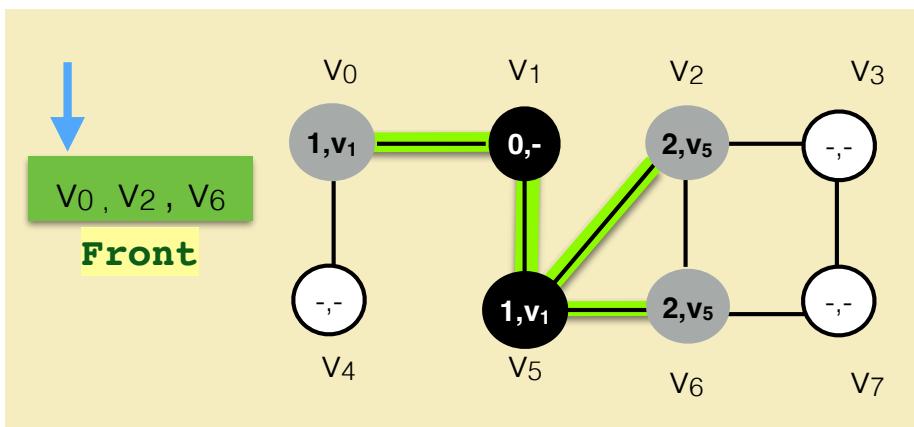
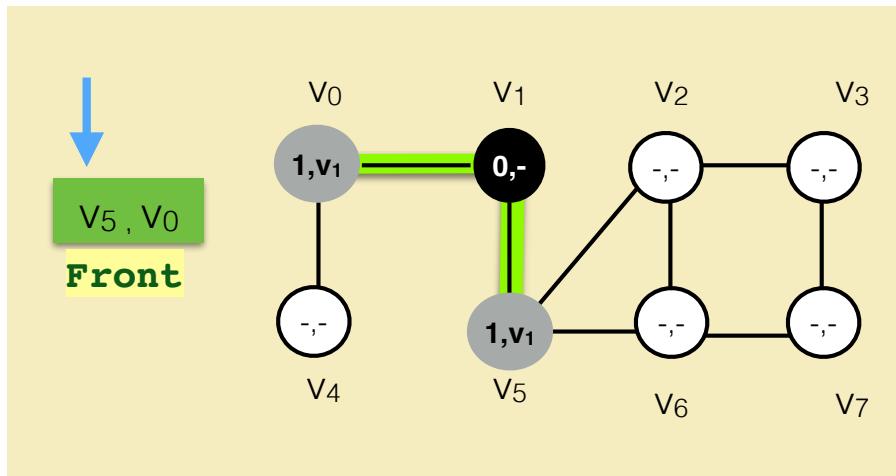
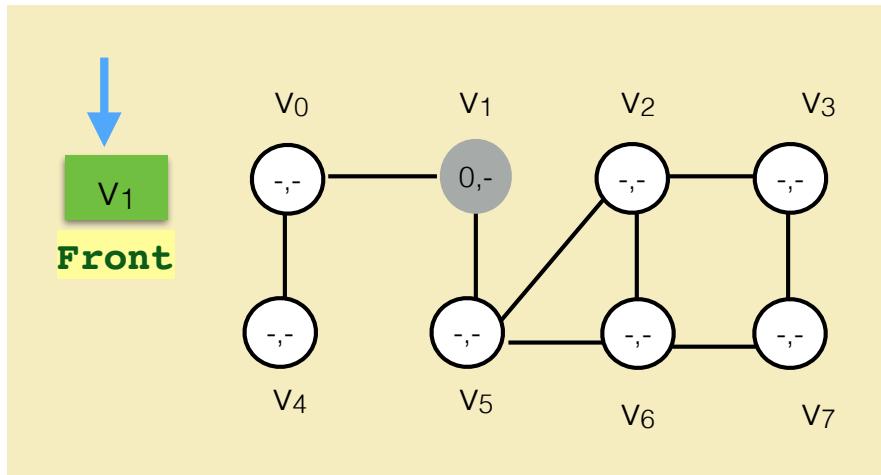
mai visitato



visitato e **sulla frontiera**

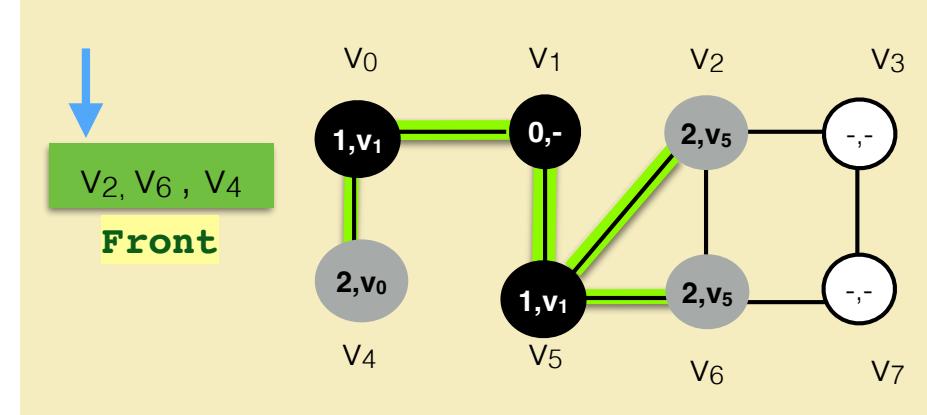


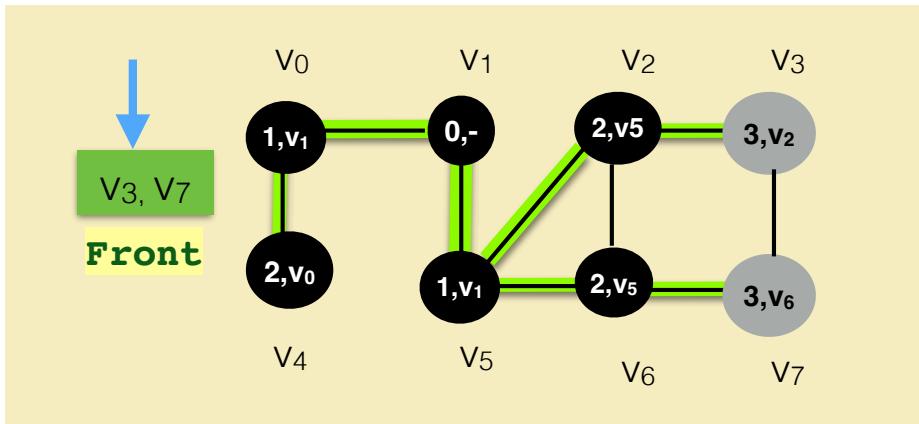
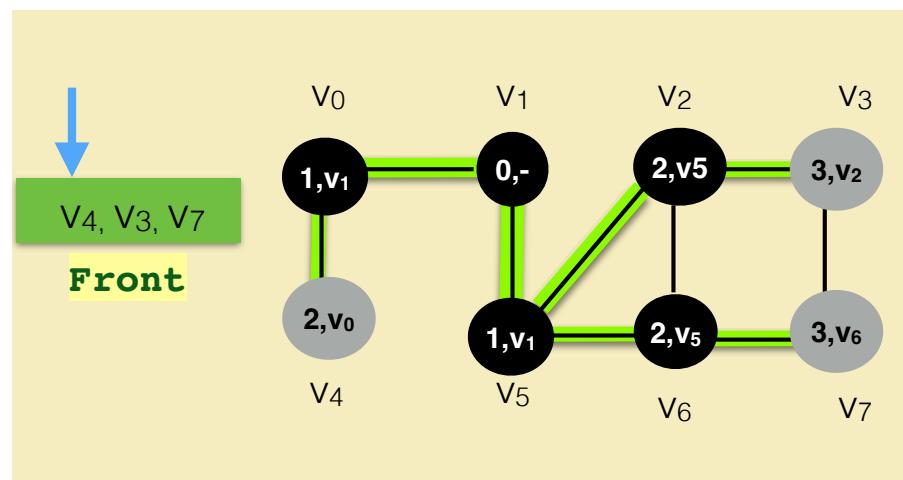
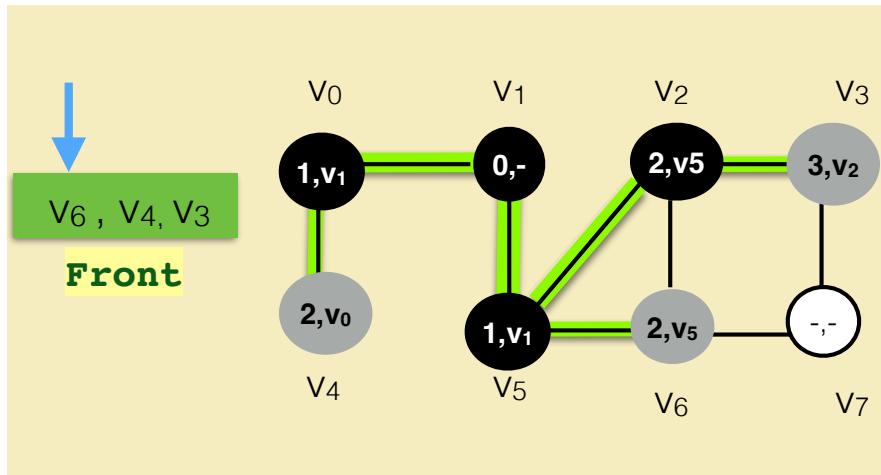
visitato e non più
su frontiera



```

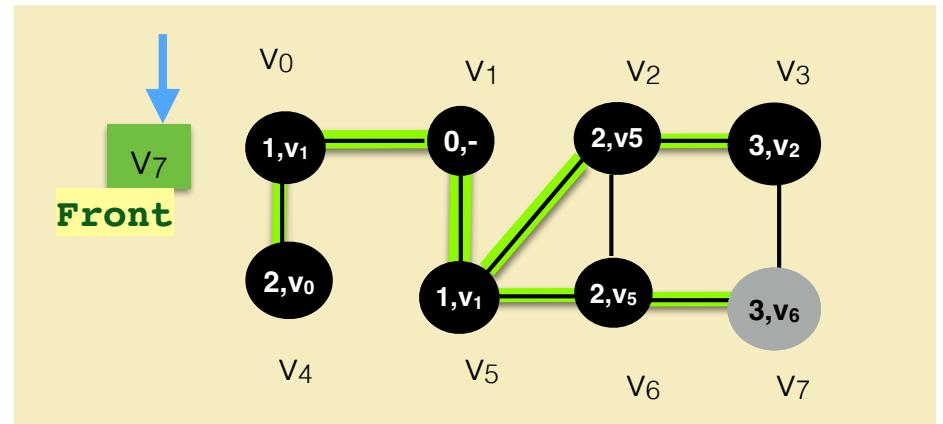
while Front.isEmpty==false
  u := Front.pop
  forEach v in AdjList(u)
    if v.color == bianco
      v.color := grigio
      v.dist := u.dist + 1
      v.padre := u
      Front.add(v)
    u.color := nero
  
```

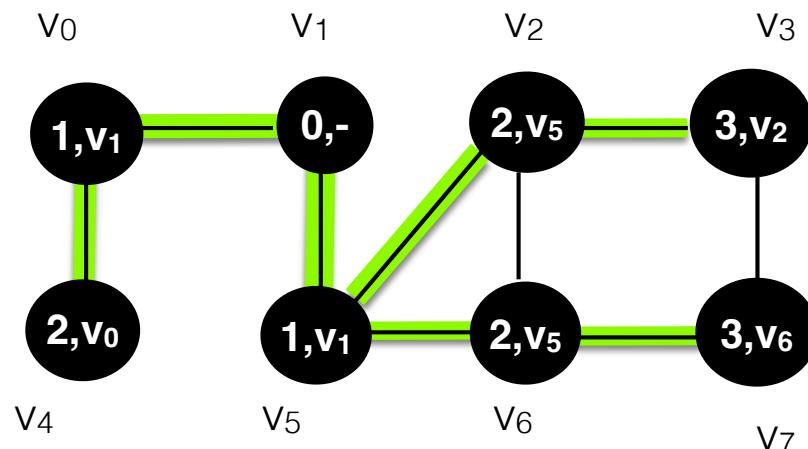




```

while Front.isEmpty==false
  u := Front.pop
  forEach v in AdjList(u)
    if v.color == bianco
      v.color := grigio
      v.dist := u.dist + 1
      v.padre := u
      Front.add(v)
    u.color := nero
  
```





```

while Front.isEmpty==false
    u := Front.pop
    forEach v in AdjList(u)
        if v.color == bianco
            v.color := grigio
            v.dist := u.dist + 1
            v.padre := u
            Front.add(v)
        u.color := nero
    
```

come facciamo a sapere che il risultato è giusto?

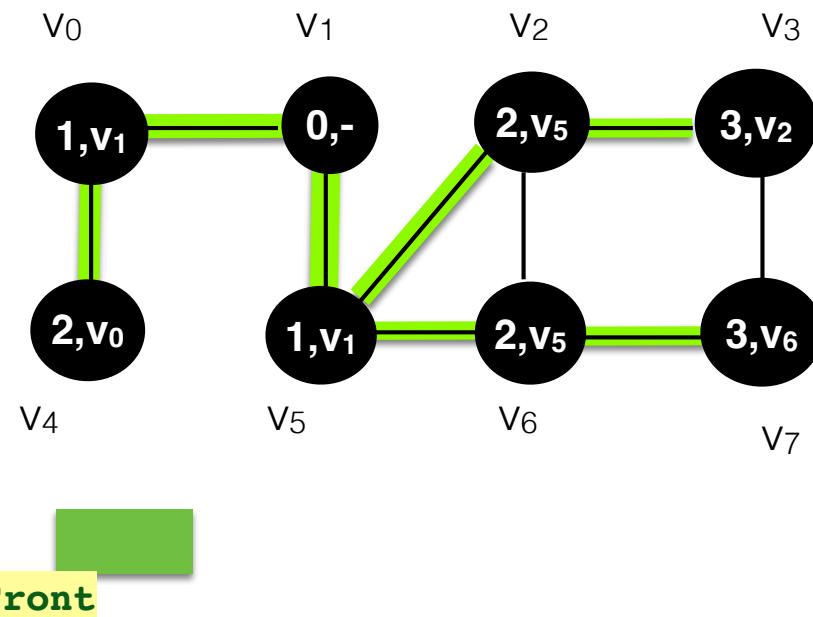
magari, se eseguito su un *altro grafo*, questo algoritmo dimentica di visitare qualche nodo, o segna un cammino che non è minimo

Proprietà di un Algoritmo

- Dato un algoritmo, scritto in pseudo-codice, si devono dimostrare due proprietà:
- **correttezza**
 - dato un *qualsiasi* grafo G ed un suo nodo s, l'algoritmo BST **termina e visita correttamente** (...) tutti i nodi
- **efficienza - complessità**
 - si fornisce una stima di **quanti passi di esecuzione** sono necessari per l'esecuzione completa dell'algoritmo
 - si fornisce una stima di **quanta memoria viene occupata**, data dalla dimensione delle strutture dati richieste

complessità
temporale

complessità
spaziale



```

while Front.isEmpty==false
  u := Front.pop
  forEach v in AdjList(u)
    if v.color == bianco
      v.color := grigio
      v.dist := u.dist + 1
      v.padre := u
      Front.add(v)
    u.color := nero
  
```

come faccio a sapere che il risultato è giusto?

- **correttezza** dell'algoritmo BFS
- **efficienza: complessità** dell'algoritmo ($O(|V|+|E|)$)

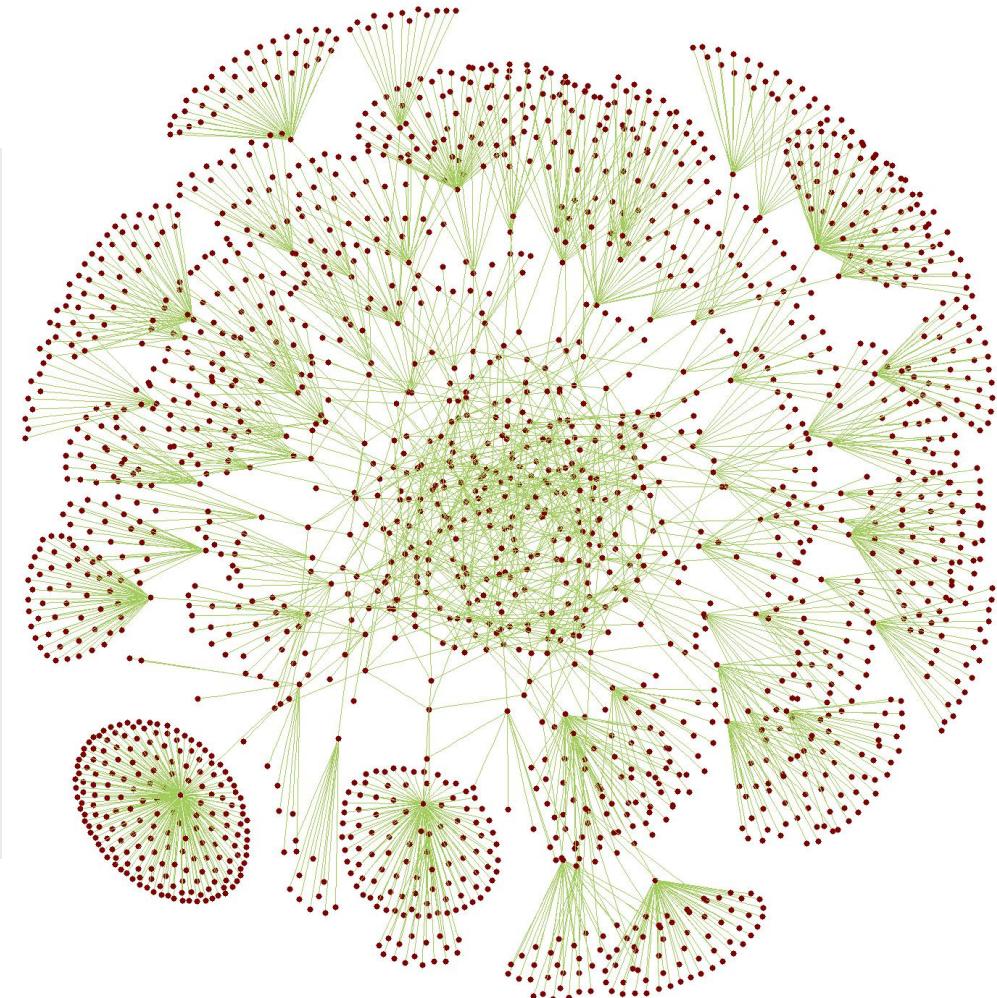
Problema: dato un grafo G e nodo iniziale s, visitare tutti i nodi e trovare i cammini minimi da s a ogni nodo



BFS

```
foreach v in V
    v.color := bianco
    v.dist := ∞
    v.padre := nil
s.color := grigio
s.dist := 0
Front.add(s)

while Front.isEmpty==false
    u := Front.pop
    foreach v in AdjList(u)
        if v.color == bianco
            v.color := grigio
            v.dist := u.dist + 1
            v.padre := u
            Front.add(v)
    u.color := nero
```

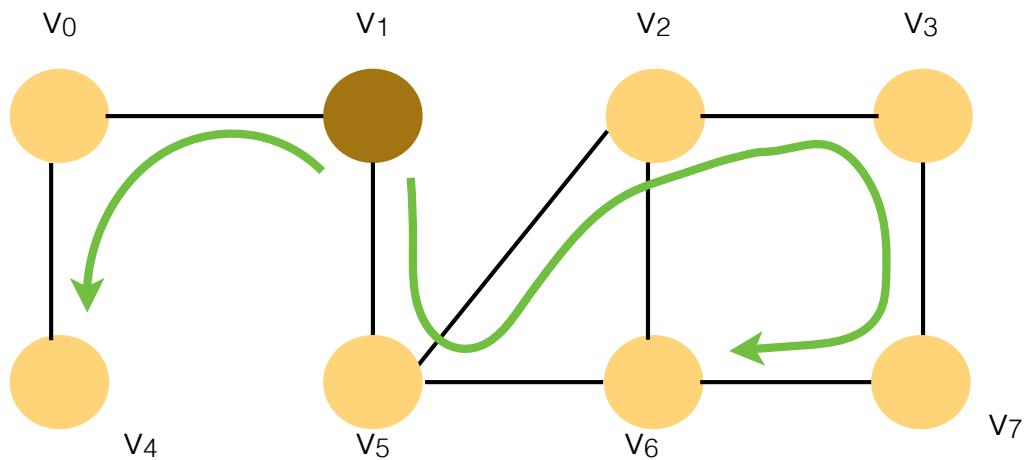


- 1. Dimostro che è corretto**
- 2. Lo implemento in Python**
- 3. Lo uso**

- La **correttezza** assicura che l'esecuzione **terminerà**
- la **complessità** fornisce una stima di **quanto durerà** l'esecuzione

Algoritmo Depth-First Search

la visita è detta in **profondità** perché, invece di fare un passo alla volta in tutte le direzioni, ora **si va avanti in una direzione finché' possibile**, cioè ogni volta si esplora un arco uscente dall'ultimo nodo raggiunto



ordine di visita: $v_1, v_5, v_2, v_3, v_7, v_6, v_0, v_4$

Problema: dato un grafo G e nodo iniziale s, visitare tutti i nodi e trovare i cammini minimi



Idea

BFS

```
forEach v in V
    v.color := bianco
    v.dist := ∞
    v.padre := nil
s.color := grigio
s.dist := 0
Front.add(s)

while Front.isEmpty==false
    u := Front.pop
    forEach v in AdjList(u)
        if v.color == bianco
            v.color := grigio
            v.dist := u.dist + 1
            v.padre := u
            Front.add(v)
    u.color := nero
```



Altra Idea !

DFS

```
forEach v in V
    ...
    ...
    ...
    ...
```

Sarà migliore di BFS oppure no?

- Su un grafo enorme meglio eseguire questo o BFS?
- Magari per certi G con una forma particolare è più veloce mentre per altri G è più lento.
- **Studio e comparo la complessità** dei due algoritmi

Attenzione

- Questa parte sugli algoritmi è più difficile, anche perché l'abbiamo trattata molto superficialmente e solo tramite esempi.
- È importante però comprendere i concetti di base, e nel caso su questi ci siano dubbi/domande:
 1. provare a rendere molto precise e puntuali le domande/i dubbi. Anche questo aiuta a migliorare lo studio!
 2. chiedere: ad un compagno di studio, nel forum del corso, al prof.

Cosa sapere sull'esecuzione di BFS

- capire **cos'è la visita in ampiezza**
- capire **cos'è la distanza e il cammino minimo** di un nodo da un altro

Esempio di esercizi:

Indichiamo con **BFS(G, n)** l'esecuzione di BFS sul grafo G a partire dal nodo n .

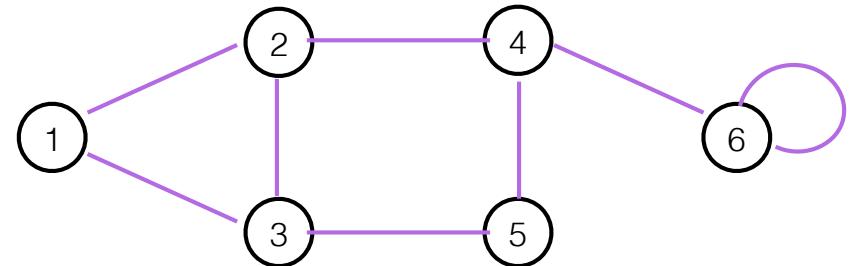
- Dato questo elenco di nodi del grafo G in figura:

1, 2, 4, 5, 6, 3

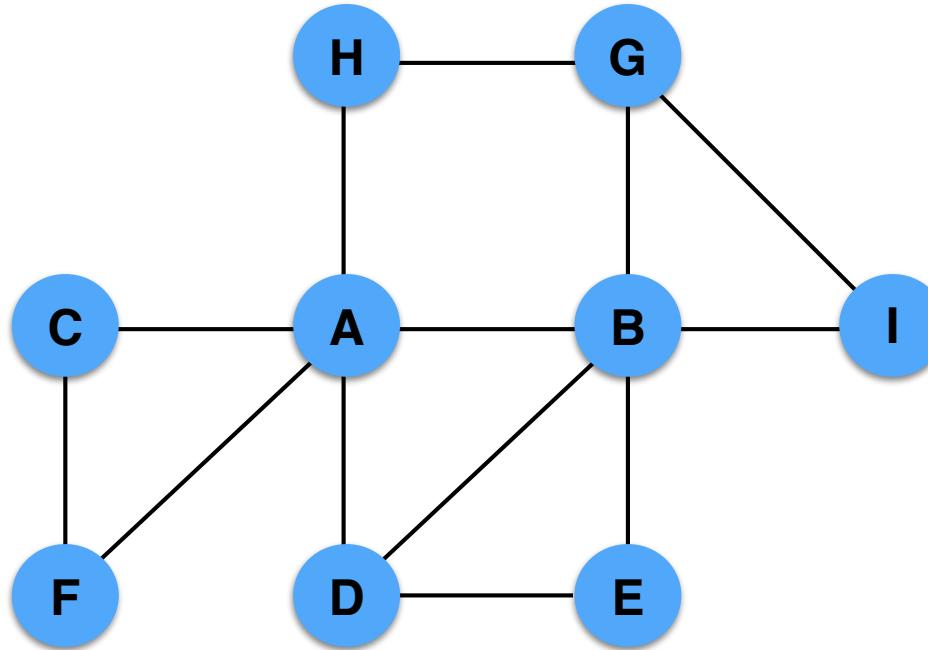
potrebbe essere l'ordine in cui **BFS($G, 1$)** visita i nodi del grafo?

- Elencare in che ordine visita i nodi l'esecuzione **BFS($G, 5$)**

N.B.: può esserci più di una soluzione: **5,4,3,...** ma anche **5,3,4,....**



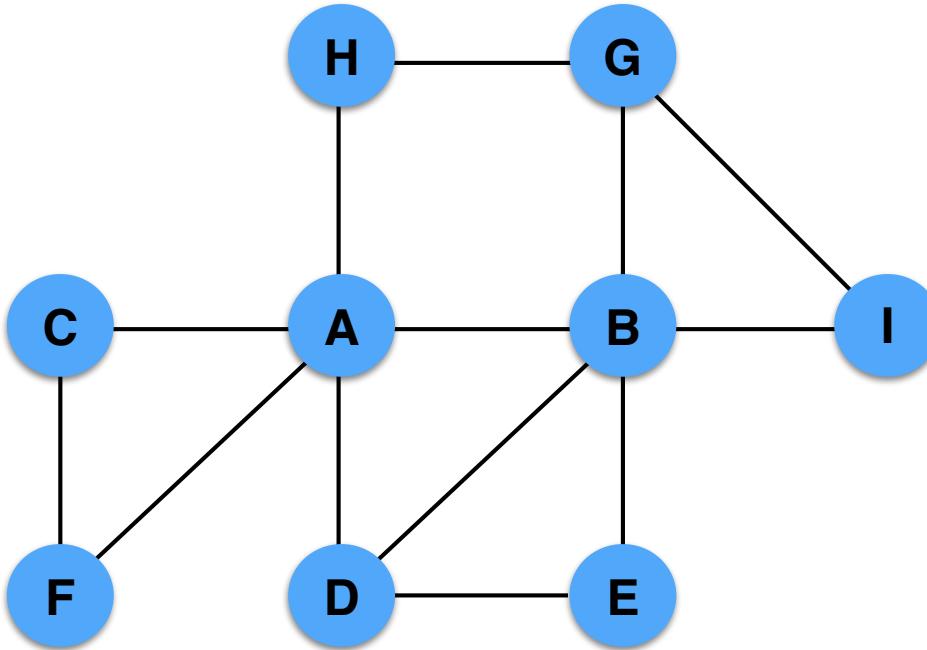
Esercizio



- Eseguire l'algoritmo Breadth-First Search a partire dal nodo A
- Eseguire l'algoritmo Breadth-First Search a partire dal nodo I

È sufficiente indicare un ordine in cui vengono visitati i nodi, e indicare per ogni nodo le 3 informazioni (colore, padre, distanza) nello stato finale.

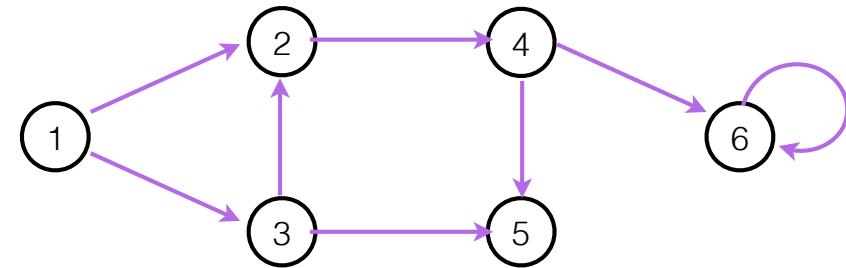
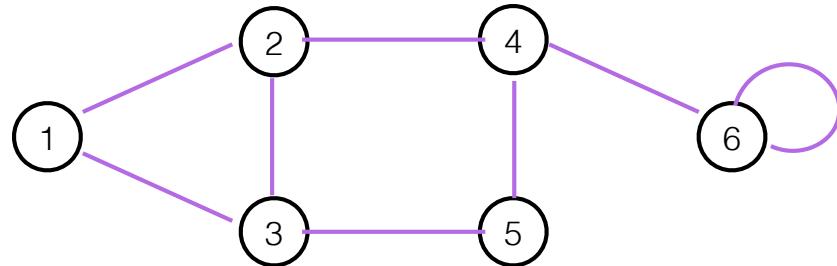
Esercizio



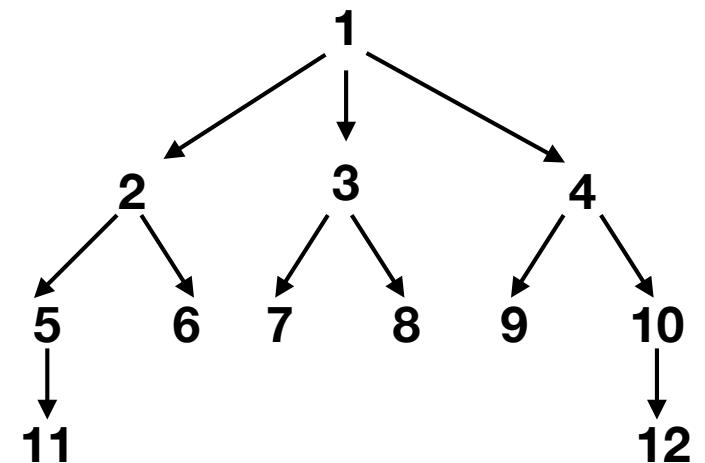
- Indicare un possibile ordine di visita Depth-first a partire dal nodo A
- Indicare un possibile ordine di visita Depth-first a partire dal nodo I

Esercizio

Eseguire i due algoritmi di visita, BFS e DFS, sui seguenti **4** grafi, indicando semplicemente l'ordine in cui vengono visitati i nodi.



Considerare il grafo G di vertici= $\{1,2,3,4,5,6\}$ e archi $\{(1,2),(2,4),(1,4),(3,4),(5,3),(4,5),(5,6),(6,1)\}$



Algoritmo Breadth-first Search

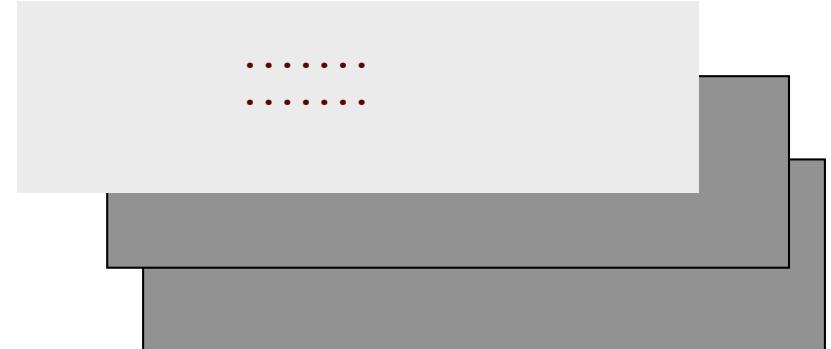
Algoritmo: BFS

```
forEach v in V
    v.color := bianco
    v.dist := ∞
    v.padre := nil
s.color := grigio
s.dist := 0
Front.add(s)

while Front.isEmpty==false
    u := Front.pop
    forEach v in AdjList(u)
        if v.color == bianco
            v.color := grigio
            v.dist := u.dist + 1
            v.padre := u
            Front.add(v)
    u.color := nero
```

questo è *pseudo-codice*:
per poterlo eseguire
va **implementato**
in un linguaggio di
programmazione

Software in Python



Software in Java



Software in C++





chiara? corretta?
completa?

“Specifica del problema”



correttezza e
complessità

“Algoritmo risolutivo”



```
int main()
{
    int palindrome, reverse=0;
    cout<<"Enter number: ";
    cin>>palindrome;

    while(palindrome > 0)
    {
        reverse = reverse * 10 + palindrome % 10;
        palindrome = palindrome / 10;
    }

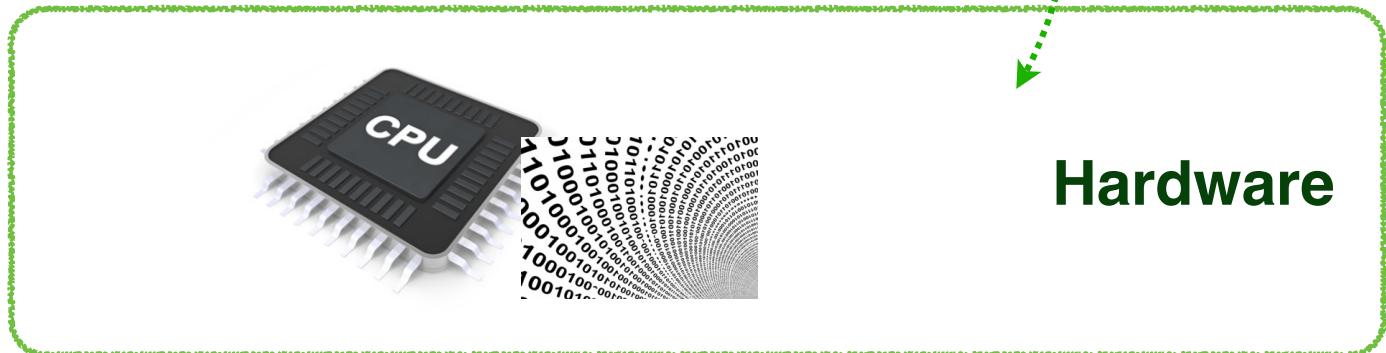
    cout<<(reverse == palindrome ? "The number is a palindrome." : "The number is not a palindrome.");
    return 0;
}
```

Software

errori sintattici e
logici

compilatore

Sistema Operativo



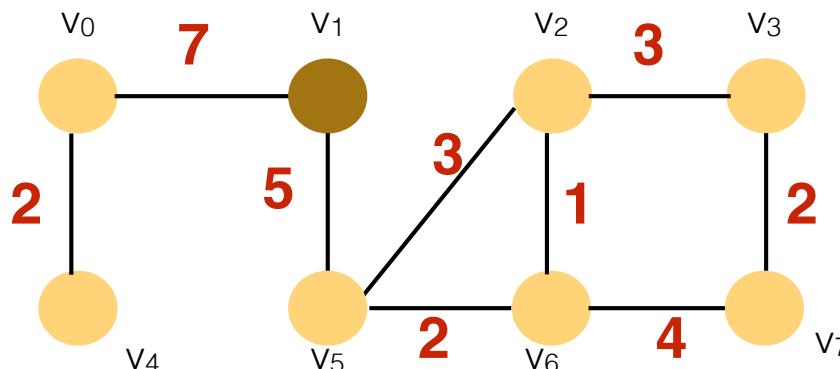
Hardware

Approfondimento NON RICHIESTO per l'esame

problemi di ottimizzazione

Tanti problemi reali si possono modellizzare come problemi su grafi:

- stoccaggio
- allocazione risorse (aerei, aule..)
- percorsi di viaggio
- ...

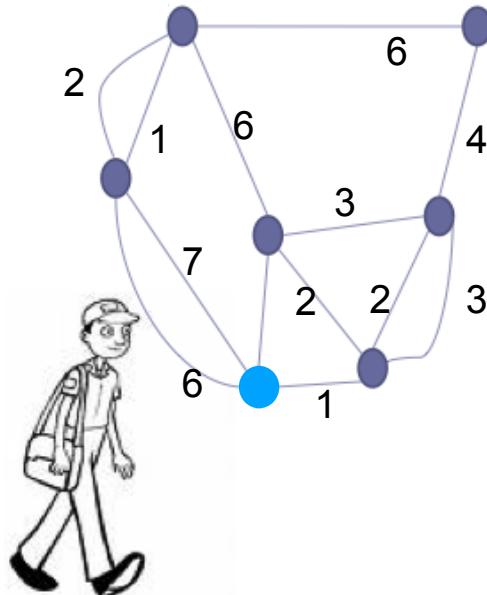


grafo pesato

Problema del postino cinese

Un postino parte dall'ufficio postale, attraversa **tutte le strade** del quartiere e **ritorna** all'ufficio postale, percorrendo la **minima** distanza possibile.

Analogo: trovare il modo per pulire tutte le strade e tornare alla base con costo minimo

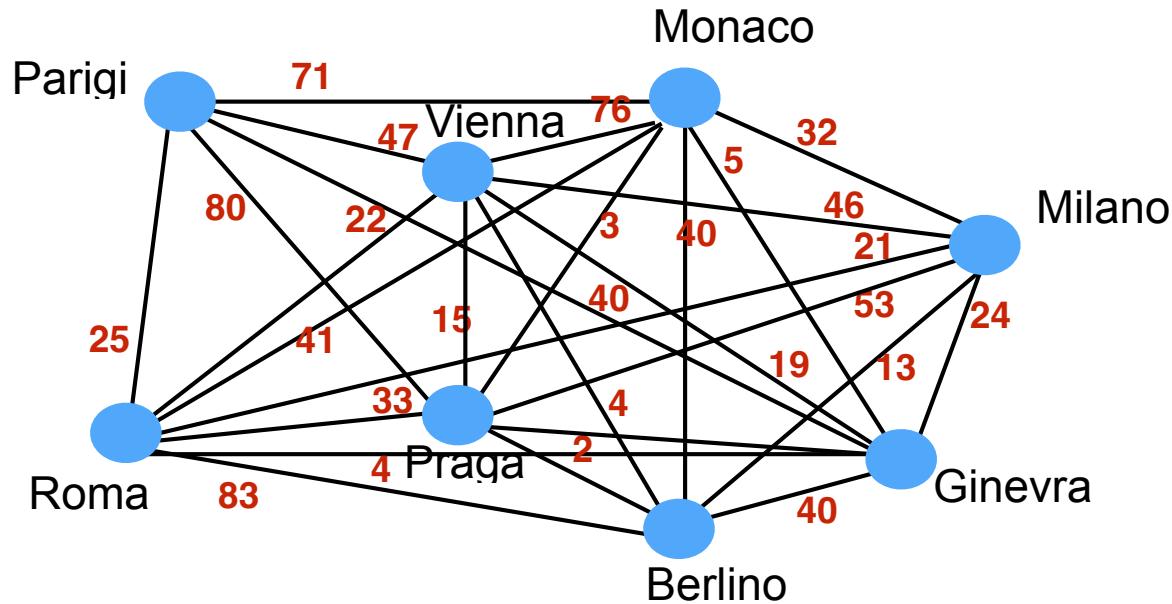


trovare un *ciclo* che passi su tutti gli archi del grafo **almeno** una volta e che sia di *lunghezza minima*

Dato questo problema, **la soluzione è un algoritmo** che, dato in input un qualsiasi grafo pesato G , fornisce in **output** un elenco di archi di G con la proprietà voluta

problema del commesso viaggiatore

TSP: Travelling Salesman Problem

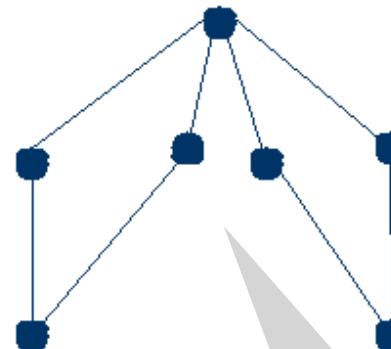
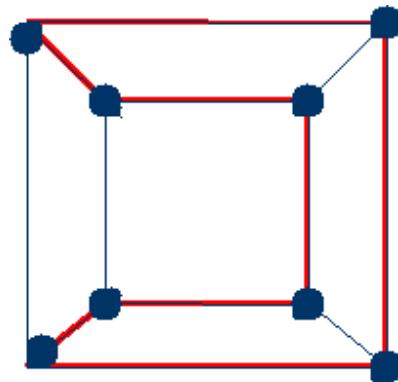


data una rete di città connesse tra loro con strade, **trovare** il percorso di **minore lunghezza** che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta **e tornare a casa**

Si cerca **un algoritmo** che, dato in input un qualsiasi grafo pesato G , fornisce in **output** un elenco di nodi di G con la proprietà voluta

problema del commesso viaggiatore

Semplifichiamo: togliamo i pesi dagli archi, e cerchiamo un ciclo che passa una e una sola volta per ogni nodo



Quindi non sempre esiste un cammino

È un problema difficile!

Più difficile del problema del postino

dovunque decida di partire, qui **non riesce** a visitare tutte le città e tornare a casa senza passare 2 volte per il nodo centrale.

Trovare il percorso del commesso viaggiatore = trovare un *ciclo hamiltoniano* in un grafo

cicli hamiltoniani e TSP

Problema 1: Dato un grafo G , G **ha** un ciclo hamiltoniano?

- è un problema di **decisione**
- **Soluzione/Algoritmo brute-force**: prendo l'elenco di **tutte** le permutazioni dei vertici di G e controllo per ognuna se è un ciclo in G
- complessità: se ho n vertici in G , ci sono $n!$ permutazioni possibili

Problema 2: Dato un grafo G e un cammino $c = v_1, \dots, v_n$, c **è** un ciclo hamiltoniano?

- è un problema di **verifica**
- **Soluzione/Algoritmo**: controllo che v_1, \dots, v_n siano tutti distinti e collegati da un arco in G
- complessità: se ho n vertici in G , $O(n)$

cicli hamiltoniani e TSP

Problema 1: Dato un grafo G, G **ha**

- è un problema di **decisione**
- **Soluzione/Algoritmo brute-force**: prende l'elenco delle permutazioni dei vertici di G e controlla per ognuna se è un ciclo in G
- complessità: se ho n vertici in G, ci sono **n!** permutazioni possibili

$$5! = 120 \sim 10^2$$

$$10! = 3.628.800 \sim 10^6$$

$$20! = 2.432.902.008.176.640.000 \sim 10^{18}$$

intrattabile!

Problema 2: Dato un grafo G e un cammino $c = v_1, \dots, v_n$,
c è un ciclo hamiltoniano?

- è un problema di **verifica**
- **Soluzione/Algoritmo**: controllo che v_1, \dots, v_n siano tutti distinti e collegati da un arco in G
- complessità: se ho n vertici in G, O(n)

Classi di Complessità

P

problemi che possono essere
decisi con un algoritmo
“veloce” (polinomiale)

NP

problemi che possono essere
verificati con un algoritmo
“veloce” (polinomiale)

Problema del postino cinese

TSP: Travelling Salesman Problem

una “decisione veloce” del TSP
non è stata **ancora** trovata

Classi di Complessità

P

problemi che possono essere
decisi con un algoritmo
“veloce” (polinomiale)

NP

problemi che possono essere
verificati con un algoritmo
“veloce” (polinomiale)

P = NP
?

una “decisione veloce” del TSP
si potrà mai trovare?

Se si dimostra che non esiste
allora P != NP

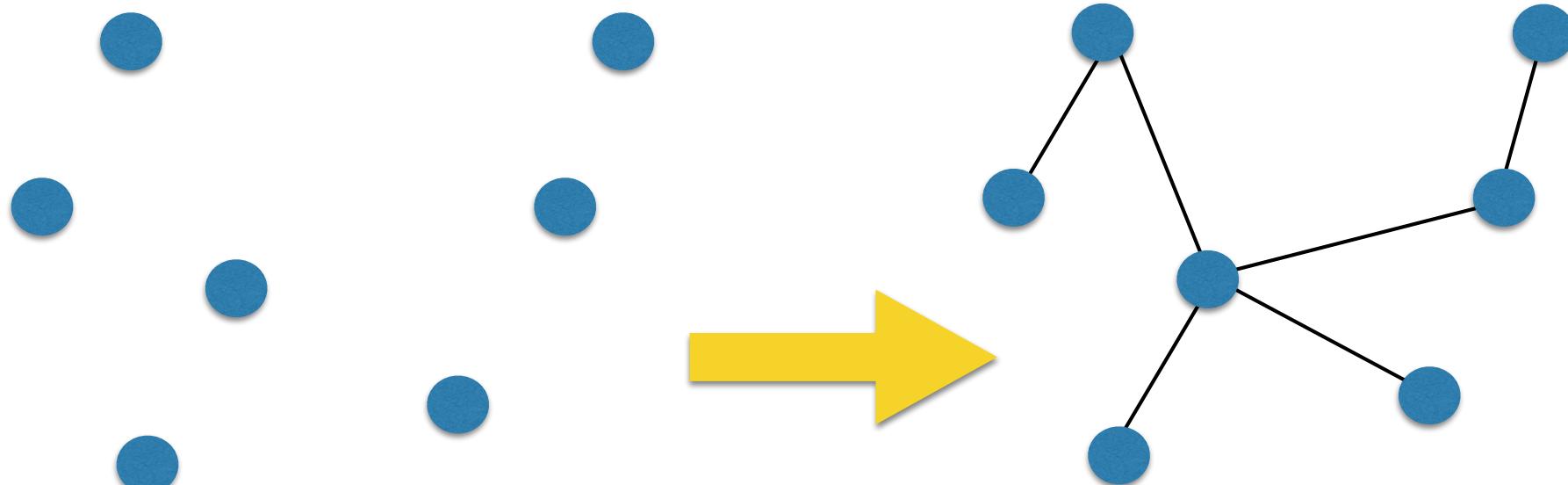


Clay Mathematics Institute

connector problem

Design a railway network connecting a number of cities, with a minimum possible construction cost.

Or electrical network, cable TV, gas, etc.



connector problem

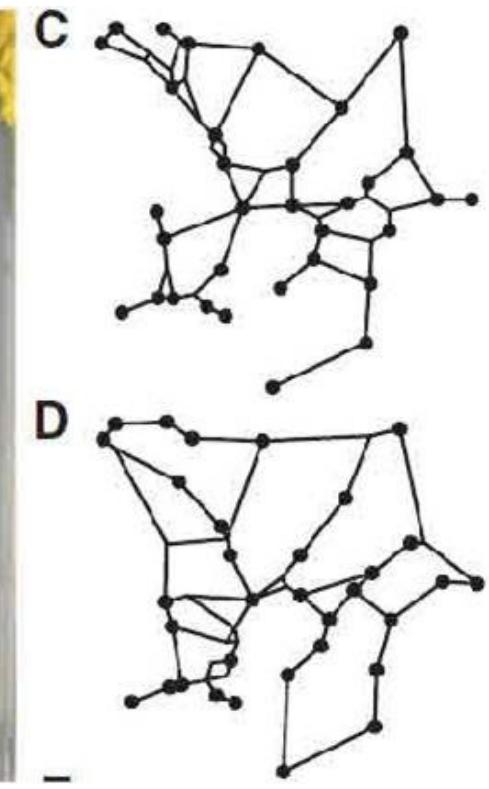
la muffa (unicellulare)
Physarium Polycephalum

reti comparabili per

- costo (lunghezza archi)
- efficienza (cammini minimi)
- resilienza (ridondanza analoga)

ma la muffa ha costruito la sua rete
in **modo distribuito e senza alcun
coordinamento globale!**

- *self-organization*
- *self-optimization*
- *self-repair*



processo di sviluppo del software

diversi livelli di astrazione



un errore può stare
su ciascuno dei livelli !!

chiara? corretta?
completa?

1. **Specific**a del programma: "Il programma prende in input un numero e ritorna in output se è primo oppure no"



2. **Algoritmo** risolutivo:

- se n è 1 oppure 2, allora n è primo
- altrimenti:
 - dividi n per 2, poi per 3, poi per 4, fino a dividere n per $n/2$
 - se una di queste divisioni ha resto 0 allora n non è primo
 - altrimenti (cioè tutte le divisioni hanno resto non 0) n è primo

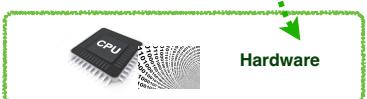
correttezza e
complessità

3. **Implementazione**

```
n = int(input())
if n==1 or n==2:
    print("n primo")
else:
    div = 2
    resto = n%div
    while (resto != 0 and div < n-1):
        div=div+1
        resto = n%div
    if resto == 0:
        print("n non primo")
    else: print("n primo")
```

errori sintattici e
logici

Sistema Operativo





Cliente



Utente

sistemi software aziendali

problemi di:

- comunicazione
- gestione processi

1. **Specifica** del programma:

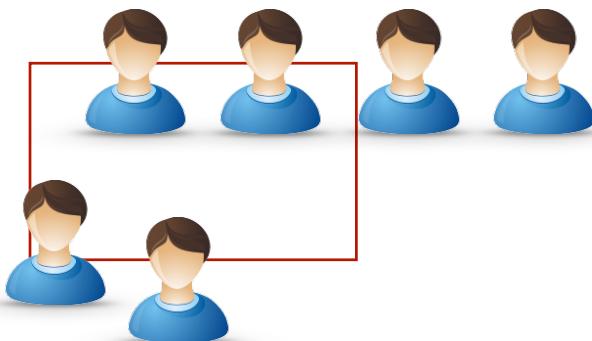
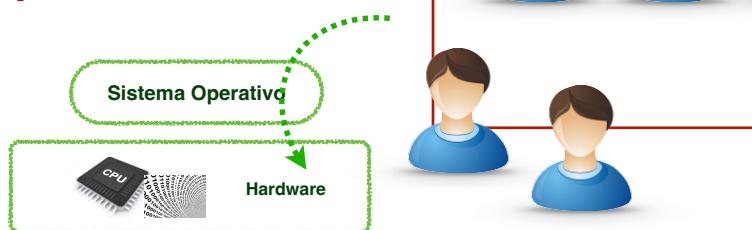


mix di aspetti
tecnici - aziendali - umani

2. **Algoritmo** risolutivo:

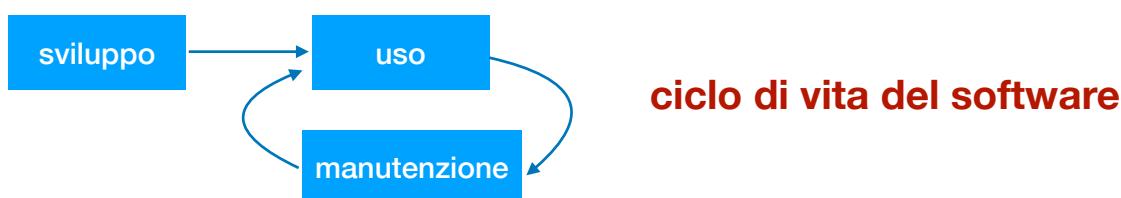


3. **Implementazione**



Ingegneria del software

- branca dell'informatica che si occupa dello **sviluppo di sistemi software grandi e complessi** (es. banca, ospedale, satellite...)
- **molte persone** che li sviluppano, vengono **mantenuti per lungo tempo**: si fanno **aggiornamenti** e possono **cambiare i requisiti** e il personale che ci lavora

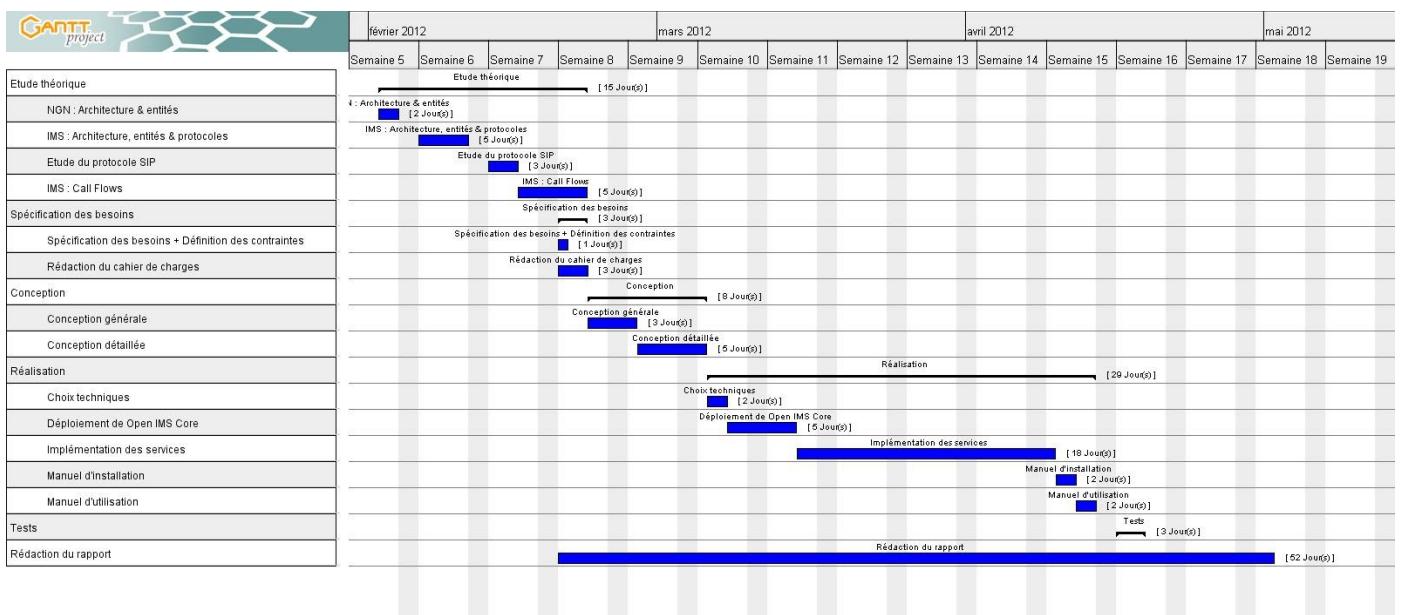


- spesso **chi fa la manutenzione non è più chi ha scritto il codice**.
- La **qualità del codice** prodotto ha a che fare anche con la **facilità di mantenerlo**: **codice chiaro e ben documentato**
- Spesso è più facile rifare un sistema che modificarlo con successo.

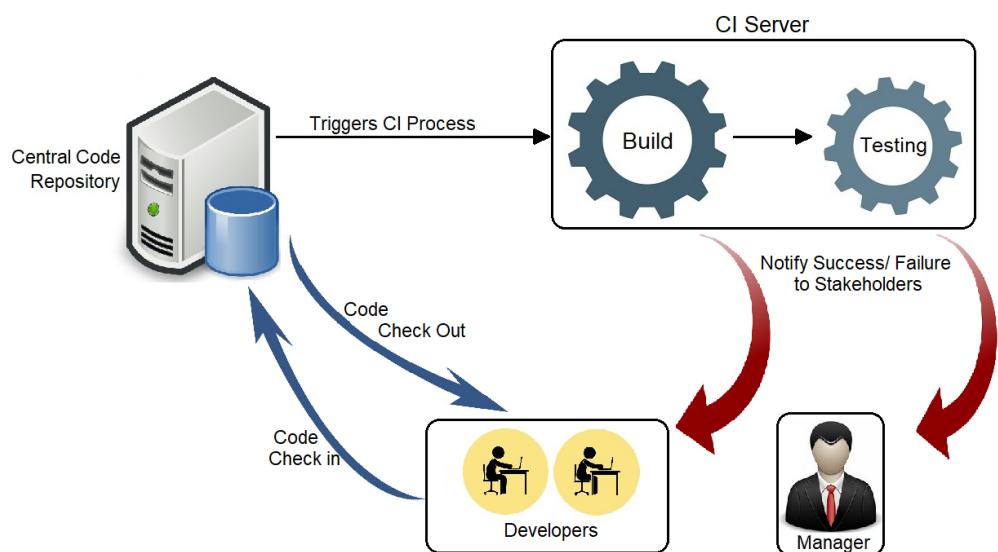
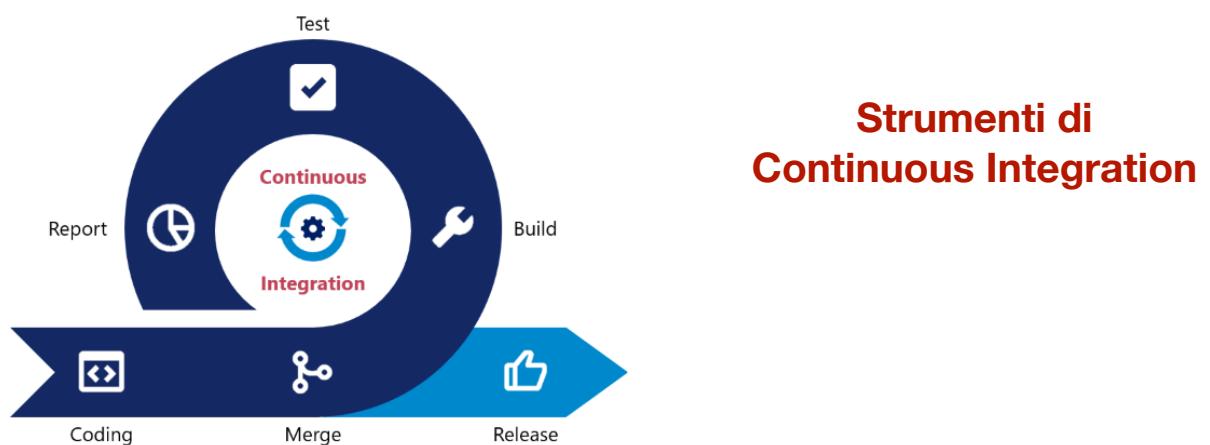
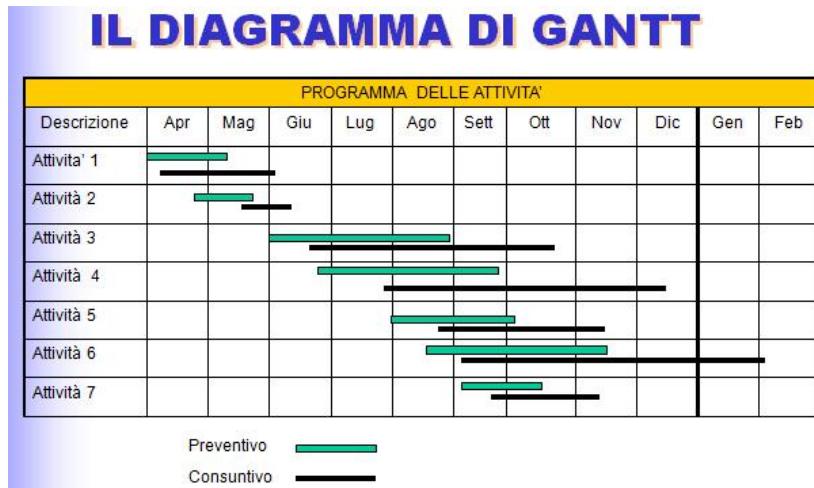
Ingegneria del software

- **Come si progetta e come si soprintende allo sviluppo?**
 - come si fa una **stima dei costi**? in termini di **denaro, tempo, altre risorse**
 - come si **suddivide il lavoro** in sottoprogetti gestibili?
 - come ci si assicura che i prodotti dei **sottoprogetti siano compatibili**? e se cambia un sottoprodotto?
 - **come comunicano** coloro che collaborano ai sottoprogetti? possono **aggiornare contemporaneamente** diversi moduli software?
 - come si misura l'avanzamento dei lavori?

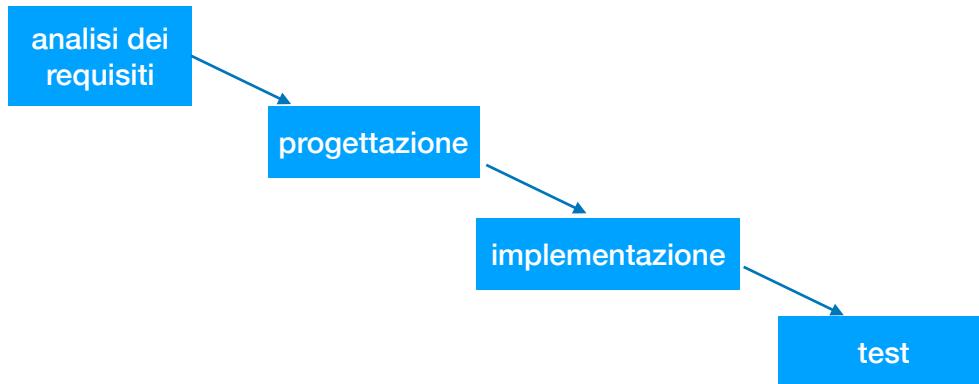
Strumenti di Pianificazione



Strumenti di Pianificazione

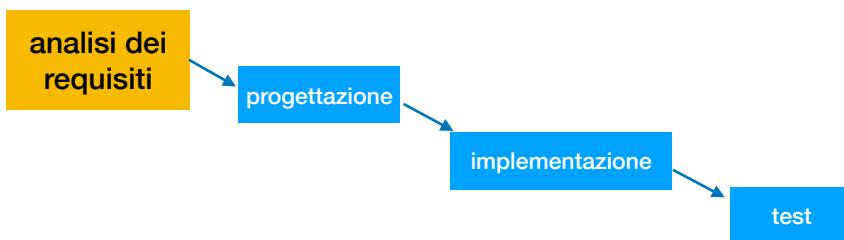


fasi di sviluppo



- L'Ingegneria del Software prevede delle **linee guida** (*standard di qualità*) **per ogni fase**
- **ogni fase deve essere documentata** in modo preciso

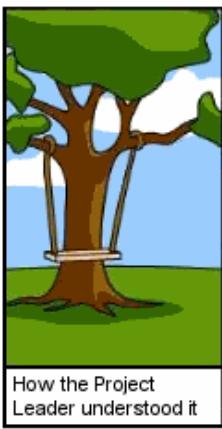
Analisi dei requisiti



- L'**analisi dei requisiti** serve a:
 - **specificare i servizi** che saranno forniti dal sistema, come ci si interagirà
 - identificare **le condizioni** di tali servizi, es. vincoli di tempo, di sicurezza..
- Prevede un significativo input da parte degli **stakeholder** (le parti interessate): gli **utenti** ma ad esempio anche l'**ufficio legale** o il **settore finanziario**.
 - I requisiti si raccolgono tramite un **questionario** per l'utilizzatore, uno **studio di fattibilità**, un'**indagine di mercato**....



How the customer explained it



How the Project Leader understood it



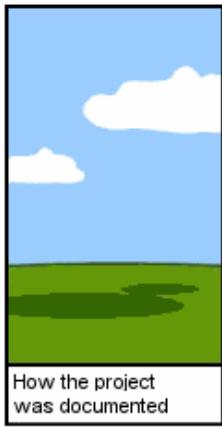
How the Analyst designed it



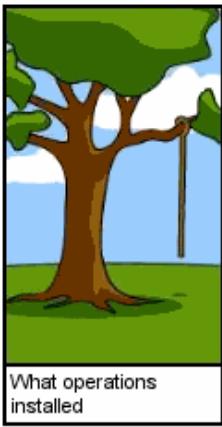
How the Programmer wrote it



How the Business Consultant described it



How the project was documented



What operations installed



How the customer was billed

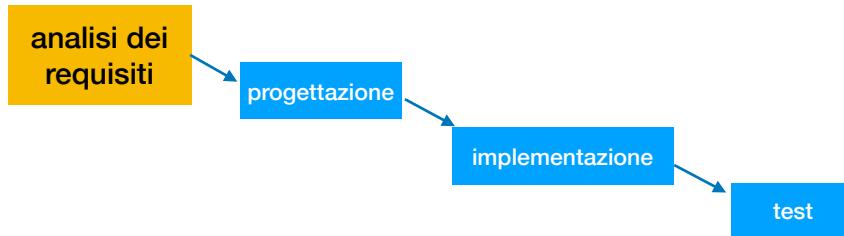


How it was supported



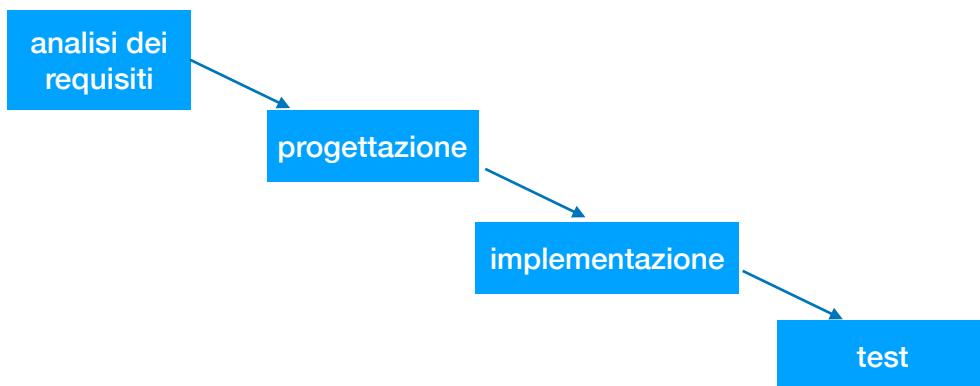
What the customer really needed

Analisi dei requisiti



- serve **negoziare compromessi tra desideri, bisogni, costi e fattibilità.**
- infine si compila un **documento di specifica dei requisiti del software**, che **costituisce un accordo scritto tra le parti:**
 - guida allo sviluppo del software e si usa per dirimere eventuali controversie
- Spessissimo i requisiti iniziali sono stati **incompresi**, vanno **modificati o integrati**, causando **ritardi, costi ed errori**
- Serve una comunicazione continua e diretta con le parti interessate al progetto

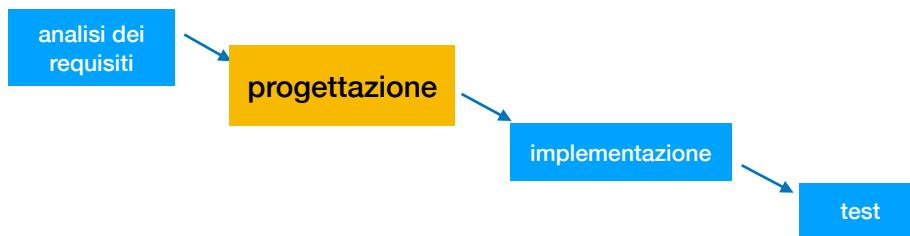
Fasi di sviluppo



Esempi: Uniweb oppure Moodle

- Provare a pensare cosa significano queste diverse fasi nello sviluppo e nella manutenzione di due software complessi come Uniweb e Moodle
- Pensare al ciclo di vita di questi software

Progettazione

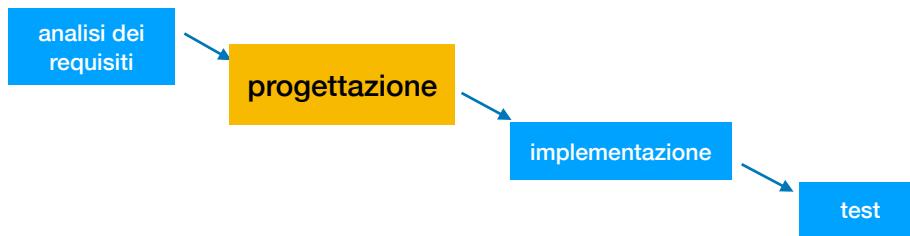


- Se l'**analisi dei requisiti** specifica il problema da risolvere, ***il che cosa***, la **progettazione** specifica la soluzione al problema, ***il come***
- in questa fase si **definisce la struttura interna del sistema software**, ad un livello di dettaglio tale da poter essere convertito poi in programma.

Esempio: è richiesto il **sito Web di un negozio di mobili**

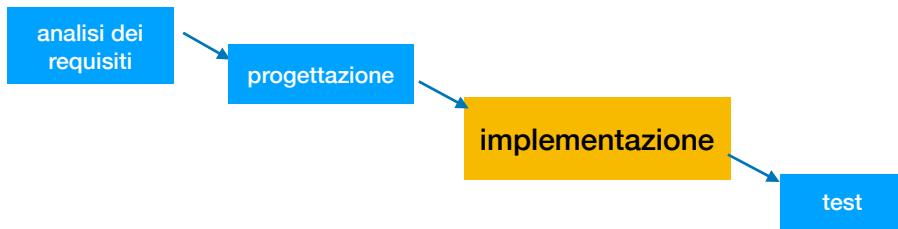
- interfaccia del sito
- un sistema di acquisto online
- integrazione con canali social
- un planner di mobile componibile

Progettazione



- Se l'**analisi dei requisiti** specifica il problema da risolvere, ***il che cosa***, la **progettazione** specifica la soluzione al problema, ***il come***
- in questa fase si **definisce la struttura interna del sistema software**, ad un livello di dettaglio tale da poter essere convertito poi in programma.
 - Ci sono **varie metodologie e notazioni** per la progettazione, es. modelli e diagrammi di vario genere, che costituiscono la **documentazione** di questa fase. Ma ***non sono notazioni stabili e standard***, come ad esempio lo sono quelle degli architetti.

Implementazione



- **Stesura del codice** vera e propria.
- Creazione di file di dati e sviluppo di **database**

test



- testing del codice: per controllare che **non ci siano errori** e che **rispetti la specifica dei requisiti**
- servono test di **accuratezza di tutte le fasi di sviluppo**: **software quality assurance**
 - seguire buone prassi di raccolta requisiti,
 - qualità e aggiornamento della documentazione,
 - revisioni periodiche tra le diverse parti coinvolte nello sviluppo.

Documentazione

1. Documentazione utente: caratteristiche del prodotto e *come si usa*

- pensata per l'utente del software, quindi non tecnica
- manuale d'uso separato oppure incluso nel software (informazioni in piccoli *help package* visualizzabili mentre si usa il software, magari che compaiono in automatico se si indugia troppo)

2. Documentazione tecnica/di sistema: caratteristiche interne del software e *come mantenerlo*

- molto più tecnica, anche informazioni su come il software va installato, configurato e mantenuto, utili al tecnico amministratore di sistema
- ci sono strumenti automatici che aiutano a generare e mantenere aggiornati i documenti tecnici

test

Program testing can only show the presence of bugs, but not their absence!

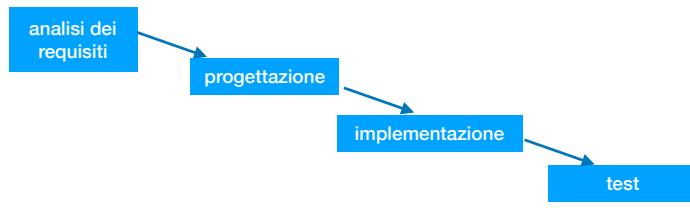


- testing del codice: per controllare che non ci siano errori e che **rispetti la specifica dei requisiti**

- servono test di **accuracy of all development phases: software quality assurance**
 - seguire buone prassi di raccolta requisiti,
 - qualità e aggiornamento della documentazione,
 - revisioni periodiche tra le diverse parti coinvolte nello sviluppo.

- nonostante il testing esaustivo restano quasi sempre errori, magari nascosti per anni ma che poi causano gravi danni

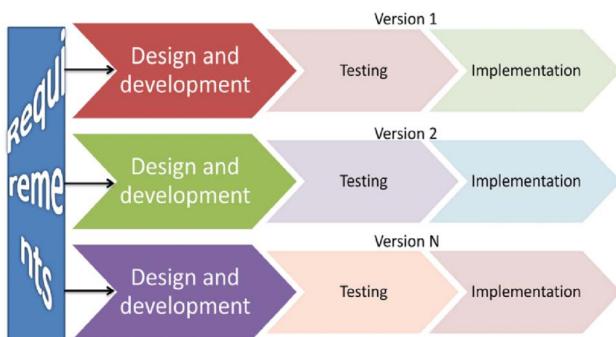
Modello a cascata



- Spesso i **requisiti iniziali** sono stati **incompresi**, vanno **modificati** o **integrati**, causando **ritardi**, **costi ed errori**
- Serve una comunicazione continua e diretta con le parti interessate al progetto

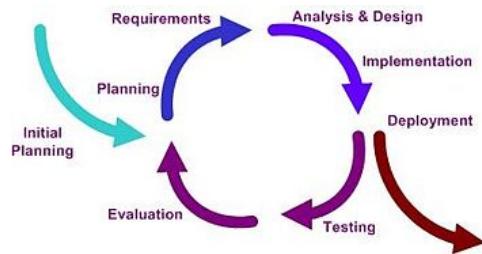
Modello incrementale

si **estende** una versione iniziale aggiungendo nuove funzionalità



Modello iterativo

si **raffina** una versione iniziale, magari partendo da un **prototipo** o un **mock-up** (versione non funzionante ma dimostrativa)



qualità del software

- estremamente complesso descrivere in maniera **precisa** ed **esaustiva**
 - **cosa deve fare** un programma (specifica/algoritmo)
 - **cosa fa** un programma e **come lo fa** (implementazione)
- e "dimostrare" che corrispondono!

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

qualità del software

- **garantire** che un programma, eseguito con *qualsiasi input*, non ha errori , è estremamente difficile:
 - **cos'è un errore?**
 - risultato errato, non terminazione, interruzione improvvisa (errore runtime), **falla di sicurezza**, **lentezza** delle prestazioni....
 - **errore rispetto a cosa?** specifica dei requisiti non facile da fare in modo chiaro, corretto, completo.
 - **qualsiasi input!** possono essere infiniti, testing non esaustivo
- **non è la stessa cosa** ad esempio **per l'ingegneria** meccanica, o altri tipi di tecnologie, che offrono maggiori garanzie di affidabilità.

Ingegneria del software

A differenza dell'ingegneria non digitale

- mancano delle metriche quantitative per **misurare le proprietà del software**:
 - **quanto è corretto? quanto è grande? quanto consuma?** hanno spesso poco senso, o un senso **poco confrontabile** con altri software

- **approccio applicativo**,
 - test empirici, **linee guida** e buone prassi, standard di **qualità di processo**
- **approccio teorico**
 - **dimostrare** (in modo automatico) l'assenza di **specifici** errori

ancora grossi problemi di affidabilità

Collection of Software Bugs

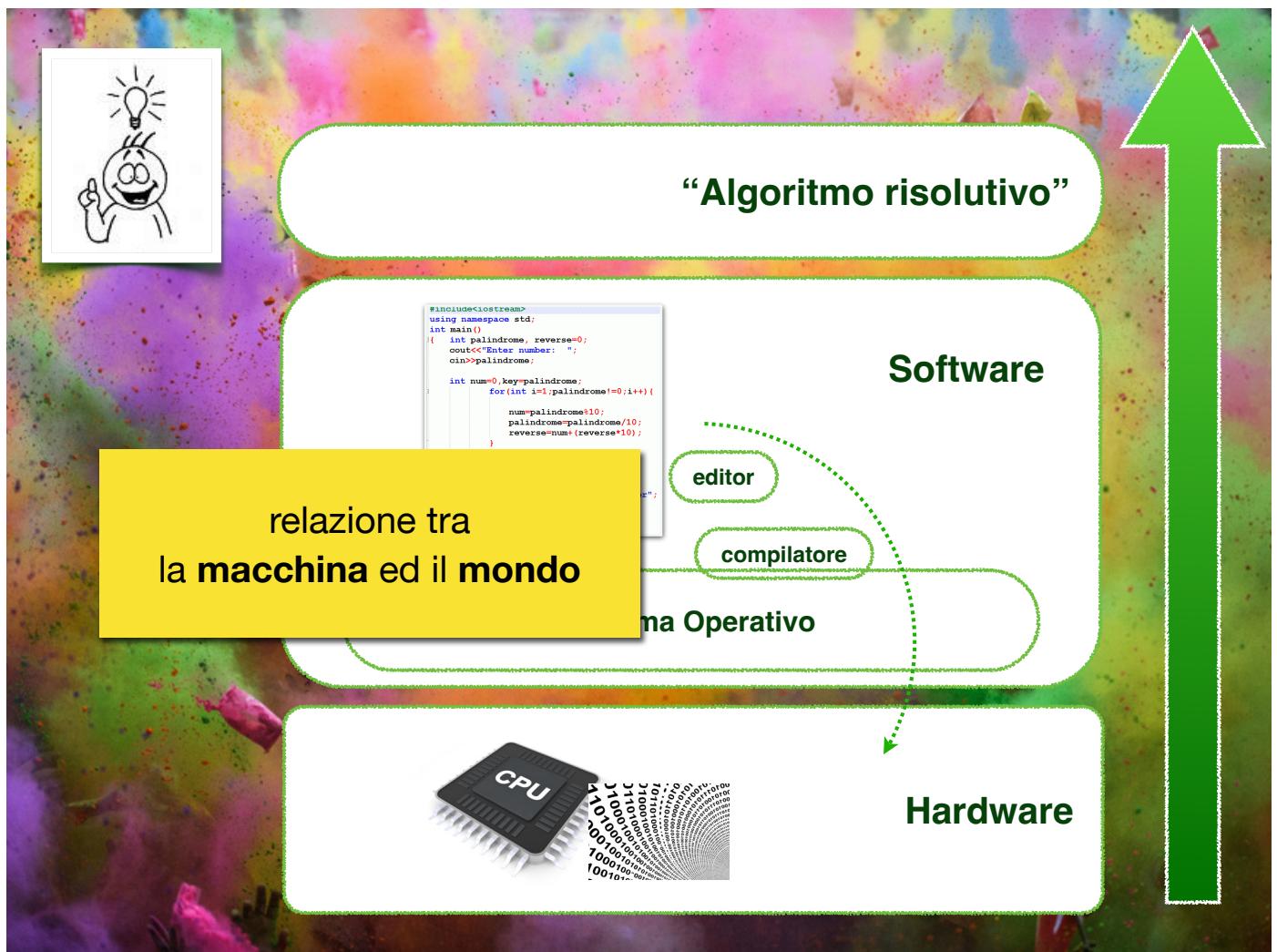
<https://www5.in.tum.de/~huckle/bugse.html>

- ingenti danni in settore bancario, aerospaziale, medicina, aviazione...
 - The DAO e smart contract ...*errore o feature del codice?*
 - **non solo bug**...anche pessime pratiche di sviluppo e uso di tool (e.g. fogli excel).
-
- **Documentazione in ogni fase!!!**



Comunicare il software

- Come si comunica efficacemente il proprio software?
 - che significa comunicare?
 - comunicare **cosa**?
 - il **codice**, la **specific**a, in che **contesto** va usato, che problemi risolve, che dipendenze ha...
 - a **chi** comunicare?
 - un programmatore, un utente, un commerciale, un giudice



Verso la conclusione...

Consapevolezza Digitale

Su cosa abbiamo riflettuto:

- che significa **sviluppare** software
- che significa **eseguire** software
- che significa **valutare** un software (correttezza, efficienza, qualità)
- che significa **comunicare** un software

La tecnologia non è neutrale

le tecnologie che usiamo per **mediare le relazioni** con gli altri *esseri umani*, gli *oggetti* e i *luoghi* che abitiamo, **influenzano** comportamenti, modi di lavorare, imparare, comunicare e divertirsi ...pensare.

Consapevolezza Digitale

Serve sviluppare due capacità importanti:

1. la capacità di guardare ai sistemi digitali con uno **sguardo attento ai diversi livelli di astrazione di cui sono composti**, distinguendo le **criticità** che dipendono:
 - dai **requisiti di progettazione**,
 - dalla specifica **logica di funzionamento**,
 - dalla correttezza **dell'implementazione**,
 - **dall'ambiente di esecuzione e manutenzione**,
 - oppure **dal contesto di uso**. (AI, "Soluzionismo Digitale")



Consapevolezza Digitale

2. la capacità di **chiedere conto di come funziona un sistema software**, al fine di comprendere:
 - **come l'informazione è rappresentata nei suoi *dati*,**
 - **che manipolazioni vengono effettuate,**
 - **cosa viene *trascurato*,**
 - **chi e come ha fatto le *scelte* progettuali e implementative,**
 - **che garanzie di *qualità* offre il software.**

Servono dunque in tutti gli ambiti professionisti capaci di approcci scientifici interdisciplinari, capaci di **riconoscere la propria corresponsabilità** nell'impatto sociale delle tecnologie digitali.

The End