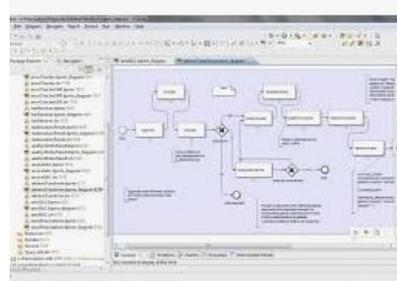




Introduzione alla programmazione informatica



```
import os, codecs, pymysql.cursors#, metro
import numpy as np
import pandas as pd
import json

if __name__ == '__main__':
    # Connect to the database
    connection = pymysql.connect(host='lo
                                user='ro
                                password
                                db='oad'
                                cursorcl
```



Emilio Sulis

Contenuti: introduzione alla programmazione

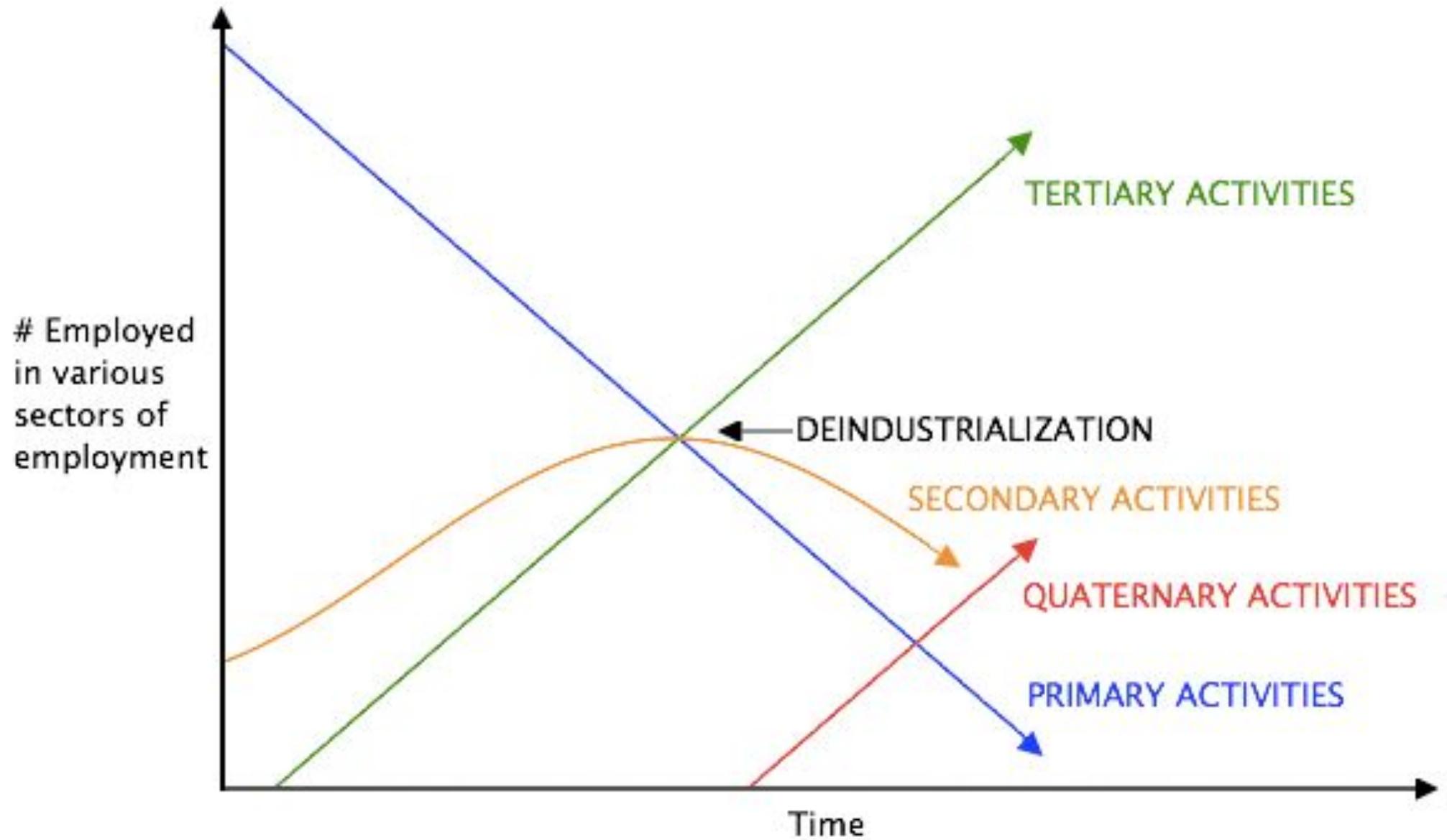
1. Pensiero computazionale - introduzione

2. La codifica dell'informazione

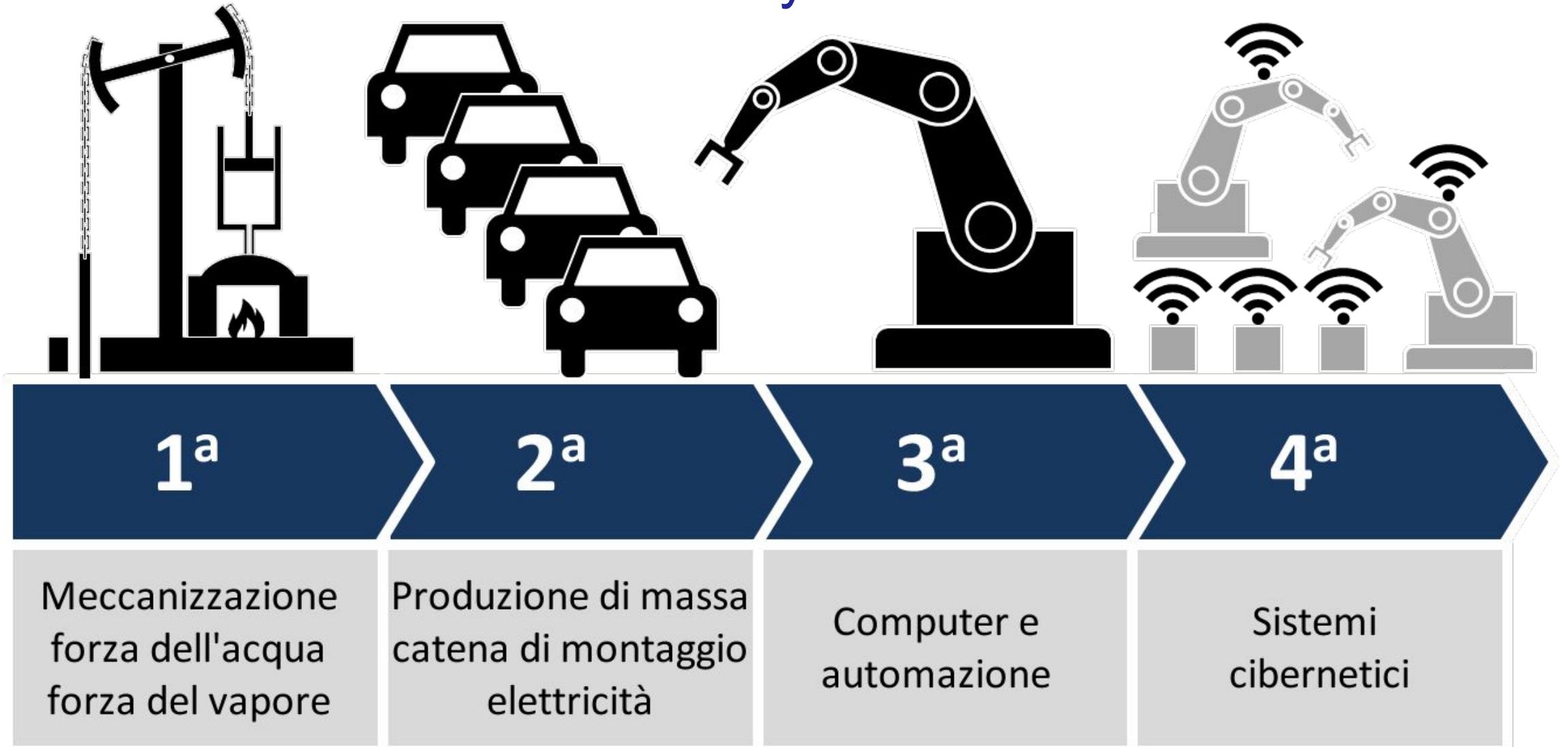
3. Coding e linguaggi visuali

4. Ingredienti principali

Attività quaternaria ?



Industry 4.0



>>> INTELLIGENZA ARTIFICIALE

Recommendation



Prediction

SEPTEMBER 30, 2013

The Iran Opportunity By Fareed Zakaria / E-Cigarettes / \$20K Homes

TIME

CAN
Google
SOLVE

DEATH?

The search giant is launching a venture to extend the human life span.

That would be crazy—if it weren't Google

By Harry McCracken and Lev Grossman

time.com

Algoritmo

>> una sequenza finita di “passi”
per risolvere un determinato problema

An **effective method** that can be expressed within a **finite** amount of **space** and **time** and in well defined **formal** language for calculating a function. Starting from an **initial state** and **initial input** (perhaps empty), the instructions describe a **computation** that, when **executed**, proceeds through a finite number of well-defined **successive states**, eventually producing «**output**» and terminating at a final **ending state**.



L'algoritmo si sostanzia in un programma

A **collection of instructions** that performs a specific task when executed by a computer.



A computer program is usually written by a **computer programmer** in a **programming language**.

Computer Programming

"A **process** that leads from an original formulation of a **computing problem** to **executable computer programs**"

fonte: *Wikipedia* *(en)*

Processo -> insieme di attività o compiti correlati tra loro, che producono un determinato prodotto o servizio [e.g. flowchart]

Problema computazionale -> problema risolvibile attraverso l'uso dei computer

Programma eseguibile -> codice, automatizzare

Computational thinking

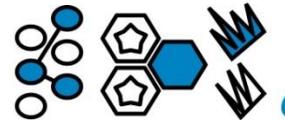
decomposition

solve a problem by **breaking** it into smaller groups



pattern recognition

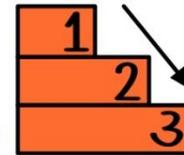
find the **order**



analyze the data

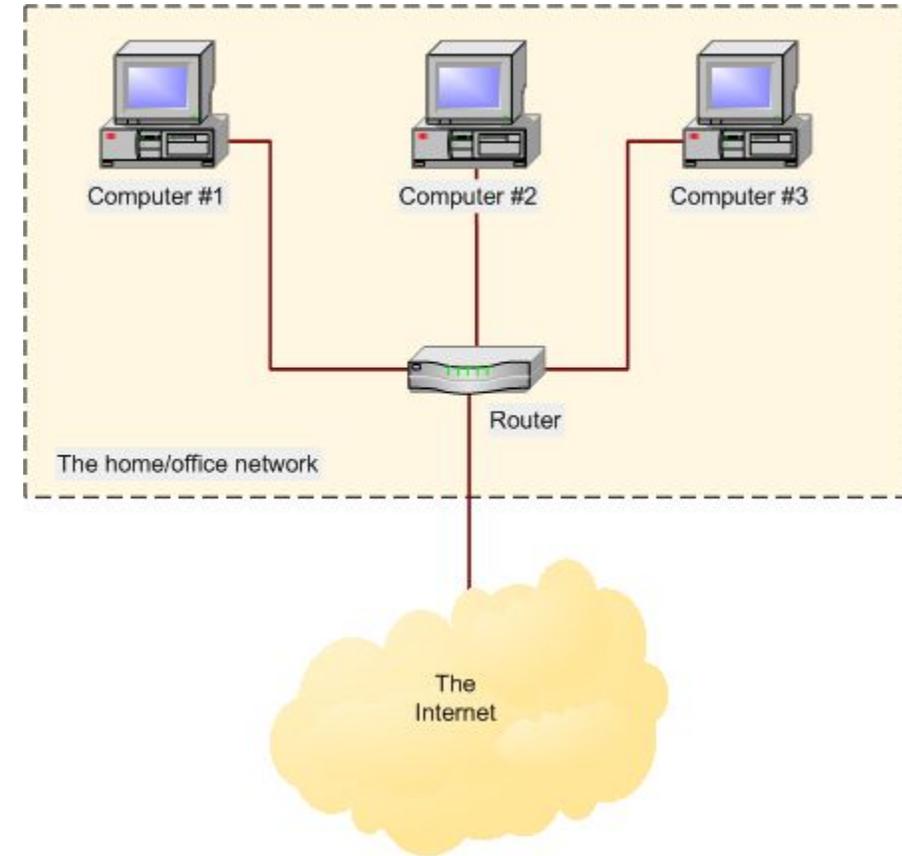
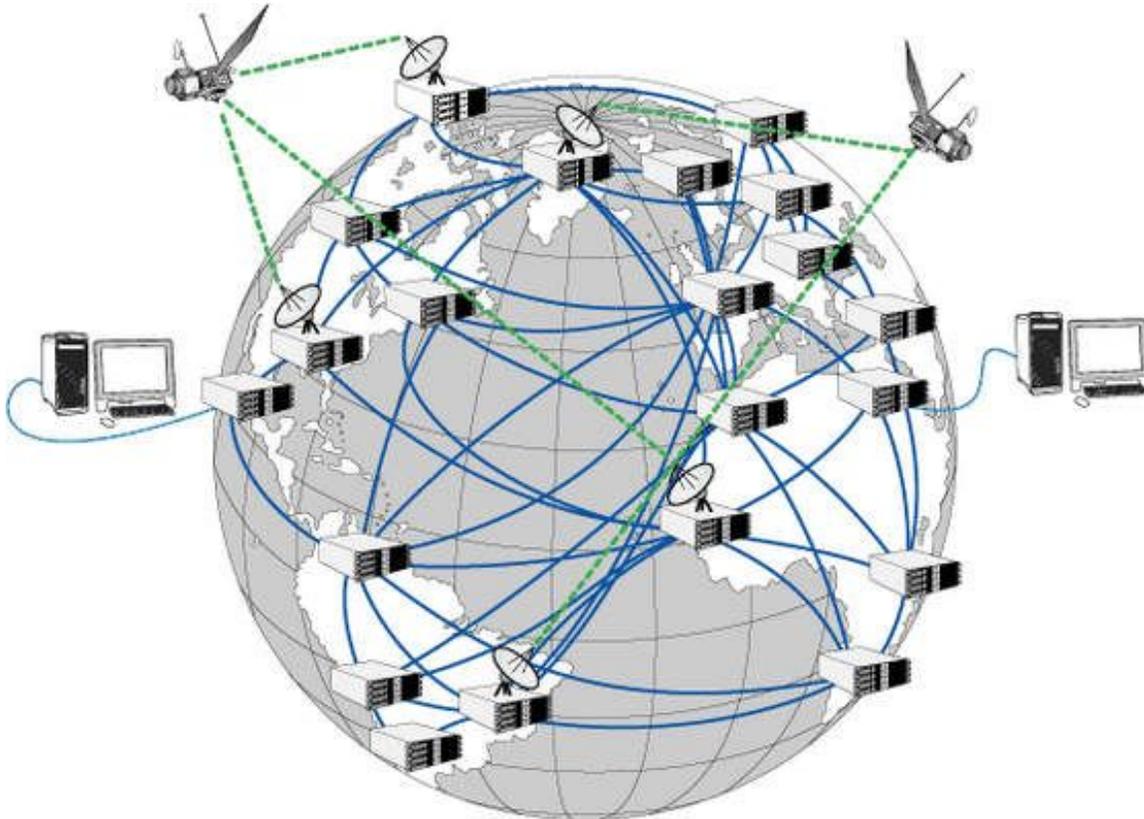
algorithmic design

creating solutions using a series of ordered **STEPS**



Internet

- ❖ **Internet è la rete fisica** di dispositivi e *calcolatori*
- ❖ La comunicazione avviene con il protocollo (TCP/IP) che permette di connettere/trasmettere dati
- ❖ il WEB è un'applicazione sulla rete Internet (altre sono la posta elettronica, condividere file - FTP ecc.)



- ❖ *Internet of Things* connessione alla rete Internet di oggetti fisici di utilizzo quotidiano (elettrodomestici, dispositivi medici o indossabili, smart ecc.)

A livello sociale: cambio di paradigma

Collaborazione, mutualità (vs segretezza, chiusura)

Condivisione delle informazioni (social web, stackoverflow.org, file sharing)

Apertura (openness, Linked Open Data, trasparenza)

Progetto GNU (vs Microsoft/Apple)

Sistema operativo Linux (vs Windows/MacOS)

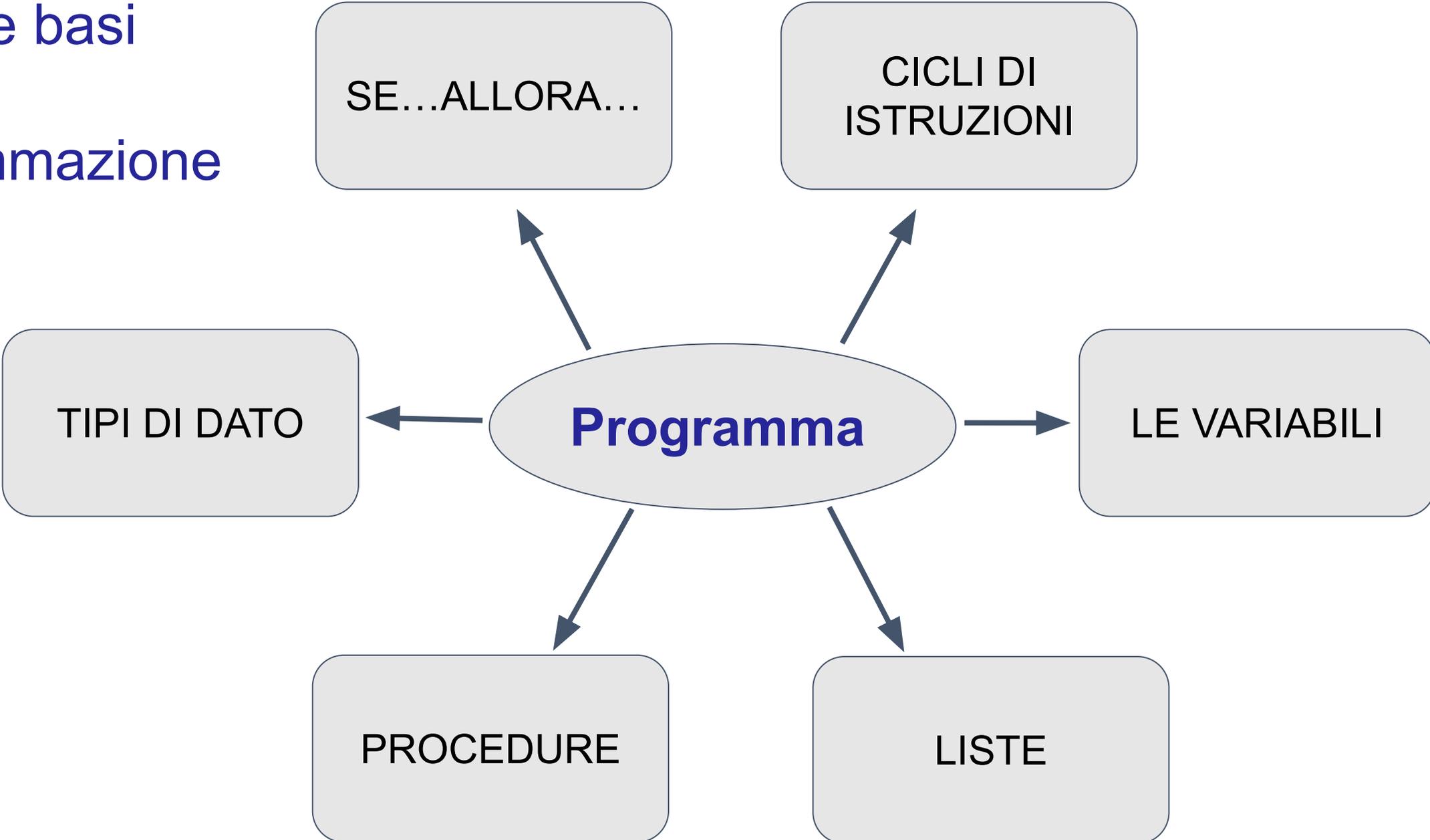
LibreOffice (vs Office)

Hacking, Wikileaks

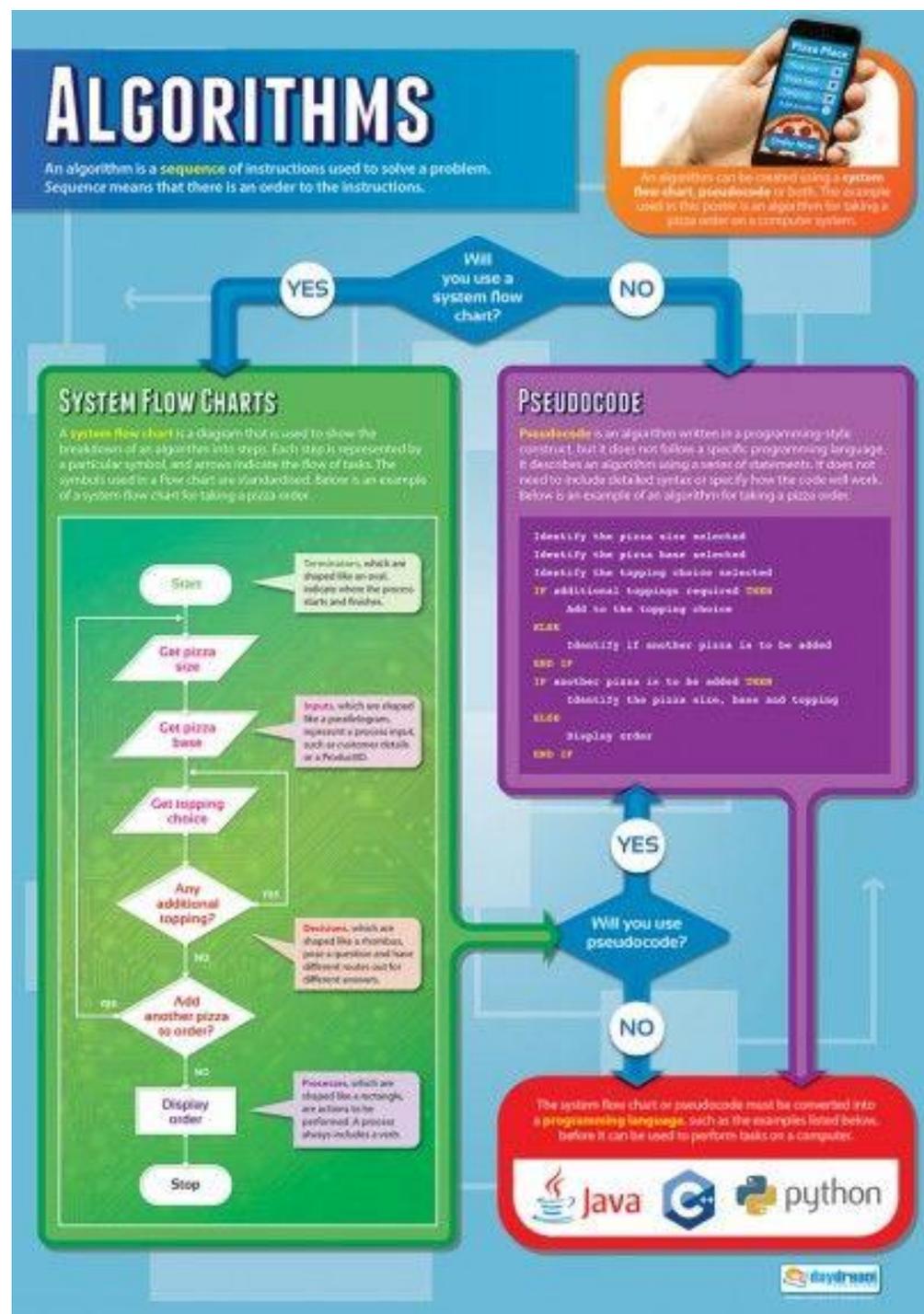
Autonomia, MOOC



Capire le basi
della
Programmazione



Capire le basi della Programmazione



E nella quotidianità?

ESEMPI

- *Una decisione, un evento, un problema da "risolvere"*

Passaggi

- *Variabili, liste, dizionari*
- *Strutture dati e di controllo, cicli, procedure*
- *Risultati attesi/eccezioni*



La programmazione informatica e la vita quotidiana

Somiglianze con un caso reale ed esempi di variabili

Caso della “*spesa al supermercato*”:

prezzo di un nuovo **prodotto** da acquistare, **somma**, **sconto**, **budget**

Implementazione? Esempi con strutture di controllo/cicli

Esempio1

Fino a quando **(somma + (prodotto - sconto)) < budget** {

somma = somma + (prodotto - sconto);

}

↘ come traduco questa espressione in un programma?

Altri casi:

La sveglia al mattino, andare al lavoro, preparare la pasta...



python™

<https://www.python.org/>

Un emulatore online (comodo, ma utile soltanto a fini didattici): <https://pythonsandbox.com>

PYTHON - vantaggi

Python è uno dei linguaggi più potenti e utilizzati oggi

Esistono distribuzioni da installare con “librerie” (es. **Anaconda**)

Si compila anche con Editor specifici (es. PyCharm, **Spyder**, ecc.)

Facilissimo da imparare/usare!

Java

```
Java
1 public class Main {
2     public static void main(String[]
   args) {
3         System.out.println("hello wor
   ld");
4     }
5 }
```

Python

```
Python
1 print("hello world");
```

<https://aulab.it/notizia/300/cose-python-e-perche-e-cosi-popolare>

La programmazione informatica e la vita quotidiana (in python)

Dichiarazione delle **variabili**

```
lista_prodotti = [15,10,14,3,7,18,12,22,16,9]
```

```
somma = 0
```

```
sconto = 1
```

```
budget = 100
```

Ciclo FOR e struttura di controllo condizionale IF

```
for prodotto in lista_prodotti:
```

```
    if (somma + (prodotto - sconto)) <= budget :
```

```
        print("Acquisto anche il prodotto che costa " + str(prodotto) + " euro!")
```

```
        somma = somma + (prodotto - sconto)
```

```
print("Ho speso in totale: ", somma)
```

>> e se lo sconto aumenta a 2 cosa succede?



Altro esempi di algoritmo e programma

La ricerca di una pagina all'interno di un libro

algoritmo >>> In genere si apre il libro a metà, si cerca di capire se la pagina di nostro interesse si trova prima o dopo quella che stiamo osservando, per poi continuare tale ricerca solo sulla parte del libro interessata.

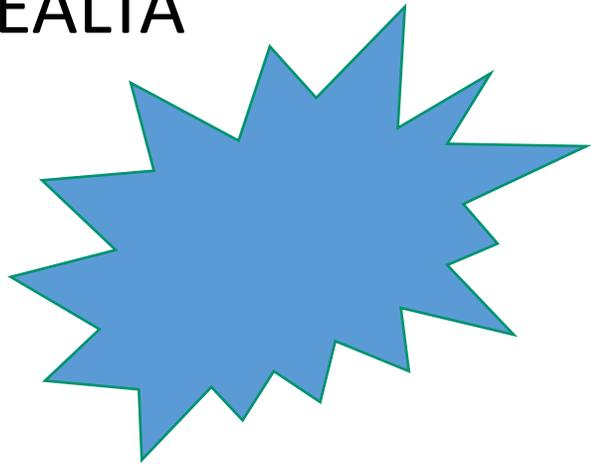
programma >>> implementa un **algoritmo di ricerca dicotomica**, che si applica a qualsiasi ricerca in un insieme di elementi ordinati (come le pagine di un libro).

Programmazione e modellazione

Programmare é preceduto da semplificare/modellare la realtà

Cosa si intende con *modellare*?

REALTA'



interpretazione



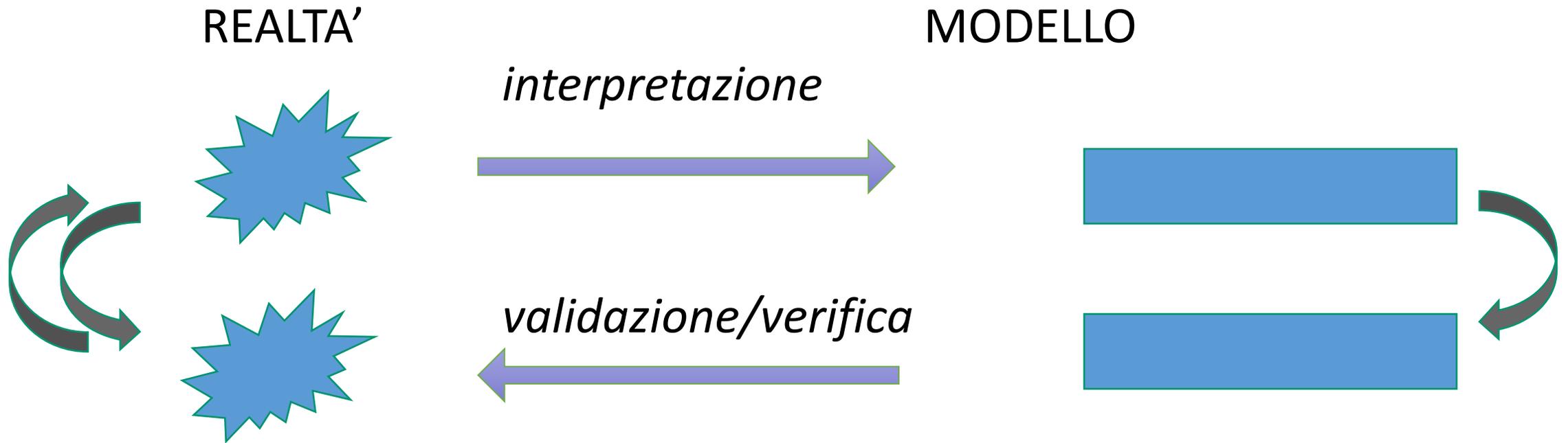
MODELLO



Dall'**osservazione** del mondo reale, si interpreta e costruisce un modello

Modellare un processo reale

Dalla realtà al modello (e ritorno):



Dove usare la modellazione?

Ciclo di vita del BPM

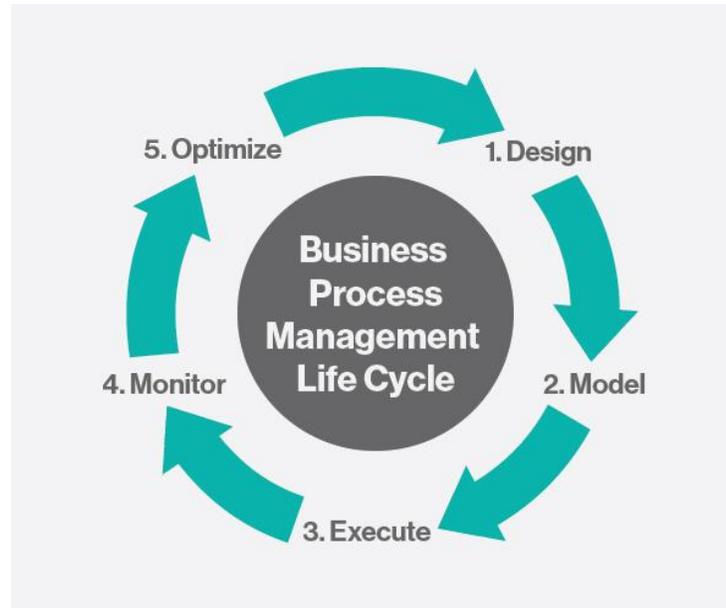
Disegno (design)

Modellazione (model)

Esecuzione (execute)

Monitoraggio (monitor)

Ottimizzazione (optimize)



Organizzazione: come modellare un processo?

Notazione BPMN 2.0

Il consorzio internazionale Object Management Group (OMG) e l'organizzazione BPMI hanno proposto notazione standard **Business Process Modeling and Notation**.

Nel 2011 viene rilasciata la versione ufficiale **BPMN 2.0**.

<http://www.omg.org/spec/BPMN/2.0/About-BPMN/>

Principali formalismi grafici della notazione BPMN 2.0

Il processo può essere scomposto in corsie (swimlanes/lanes)

Corsie (pool e lane)

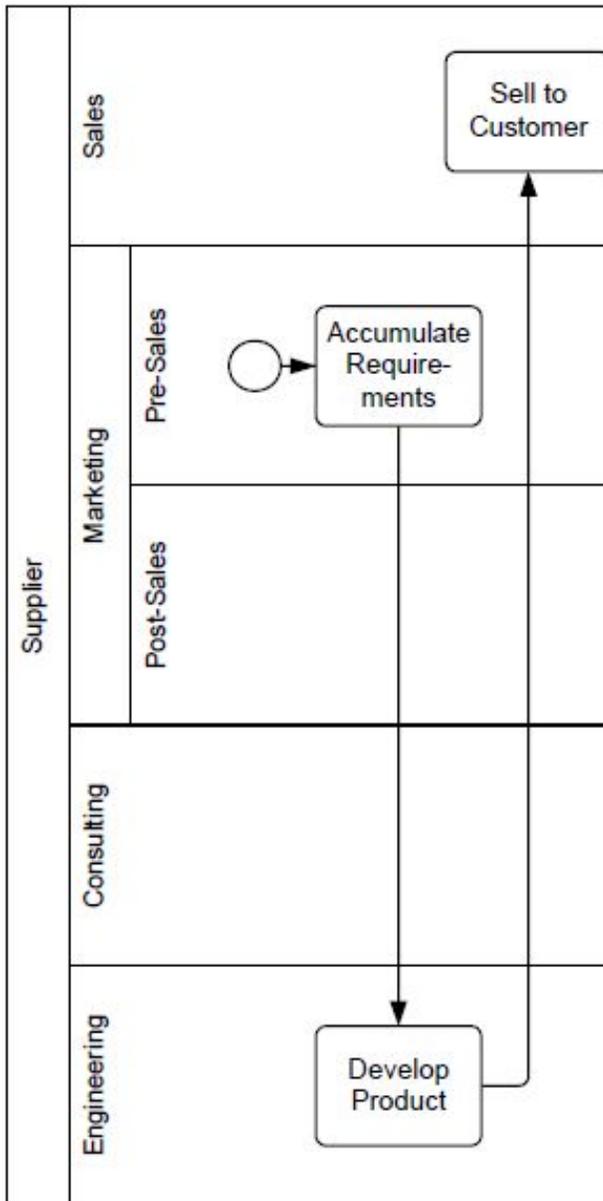
Le forme principali distinguono:

Attività (task)

Eventi (events)

Decisioni (gateway)

Il flusso tra gli elementi grafici (forme) è gestito da **frecce**



Pool e lane

Un'unità organizzativa (**pool**) può essere rappresentata da un'area rettangolare (es.: Supplier).

Le corsie (**lane**) sono a loro volta aree rettangolari che raggruppano le attività simili, quali reparti/aree funzionali di un'azienda ecc.

Ogni lane contiene quanto si verifica in quell'unità organizzativa. (es. Sales, Marketing, Consulting ecc.), separandolo da quanto è responsabilità di altri e sarà quindi contenuto in altre corsie.

Eventi (events)



Inizio (start) del processo, con l'arrivo di una *transazione* (Es. un'istanza, un utente, una chiamata ecc.)

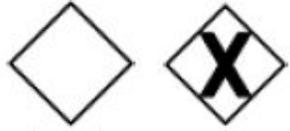


Ritardo (timer), evento che interrompe il flusso. Attività che riguardano personale diverso da quello del processo in esame (es. Ritardo per l'attesa dei risultati di laboratorio sugli esami del sangue)



Fine: evento conclusivo del processo

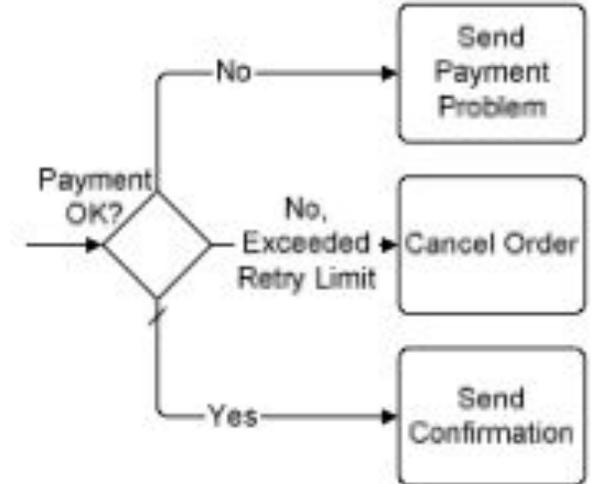
Decisioni (gateway)



Decisione esclusiva (**gateway esclusivo**) tra due o più uscite

Il flusso deve seguire soltanto una delle uscite

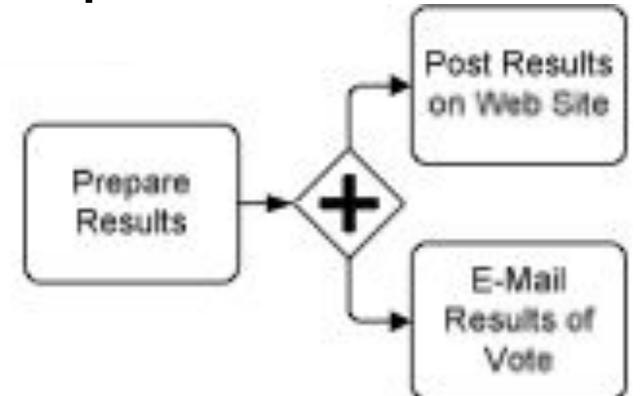
Esempio: decide in base all'avvenuto pagamento



Decisione inclusiva (**gateway inclusivo**) tra due o più uscite

Il flusso deve proseguire in tutte le uscite

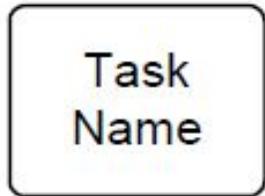
Esempio: effettua le due attività previste



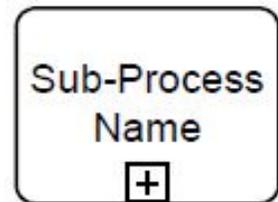
Attività (task)



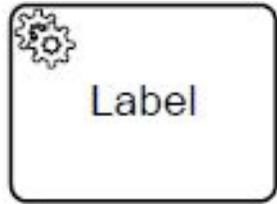
Indica quanto viene svolto durante l'esecuzione di un processo



*Possono essere **attività semplici** o sotto-processi:*
Attività “atomiche”, non ulteriormente scomponibili



Sotto-processi (composti da più task)



Indica un'attività di servizio (svolta da un servizio automatico, una funzione di un programma). *Esempio: il calcolo di un tasso di credito ottenuto da un'agenzia.*



Attività che prevede l'invio di un messaggio



Attività che prevede la ricezione di un messaggio



Attività che l'utente deve necessariamente completare prima che sia possibile proseguire con le altre attività. *Esempio: controllare fattura, approvare richiesta*

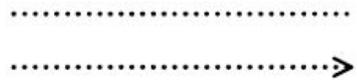
Gestire il flusso tra gli elementi grafici



Indicano la sequenza del flusso tra task



Indicano il flusso di un messaggio tra lane



Associazione tra altri elementi grafici

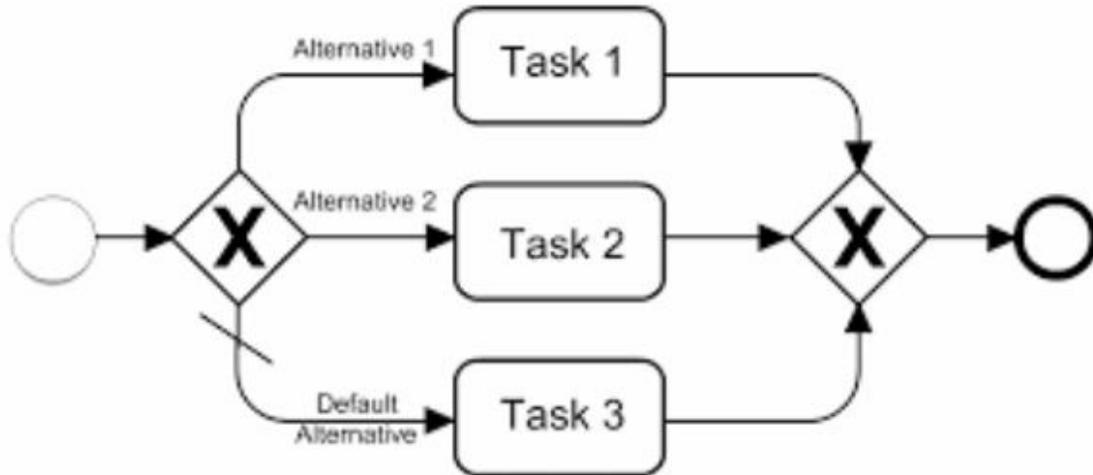
Come modellare ?

<https://bpmn.io/>

○
Inizio

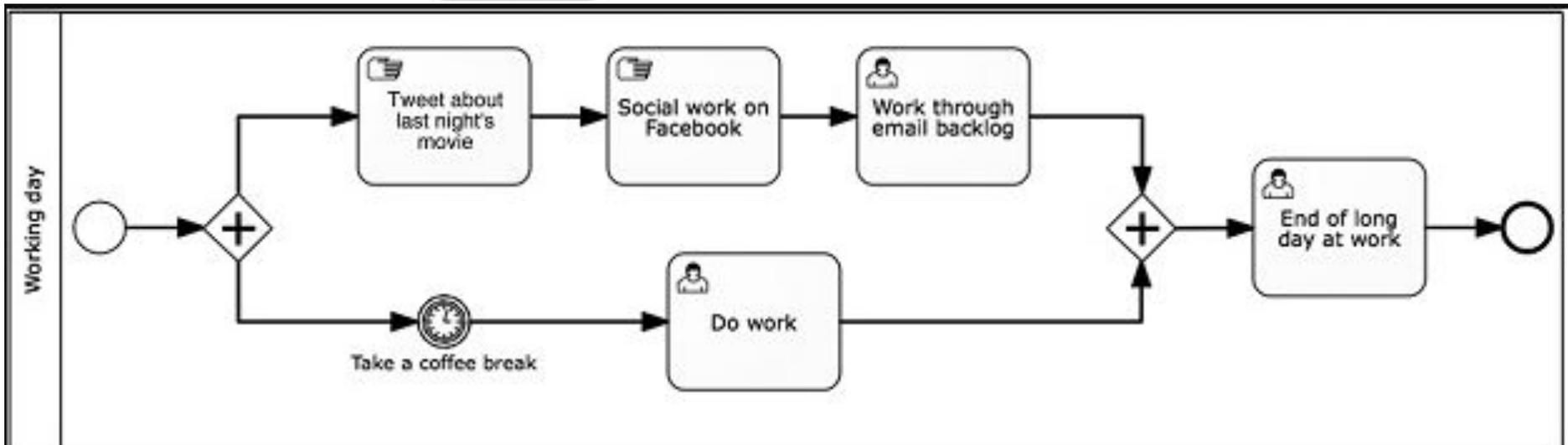
○
Fine

Task Name
Task



E per simulare?

<https://bimp.cs.ut.ee/simulator>



Esercizio0

Programmare le istruzioni per il calcolo dell'Area di un quadrato

Chiedendo il Formato

se é un quadrato, procedi...

Chiedendo il Lato

se positivo, calcola l'area

Stampa il risultato



Esercizio0

Programmare le istruzioni per il calcolo dell'Area di un quadrato

Chiedendo il Formato

in PYTHON

es. online: <https://www.programiz.com/python-programming/online-compiler/>

se é un quadrato, procedi...

```
print("Esercizio 0 - Calcolare l'area")
```

```
formato = input("Inserire il formato:\n")
```

Chiedendo il Lato

se positivo, calcola l'area

```
if (formato == "quadrato"):
```

```
    lato = input("Inserire la lunghezza del lato: ")
```

```
    if lato > 0:
```

```
        area = int(lato) * int(lato)
```

```
        print("Area = ", area)
```

Stampa il risultato

Dal modello al programma: cassetta degli attrezzi

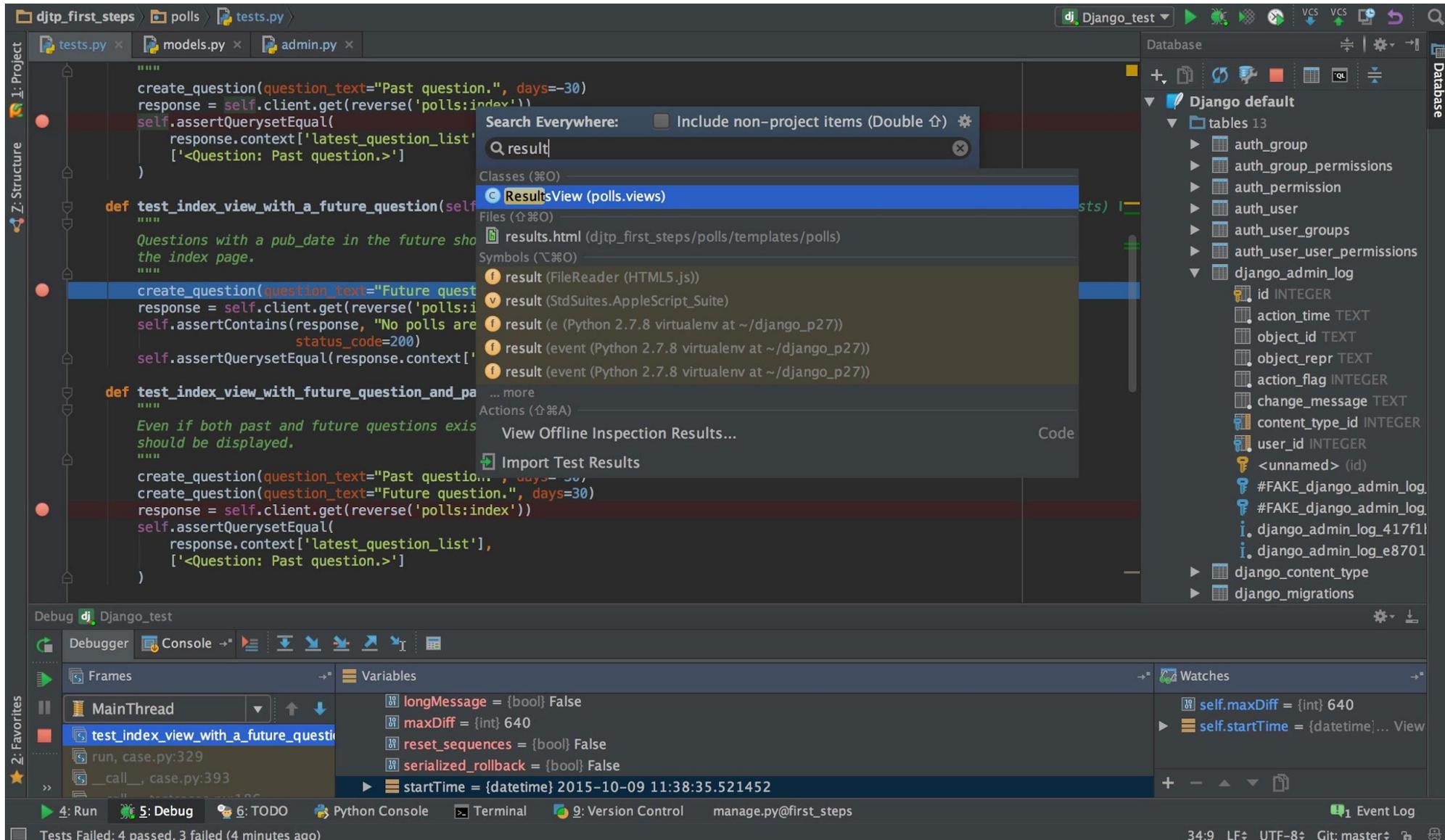
- Funzionamento della “macchina” di calcolo
- Quali linguaggi?
- Quali tools ?
- Come si scrivono i programmi
- Gli elementi di un linguaggio di programmazione

Programmazione

Come si scrivono i programmi?

- **Editor di testo**
 - Blocco note (!)
- **Tools specifici (IDE - Integrated Development Environment)**
 - Editor *multipurpose*:
 - Visual Studio Code, NetBeans, Eclipse
 - Python → PyCharm, Spyder
 - R → RStudio
- **Database**
 - SQL, NoSQL

Esempio di editor



come mai è nero ?

Esecuzione di un programma

Linguaggi di basso livello

→ Linguaggi “macchina” (es. Assembler)

Linguaggi di alto livello

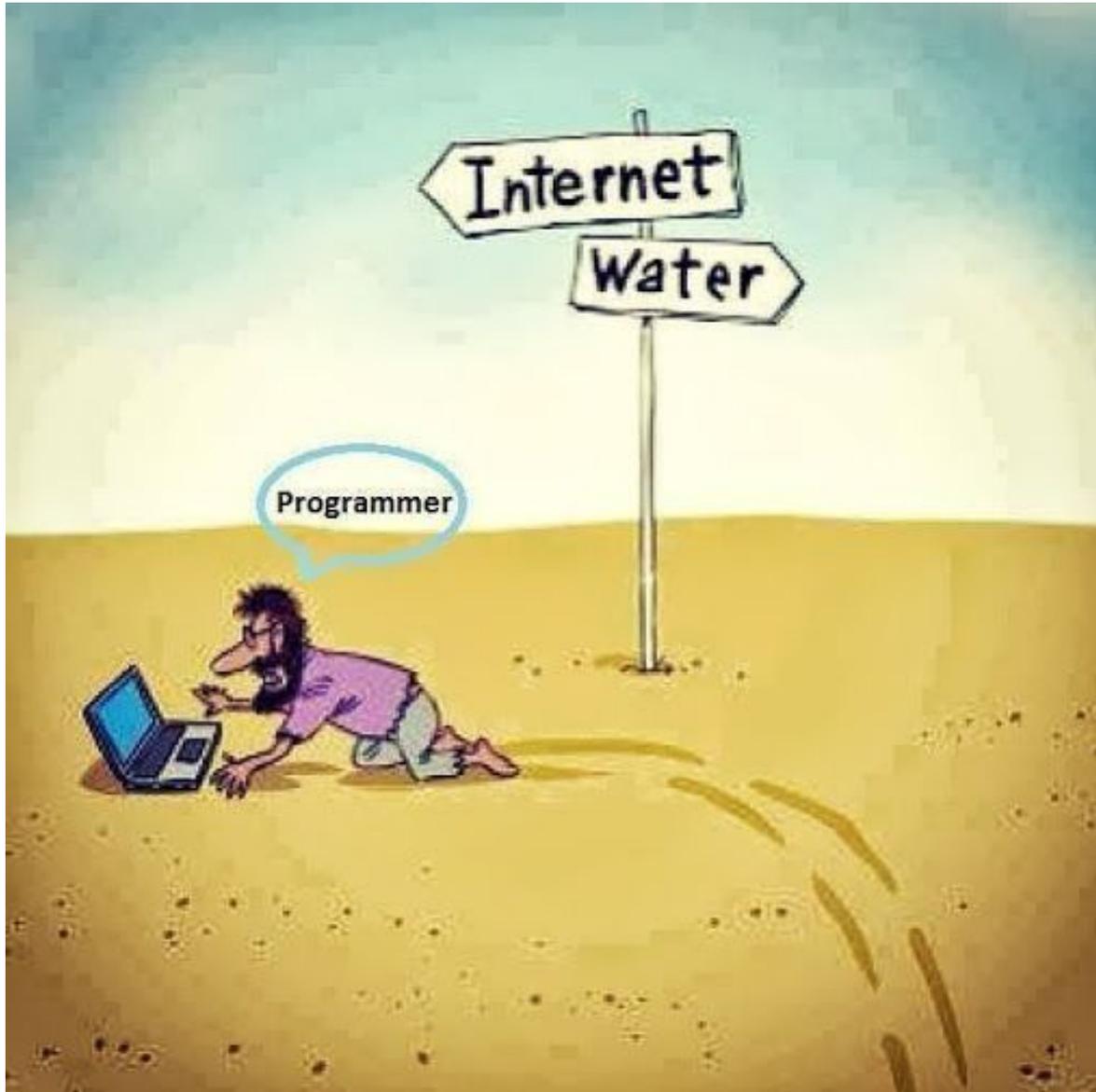
→ Facilitati, user-friendly (es. C, Java, **Python**, R, ecc.)

Un programma (codice) va compilato ed eseguito

Online ci sono dei “simulatori”, ad esempio:
pythonsandbox (già visto) oppure

<https://www.programiz.com/python-programming/online-compiler/>

obiettivi/motivazioni



Programmatore
in cerca di lavoro



Programmatore
con un lavoro



Contenuti: introduzione alla programmazione

1. Pensiero computazionale - introduzione
- 2. La codifica dell'informazione**
3. Coding e linguaggi visuali
4. Ingredienti principali

La codifica dell'informazione

Con **informazione** si intende tutto ciò che possiede un **significato** e che viene conservato o comunicato in vista di un'utilità pratica, immediata o futura.

Riguarda un fenomeno del mondo reale e un passaggio di conoscenza.

Da colui che possiede una conoscenza (emittente) a chi la riceve.

La trasmissione dell'informazione si basa su un **insieme di regole comuni** all'emittente e al ricevente, chiamate **codice**.

Esempi di codice? Codice morse, la lingua naturale (la lingua italiana)
Per i calcolatori si usa il codice binario

La codifica dell'informazione



Elementi:

- Un **emittente** che vuole comunicare un'informazione
- Avviene una trasformazione (**codifica**) del messaggio prima di trasmetterlo
- Il ricevente deve effettuare la trasformazione inversa (**decodifica**)
- Infine, il ricevente comprende il significato dell'informazione

Emittente e destinatario possono essere sia una persona o un computer

L'informazione è un insieme di **dati** elaborati e trasmessi

Elaborazione dei dati



L' **elaborazione** riguarda il trattamento dei dati

Esempi:

- la trascrizione di dati (copiatura);
- il calcolo matematico, su dati numerici e regole predefinite;
- tipi più complessi sono: la trasformazione di figure piane, la definizione di uno schema, la costruzione di una teoria scientifica...

Numeri binari

Nella programmazione:

l'output del sistema di elaborazione dev'essere un numero in codice binario

Sistema binario: rappresentazione con due sole cifre: 0 e 1

I computer, come tutti i circuiti elettrici, “capiscono” soltanto questi due stati:

1 - Presenza di tensione, in genere 5 volt

0 - Assenza di tensione, quindi 0 volt

Sistema binario

Sistema posizionale: ogni cifra ha diverso valore a seconda della posizione

La **posizione** di destra è quella meno significativa. Es. unità, nel sistema decimale. L'**ordine** in cui si trova la cifra è quindi fondamentale.

Base è il numero di cifre che costituisce l'alfabeto: nel sistema binario sono due

Si usa indicarla come *pedice*, es: **100101₂**

(nel sistema decimale le cifre sono 10, si dice 'base 10' e non si indica il pedice)

Il valore associato ad una cifra corrisponde al valore ottenuto moltiplicando tale cifra per la base elevata all'esponente dato dall'ordine (posizione – 1)

Numeri binari

Nel sistema (posizionale) di numerazione decimale disponiamo di **9 cifre** e lo **zero**:

10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1

Es.: **20352** \rightarrow **2** **0** **3** **5** **2**

$2 * 10000$ $0 * 1000$ $3 * 100$ $5 * 10$ $2 * 1$

\downarrow \downarrow \downarrow \downarrow \downarrow

20000 + **0** + **300** + **50** + **2** = **20352**

...nel sistema binario (2 cifre: 0 e 1) la prima posizione (a destra) è 2^0 , quindi 2^1 ecc.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

Es.: **10101** \rightarrow **1** **0** **1** **0** **1**

$1 * 16$ $0 * 8$ $1 * 4$ $0 * 2$ $1 * 1$

\downarrow \downarrow \downarrow \downarrow \downarrow

16 + **0** + **4** + **0** + **1** = **21**

La codifica dell'informazione

La rappresentazione delle informazioni nell'elaboratore si basa sul calcolo binario

Elementi fondamentali **0 e 1**: sono le cifre binarie denominate ***bit* (binary digits)**

Le cifre binarie sono trattate a gruppi di 4 bit (*nibble*) oppure di **8 bit = 1 byte**

Le sequenze o stringhe formate da un numero fisso di bit/byte sono dette **parole**

Informazioni:

- numeriche: viene allocato spazio di memoria (adeguato)
- testuali: ogni simbolo viene codificato in un numero

La codifica del testo

Codifica di base: **ASCII** – **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

Codifica i caratteri in base a 7 bit: **ad es.: 'A' = 01000001** (65 in binario)

ASCII: i primi 31 -> Caratteri non stampabili (spazio vuoto, canc,..)

da 48 a 57 -> Cifre da 0 a 9

da 65 a 90 -> Lettere maiuscole (dalla 'A' alla 'Z')

Estensione: 8 bit (ASCII esteso) → 2^8 caratteri = 256 caratteri

Dal 1991: nuova codifica **Unicode** (utilizzata in Internet, per il WWW):

- Versioni UTF-8 (a 8 bit), UTF-16 (a 16 bit) e UTF-32 (a 32 bit)

La codifica del testo

*Esempio: la stringa **CIAO** in ASCII code*

C -> 01000011

I -> 01001001

A -> 01000001

O -> 01001111

C I A O

01000011010010010100000101001111

LETTER	ASCII VALUES	BINARY VALUES	LETTER	ASCII VALUES	BINARY VALUES
A	65	01000001	A	97	01100001
C	67	01000011	C	99	01100011
D	68	01000100	D	100	01100100
E	69	01000101	E	101	01100101
F	70	01000110	F	102	01100110
G	71	01000111	G	103	01100111
H	72	01001000	H	104	01101000
I	73	01001001	I	105	01101001
J	74	01001010	J	106	01101010
K	75	01001011	K	107	01101011
L	76	01001100	L	108	01101100
M	77	01001101	M	109	01101101
N	78	01001110	N	110	01101110
O	79	01001111	O	111	01101111
P	80	01010000	P	112	01110000
Q	81	01010001	Q	113	01110001
R	82	01010010	R	114	01110010
S	83	01010011	S	115	01110011
T	84	01010100	T	116	01110100
U	85	01010101	U	117	01110101
V	86	01010110	V	118	01110110
W	87	01010111	W	119	01110111
X	88	01011000	X	120	01111000
Y	89	01011001	Y	121	01111001
Z	90	01011010	Z	122	01111010

Contenuti: introduzione alla programmazione

1. Pensiero computazionale - introduzione
2. La codifica dell'informazione
- 3. Coding e linguaggi visuali**
4. Ingredienti principali

Coding e linguaggi visuali

Per imparare a programmare sono disponibili dei tool pensati per età 8-16 anni

Il MediaLab del MIT ha sviluppato Scratch: <https://scratch.mit.edu/>

The Scratch logo is displayed in a large, stylized, orange font with a white outline. The letters are rounded and have a playful, bubbly appearance.

Coding e linguaggi visuali

SNAP! (dal precedente BYOB - Build Your Own Block)

<https://snap.berkeley.edu/snap/snap.html>

Creato dall'università di Berkeley

Si basa su browser: usa html5 e javascript

Funziona anche su iPad e android

Gestione delle variabili tipizzate

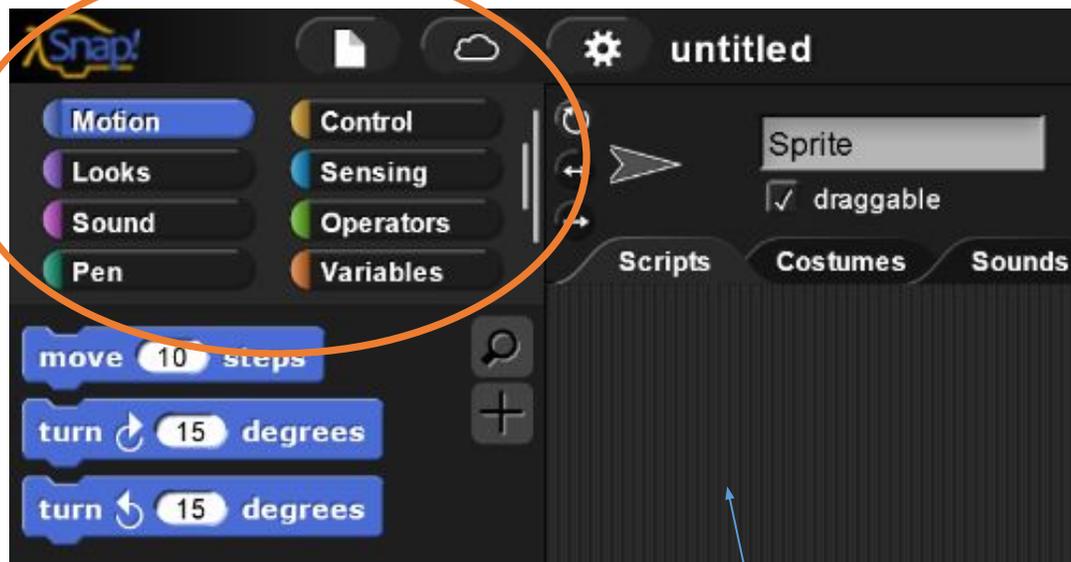
Maggiore possibilità di usare le funzioni matematiche



Coding e linguaggi visuali

SNAP!

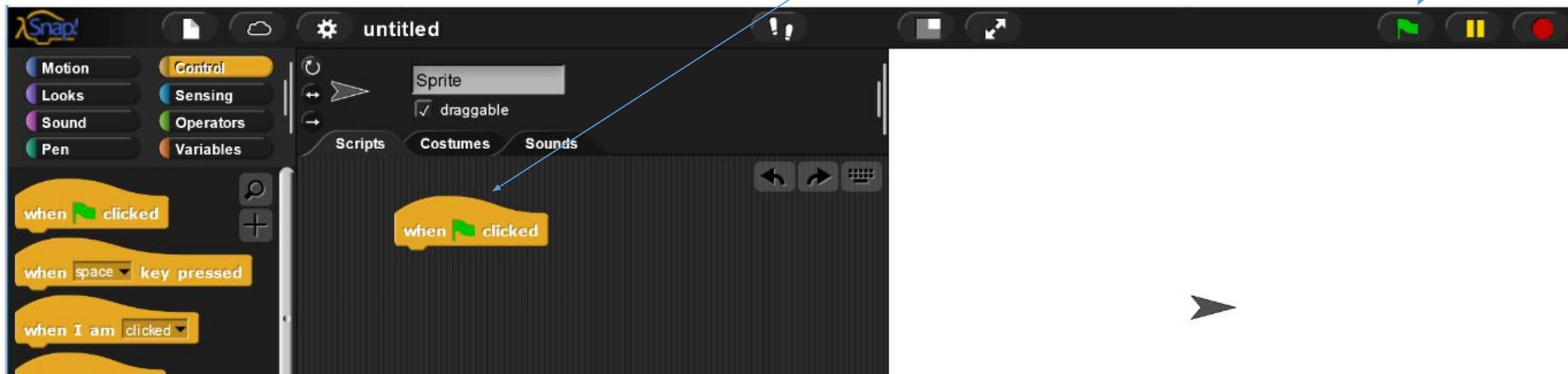
Movimento
Controllo
Aspetto
Sensori
Suoni
Operatori
Penna
Variabili



i blocchi vanno trascinati nell'area SCRIPTS

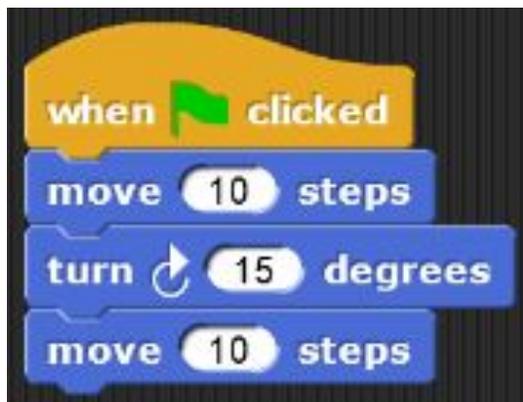
Coding e linguaggi visuali

Il primo blocco nello Script dev'essere lo Start che si attiva cliccando la bandiera



Dopo lo start si possono agganciare altre istruzioni, cicli, creare variabili, ecc.

Esempio:



Si possono cambiare i valori e vedere l'esito

Esercizio1

Programmare la freccina (Sprite) in modo che svolga le seguenti istruzioni:

Ripetere 10 volte

Fare 10 passi

Girarsi di 180 gradi

Ripetere 10 volte

Fare 10 passi

Esercizio2

Creare una variabile di nome 'numero'

Inizializzare la variabile numero a 0

Ripeti fino a quando numero non diventa maggiore di 5 le istruzioni:

avanza di 5 passi

incrementa numero di uno

aspetta un secondo

Contenuti: introduzione alla programmazione

1. Pensiero computazionale - introduzione
2. La codifica dell'informazione
3. Coding e linguaggi visuali
- 4. Ingredienti principali**

Modello procedurale

Modellazione: rappresentazione schematica di un particolare aspetto della realtà.

La formalizzazione del modello individua gli elementi principali della realtà osservata e tratta tali elementi come **entità astratte**, con due caratteristiche:

- devono essere registrabili in una memoria
- devono essere messe in relazione tra loro

L'**implementazione** di un problema si ha quando la sua soluzione viene affrontata attraverso la costruzione di un modello formalizzato, in modo tale che il problema possa essere rappresentato e gestito con un sistema di elaborazione automatica.

Fasi della programmazione

1. Individuare un **problema**: analisi del problema per individuare regole, dati e obiettivi; eliminare dettagli inutili/ambigui
2. Costruire un **modello**
3. Per ogni **entità** individuare le relative **proprietà** e ragionare sui valori che potranno assumere (tipi di dato) -> sono le **variabili** (e le **costanti**) del nostro programma, definite mediante **identificatori**
4. Recuperare i **dati** del problema
5. Definire le **azioni** per ottenere il risultato desiderato (potrebbe essere necessario elaborare un **algoritmo**)

Esempio di implementazione

Una volta individuato un problema, si trova una soluzione che può essere affrontata mediante un sistema di elaborazione automatico, utilizzando uno specifico linguaggio di programmazione

Esempio: calcolare il costo di un viaggio in auto

Serve conoscere:

>> il numero di KM percorsi, il consumo dell'auto, la velocità

In seguito: svolgere un calcolo sulla base di tali dati (chiamati *variabili*)

Variabili

Sono **proprietà** relative alle entità del problema considerato: possono assumere valori diversi, per diversi esemplari della stessa entità.

- Es: $A = b \times h$

A è una variabile che conterrà **l'Area** che vogliamo calcolare

b è una variabile relativa alla lunghezza della **base** del rettangolo

h è una variabile che rappresenta la proprietà '**altezza**' del rettangolo

Ogni variabile assumerà diversi valori, per diversi esemplari di rettangolo

Variabili

Possono rappresentare dati di ingresso (variabili di input), dati di uscita (variabili di output), o elementi utili nel programma (variabili di lavoro)

Es. Date le misure di due cateti, calcolare il perimetro del triangolo rettangolo

Variabili di INPUT	Variabile di OUTPUT	Variabile di LAVORO
Cateto1, Cateto2	Perimetro	Ipotenusa

Assegnare un valore ad una variabile

Può essere un nuovo valore:

nome_variabile = valore

(es., **base = 10**, oppure: **altezza = 5**)

Può essere il risultato di un'espressione:

nome_variabile2 = valore1 * valore2

(es., **area = base * altezza**)

Esempio

Assegnare un valore a variabili (base e altezza):

```
base = 5
```

```
altezza = 10
```

```
print(base * altezza)
```

Esercizio3: calcolare e stampare l'area

Dopo aver assegnato i valori alle variabili (base e altezza)

Calcolare l'area in una nuova variabile e visualizzare il risultato

Costanti

Una costante è una proprietà che non cambia considerando problemi con variabili diverse.

Può essere definita dal programmatore all'inizio del programma.

Oppure può essere già prevista dal linguaggio di programmazione.

Esempio: il **pi-greco**

[rappresenta il rapporto tra la circonferenza e il diametro di un cerchio]

Altro esempio: **g**

[costante del moto uniformemente accelerato]

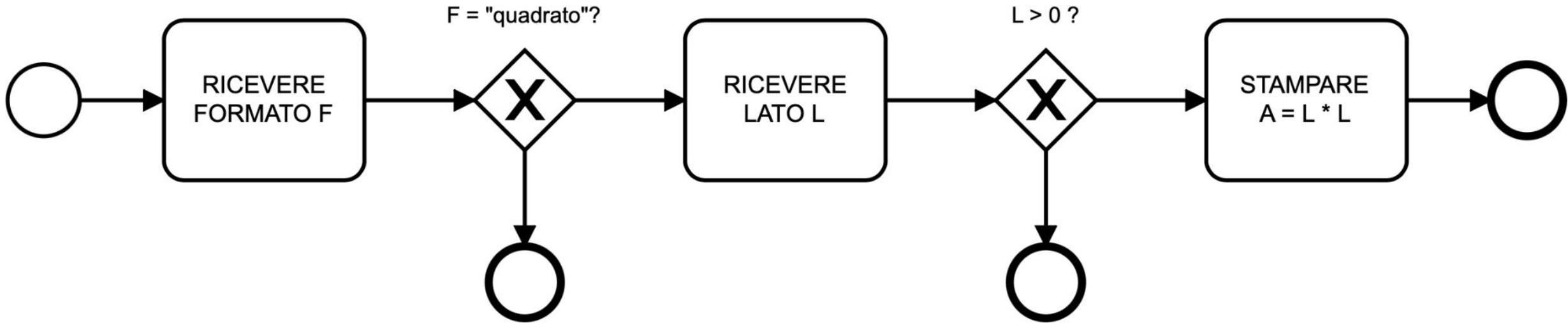
Esercizio4

Stampare i numeri da 100 a 300 con un ciclo 'for'

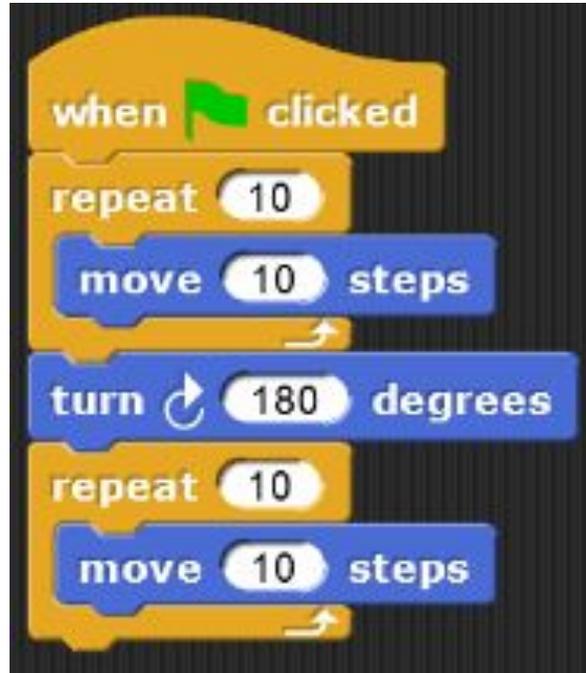
```
for i in range(100,300):
```

```
| ?
```

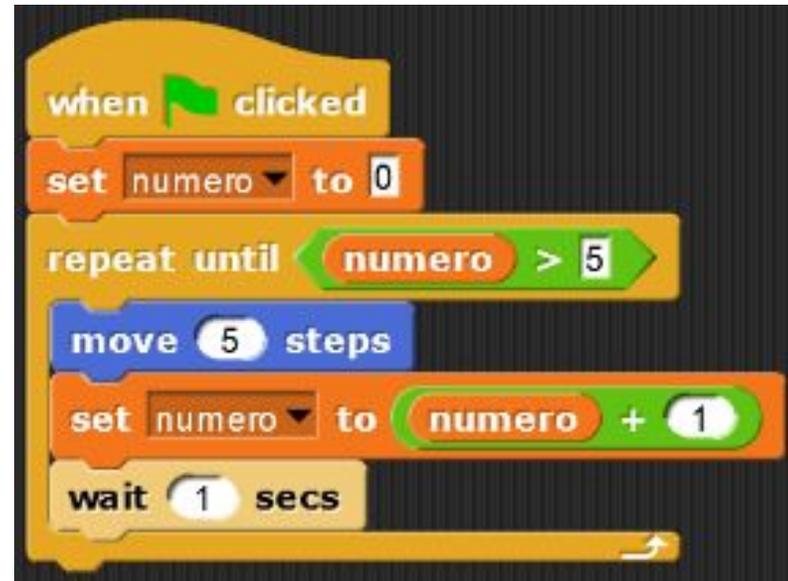
Soluzione Esercizio0



Soluzione Esercizio1



Soluzione Esercizio2



Soluzione Esercizio3

base = 5

altezza = 10

area = base * altezza

print(area)

Soluzione Esercizio4

```
# numeri da 100 a 300
```

```
for i in range(100,300):  
    print(i)
```