

CIVIFORM

CODING E ROBOTICA PER L'INNOVAZIONE SOCIALE

Cividale, gennaio-febbraio 2022

PRESENTAZIONI

PRESENTAZIONI

Comincio io...

PRESENTAZIONI

Comincio io...

- programmatore per passione



ZX Spectrum 48K, 1982



Clive Sinclair con uno ZX Spectrum

PRESENTAZIONI

Comincio io...

- programmatore per passione

PRESENTAZIONI

Comincio io...

- programmatore per passione e mestiere
- laureato in Scienze dell'informazione

PRESENTAZIONI

Comincio io...

- programmatore per passione e mestiere
- laureato in Scienze dell'informazione
- membro di CoderDojoFVG



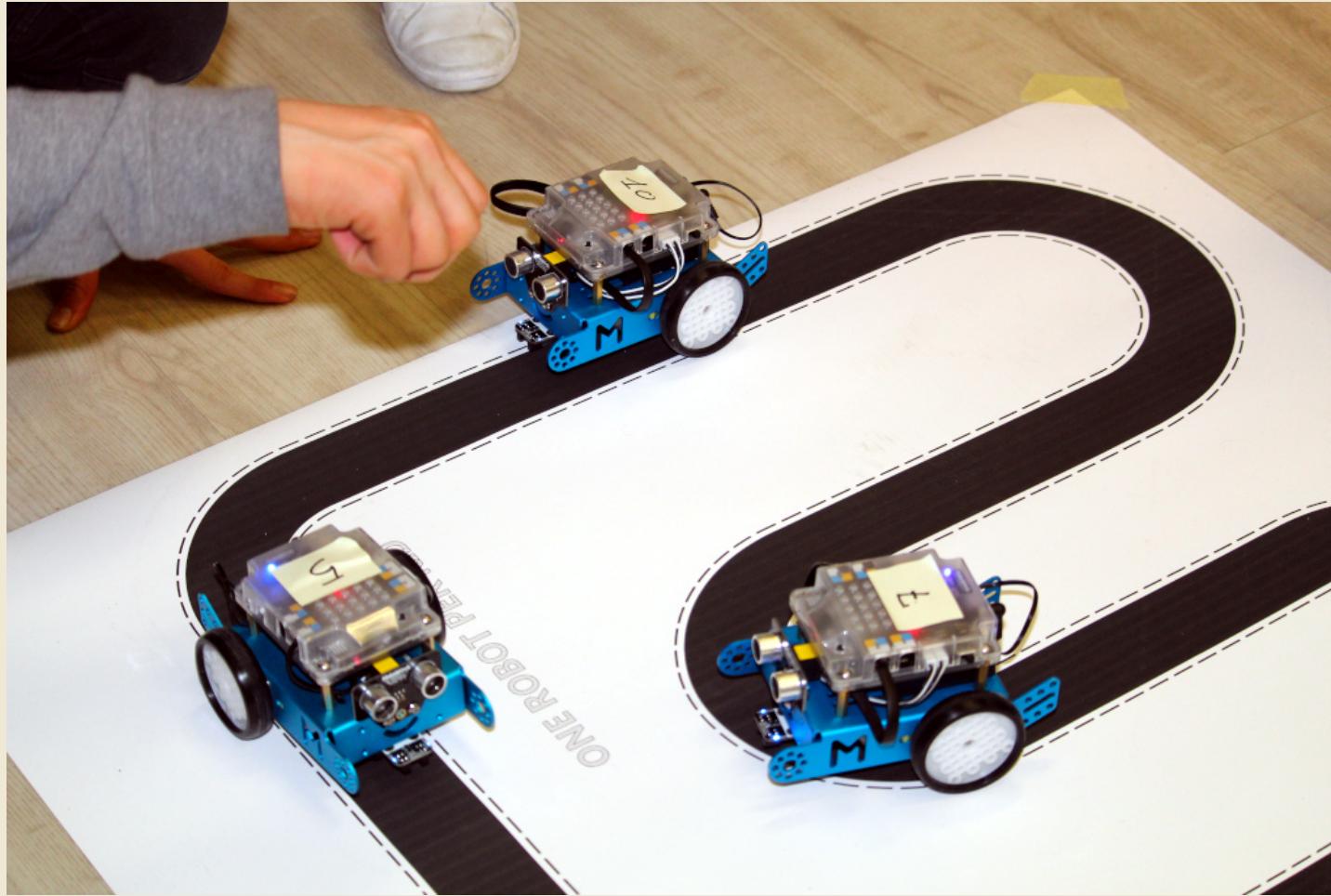
CoderDojo



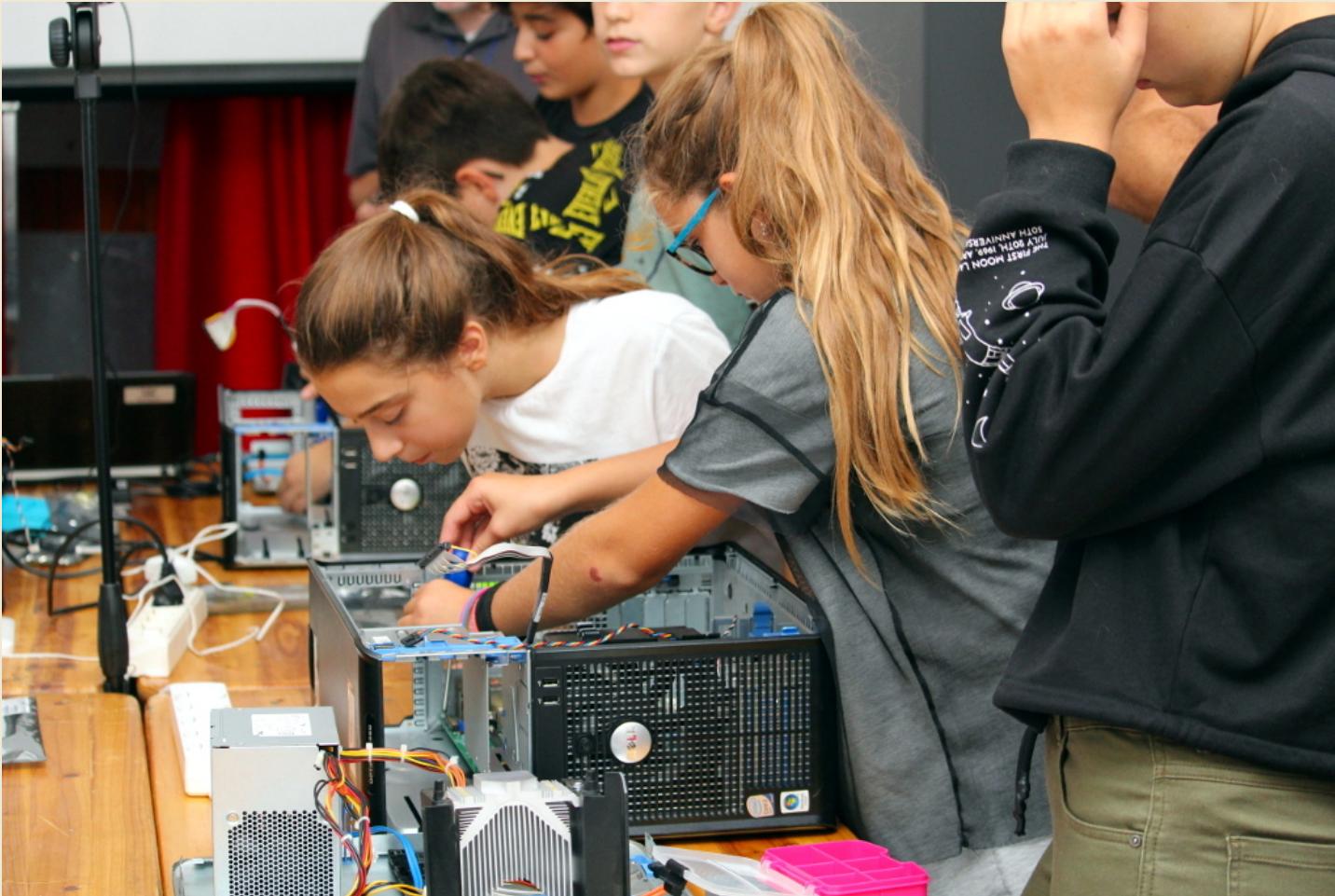
Dojo@School



TeacherDojo



RoboDojo



HardwareDojo

PRESENTAZIONI

Tocca a voi!

PRESENTAZIONI

Tocca a voi!

- che attività svolgete?
- motivo dell'interesse per il coding?
- esperienze pregresse?
- quali aspettative avete?

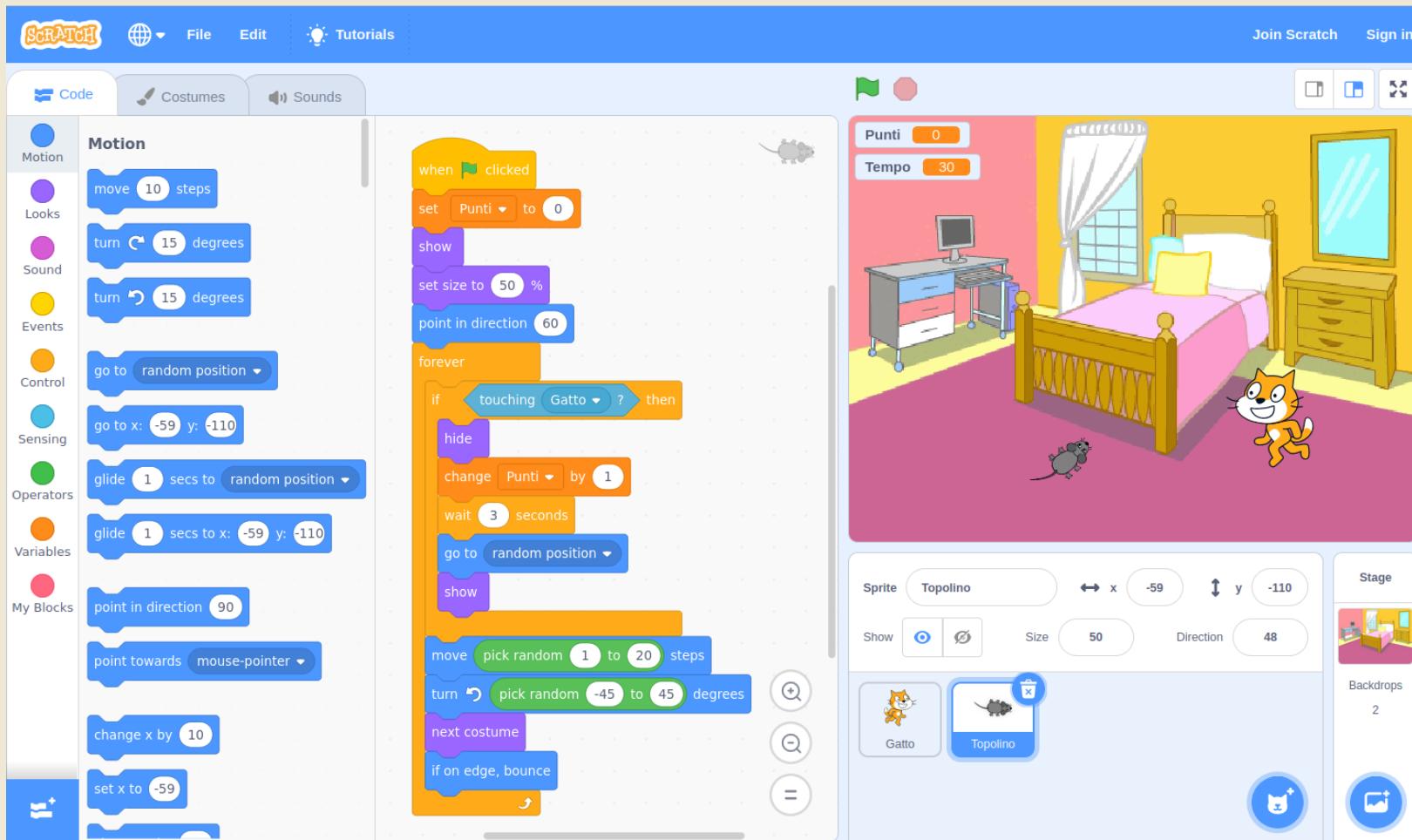
STRUTTURA DEL CORSO

STRUTTURA DEL CORSO

- breve introduzione ai concetti base (oggi)

STRUTTURA DEL CORSO

- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)



Scratch 3.0

STRUTTURA DEL CORSO

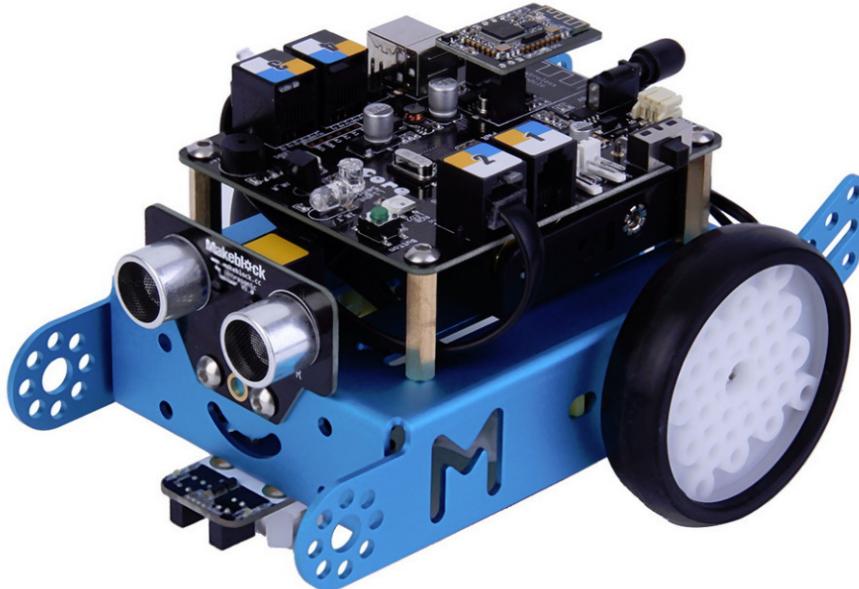
- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)

STRUTTURA DEL CORSO

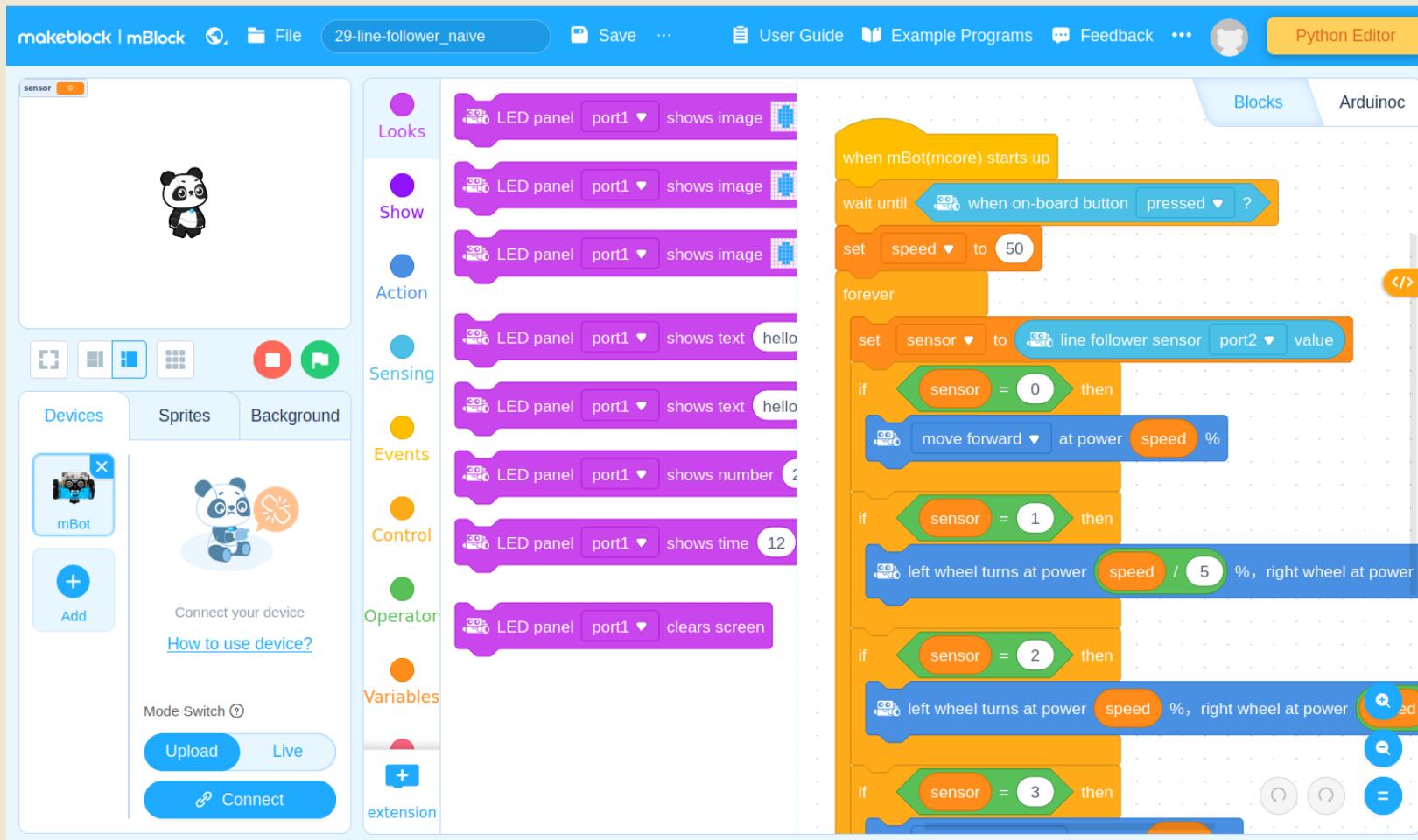
- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)
- principi di programmazione con Scratch (4-5 lezioni)

STRUTTURA DEL CORSO

- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)
- principi di programmazione con Scratch (4-5 lezioni)
- robotica con mBot



mBot



mBlock 5.4

STRUTTURA DEL CORSO

- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)
- principi di programmazione con Scratch (4-5 lezioni)
- robotica con mBot

STRUTTURA DEL CORSO

- breve introduzione ai concetti base (oggi)
- presentazione dell'ambiente Scratch (oggi)
- principi di programmazione con Scratch (4-5 lezioni)
- robotica con mBot
- varie ed eventuali

ATTIVITÀ

ATTIVITÀ

- incentrata sulla pratica

ATTIVITÀ

- incentrata sulla pratica
- orientata al progetto

ATTIVITÀ

- incentrata sulla pratica
- orientata al progetto
- difficoltà gradualmente crescente

ATTIVITÀ

- incentrata sulla pratica
- orientata al progetto
- difficoltà gradualmente crescente
- i suggerimenti sono benvenuti!

MATERIALE DI SUPPORTO

MATERIALE DI SUPPORTO

- tantissime risorse in rete

MATERIALE DI SUPPORTO

- tantissime risorse in rete
- libri e manuali:

MATERIALE DI SUPPORTO

- tantissime risorse in rete
- libri e manuali:
 - Coding - programmare è un gioco (DeAgostini)

MATERIALE DI SUPPORTO

- tantissime risorse in rete
- libri e manuali:
 - Coding - programmare è un gioco (DeAgostini)
 - Imparare a programmare con Scratch (Apogeo)

MATERIALE DI SUPPORTO

- tantissime risorse in rete
- libri e manuali:
 - Coding - programmare è un gioco (DeAgostini)
 - Imparare a programmare con Scratch (Apogeo)
 - TECHNOLogica - coding e robotica (DeAgostini)

CODING

CODING

Scrittura di programmi per calcolatori.

CODING

Scrittura di programmi per calcolatori.

- codifica

CODING

Scrittura di programmi per calcolatori.

- codifica
- sviluppo (software)

CODING

Scrittura di programmi per calcolatori.

- codifica
- sviluppo (software)
- implementazione (di programmi)

CODING

Scrittura di programmi per calcolatori.

- codifica
- sviluppo (software)
- implementazione (di programmi)
- programmazione

CODING

Frutto del **coding** è il **programma**.

programma

CODING

Frutto del **coding** è il **programma**.

programma

Insieme coordinato e strutturato di istruzioni, codificato secondo un opportuno **linguaggio di programmazione**, atte a risolvere un dato problema mediante un calcolatore elettronico [Treccani].

CODING

Un **programma** nasce per essere eseguito da un computer.

programma → **computer** → **risposta**

CODING

Come nasce un **programma**?

? → **programma**

CICLO DI VITA DEL SOFTWARE

L'esigenza nasce da un problema...

problema → ... → programma

CICLO DI VITA DEL SOFTWARE

... del quale si conosce una soluzione...

problema → soluzione → ... → programma

CICLO DI VITA DEL SOFTWARE

... che è stata **formalizzata** in un **algoritmo**.

problema → **soluzione** → **algoritmo** → **programma**

Programming is not about typing, it's about thinking.

— Rich Hickey

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

1. analisi

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

- 1. analisi**
- 2. progettazione**

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

1. analisi
2. progettazione
3. coding

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

1. analisi
2. progettazione
3. coding
4. collaudo

FASI DEL CICLO DI VITA DEL SOFTWARE

problema → soluzione → algoritmo → programma

1. analisi
2. progettazione
3. coding
4. collaudo

ANALISI

ANALISI

Richiede l'uso di tecniche di **problem solving**.

problema → soluzione

ANALISI

Richiede l'uso di tecniche di **problem solving**.

problema → idea → soluzione

ANALISI

Richiede l'uso di tecniche di **problem solving**.

problema → idea → soluzione

**Da cosa
nasce cosa**



Munari



Bruno Munari (1981)

ANALISI

problema → definizione

ANALISI

problema → definizione

Che tipo di soluzione è desiderabile?

ANALISI

problema → definizione

Che tipo di soluzione è desiderabile?

- completa, definitiva?

ANALISI

problema → definizione

Che tipo di soluzione è desiderabile?

- completa, definitiva?
- provvisoria, approssimativa?

ANALISI

problema → definizione

Che tipo di soluzione è desiderabile?

- completa, definitiva?
- provvisoria, approssimativa?
- di minor costo?

ANALISI

problema → definizione

Che tipo di soluzione è desiderabile?

- completa, definitiva?
- provvisoria, approssimativa?
- di minor costo?
- ...

ANALISI

definizione → scomposizione

ANALISI

definizione → scomposizione

Scomposizione del problema generale nelle sue componenti.

ANALISI

definizione → scomposizione

Scomposizione del problema generale nelle sue componenti.

Moltiplica il numero di problemi da risolvere.

ANALISI

definizione → scomposizione

Scomposizione del problema generale nelle sue componenti.

Moltiplica il numero di problemi da risolvere.

Riduce la complessità di ogni singolo sottoproblema.

ANALISI

scomposizione → raccolta dati

ANALISI

scomposizione → raccolta dati

Analisi delle soluzioni esistenti.

ANALISI

scomposizione → raccolta dati

Analisi delle soluzioni esistenti.

Verifica dell'applicabilità delle soluzioni trovate.

ANALISI

scomposizione → raccolta dati

Analisi delle soluzioni esistenti.

Verifica dell'applicabilità delle soluzioni trovate.

Soluzioni non applicabili diventano fonte d'ispirazione.

ANALISI

raccolta dati → integrazione

ANALISI

raccolta dati → integrazione

Conciliazione delle soluzioni parziali con il progetto generale.

ANALISI

raccolta dati → integrazione

Conciliazione delle soluzioni parziali con il progetto generale.

La creatività che nasce dalla metodica ricombinazione delle soluzioni si sostituisce all'idea dell'"illuminazione geniale".

problema



definizione



scomposizione



raccolta dati



integrazione



soluzione

PROGETTAZIONE

PROGETTAZIONE

Formalizzazione della soluzione.

soluzione → algoritmo

ESEMPIO

problema

ESEMPIO

problema

Preparare della pasta al sugo per due persone.

ESEMPIO

soluzione

ESEMPIO

soluzione

- far bollire l'acqua

ESEMPIO

soluzione

- far bollire l'acqua
- aggiungere il sale

ESEMPIO

soluzione

- far bollire l'acqua
- aggiungere il sale
- buttare la pasta

ESEMPIO

soluzione

- far bollire l'acqua
- aggiungere il sale
- buttare la pasta
- quando sembra cotta, scolare

ESEMPIO

soluzione

- far bollire l'acqua
- aggiungere il sale
- buttare la pasta
- quando sembra cotta, scolare
- aggiungere il sugo e servire

ESEMPIO

soluzione

- far bollire l'acqua - **come, dove?**
- aggiungere il sale
- buttare la pasta
- quando sembra cotta, scolare
- aggiungere il sugo e servire

ESEMPIO

soluzione

- far bollire l'acqua - **come, dove?**
- aggiungere il sale - **quanto?**
- buttare la pasta
- quando sembra cotta, scolare
- aggiungere il sugo e servire

ESEMPIO

soluzione

- far bollire l'acqua - **come, dove?**
- aggiungere il sale - **quanto?**
- buttare la pasta - **quanta?**
- quando sembra cotta, scolare
- aggiungere il sugo e servire

ESEMPIO

soluzione

- far bollire l'acqua - **come, dove?**
- aggiungere il sale - **quanto?**
- buttare la pasta - **quanta?**
- quando sembra cotta, scolare - **sembra!?**
- aggiungere il sugo e servire

ESEMPIO

soluzione

- far bollire l'acqua - **come, dove?**
- aggiungere il sale - **quanto?**
- buttare la pasta - **quanta?**
- quando sembra cotta, scolare - **sembra!?**
- aggiungere il sugo e servire - **una cosa alla volta!**

ESEMPIO

algoritmo

ESEMPIO

algoritmo

- prendere una pentola

ESEMPIO

algoritmo

- prendere una pentola
- versarci 4 litri d'acqua dentro

ESEMPIO

algoritmo

- prendere una pentola
- versarci 4 litri d'acqua dentro
- mettere la pentola sul fornello

ESEMPIO

algoritmo

- prendere una pentola
- versarci 4 litri d'acqua dentro
- mettere la pentola sul fornello
- accendere il fuoco

ESEMPIO

algoritmo

- prendere una pentola
- versarci 4 litri d'acqua dentro
- mettere la pentola sul fornello
- accendere il fuoco
- attendere 5 minuti

ESEMPIO

algoritmo

- versarci 4 litri d'acqua dentro
- mettere la pentola sul fornello
- accendere il fuoco
- attendere 5 minuti
- se l'acqua non bolle, tornare al passo precedente

ESEMPIO

algoritmo

- mettere la pentola sul fornello
- accendere il fuoco
- attendere 5 minuti
- se l'acqua non bolle, tornare al passo precedente
- aggiungere due cucchiai di sale grosso

ESEMPIO

algoritmo

- accendere il fuoco
- attendere 5 minuti
- se l'acqua non bolle, tornare al passo precedente
- aggiungere due cucchiai di sale grosso
- buttare 160 grammi di pasta nella pentola

ESEMPIO

algoritmo

- attendere 5 minuti
- se l'acqua non bolle, tornare al passo precedente
- aggiungere due cucchiai di sale grosso
- buttare 160 grammi di pasta nella pentola
- predisporre due piatti fondi puliti

ESEMPIO

algoritmo

- se l'acqua non bolle, tornare al passo precedente
- aggiungere due cucchiai di sale grosso
- buttare 160 grammi di pasta nella pentola
- predisporre due piatti fondi puliti
- attendere il tempo riportato nella confezione

ESEMPIO

algoritmo

- se l'acqua non bolle, tornare al passo precedente
- aggiungere due cucchiai di sale grosso
- buttare 160 grammi di pasta nella pentola
- predisporre due piatti fondi puliti
- assaggiare la pasta

ESEMPIO

algoritmo

- aggiungere due cucchiai di sale grosso
- buttare 160 grammi di pasta nella pentola
- predisporre due piatti fondi puliti
- assaggiare la pasta
- se non è cotta attendere un minuto e ripetere

ESEMPIO

algoritmo

- buttare 160 grammi di pasta nella pentola
- predisporre due piatti fondi puliti
- assaggiare la pasta
- se non è cotta attendere un minuto e ripetere
- spegnere il fornello

ESEMPIO

algoritmo

- predisporre due piatti fondi puliti
- assaggiare la pasta
- se non è cotta attendere un minuto e ripetere
- spegnere il fornello
- scolare la pasta

ESEMPIO

algoritmo

- assaggiare la pasta
- se non è cotta attendere un minuto e ripetere
- spegnere il fornello
- scolare la pasta
- suddividere la pasta nei due piatti

ESEMPIO

algoritmo

- se non è cotta attendere un minuto e ripetere
- spegnere il fornello
- scolare la pasta
- suddividere la pasta nei due piatti
- versare due cucchiai di sugo in ognuno dei due piatti

ESEMPIO

algoritmo

- spegnere il fornello
- scolare la pasta
- suddividere la pasta nei due piatti
- versare due cucchiai di sugo in ognuno dei due piatti
- servire

ALGORITMO

Strategia di risoluzione di un **problema**.

ALGORITMO

Strategia di risoluzione di un **problema**:

- finito

ALGORITMO

Strategia di risoluzione di un **problema**:

- finito
- deterministico

ALGORITMO

Strategia di risoluzione di un **problema**:

- finito
- deterministico
- non ambiguo

ALGORITMO

Strategia di risoluzione di un **problema**:

- finito
- deterministico
- non ambiguo
- generale

ALGORITMO

Strategia di risoluzione di un **problema**:

- finito
- deterministico
- non ambiguo
- generale

Deriva dal nome del matematico persiano **al-Khwarizmi**.

ALGORITMO

Spesso espresso in **pseudocodice**:

```
inizio
imposta la velocità del robot a 100
per sempre
    DISTANZA = distanza dall'ostacolo più vicino
    se DISTANZA > 40 allora
        prosegui dritto
    altrimenti
        svolta a sinistra o destra
fine
```

ALGORITMO

A volte rappresentato con un [diagramma di flusso](#)...

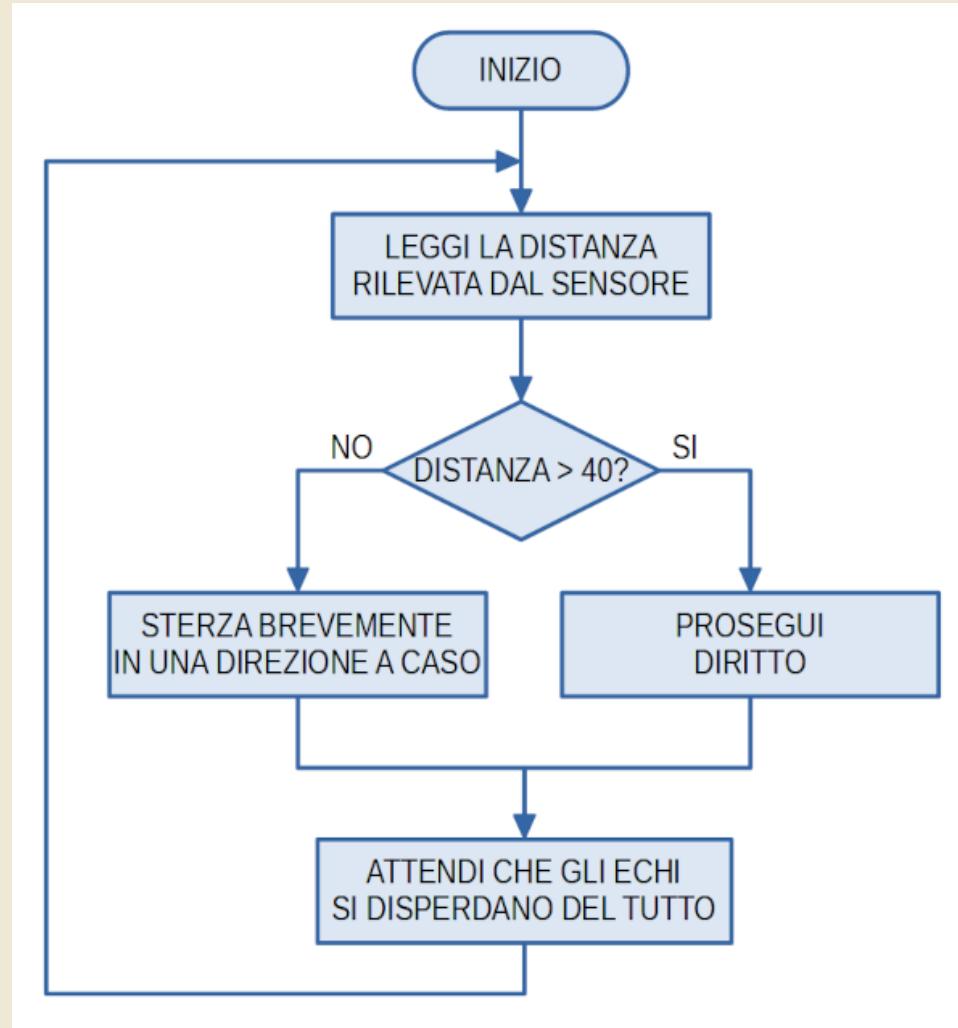


Diagramma di flusso

CODING

Scrittura di programmi per calcolatori.

algoritmo → programma

CODING

Scrittura di programmi per calcolatori.

algoritmo → programma

Transcodifica di un **algoritmo** in una forma comprensibile ad un calcolatore affinché lo possa eseguire autonomamente.

CODING

Scrittura di programmi per calcolatori.

algoritmo → programma

Transcodifica di un **algoritmo** in una forma comprensibile ad un calcolatore affinché lo possa eseguire autonomamente.

Lo strumento utilizzato è il **linguaggio di programmazione**.

LINGUAGGIO DI PROGRAMMAZIONE

Linguaggio **formale** per la programmazione di un calcolatore.

LINGUAGGIO DI PROGRAMMAZIONE

Linguaggio **formale** per la programmazione di un calcolatore.

Dotato di **sintassi** e **semantica** ben definite.

LINGUAGGIO DI PROGRAMMAZIONE

Linguaggio **formale** per la programmazione di un calcolatore.

Dotato di **sintassi** e **semantica** ben definite.

- **sintassi:** definisce le istruzioni e le regole di ricombinazione

LINGUAGGIO DI PROGRAMMAZIONE

Linguaggio **formale** per la programmazione di un calcolatore.

Dotato di **sintassi** e **semantica** ben definite.

- **sintassi**: definisce le istruzioni e le regole di ricombinazione
- **semantica**: specifica l'effetto sortito da ogni istruzione

LINGUAGGIO DI PROGRAMMAZIONE

Linguaggio **formale** per la programmazione di un calcolatore.

Dotato di **sintassi** e **semantica** ben definite.

- **sintassi**: definisce le istruzioni e le regole di ricombinazione
- **semantica**: specifica l'effetto sortito da ogni istruzione

È il mezzo di comunicazione tra programmatore e calcolatore.

LINGUAGGIO DI PROGRAMMAZIONE

LINGUAGGIO DI PROGRAMMAZIONE

- Linguaggi di basso livello

LINGUAGGIO DI PROGRAMMAZIONE

- **Linguaggi di basso livello**, orientati alla macchina

LINGUAGGI DI BASSO LIVELLO

```
00000110 00000101  
00111110 00001000  
00100001 10011111 01000011  
01110111  
00010000 11111100  
11001101 11100100 01000000  
11111110 00000011  
00100000 11101110  
00100010 00000111 01000000  
01011101  
11001101 00101101 01000001  
00100001 10100001 01000011  
...
```

Codice binario Z80

LINGUAGGI DI BASSO LIVELLO

4082	06	05	DRIVER	ld b,05
4084	3E	08		ld a,08
4086	21	9F 43		ld hl,439F
408A	77			ld (hl),a
408B	10	FC		djnz DRIVER1
408D	CD	E4 40		call KYBD
4090	FE	03		cp 03
4092	20	EE		jr nz,DRIVER
4094	22	07 40		ld (PPC),hl
4097	5D			ld e,l
4098	CD	2D 41		call MOVE
409B	21	A1 43		ld hl,43A1
...				

LINGUAGGIO DI PROGRAMMAZIONE

È il mezzo di comunicazione tra programmatore e calcolatore.

- **Linguaggi di basso livello**, orientati alla macchina

LINGUAGGIO DI PROGRAMMAZIONE

È il mezzo di comunicazione tra programmatore e calcolatore.

- **Linguaggi di basso livello**, orientati alla macchina
- **Linguaggi di alto livello**

LINGUAGGIO DI PROGRAMMAZIONE

È il mezzo di comunicazione tra programmatore e calcolatore.

- **Linguaggi di basso livello**, orientati alla macchina
- **Linguaggi di alto livello**, orientati al programmatore

LINGUAGGI DI ALTO LIVELLO

```
class RecentlyUsedList {
    std::vector<std::string> items_;
public:
    bool empty() const { return items_.empty(); }
    size_t size() const { return items_.size(); }
    std::string operator[](size_t index) const {
        if (index >= size())
            throw std::out_of_range("invalid subscript");
        return items_[size() - index - 1];
    }
};
```

Codice C++

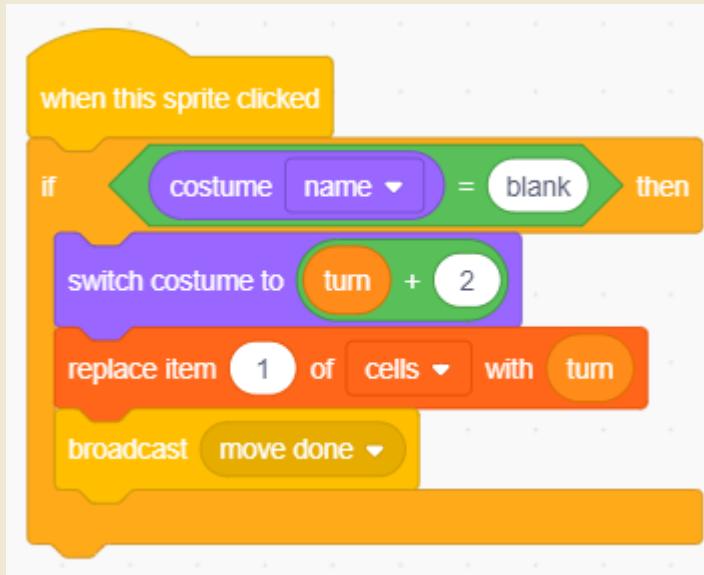
LINGUAGGI DI ALTO LIVELLO

```
class NamedOutliers(dict):

    def __init__(self, metric, items, threshold):
        self.metric = metric
        self.threshold = float(threshold)
        self.details = {}
        for name, value in items.items():
            if value > threshold:
                self.details[name] = value
        self.score = sum(self.details.values()) / self.threshold
        self.has_details = True
```

Codice Python

LINGUAGGI DI ALTO LIVELLO



Codice Scratch

COLLAUDO

No one in the brief history of computing has ever written a piece of perfect software. It's unlikely that you'll be the first.

— Andy Hunt

COLLAUDO

In caso di non conformità a seconde dei casi può essere necessaria la revisione del **programma**, dell'**algoritmo** o della **soluzione**.

COLLAUDO

In caso di non conformità a seconde dei casi può essere necessaria la revisione del **programma**, dell'**algoritmo** o della **soluzione**.

La ricerca della causa degli errori prende il nome di **debug**...

9/9

0800 Antran started
 1000 " stopped - antran ✓
 13°SC (032) MP - MC
 (033) PRO 2
 convkt

$$\left\{ \begin{array}{l} 1.2700 \cdot 9.037847025 \\ 9.037846995 \text{ convkt} \\ \hline 1.982647000 \\ 2.130476415 \end{array} \right.$$

~~(033) 4.615925059(-2)~~

2.130476415

Relays 6-2 in 033 failed special speed test
 in relay " 10.00 test .

Relay
 2145
 Relay 3370

1700 Started Cosine Tape (Sine check)
 1525 Started Multi Adder Test.

1545



Relay #70 Panel F
 (moth) in relay.

1630 Antran started.
 1700 closed down .

Il primo bug

FASI DEL CICLO DI VITA DEL SOFTWARE

- analisi
- progettazione
- codifica
- collaudo

FASI DEL CICLO DI VITA DEL SOFTWARE

- analisi
- progettazione
- codifica
- collaudo

... dunque perché fare **coding**?

PENSIERO COMPUTAZIONALE

Il **coding** abilita il **pensiero computazionale**, cioè:

[...] i processi mentali coinvolti nel formulare problemi e le loro soluzioni in modo che le soluzioni possano essere rappresentate in una forma che può essere efficacemente eseguita da un agente di elaborazione dell'informazione.

— Jeannette Wing

PENSIERO COMPUTAZIONALE

Il lato scientifico-culturale dell'informatica, definito anche **pensiero computazionale**, aiuta a sviluppare competenze logiche e capacità di risolvere problemi in modo creativo ed efficiente, qualità che sono importanti per tutti i futuri cittadini. Il modo più semplice e divertente di sviluppare il pensiero computazionale è attraverso la **programmazione** in un contesto di gioco.

— MIUR, 2014

PENSIERO COMPUTAZIONALE

- analisi
- progettazione
- codifica
- collaudo

PENSIERO COMPUTAZIONALE

- **analisi** → esercitare il pensiero
- **progettazione**
- **codifica**
- **collaudo**

PENSIERO COMPUTAZIONALE

- **analisi** → esercitare il pensiero
- **progettazione** → formalizzare il pensiero
- **codifica**
- **collaudo**

PENSIERO COMPUTAZIONALE

- **analisi** → esercitare il pensiero
- **progettazione** → formalizzare il pensiero
- **codifica** → automatizzare il pensiero
- **collaudo**

PENSIERO COMPUTAZIONALE

- **analisi** → esercitare il pensiero
- **progettazione** → formalizzare il pensiero
- **codifica** → automatizzare il pensiero
- **collaudo** → validare del pensiero

PENSIERO COMPUTAZIONALE

- **analisi** → esercitare il pensiero
- **progettazione** → formalizzare il pensiero
- **codifica** → automatizzare il pensiero
- **collaudo** → validare del pensiero

Il riscontro oggettivo e immediato della bontà della soluzione corrente permette di instaurare un ciclo virtuoso che accelera il raggiungimento della soluzione cercata

SCRATCH

The Scratch logo consists of the word "SCRATCH" in a bold, white, sans-serif font. The letters are filled with a bright orange color and have a thick, rounded black outline. The entire word is set against a white, cloud-like background that has a scalloped, bubbly edge. The entire logo is centered on a plain, light beige background.

SCRATCH

ORIGINI

ORIGINI

1967 – Seymour Papert inventa il linguaggio LOGO al MIT

ORIGINI

1967 – Seymour Papert inventa il linguaggio LOGO al MIT

L'apprendimento avviene per ricreazione, non trasmissione.

ORIGINI

1967 – Seymour Papert inventa il linguaggio LOGO al MIT

L'apprendimento avviene per ricreazione, non trasmissione.

L'apprendimento è stimolato dalla manipolazione di oggetti reali.

ORIGINI

1967 – Seymour Papert inventa il linguaggio LOGO al MIT

Uso del computer come supporto per l'apprendimento

LOGO

LOGO

Linguaggio di programmazione dedicato ai bambini.

LOGO

Linguaggio di programmazione dedicato ai bambini.

Prima occasione per loro di sperimentare la programmazione.

LOGO

Linguaggio di programmazione dedicato ai bambini.

Prima occasione per loro di sperimentare la programmazione.

Consente di ottenere subito risultati rapidi e concreti.

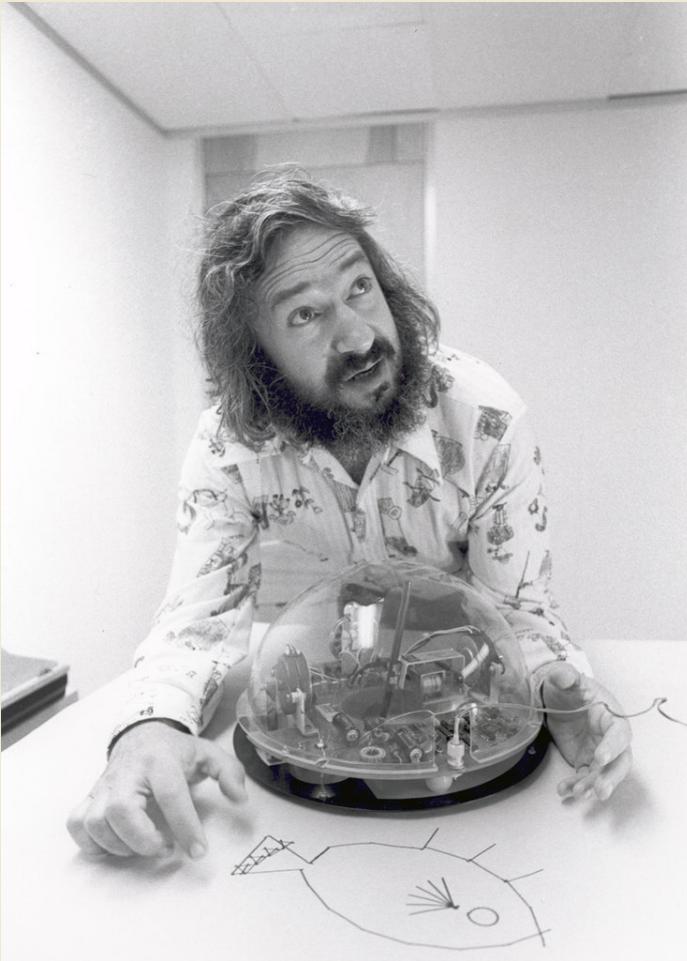
LOGO

Linguaggio di programmazione dedicato ai bambini.

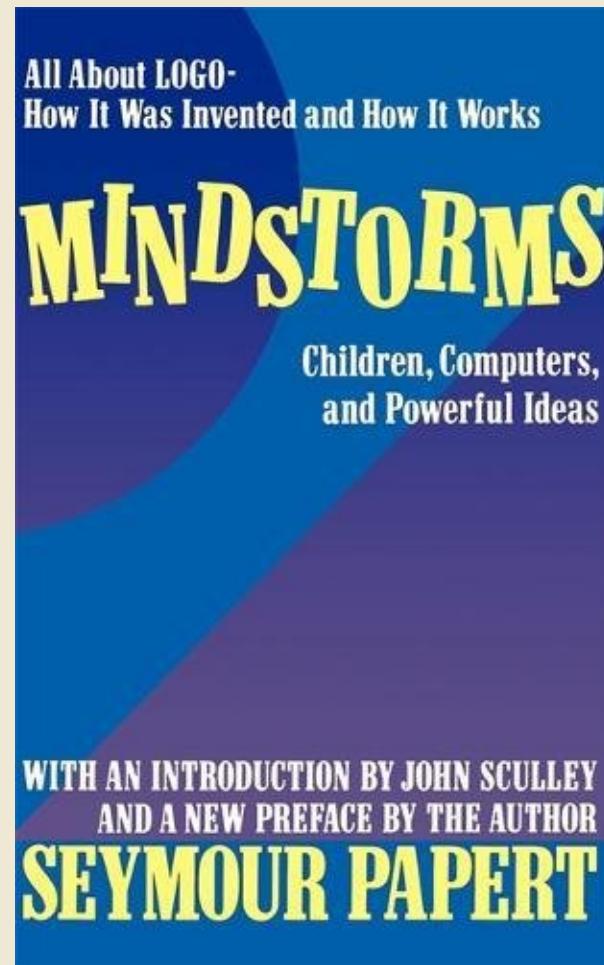
Prima occasione per loro di sperimentare la programmazione.

Consente di ottenere subito risultati rapidi e concreti.

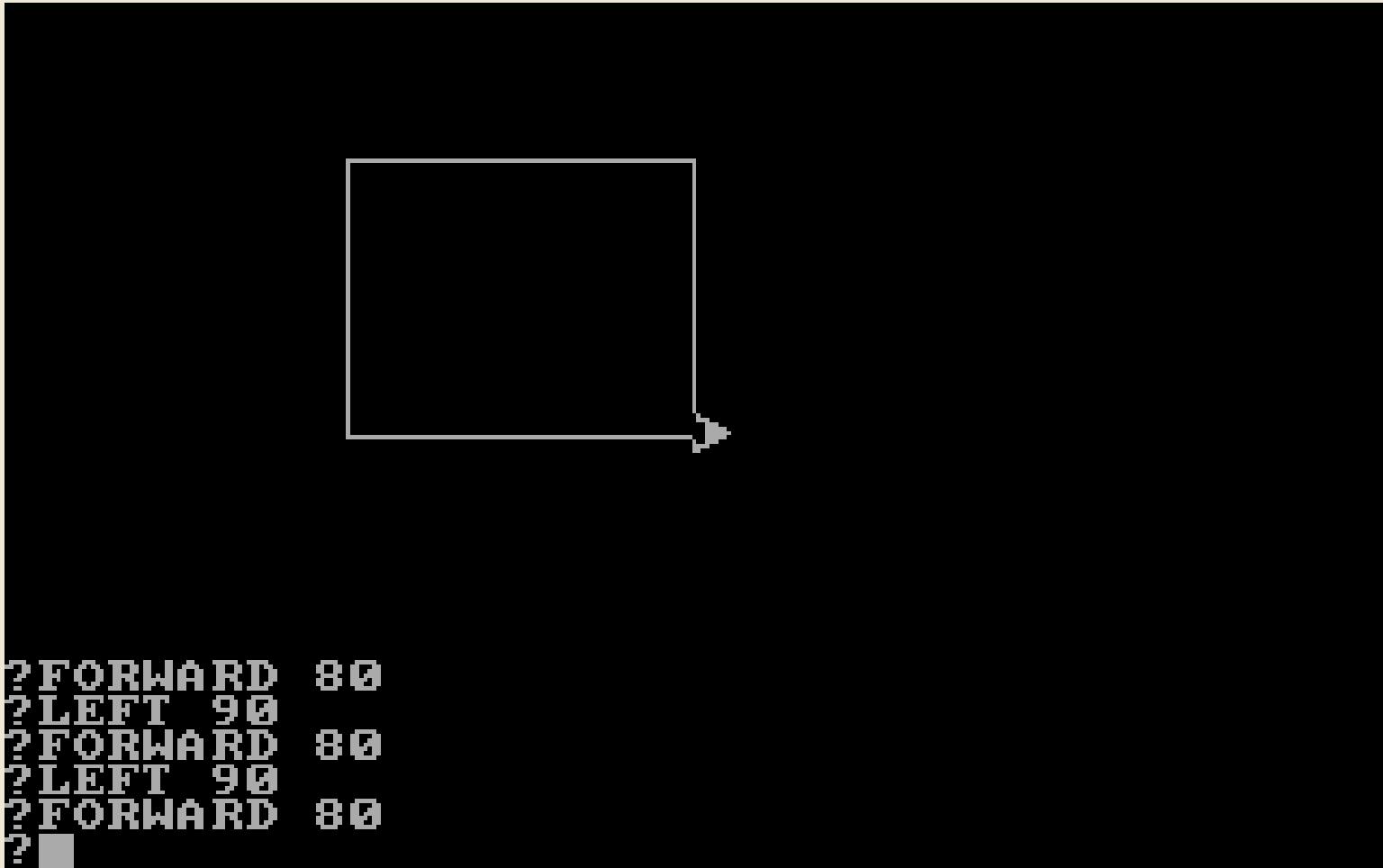
Il modulo più celebre è il **turtle graphics**.



Seymour Papert, 1969



Mindstorms, 1980



```
?FORWARD 80  
?LEFT 90  
?FORWARD 80  
?LEFT 90  
?FORWARD 80  
?■
```

LOGO per DOS, 1983

ORIGINI

1985 – Mitchell Resnick e Papert fondano il MIT Media Lab

Quando impariamo a scrivere e a contare, non lo facciamo per diventare scrittori o matematici. [...]. Ragionare come un programmatore ci permette di scomporre qualunque problema per gestirlo nel migliore dei modi, procedendo per astrazione, automazione e analisi fin dalla tenera età.

— Mitchell Resnick



Spirale dell'apprendimento creativo

4 P

4 P

- Projects
- Peers
- Passion
- Play

4 P

- **Projects:** lavorare su progetti piuttosto che esercizi
- **Peers**
- **Passion**
- **Play**

4 P

- **Projects:** lavorare su progetti piuttosto che esercizi
- **Peers:** promuovere il confronto e la collaborazione
- **Passion**
- **Play**

4 P

- **Projects:** lavorare su progetti piuttosto che esercizi
- **Peers:** promuovere il confronto e la collaborazione
- **Passion:** l'interesse innesca l'impegno e la perseveranza
- **Play**

4 P

- **Projects:** lavorare su progetti piuttosto che esercizi
- **Peers:** promuovere il confronto e la collaborazione
- **Passion:** l'interesse innesca l'impegno e la perseveranza
- **Play:** l'aspetto ludico favorisce la sperimentazione

ORIGINI

1985 – Mitchell Resnick e Papert fondano il MIT Media Lab

ORIGINI

1985 – Mitchell Resnick e Papert fondano il MIT Media Lab

Patrocinato dalla LEGO (cfr. progetto LEGO Mindstorms).

ORIGINI

1985 – Mitchell Resnick e Papert fondano il MIT Media Lab

Patrocinato dalla LEGO (cfr. progetto LEGO Mindstorms).

Inizia lo sviluppo di un linguaggio di programmazione a blocchi.

ORIGINI

1985 – Mitchell Resnick e Papert fondano il MIT Media Lab

Patrocinato dalla LEGO (cfr. progetto LEGO Mindstorms).

Inizia lo sviluppo di un linguaggio di programmazione a blocchi.

Incentrato sulla creazione di storie digitali, giochi e animazioni.

STORIA

STORIA

2007 – Scratch v. 1.0, prima versione

STORIA

2007 – Scratch v. 1.0, prima versione

2009 – Scratch v. 1.4, la grande diffusione

STORIA

2007 — Scratch v. 1.0, prima versione

2009 — Scratch v. 1.4, la grande diffusione

2013 — Scratch v. 2.0, versione on-line

STORIA

2007 – Scratch v. 1.0, prima versione

2009 – Scratch v. 1.4, la grande diffusione

2013 – Scratch v. 2.0, versione on-line

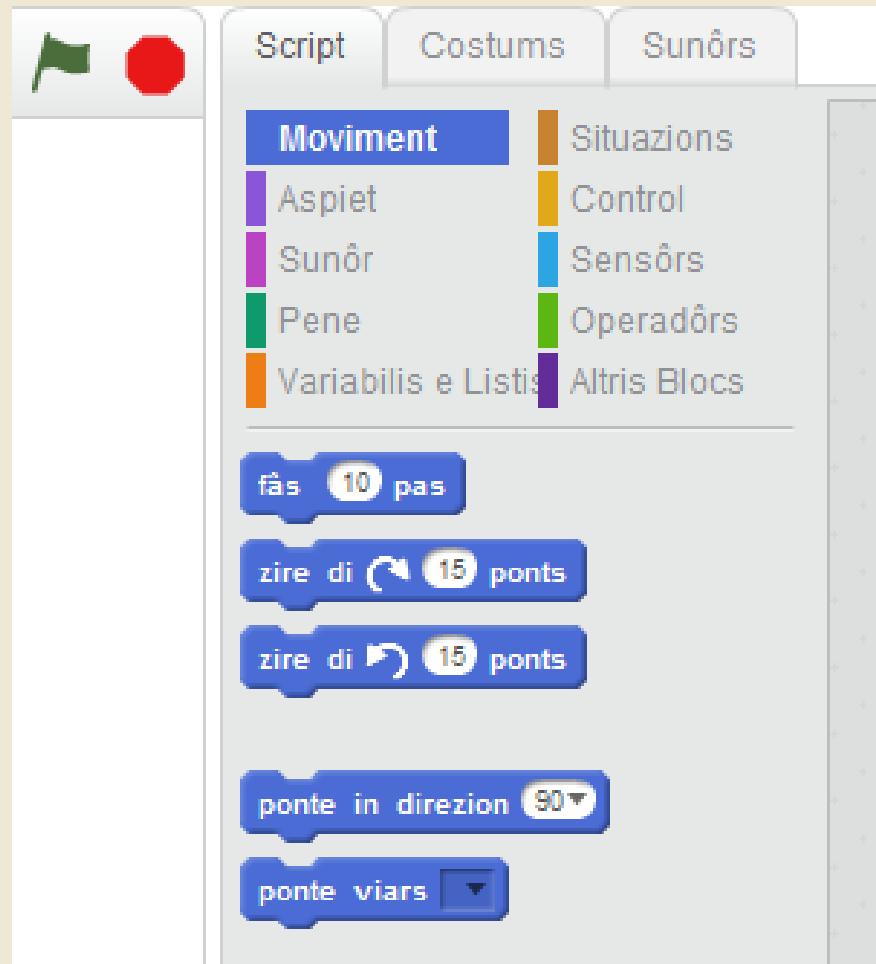
2019 – Scratch v. 3.0, Scratch Foundation

SCRATCH

- Doppia versione, online e offline

SCRATCH

- Doppia versione, online e offline
- Gratuito e Open Source



Scratch par furlan, 2016

Il friulano ammesso tra le lingue ufficiali per programmare App

Via libera dal Mit di Boston all'iniziativa partita in Fvg
L'idea nata spiegando ai bambini come sviluppare software

di Davide Francescutti
■ UDINE

Un pezzo di Friuli finisce sul sito web del Mit di Boston: il prestigioso Massachusetts Institute of Technology ha accettato la versione in marlenghe del proprio linguaggio di programmazione Scratch, fruibile liberamente da tutti nel mondo. Il merito è del gruppo di programmatore e insegnanti che ruota attorno al mondo dei Coderdojo Friuli Venezia Giulia, vere e proprie palestre digitali in cui i bambini possono imparare a programmare da soli piccoli programmi, app e giochi didattici proprio con questo linguaggio open source sviluppato dall'istituto statunitense.

«Usiamo da sempre Scratch - spiega a nome del team di lavoro friulano l'informatico Gianfranco Zuliani, 46 anni di Bressa - per la sua facilità d'uso, per lo spirito che ani-

ma la comunità di utilizzatori e non ultimo per il fatto che si tratta di un software gratuito». Zuliani insieme agli altri "maestri" del dojo tiene lezioni itineranti sia nei paesi regionali che nelle scuole e proprio in una di queste, lo Zancr di Udine, in occasione del "Tablet School Day" nel 2015, ci fu un incontro con insegnanti di friulano nelle scuole primarie del territorio.

«Notando la quantità di lingue in cui è possibile configurare l'interfaccia del software - aggiunge Zuliani - mi fu chiesto se sarebbe stato possibile inserire anche il friulano, perché ciò avrebbe permesso di usare Scratch durante le lezioni».

Tramite il contatto dell'altro membro del dojo Matteo Troia, s'iniziò a lavorare sul progetto. La "sfida" era quella di realizzare una versione del programma utilizzabile anche off line scaricandolo nel proprio computer, visto che

non sempre nelle scuole si può contare su una connessione stabile. «Nel giro di qualche serata - ricorda l'informatico - sono venuto a capo della cosa: ho predisposto i file necessari e li ho passati all'insegnante che si è occupata della traduzione di gran parte degli oltre 500 testi. Io mi sono concentrato sulla parte più tecnica del lavoro, dal formato dei file alle codifiche».

Da qui la prima fase ci test lungo lo scorso anno scolastico, in cui gli alunni hanno programmato in friulano per la prima volta. «Rimaneva però un piccolo problema - scatolina Zuliani -: ogni volta dovevano fare il caricamento manuale dei testi in quanto il friulano non era tra le lingue ufficialmente supportate. Quest'estate ho quindi contattato direttamente il Mit a Boston, seguendo le loro indicazioni per ottenere finalmente il riconoscimento della nostra versione».



In alto alcuni ragazzi del Coderdojo e sotto la schermata approvata dal Mit

Sono state rese necessarie ulteriori 200 parole ma alla fine, dal primo settembre scorso, l'editor in friulano di Scratch è disponibile sul sito scratch.mit.edu. I più felici, in questa operazione digitale, sono stati i bambini. «Siamo - conclude Zuliani - soddisfatti dei risultati: le frasi semplici favoriscono la comprensione

anche di chi conosce poco la lingua».

I Coderdojo, avviati nel 2014 dall'esperto digitale Giampiero Riva e da Simone Puksic (poi diventato presidente di Insell), ripartiranno in autunno con nuovi appuntamenti negli istituti scolastici.

OPPRODUZIONE RISERVATA

Scratch par furlan, 2016

SCRATCH

- Doppia versione, online e offline
- Gratuito e Open Source

SCRATCH

- Doppia versione, online e offline
- Gratuito e Open Source
- Ricca e vivace comunità virtuale

SCRATCH

- Doppia versione, online e offline
- Gratuito e Open Source
- Ricca e vivace comunità virtuale
- Consente la condivisione e il remix dei progetti

ALTRI STRUMENTI

ALTRI STRUMENTI

- ScratchJr, per bambini in età prescolare



ScratchJr

ALTRI STRUMENTI

- ScratchJr, per bambini in età prescolare

ALTRI STRUMENTI

- ScratchJr, per bambini in età prescolare
- ScratchEd, risorse per insegnanti (fermo dal 2019)

ALTRI STRUMENTI

- ScratchJr, per bambini in età prescolare
- ScratchEd, risorse per insegnanti (fermo dal 2019)
- Teaching with Scratch Discussion Group

www.scratch.mit.edu