It's coding time

Pensiero Computazionale a scuola

Borgotaro 25 Novembre 2015

CHI SIAMO

Michael Lodi

lodi.ml

Docente di Informatica al Liceo e Tutor all'Università

Formatore USR-ER e Mentor di CoderDojo Bologna

Giovanni Govoni

@giovannigovoni

Servizio Marconi TSI USR-ER

Vicario I.C. San Pietro in Casale (BO)

“...CODING COME UN NUOVO MODO, PER LE PERSONE, DI ORGANIZZARSI, ESPRIMERSI E CONDIVIDERE LE PROPRIE IDEE.”

Mitchell Resnick MIT Media Lab Boston

TUTTO PARTE DA QUI?

OPPURE DA QUI?

O DA QUI?

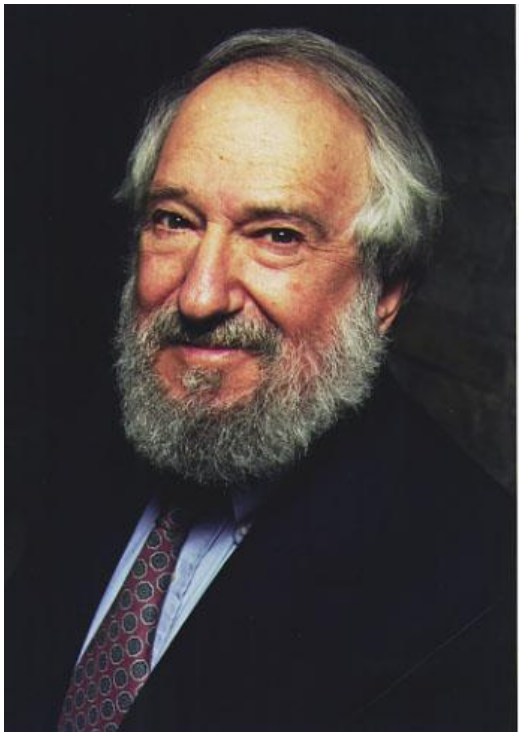
E DA QUI?

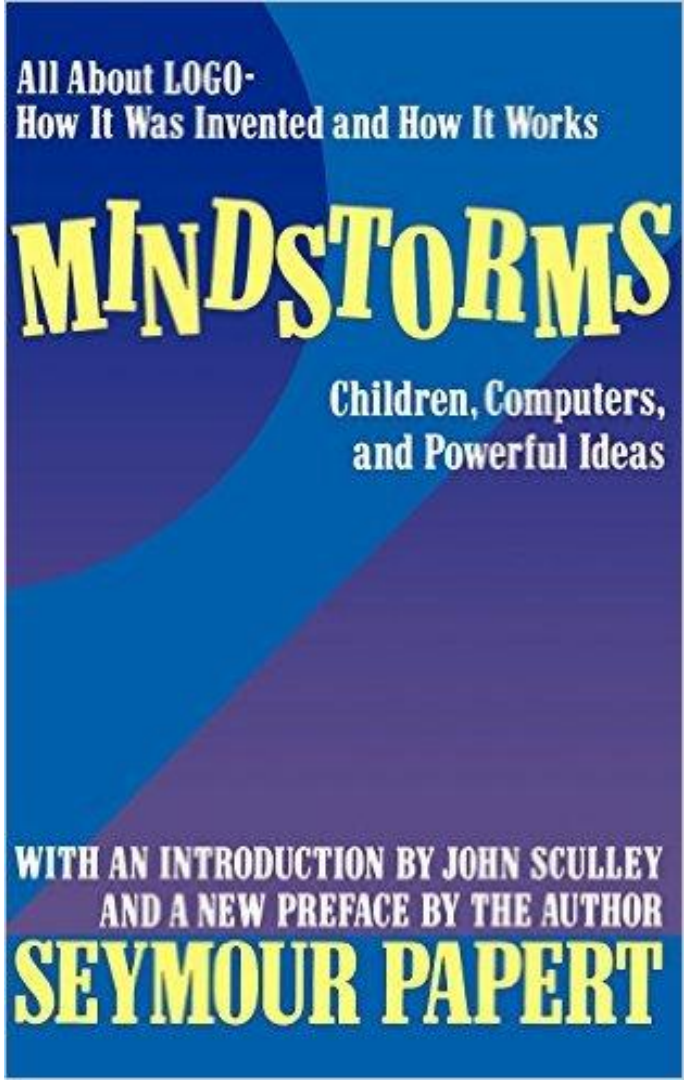
SCUOLA DI BARBIANA

LETTERA A UNA PROFESSORESSA

LIBRERIA
EDITRICE
FIORENTINA

SICURAMENTE PASSA DA QUI ...





E PERCHE' NON DA QUI?


STAR WARS
Building a Galaxy with Code
Blocks
Use drag-drop blocks.
Many languages | Modern browsers, smartphones, tablets | Ages 6-10
Coming soon
JavaScript
Use drag-drop blocks and JavaScript.
English only | Modern browsers | Ages 11+
Try now


MONTH
DAY
YEAR
PM
HOUR
MIN
OCT
26
1985
09:00
DESTINATION TIME
MONTH
DAY
YEAR
PM
HOUR
MIN
OCT
21
2015
07:28
PRESENT TIME
MONTH
DAY
YEAR
PM
HOUR
MIN
NOV
12
1955
06:38

Perché?

- essere fluenti con le nuove tecnologie
  - capire il mondo che ci circonda (così come le altre materie)
  - per “uguaglianza sociale”
  - per trovare lavoro
- pensare computazionalmente
  - risolvere problemi
  - insegnare (al computer) per imparare meglio
    Learn to code - code to learn
- creatività
  - da utenti passivi a creatori attivi (saper “scrivere” oltre che “leggere”)
  - esprimere se stessi

Pensare il coding in maniera pedagogica

- funzionalistico (imparare a programmare)
- aspetto espressivo (CoderDojo+Maker)
- aspetto interpretativo (il codice come linguaggio)
- aspetto emancipatorio (etica hacker, ripensare le interfacce e le interazioni)

appunti da cit. prof. Pier Cesare Rivoltella

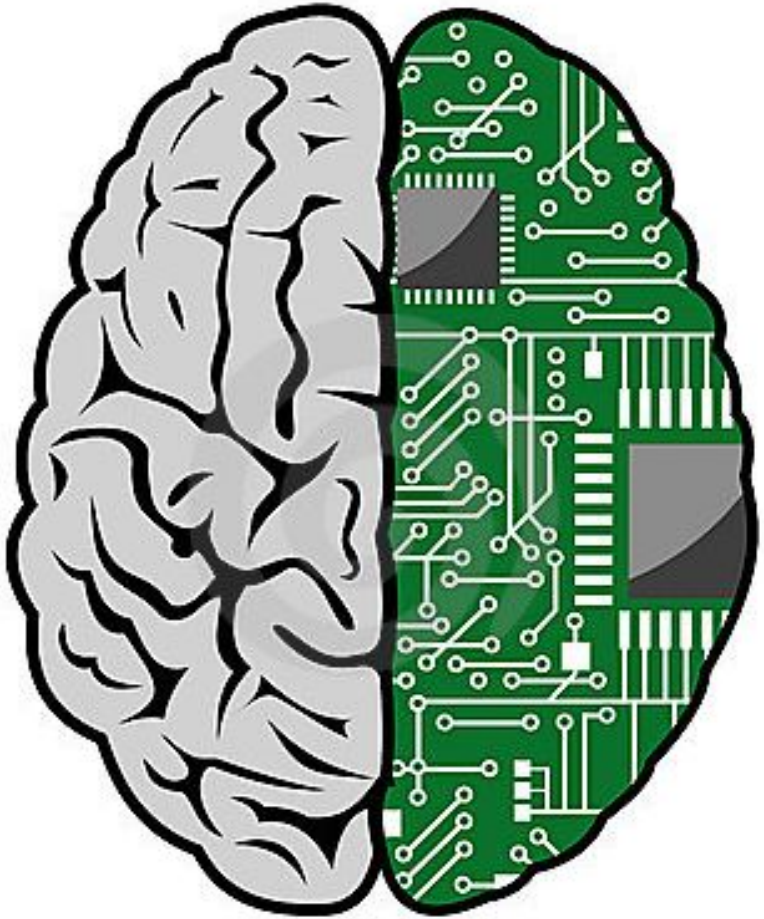
PLUGGED O UNPLUGGED ?

CODING TRA L’ANALOGICO ED IL DIGITALE

Pensare **come un informatico** per risolvere problemi (in ogni ambito della vita)

Processo di **formulazione di problemi e di soluzioni** in una forma che sia eseguibile da un "agente che processa informazioni"

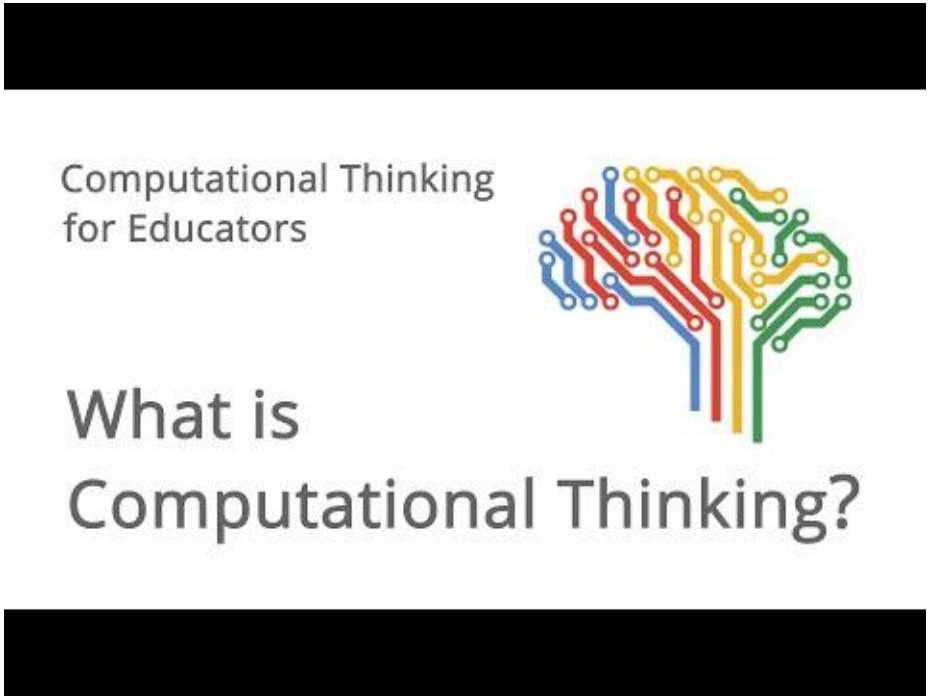
NON È PENSARE COME UN COMPUTER!




100% HUMAN!

“Ma io non insegno Informatica...”

Pensiero Computazionale ≠ Informatica



Origini

Alan Perlis (1962)

Gli studenti di tutte le discipline dovrebbero imparare la programmazione e la teoria della computazione.

Programmare favorisce il pensiero procedurale, da applicare a tutti gli altri aspetti della vita.

Seymour Papert (1996)

Jeannette Wing (2006)

Oltre a leggere, scrivere e calcolare, bisogna insegnare il pensiero computazionale ad ogni bambino.

Una possibile definizione

Concetti

Pratiche

Prospettive

Concetti
(che gli sviluppatori utilizzano quando programmano)

sequenze

condizionali

ripetizioni

eventi

parallelismo

operatori

dati (collezione, analisi, rappresentazione)

Pratiche
(che si apprendono programmando)

(ma anche studiando informatica)

essere incrementali e iterativi

testing e debugging

riuso e remixing

astrazione

generalizzazione e riconoscimento di pattern

decomposizione

automazione

simulazione

efficienza (calcolabilità e complessità)

Prospettive (i modi di vedere il mondo e se stessi che si sviluppano programmando)

esprimere se stessi (creare)

connettersi (collaborare)

farsi domande (riflettere)

saper gestire la complessità e i problemi difficili

tolleranza per l’ambiguità e i problemi aperti

… essere felici? :)

2-12-14

5 Things I have learnd about Programing

1.) that computers are really dumb, they only do what you tell it to do.

2.) It's very hard! (you have to be very specific)

3.) what you tell it to do has to be in the right order.

4.) you can do many differen things with a computer.

5.) You always have to check your work

Come insegnarlo?

- Insegnando a programmare
  - Così hanno imparato gli informatici
  - Forse è imprescindibile

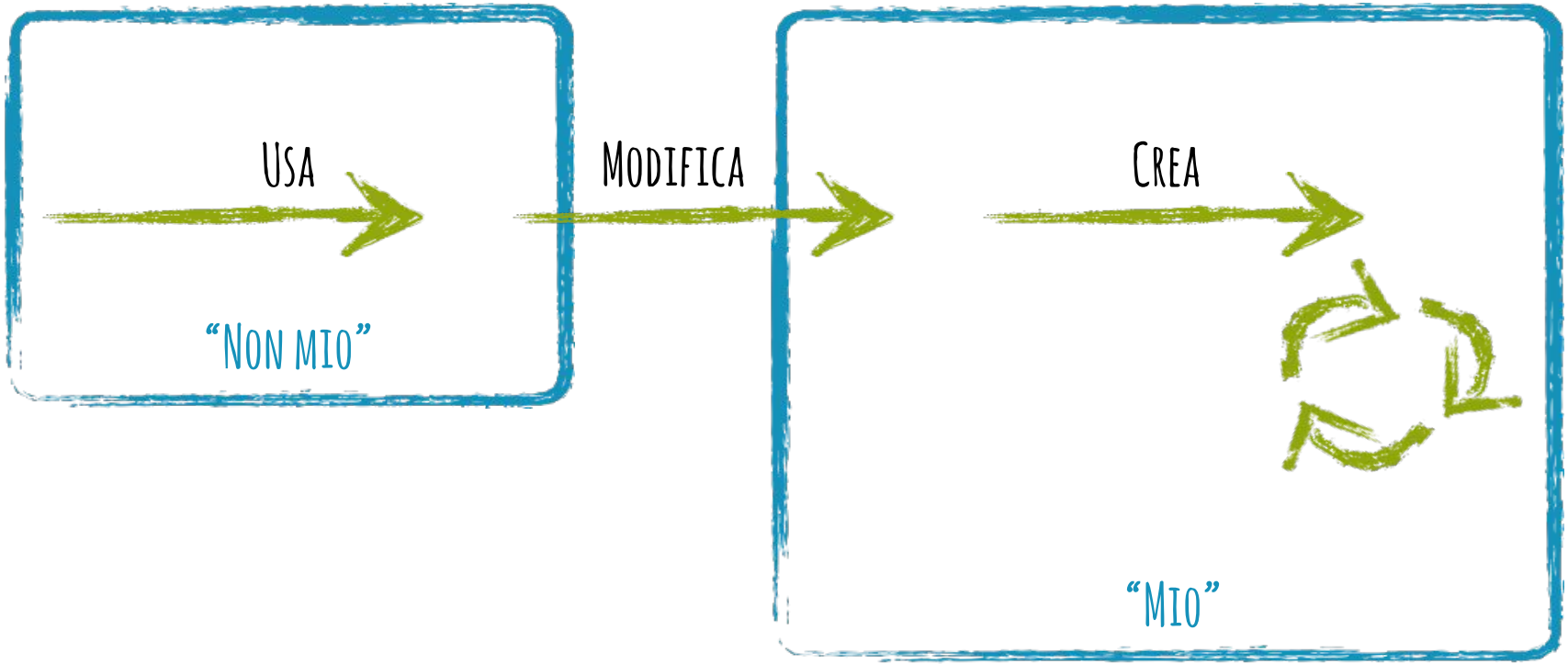
- Insegnandolo in altre attività
  - Concetti e pratiche sono trasversali

In pratica? - Programmare

- Approccio più “scolastico”
  - programmailfuturo.it (e molti altri…)
  - Un punto di partenza, focus sul “problem solving”

- Approccio “alla CoderDojo”
  - Scratch/Snap!, AppInventor, Arduino, Python, Android...
  - Ricerca pedagogica dell MIT, focus sulla “creatività”

Usa, modifica, Crea



In pratica? - Altre attività

- Attività “unplugged” (“senza rete”)
  - Prediligere quelle in cui gli studenti “impersonano” agenti computazionali

- Favorire l’apprendimento di concetti e pratiche “mentre si insegna altro”
  - La programmazione diventa strumento

E I ROBOT?

- Il “cervello” dei Robot va programmato!
  - Spesso con linguaggi a blocchi… almeno all’inizio

- Le istruzioni potrebbero essere diverse…
  - ...ma i concetti del pensiero computazionale (es. Sequenza, ripetizioni, condizionali) non cambiano!

Riferimenti

PENSIERO COMPUTAZIONALE

- Wing, J. M. (2006). Computational Thinking. Communications of the ACM, 49(3), 33-35.
- Computational thinking with Scratch
- Programmailfuturo.it
- Una bella tesi di laurea… ;)
- Presentazione di Dr. Scratch.

Riferimenti

SCRATCH

- Risorse da CoderDojo:
  http://kata.coderdojo.com/wiki/Learning_Resource
- Forum Italiano di Scratch
  http://scratch.mit.edu/discuss/21/
- Ottime risorse “scolastiche” per docenti e studenti (scuole superiori - ma con materiale utile a tutti)
  https://it.wikibooks.org/wiki/File:Diderot_2014_Guida_Studenti.pdf
  https://it.wikibooks.org/wiki/File:Diderot_2014_Guida_docenti.pdf

Riferimenti

MIT CREATIVE *(computing, learning, etc.)

- *S. Papert,* Mindstorms: Children, Computers, and Powerful Ideas (1980)
- Guida al corso di Informatica Creativa, ScratchEd
- Apprendimento creativo al MIT MediaLab
- Costruttivismo e programmazione informatica dalle teorie di Piaget all'esperienza di CoderDojo (A. Lombardo, C. Presicce). In pubblicazione.
- M.Resnick, D.Siegel, Un approccio diverso al Coding Come i bambini costruiscono e ricostruiscono se stessi da zero