Il coding e le sue potenzialità didattiche

Laura Lana* e Vittorio Mazzoli**

*Laureata in Lettere Moderne e giornalista pubblicista. **Laureato in Consulenza pedagogica e coordinamento di interventi formativi ed è esperto di informatica. Entrambi hanno conseguito una seconda laurea in Scienze della formazione primaria e sono docenti abilitati per la scuola dell'infanzia e primaria.

L'articolo offre un approfondimento sul coding e sulle potenzialità didattiche del pensiero computazionale attraverso un excursus sulla letteratura di riferimento e una dettagliata lettura dei documenti ministeriali. Infine, fornisce una breve analisi per comprendere i concetti base dei linguaggi informatici e del funzionamento dei principali software usati a scuola come Scratch, che si avvalgono della cosiddetta "programmazione visuale a blocchi".

Introduzione

Coding è un termine inglese al quale corrisponde l'italiano "programmazione", cioè la scrittura di una sequenza di istruzioni che fanno eseguire ad un calcolatore determinate operazioni.

Non si pensi, però, soltanto a computer e software. Il nostro stesso cervello lavora così: individuazione di un problema, scelta di una soluzione, attivazione di una serie di istruzioni per metterla in atto. Quello che, in termini tecnici, viene definito "pensiero computazionale".

I processi propri del pensiero computazionale vengono messi in atto tutti i giorni, per esempio quando stabiliamo il percorso più breve per raggiungere una destinazione oppure dobbiamo preparare un piatto di pasta: la procedura per passare dalla situazione iniziale (ingredienti della ricetta) a quella finale (piatto di pasta pronto da mangiare) è data da un algoritmo (procedimento che risolve un determinato problema attraverso un numero definito di passi elementari) che viene eseguito dall'individuo.

Proprio per questo, lo sviluppo del pensiero computazionale può essere stimolato da attività logiche svolte utilizzando comuni materiali ed oggetti, da soli o in gruppo, chiamate attività *unplugged* poiché utilizzano strumenti non digitali (Bonaiuti et al., 2017).

Un discorso a parte può essere fatto per il *Coding* in quanto concetto estremamente legato a quello del pensiero computazionale. Infatti, secondo la definizione che ne dà Bogliolo (2016) nel suo testo "*Coding in your*

classroom, now!", il *coding* non può essere semplicemente tradotto in italiano con il termine "programmazione", ma in modo più complesso come «l'uso di strumenti e metodi di programmazione visuale a blocchi per favorire lo sviluppo del pensiero computazionale» (p. 13).

Alfieri (2016, p. 3) aggiunge: «il *coding* è la palestra del pensiero computazionale che va stimolato e allenato sin da piccoli» e, poiché consente di apprendere le basi della programmazione informatica in modo pratico e divertente, è lo strumento didattico più utilizzato per educare i bambini al pensiero computazionale.

Va precisato, però, che il *coding* non può essere totalmente identificato con il pensiero computazionale, anche se attualmente rimane lo strumento più diffuso per favorirne l'acquisizione (Giordano e Moscetti, 2016). Infatti, come sottolineano Bocconi et al. (2016), il pensiero computazionale utilizza strategie come algoritmi, astrazione e *debugging* (individuare e correggere *bug*, errori), che sono necessarie anche nel *coding*, il quale ha in più il merito di rendere concreti dei concetti altrimenti molto astratti chiedendo agli alunni di mettere in gioco in maniera pratica e tangibile le proprie competenze di logica e programmazione.

«Il pensiero computazionale è più della codifica di una soluzione eseguibile da un computer attraverso la programmazione» (Grover e Pea, 2017, p. 13). Significa pensare come un informatico, inteso però come abilità generale che chiama in causa astrazione, rappresentazione e capacità di analisi e che non va ridotta alla sola capacità di programmare utilizzando uno specifico linguaggio. L'informatico, nello sviluppo di un software, non si limita alla sola scrittura del codice, ma progetta la struttura del programma dopo aver compreso e definito il problema (es. esigenza del cliente) attraverso una sua analisi ed esplorazione, consapevole che non esiste un'unica soluzione, ma tante, e che differiscono tra loro soprattutto in termini di efficienza e adattabilità a nuove situazioni.

Perché fare *coding* a scuola

Nei documenti ufficiali della Commissione Europea si afferma che l'acquisizione di competenze digitali, incluso il *coding*, è fondamentale per lo sviluppo, la competitività e per il mercato del lavoro, ma l'educazione dovrebbe invece puntare sullo sviluppo di capacità generali relative ad identificazione, rappresentazione, comunicazione e condivisione di situazioni problematiche e relative soluzioni, per poter far fronte alla nostra società in continua trasformazione (Olimpo, 2017).

Per questo motivo, da docenti occorre approcciarsi all'insegnamento del *coding* consapevoli di tutte le sue potenzialità trasversali, in quanto:

- favorisce lo sviluppo della creatività perché un problema può essere affrontato e risolto in differenti modi;
- è proficuo perché produce risultati visibili;
- aiuta a saper affrontare la complessità perché imparare a risolvere problemi informatici aiuta ad essere capaci di risolvere problemi in altri ambiti;
- contribuisce allo sviluppo della precisione poiché un programma per funzionare bene necessita della correttezza di tutte le istruzioni e quindi di individuare e correggere eventuali errori (Alfieri, 2016).

Inoltre, secondo Giordano e Moscetti (2016, p. 7) programmare:

- "è un potente strumento di pensiero": programmare un computer per fargli risolvere un problema richiede di aver analizzato il problema ed aver pianificato in modo dettagliato una soluzione, e ciò permette di apprendere qualcosa di nuovo anche indipendentemente dalla programmazione stessa. L'unico vincolo è che se si vuole programmare per imparare è necessario prima aver imparato a programmare;
- "può essere un potentissimo strumento di espressione personale": programmare permette di produrre qualcosa di personale in modo creativo, da una presentazione, al videogioco, all'animazione digitale;
- "può essere uno strumento di crescita personale": quando si programma si procede in modo incrementale, cioè si scrive del codice, si verifica se viene correttamente eseguito e, se così non avviene, si modifica il codice, procedendo in modo ciclico (codice-esecuzione-codice) per perfezionare continuamente il programma. Operare in questo modo permette di sviluppare l'intelligenza, grazie all'impegno nell'affrontare le difficoltà, e una forma mentis aperta all'imprevisto, che vede l'imperfezione e l'errore come occasione per migliorare.

Non è un caso, infatti, che "*Learn to code, code to learn*" sia lo slogan di Scratch, uno dei più noti programmi utilizzati per il *coding*, a significare che imparando come si programma "si impara ad imparare": si apprendono strategie per risolvere problemi, fare progetti e comunicare idee, utili a tutti, in modo iidipendente dall'età, dall'occupazione o dal livello socio-economico-culturale (Resnick, 2013).

Su questa convinzione è nata nel 2013 negli Stati Uniti "Code.org" (https://code.org/), organizzazione non-profit a cui partecipano milioni di studenti e insegnanti di tutto il mondo, desiderosi di imparare a programmare. L'obiettivo è quello di aumentare l'accesso all'informatica, la partecipazione delle donne e di minoranze in modo da sviluppare le competenze che aiutino gli allievi di tutto il mondo a dialogare con i computer e a non essere solo dei semplici consumatori, ma creatori e cittadini attivi. Code.org, attraverso corsi suddivisi per età, a partire dai quattro anni, non vuole insegnare uno specifico linguaggio di programmazione, ma i concetti che ne sono alla base in quanto comuni a tutti i linguaggi di programmazione.

Il *coding* nella normativa italiana

Anche l'Italia si sta muovendo da qualche anno in merito all'introduzione di attività di pensiero computazionale e *coding* all'interno della realtà scolastica.

A partire dall'anno scolastico 2014-15 è stato avviato il progetto "Programma il Futuro" avviato dal Consorzio Interuniversitario Nazionale per l'Informatica (CINI) in collaborazione con il Ministero dell'Istruzione, Università e Ricerca (MIUR) con l'obiettivo di inserire un'adeguata educazione al pensiero computazionale come disciplina nei vari ordinamenti scolastici italiani.

A fine ottobre 2015 è stato, inoltre, adottato il Piano Nazionale Scuola Digitale (PNSD), che ha introdotto l'insegnamento del pensiero computazionale nella scuola primaria.

L'obiettivo non è far diventare tutti dei programmatori informatici, ma far capire i princìpi alla base del funzionamento dei sistemi e della tecnologia informatica. Oggi i computer sono ovunque, una comprensione dei concetti di base dell'informatica è indispensabile per tutti i futuri cittadini e lavoratori.

Il 16 luglio 2015 è entrata in vigore la legge n. 107 del 13 luglio 2015 recante: «Riforma del sistema nazionale di istruzione e formazione e delega per il riordino delle disposizioni legislative vigenti» detta "La Buona Scuola". Uno dei punti della riforma ha riguardato proprio le cosiddette competenze digitali e, in particolare, il pensiero computazionale e il *coding* (punto *h*, p. 2).

Si legge, infatti, al comma 56:

> *al fine di sviluppare e di migliorare le competenze digitali degli studenti e di rendere la tecnologia digitale uno strumento didattico di costruzione delle competenze in generale, il Ministero dell'istruzione, dell'università e della ricerca adotta il Piano Nazionale per la Scuola Digitale*

per il quale il comma 58 specifica i seguenti obiettivi:

- sviluppo delle competenze digitali degli studenti, anche collaborando con università, associazioni, organismi del terzo settore e imprese;
- migliorare la formazione e l'innovazione delle istituzioni scolastiche, potenziando gli strumenti didattici e laboratoriali;
- favorire la governance, la trasparenza e la condivisione di dati (anche relativamente alle migliori esperienze delle istituzioni scolastiche);
- formazione dei docenti, dei dirigenti, degli assistenti amministrativi e degli assistenti tecnici;
- potenziamento delle infrastrutture di rete, in particolare la connettività nelle scuole;
- adottare testi didattici in formato digitale, anche prodotti autonomamente dagli istituti scolastici.

Il Piano Nazionale per la scuola digitale[1] prevede 35 azioni di cui l'azione #15[2], relativa a

> *Scenari innovativi per lo sviluppo di competenze digitali applicate" specifica che "nel quadro più ampio rivolto allo sviluppo del pensiero computazionale […] piattaforme e linguaggi diversi, con o senza il computer [...] la robotica educativa, i percorsi unplugged (senza l'uso del PC), le interazioni tra programmazione a blocchi e schede, la programmazione di droni o stampanti 3D possono essere efficacemente integrati in percorsi didattici interdisciplinari per lo sviluppo delle competenze"* (p. 77).

Per favorire la diffusione del *coding* a scuola, nell'ambito dell'azione #17, relativa a "Portare il pensiero computazionale a tutta la scuola primaria", si cita espressamente "Programma il futuro" (https://programmailfuturo.it/) come iniziativa di riferimento per portare nelle scuole i concetti di base dell'informatica grazie a strumenti semplici e divertenti. L'obiettivo è che tutti gli studenti della scuola primaria svolgano 10 ore annuali di logica e pensiero computazionale.

L'azione #17 prevede che "Programma il Futuro" sia esteso a tutte le scuole e che siano avviate attività sperimentali più ampie e di tipo laboratoriale, coinvolgendo anche la scuola dell'infanzia in azioni dedicate. Il modo più semplice e divertente di sviluppare il pensiero computazionale è attraverso la programmazione (coding) in un contesto di

gioco" (circolare del MIUR del 8/10/2015[3], p.2).

"Programma il futuro" si rifà alle attività e all'esperienza del portale Code.org e fornisce supporto agli insegnanti sui concetti di base della programmazione, tramite percorsi strutturati, videolezioni (canale YouTube), forum di discussioni e attività di formazione.

Sono previsti due percorsi: uno base, che consente la partecipazione all'"Ora del Codice" svolgendo attività di avviamento al pensiero computazionale in un'ora, e l'altro avanzato che consiste nel far seguire percorsi più strutturati in base alla fascia d'età (dai 4 anni in su). Le attività vengono svolte accedendo al sito https://studio.code.org/: non richiedono nessuna conoscenza avanzata nell'uso del computer.

I percorsi sono caratterizzati dalla presenza di "Lezioni tecnologiche" e di "Lezioni tradizionali" (senza l'utilizzo del computer), organizzate in modo sequenziale e strutturate in corsi di diverso tipo che l'insegnante può gestire in relazione alla propria classe.

Nelle Indicazioni Nazionali (MIUR, 2012) si parla dell'introduzione ai linguaggi di programmazione nella parte dedicata alla Tecnologia con l'obiettivo di sviluppare la creatività, realizzare progetti e comprendere il rapporto diretto esistente tra codice sorgente e risultato visibile.

L'importanza dello sviluppo del pensiero computazionale è ribadita anche nel documento "Indicazioni nazionali e nuovi scenari" (MIUR, 2017), a cura del Comitato Scientifico Nazionale per le Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione, in cui si afferma: "Nei contesti attuali, in cui la tecnologia dell'informazione è così pervasiva, la padronanza del *coding* e del pensiero computazionale possono aiutare le persone a governare le macchine e a comprenderne meglio il funzionamento, senza esserne invece dominati e asserviti in modo acritico" (p.13).

Capire il *coding*

Non è facile comprendere il *coding* per dei "non addetti ai lavori" come sono gran parte dei docenti. Ma è fondamentale conoscere almeno le basi del suo funzionamento per poterlo portare nelle classi nella maniera più consapevole ed efficace possibile.

Alla base del *coding* ci sono i concetti fondamentali dei linguaggi informatici, che di seguito vengono riassunti (Brennan e Resnick, 2012; Lodi e Marchignoli, 2016; Lodi, 2016) e che spiegano come opera qualsiasi programma informatico che ci troviamo ad utilizzare:

- si rispetta sempre una *sequenza*, ovvero un ordine in cui vengono scritte le istruzioni, nulla è casuale;
- c'è una *condizione*, ovvero il programma esegue differenti istruzioni in relazione a specifici casi;
- c'è una *ripetizione (ciclo)*: un programma può eseguire le stesse istruzioni per un certo numero di volte;
- tutto nasce da un *evento*: un programma utilizza alcune istruzioni solo nel momento in cui accade qualcosa (es. viene cliccato un pulsante con il mouse);
- può verificarsi un *parallelismo*: più istruzioni possono essere eseguite contemporaneamente;
- si chiamano *operatori* le espressioni logiche e matematiche utilizzate all'interno delle istruzioni;

- si parla di *dati* quando si analizzano, memorizzano e rappresentano informazioni.

I "modi di lavorare, pensare e approcciare i problemi" nel mondo informatico e, di conseguenza, nell'ambito del *coding* (Lodi e Marchignoli, 2016, p. 27) sono:

- incrementali e interattivi: è molto più semplice lavorare in modo incrementale su un progetto, testandolo e variandolo, piuttosto che tentare di realizzarlo perfetto al primo tentativo;
- si lavora per *testing* e *debugging*, in quanto per determinare se un programma informatico funzioni è utile provarlo alla ricerca di eventuali errori e cercare soluzioni per correggerli;
- si usa il riuso e *mixing*: è più comodo riutilizzare del codice già scritto (anche da altri) piuttosto che scrivere ogni volta un programma da zero;
- si mettono in campo attenzione ad efficienza, calcolabilità e complessità: il risultato, cioè la soluzione di un problema, deve essere il migliore ma usando meno risorse possibili. A questi Lodi (2016) aggiunge la scomposizione, cioè giungere alla soluzione dell'intero problema suddividendolo in parti più semplici e facilmente risolvibili, e l'astrazione, cioè concentrarsi sugli aspetti importanti per la soluzione ed il riconoscimento di pattern e generalizzazione, cioè riconoscere che alcune parti della soluzione possono essere utilizzate in problemi simili.

I linguaggi di programmazione

In informatica si definisce *script* il codice scritto utilizzando un determinato linguaggio di programmazione, detto appunto linguaggio di *scripting*. Un linguaggio di programmazione è un linguaggio formale attraverso cui un programmatore specifica le istruzioni, cioè il codice sorgente di un programma, che possono essere usate da un computer per produrre dati in output. Sono esempi di linguaggi di *scripting* JavaScript, Python, Visual Basic, ActionScript e PHP e sono di tipo testuale.

I linguaggi di programmazione possono essere di due tipi: testuali (più complessi, come quelli nominati poco sopra) o visuali (più intuitivi e adatti anche ai principianti ed in particolare per l'utilizzo nella scuola dell'infanzia e primaria).

Nel caso dei linguaggi testuali il codice viene scritto usando un editor di testo (ve ne sono anche alcuni con la funzione di autocompletamento che "suggerisce" il codice e i parametri semplificando e velocizzando la scrittura) e le istruzioni sono sequenze di parole (e relativi parametri) tipicamente in inglese e salvati in un file che verrà compilato o interpretato per essere eseguito. La compilazione permette di creare un file eseguibile (ad esempio .exe o .msi in Windows), mentre nell'interpretazione il codice viene direttamente eseguito da un'applicazione (che funge da interprete traducendo le istruzioni in linguaggio macchina, che è di tipo binario).

I linguaggi visuali hanno come punto di forza la possibilità di scrivere del codice senza preoccuparsi della sintassi (correttezza della scrittura di istruzioni e parametri), risultando meno flessibili ma permettendo di concentrarsi sull'apprendimento dei concetti di base della programmazione (eventi, cicli, variabili ecc.) (Faccioli, 2019).

Il vantaggio della programmazione a blocchi è che non richiede di conoscere uno specifico codice di programmazione, ma è sufficiente manipolare degli oggetti (blocchi

colorati di forme e dimensioni diverse), spostandoli, unendoli e specificandone dei parametri all'interno di un programma dedicato, detto ambiente di programmazione visuale, in cui ad ogni blocco corrisponde una o più righe di codice, cioè un comando, un'istruzione.

Per questo motivo software come Scratch e Blockly scelgono la programmazione a blocchi, che è molto adatta con gli studenti.

Nel suo libro "*Coding in your classroom, now!*" Bogliolo (2016) afferma: "La programmazione visuale non è nient'altro che un metodo di rappresentazione che ci permette di esprimere un procedimento come concatenazione di blocchi colorati che ne rappresentano i passi elementari, o le istruzioni che li descrivono. I blocchi hanno di solito piccoli incastri che ne suggeriscono le possibilità di connessione in modo da renderne intuitiva la composizione e impedire combinazioni prive di senso" (ibidem, p.14).

Questi programmi sono utilizzati per il coding, perché permettono ad esempio di far muovere un personaggio virtuale, ma anche per programmare i robot, cioè nella robotica educativa.

Nella robotica educativa, infatti, la programmazione dei comportamenti del robot è, insieme alla costruzione, la fase fondamentale dell'esperienza per quanto concerne l'apprendimento, perché permette l'acquisizione di abilità e competenze logiche e progettuali (per risolvere dei problemi si ipotizzano soluzioni e se ne valuta l'efficacia risolutiva), ma richiede l'utilizzo di un ambiente di programmazione utilizzabile anche da chi non è esperto nella scrittura del codice utilizzando linguaggi specifici e di tipo testuale (Garbati, 2011).

Conclusioni

In conclusione, appare chiaro come il pensiero computazionale venga oggi riconosciuto come una competenza fondamentale per avere successo nelle discipline STEM, ma anche negli altri ambiti disciplinari. Programmare è un potente strumento di pensiero, di espressione e di crescita personale perché, imparando come si programma, si impara ad imparare.

A scuola gli insegnanti sono invitati ad utilizzare il *coding*, che è il modo più diffuso per favorire l'acquisizione del pensiero computazionale.

Riferimenti bibliografici

Alfieri, P. (2016), *Coding e il pensiero computazionale*, Lecce: Ed. Youcanprint.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*; EUR 28295 EN; doi:10.2791/792158. Https://publications.europa.eu/en/publication-detail/-/publication/093eadcc-c820-11e6-a6db-01aa75ed71a1/language-en (ultima consultazione 04/06/2019).

Bogliolo (2016), *Coding in your classroom, now!*, Firenze: Ed. Giunti.

Bonaiuti, G., Calvani, A., Menichetti, L., Vivanet, G. (2017), *Le tecnologie educative. Criteri per una scelta basata su evidenze*, Roma: Ed. Carocci.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Proceedings of the 2012 Annual Meeting of the American Educational Research Association in Vancouver, BC, Canada. Https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf (ultima consultazione 04/06/2019).

Faccioli, A., (2019), *Coding a blocchi o testuale?*, Rivista Bricks - anno 9 - numero 1 Http://www.rivistabricks.it/wp-content/uploads/2019/03/2019_1_10_Faccioli.pdf (ultima consultazione 04/06/2019).

Garbati, M., (2011), *Robotica Educativa*, Napoli: The Boopen Editore.

Giordano, M., Moscetti, C. (2016), *Coding e pensiero computazionale nella scuola primaria*, Loreto: Ed. La Spiga.

Grover, S., Pea, R. (2017). *Computational Thinking: a competency whose time has come*. Https://www.researchgate.net/publication/322104135_Computational_Thinking_A_Competency_Whose_Time_Has_Come/download, (ultima consultazione 04/06/2019).

Lodi, M., (2016), *Prefazione* in "*Coding e pensiero computazionale nella scuola primaria*", Loreto: Ed. La Spiga.

Lodi, M., Marchignoli, R. (2016), *EAS e pensiero computazionale. Fare coding nella scuola primaria*. Brescia: Ed. La Scuola.

MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) (2012), *Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, Annali della Pubblica Istruzione*, Le Monnier, Firenze. MIUR (2012). Http://www.indicazioninazionali.it/documenti_Indicazioni_nazionali/indicazioni_nazionali_infanzia_primo_ciclo.pdf, (ultima consultazione 04/06/2019).

MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca), Comitato Scientifico Nazionale per le Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione, (2017), *Indicazioni nazionali e nuovi scenari*, https://www.orizzontescuola.it/wpcontent/uploads/2018/02/Indicazioni-nazionali-e-nuovi-scenari.pdf, (ultima consultazione 04/06/2019).

Olimpo, G. (2017). *Dal mestiere dell'informatico al pensiero computazionale*. Italian Journal of Educational Technology, 25(2), 15-26. doi:10.17471/2499-4324/918, https://ijet.itd.cnr.it/article/download/918/875/, (ultima consultazione 04/06/2019).

Resnick M. (2013), *Learn to Code, Code to Learn*, https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn, (ultima consultazione 04/06/2019).

Note

[1] http://www.istruzione.it/scuola_digitale/allegati/Materiali/pnsd-layout-30.10-WEB.pdf

[2] L'azione #15 e #17 usufruiscono dei fondi PON FSE "Per la Scuola" 2014-2020. Il Programma Operativo Nazionale (PON) del Ministero dell'Istruzione, dell'Università e della Ricerca, intitolato "Per la Scuola - competenze e ambienti per l'apprendimento", finanziato dai Fondi Strutturali Europei contiene le priorità strategiche del settore istruzione e ha una durata settennale, dal 2014 al 2020 (http://www.istruzione.it/pon/ilpon.html#sec_pro).

[3] La circolare ha come oggetto: "Il pensiero computazionale a scuola – al via il secondo anno dell'iniziativa "Programma il Futuro": insegnare in maniera semplice ed efficace le basi dell'informatica."