

BRICKS | TEMA

IL LABORATORIO VIRTUALE DI CODING, PER UNA DIDATTICA DEI LINGUAGGI DI PROGRAMMAZIONE EFFICACE ANCHE A DISTANZA

a cura di:
Giuliana Barberis



Innovazione, Ambiente Digitale di Apprendimento, Coding, Scuola Secondaria, Istruzione Universitaria, Informatica, E-learning, Aula Virtuale, DAD, DDI, Plugin

Introduzione

Per quanto riguarda la condivisione di materiali, la progettazione di lezioni che prevedano un percorso ben definito, l'erogazione di verifiche, l'automatizzazione delle consegne degli elaborati, Moodle è l'ambiente ideale, soprattutto se si vuole adottare un approccio costruttivista nell'insegnamento.

Le caratteristiche percettive e strutturali dell'apprendimento così come le descrivono le teorie costruttiviste, sono perfette per la didattica dell'Informatica, specie nel coding, dove, nelle attività di problem solving, è indispensabile la maturazione di un pensiero creativo computazionale.

Utilizzando una piattaforma Moodle, sono automaticamente disponibili, una serie di strumenti che possono essere utilizzati al servizio di una didattica che segua tali teorie, che sono state proprio per questo di possibile attuazione anche nel periodo della Didattica a Distanza. Infatti una delle difficoltà sorte con la DAD è stata sicuramente quella di riuscire a insegnare a programmare a studenti che, a casa propria, possono usufruire di dispositivi diversi tra loro: pc, tablet o anche solo smartphone.

Questi device sono muniti da molteplici sistemi operativi e quindi gli ambienti di sviluppo installabili risultano di fatto molto diversi tra loro, generando una classe multiforme, molto complicata da gestire, specie a distanza. In poche parole è diventata pressante l'esigenza di avere un ambiente di sviluppo, con editor e compilatore o interprete, integrato in Moodle, almeno per i linguaggi di programmazione più diffusi. Un plugin che soddisfa questa esigenza è senz'altro CodeRunner: un software creato da Richard Lobb (University of Canterbury, New Zealand) e Tim Hunt (The Open University, UK). La piattaforma Moodle si trasforma così in un laboratorio di Informatica a distanza, che è stato indispensabile in DAD, e si sta rivelando anche molto utile nella didattica consueta. Con questo plugin, l'Aula Virtuale, replica efficacemente l'ambiente di esercitazione del laboratorio di informatica, si possono assegnare esercizi di programmazione di difficoltà variabile e progressiva, che possono essere svolti da tutti gli studenti, con qualsiasi dispositivo personale, purché abbia un browser. L'insegnante può entrare sulla "postazione" di ciascuno e verificare lo svolgimento dell'esercizio, dando consigli e sostegno ad hoc.

Le stesse esercitazioni possono anche essere svolte in autonomia, sempre nell'ottica di una didattica di tipo costruttivista del fare per imparare. Lo studente è chiamato a realizzare praticamente l'esercizio di coding, potendo far verificare anche molte volte la propria soluzione dalla piattaforma stessa, in modo da ottenere un feedback che gli consentirà di essere guidato alla soluzione ottimale per tentativi successivi.

Questo comune ambiente di programmazione può essere efficacemente utilizzato anche per le verifiche, per le quali è essenziale che tutti gli studenti siano messi nelle medesime condizioni per una valutazione equa. Inoltre le verifiche pratiche svolte mediante l'utilizzo del plugin CodeRunner beneficiano di tutti i vantaggi offerti dallo strumento Quiz di Moodle, dando la possibilità di alterare l'ordine delle domande e di sceglierne casualmente ciascuna in un serbatoio di domande simili ma non uguali, il che impedisce copiatore e rende quindi consistente la valutazione anche a distanza.

Insegnare a pensare: un approccio costruttivista

L'insegnamento del coding non può prescindere dall'attivare negli studenti un particolare tipo di pensiero, il pensiero computazionale, che è un tipo di ragionamento indispensabile per analizzare e risolvere problemi, ed è un prerequisito essenziale per imparare a codificare la soluzione ad un esercizio in un linguaggio di programmazione o in pseudocodifica.

Dobbiamo quindi insegnare agli studenti a pensare come "pensa" un elaboratore elettronico, per raggiungere in modo efficace questo obiettivo, possiamo analizzare l'evoluzione delle tecniche didattiche tra il novecento e oggi.

Dall'inizio del novecento gli studi sui metodi di insegnamento transitano dalla filosofia alla pedagogia e alla psicologia, a dimostrazione del progresso raggiunto dal mondo della didattica; quindi per capire come si può insegnare al meglio, si inizia a studiare la mente e il suo sviluppo durante la fase di scolarizzazione.

Con John Dewey e la sua "scuola attiva" si mette il discente al centro di qualsiasi interesse, promuovendo lo sviluppo delle sue capacità critiche. Questa corrente di pensiero è molto importante per quanto riguarda la comprensione dei problemi legati all'insegnamento del pensiero creativo e del problem solving, e sarà capace di generare tutta una serie di studi sulle caratteristiche percettive e strutturali della mente.

Negli anni successivi, la pedagogia fa un altro sostanziale passo avanti, con l'apporto teorico dato dal costruttivismo, secondo il quale occorre creare ambienti per l'apprendimento in cui la conoscenza si acquisisca anche attraverso la costruzione delle cose e delle idee.

"Una didattica costruttivista deve essere caratterizzata dalla costruzione e non dalla riproduzione di conoscenza, una costruzione inevitabilmente caratterizzata dallo stile cognitivo e dal tipo di intelligenza prevalente del discente (cfr Gardner 1994). Una didattica che non deve semplificare ma rendere invece visibile la complessità della realtà e le sue multiprospettive rappresentazioni, sviluppando situazioni di apprendimento basate su casi reali." ¹

"Se ascolto dimentico, se vedo ricordo, se faccio capisco". (Confucio)

Per quanto riguarda la Matematica, e anche le lingue straniere, il metodo prevalente di insegnamento è quello di iniziare un argomento presentando i concetti teorici, magari dandone anche qualche esempio di applicazione. Solo in seguito è richiesto allo studente di applicare le tecniche apprese, ma solo per risolvere batterie di esercizi simili fra loro.

Anche l'insegnamento del Coding si presta a questa metodologia, nella fase di applicazione si ottiene un programma, che può essere assimilabile ad un artefatto cognitivo, tuttavia, se non si stimolano i ragazzi a programmare in modo autonomo, si tratta sempre di riproduzione di conoscenza, cosa che non rispetta la teoria della didattica costruttivista.

L'insegnamento del coding per competenze

"Coding significa letteralmente "l'azione di scrivere codice sorgente", che è un'attività della programmazione informatica. La programmazione, in informatica, è l'insieme delle attività e delle tecniche che una o più persone specializzate, programmatori o sviluppatori (developer), svolgono per creare un programma, ossia un software da far eseguire a un computer, scrivendo il relativo codice sorgente in un certo linguaggio di programmazione".²

Imparare a realizzare programmi è molto importante per la maturazione di un pensiero logico creativo e si possono coinvolgere in questa attività anche studenti molto giovani, addirittura della scuola elementare, Sulla rivista Brick del 21 gennaio del 2019, è stato riportata una parte di un un articolo di Jan Lepeltack ("In molti paesi europei il pensiero computazionale/coding diventa una materia scolastica obbligatoria"³) dal quale si evince che in Europa si sta facendo un certo sforzo per introdurre questo insegnamento nelle scuole primarie e secondarie e c'è un documento, elaborato a Bruxelles nel corso di un seminario del 2019, dal titolo "Informatics for all"⁴, che attesta questo impegno.

¹ Andrea Varani Didattica costruttivista e Tecnologie dell'Informazione e della Comunicazione: una sinergia potente, <http://www.icferraripontremoli.it/materiale/2marzo/Nuova%20cartella/3%20Costruttivismo.pdf>

² Rivista Bricks del 3 Marzo 2020, articolo "Integrare Coding e Pensiero Computazionale nella didattica", <http://www.rivistabricks.it/2020/03/03/integrare-coding-e-pensiero-computazionale-nella-didattica/>

³ Rivista Bricks del 21 gennaio 2019, articolo "In molti paesi europei il pensiero computazionale/coding diventa una materia scolastica obbligatoria", http://www.rivistabricks.it/wp-content/uploads/2019/03/2019_1_21_Lepeltak.pdf

⁴ ACM Europe Council, <https://europe.acm.org/i4all>

L'iniziativa a livello mondiale "The Hour of Code", promossa dalla piattaforma Code.org⁵, ha portato alla nascita in Italia del progetto "Programma il futuro"⁶, patrocinato anche dal Ministero dell'Istruzione e dal CINI (Consorzio Interuniversitario Nazionale per L'Informatica). Tutte queste azioni sono ideate e promosse per raggiungere l'obiettivo comune di introdurre il pensiero computazionale nella cultura collettiva.

Ai più piccoli si può insegnare Coding anche in modo unplugged, o con l'aiuto di linguaggi di programmazione a blocchi (come Scratch o Trinket di Python), ma per gli studenti delle scuole superiori è fondamentale approcciarsi ad un linguaggio testuale (C, C++, JavaScript o Python).

Nella Raccomandazione del Parlamento Europeo del Consiglio del 23 aprile del 2008, riguardo al Quadro Europeo delle Qualifiche, modello EQF⁷, si stabilisce che le competenze possano essere descritte come espressione di responsabilità e autonomia. La didattica di un linguaggio di programmazione è sicuramente più efficace se svolta per competenze. Occorre dare agli studenti sufficiente autonomia per cercare la propria soluzione ai problemi proposti. Per aumentare la responsabilità del discente si dovrebbe cercare di sottoporre quanto più possibile problemi di realtà, molto più interessanti e motivanti da studiare.

Un discorso a parte merita l'importanza dell'errore nel processo di apprendimento⁸: soprattutto nell'apprendimento della programmazione è importantissimo che gli studenti non siano spaventati dalla possibilità di sbagliare, l'errore infatti dà la possibilità di allenarsi a riparare. Uno studente che commette molti errori scrivendo il codice, impara ben presto le tecniche necessarie a correggersi, sarà portato a esplorare nuove possibilità e curioso di nuovi metodi, uno studente che è stato guidato in modo troppo preciso, nel suo processo di problem solving, tenderà a non commettere errori e quindi perderà la possibilità di imparare a correggere il codice..

L'errore è il modo di imparare per approssimazioni successive, in sostanza si impara ad imparare, è importante però avere uno strumento che impedisca al discente di sentirsi disorientato quando si trova in una situazione senza uscita, quando sente di non avere abbastanza strumenti per proseguire, è allora fondamentale avere la possibilità di fornirgli un feedback che lo guidi nella direzione giusta.

La necessità di un laboratorio virtuale

Il plugin CodeRunner

Insegnare a programmare può essere un impegno creativo e il risultato dà sicuramente molta soddisfazione, sia al docente che agli studenti. Moodle è uno strumento indispensabile nella didattica dell'Informatica e in particolare del Coding per tanti aspetti, dal più banale che permette di condividere materiali di tutti i tipi in un corso visibile a tutta la classe, al più utile, che si riferisce alla possibilità di somministrare verifiche a correzione automatica, che possono essere considerate davvero valide per una valutazione realistica del processo di apprendimento.

Per quanto riguarda l'insegnamento di un linguaggio di programmazione, per esempio Python, si possono però solo fare esercitazioni (e verifiche) di "interpretazione", come nell'esempio in Figura 1.

⁵ Piattaforma Code.org <https://code.org/>

⁶ Progetto Programma il futuro <https://programmailfuturo.it/>

⁷ Il Quadro europeo delle qualificazioni (EQF) dell'Unione Europea <https://europa.eu/europass/it/european-qualifications-framework-eqf>

⁸ Camerlengo A. Il pensiero Computazionale: logica e problem solving dallo studente al manager informatico, (2021). Dario Flaccovio Editore

dato il seguente frammento di codice:

```

a=1
b=3
cont=0
while a >= 1:
    if a<b:
        c=a
        a=b
        b=c
    a = a-b
    cont = cont + 1
print (cont)

```

quale sarà il numero in output?

Risposta:










Figura 1 - Esempio di domanda di interpretazione

Questi esercizi possono stimolare il pensiero computazionale, ma a posteriori, senza dare allo studente la sensazione di costruire una propria soluzione ad un problema, non un vero learning by doing quindi. Serve un'estensione di Moodle che permetta di avere un tipo di domanda nella quale lo studente possa rispondere scrivendo un programma in un linguaggio di programmazione, e la valutazione della risposta sia relativa all'effettiva esecuzione del codice scritto dallo studente e al confronto dell'output di tale codice con i risultati attesi, potendolo testare anche con input diversi, per garantire la generalità della soluzione trovata.

CodeRunner è un plugin di Moodle, free e open-source, creato da Richard Lobb (University of Canterbury, New Zealand) e Tim Hunt (The Open University, UK), con questa estensione l'attività quiz di Moodle si arricchisce di un nuovo tipo di domanda, denominato appunto CodeRunner, come si vede in Figura 2.

Scegli un tipo di domanda da aggiungere

DOMANDE

-  Risposta multipla
-  Vero/Falso
-  Corrispondenza
-  Risposta breve
-  Numerica
-  Componente
-  Calcolata
-  Calcolata semplice
-  CodeRunner

Scegli un tipo di domanda per vederne la descrizione.

Figura 2 - Il tipo di domanda CodeRunner

Scegliendo il tipo di domanda CodeRunner si può mettere in un quiz un quesito che chieda di scrivere del codice in uno fra molti linguaggi di programmazione disponibili: C, C++, Java, Nodejs, Octave, Pascal, PHP, Python, Sql, come nell'esempio in Figura 3

Categoria in uso: praticaPythonFinoCicliVerifica (18) Usa questa categoria

Salva nella categoria: praticaPythonFinoCicliVerifica (18)

Nome della domanda: **!** CicloEnumerativoVerifica

Testo della domanda: **!** Paragrafo **B** **I** **☰** **☰** **🔗** **🔗** **🔗** **🖼️** **😊** **📺** **📄**
scrivi un programma che scriva in output la tabellina del 4 da 16 a 160 compresi

Figura 3 - Esempio di domanda di tipo CodeRunner

Ogni domanda dovrà essere corredata dal modello di risposta, costituito dal codice che produce il risultato atteso, come in Figura 4.

▼ Answer

Answer **?**

```
1 - for i in range(16, 161, 4):
2 -     print(i)
```

Figura 4 - Esempio di modello di risposta

Dovranno anche essere esplicitati i casi di prova, che possono essere anche numerosi, e consistono nei dati di input (anche file), se sono necessari, o, come in Figura 5, solo dall'output.

▼ Test cases

Test case 1 **?**

Standard Input **?**

Expected output **?** 16
20
24
...

Extra template data **?**

Test properties: **?** Use as example Display Show Hide rest if fail Mark 1.000 Ordering 10

Figura 5 - Esempio di caso di prova

Se vengono dati più casi di prova allora la valutazione verrà fatta sulla base della percentuale dei casi superati dal codice scritto dallo studente.

Si può trovare molta documentazione sulla pagina Web del plugin [9], dove si precisa che questa estensione è pensata principalmente per l'uso nei corsi di Coding, sebbene possa essere impiegata per valutare qualsiasi domanda per la quale la risposta sia testuale.

Quando lo studente trova una domanda di questo tipo nel suo quiz, ha a disposizione una finestra per digitare il codice (Figura 6) nella quale deve scrivere la sua soluzione, questa casella beneficia di un vero e proprio editor specifico del linguaggio di programmazione, si è quindi facilitati dalla syntax highlighting.

scrivi un programma che scriva in output la tabellina del 4 da 16 a 160 compresi

Answer: (penalty regime: 5, 10, 15, 20, ... %)

```

1 | for i in range(16, 160, 4):
2 |     print(i)

```

Verifica risposta

Figura 6 - Casella di editing per lo studente

Cliccando sul tasto "verifica risposta" viene mandato in esecuzione il codice scritto dallo studente e il suo output viene confrontato con i risultati attesi, lo studente riceve quindi un feedback (Figura 7) che gli consente di confrontare quello che ha ottenuto (Got) con il risultato atteso (Expected). Se non appare immediatamente evidente la differenza tra il risultato atteso e quello che lui ha ottenuto con il suo programma, può cliccare sul tasto "Show differences".

	Expected	Got	
×	16	16	×
	20	20	
	24	24	
	28	28	
.			
	152	152	
	156	156	
	160		

Show differences

Figura 7 - Feedback

Se invece il codice contiene errori di sintassi e per questo non può essere eseguito, si riceve in risposta un adeguato messaggio di errore, con l'indicazione del numero della riga alla quale si è fermata la compilazione o l'interpretazione del codice (Figura 8).

	Input	Expected	Got	
✘	pere// banane// cipolle// sed ano	['pere', 'banane', 'cipolle', 'sedano']	***Run error*** Traceback (most recent call last): File "__tester__.python3", line 2, in <module> aggettivi=input("") EOFError: EOF when reading a line	✘

Testing was aborted due to error.

Show differences

Figura 8 - Errore di sintassi

In questo modo lo studente ottiene il feedback che gli consente di correggere l'errore e rieseguire il programma, proprio come farebbe utilizzando un editor specifico.

Se la risposta dà come output i risultati attesi la verifica risponde con una finestra con sfondo verde, ed è evidente che la risposta è esatta (Figura 9).

	Expected	Got	
✓	16	16	✓
	20	20	
	24	24	
	28	28	

Figura 9 - Risposta esatta

Si possono leggere direttamente in un articolo di Richard Lobb e Jenny Harlow dell'Università di Canterbury⁹ queste potenzialità di CodeRunner.

Nella pagina web dell'estensione¹⁰ si vede che ad oggi oltre 2000 piattaforme Moodle hanno installato il plugin CodeRunner, questa pagina è a sua volta un corso Moodle, al quale ci si può iscrivere per consultare i manuali per l'uso e l'installazione.

Installare CodeRunner sulla piattaforma Moodle

La documentazione tecnica dell'estensione è disponibile su GitHub¹¹, una piattaforma che offre un servizio di hosting per aziende e privati, per scaricare i materiali da questa piattaforma occorre essere registrati.

Consultando la documentazione presente su GitHub si evince che CodeRunner richiede in realtà l'installazione di due plugin, uno per poter utilizzare questo particolare tipo di domanda e l'altro per configurare il comportamento adattivo che deve essere attivato per l'utilizzo dell'editor specifico.

Si possono effettuare le installazioni in due modi, per entrambi occorre avere accesso alla cartella del proprio spazio Web, dove è stata installata la piattaforma Moodle:

- si possono scaricare due file zip da GitHub estraendoli in apposite sottocartelle della cartella Moodle
- si possono utilizzare due comandi specifici di Git, da copiare e incollare per essere eseguiti sempre nello spazio Moodle

Gli editor dei linguaggi e i motori per eseguire i comandi utilizzano un server a parte, diverso da quello della propria piattaforma Moodle, denominato Jobe Server. Nella configurazione iniziale del plugin viene collegato il Jobe Server dell'Università di Canterbury, ma la documentazione chiarisce che si è autorizzati a farne uso solo per i test iniziali del sistema, essendo posto il limite di utilizzo dell'API-Key solo per 100 utenti per ciascuna ora (nel mondo).

⁹ Richard Lobb Jenny Harlow, Coderunner: A Tool for Assessing Computer Programming Skills
https://coderunner.org.nz/pluginfile.php/1746/mod_resource/content/2/CodeRunnerArticlePublishedForm.pdf

¹⁰ Piattaforma Moodle su CodeRunner <https://coderunner.org.nz/>

¹¹ GitHub con i manuali e i file per l'installazione del plugin https://github.com/trampgeek/moodle-qtype_coderunner#code-runner

Per poter utilizzare questa estensione con i propri studenti, quindi occorre configurare un proprio Jobe Sandbox. Le istruzioni per implementare il proprio servizio, sono fornite in uno spazio GitHub apposito¹². Una volta configurato il proprio Jobe Server, occorre entrare nello spazio dell'amministratore della propria piattaforma Moodle e cambiare il nome del server, il numero della porta logica ed eventualmente l'API-Key relativi.

Dallo spazio GitHub è possibile accedere a una video-guida per facilitare il processo di configurazione di un Jobe Server in un servizio hosting di DigitalOcean (ma si può utilizzare un qualsiasi web hosting che garantisca le autorizzazioni opportune, per esempio AWS Amazon).

L'approccio didattico laboratoriale

Una volta che si sia installata l'estensione, si può procedere nella realizzazione di quiz con domande pratiche, tratte da problemi di realtà, che possono essere somministrati alla propria classe.



<input type="checkbox"/>		Nome / Cognome	Stato	Iniziato
<input type="checkbox"/>		Leonardo [redacted] Rivedi tentativo	Completato	19 ottobre 2021 11:25
<input type="checkbox"/>		Gabriele [redacted] Rivedi tentativo	Completato	19 ottobre 2021 11:28
<input type="checkbox"/>		Edoardo [redacted] Rivedi tentativo	Completato	19 ottobre 2021 11:29
<input type="checkbox"/>		Manuel [redacted] Rivedi tentativo	Completato	19 ottobre 2021 11:29

Figura 10 - Report dei tentativi del quiz

Intanto che gli studenti si esercitano, l'insegnante può consultare, in tempo reale, dalla propria postazione il report dei tentativi del quiz prodotto da Moodle, come in figura 10.

Cliccando su "Rivedi tentativo", l'insegnante può vedere a che punto è uno specifico studente nella soluzione dell'esercizio (Figure 11 e 12) ed eventualmente dare dei suggerimenti se ne riscontra la necessità.

¹² GitHub con i file per l'installazione del JobeServer <https://github.com/trampgeek/jobee>

Scrivi un programma che riceva in input (senza prompt) un testo in un'unica variabile composto da un elenco di informazioni separate da un doppio simbolo del dollaro (i due dollari separano gli elementi della lista, dopo i due simboli del dollaro può esserci o no uno spazio) e restituisca in output una lista nella quale in ogni elemento ci sia un'informazione senza le eventuali virgole che possono essere presenti (lasciare al loro posto gli spazi).

visualizzare la lista nella shell senza formattazioni

ad esempio il tuo testo di input potrebbe essere il seguente:

```
Gatto$$Gatto$$CANE $$coco,,rita
```

Answer: (penalty regime: 5, 10, 15, 20, ... %)

```
1 testo=input("")
2 lista=[]
3 for carattere in testo:
4     testo.replace("$$", " ")
5     lista=lista.append(testo)
6     print(lista)
7
```

Figura 11 - Risposta dello studente

	Input	Expected	Got	
✘	pere\$\$ banane\$\$ cipolle\$\$ sed,,ano	['pere', ' banane', ' cipolle', ' sedano']	None ***Run error*** Traceback (most recent call last): File "__tester__.python3", line 5, in <module> lista=lista.append(testo) AttributeError: 'NoneType' object has no attribute 'append'	✘

Testing was aborted due to error.

Show differences

Figura 12 - Errore rilevato dall'interprete/compiler

In questo modo l'ambiente di sviluppo è indipendente sia dal dispositivo utilizzato dallo studente (pc, tablet o smartphone) e quindi anche dal sistema operativo: è sufficiente avere una rete e un browser. Inoltre si lascia a ciascun discente la possibilità di ricevere i feedback automatici del sistema o si può intervenire per dare un feedback personalizzato. Se si è in laboratorio, o anche in una sessione di DAD, si può presentare lo schermo del docente a tutti gli studenti per mostrare la soluzione trovata da uno di essi o una particolare tecnica per correggere un errore, in modo che diventi patrimonio collettivo.

Alla conclusione del quiz, cliccando su Termina il tentativo, lo studente vede la revisione del proprio invio, con la valutazione per ciascun esercizio, può inoltre vedere il codice proposto dal docente come modello di soluzione.

Le domande mostrate nelle immagini delle pagine precedenti, sono relative al linguaggio Python, l'estensione però permette di usare anche altri linguaggi: C, C++, Java, Nodejs, Octave, Pascal, PHP, Python, Sql. Per linguaggi "tradizionali" come Python, Pascal, C e C++ si può richiedere agli studenti di scrivere come risposta programmi o anche funzioni.

Una possibilità veramente molto interessante è che si possano fare domande da svolgere in linguaggio SQL, come nella Figura 13.

Questa possibilità è interessante perché il linguaggio SQL, per poter essere processato, necessita di un server con "motore SQL" attivo (MySQL, MariaDB), in questo caso il JobeServer è automaticamente dotato anche di questa proprietà, rendendo disponibile un ambiente di prova altrimenti scomodo da configurare.

Nome della domanda ! dbAbbonamentiLeftJoin

Testo della domanda !

Paragrafo **B** *I* [List] [List] [Link] [Image] [Smiley] [Video] [File]

Visualizzare il titolo delle riviste per le quali non ci siano abbonamenti

```

    graph TD
      rivista[abbonamenti_rivista] -- codRivista --> abbonamento[abbonamenti_abbonamento]
      abbonamento -- codAbbonato --> abbonato[abbonamenti_abbonato]
      abbonamento -- codRivista --> abbonato
      abbonamento -- codRivista --> citta[abbonamenti_citta]
      abbonamento -- codRivista --> distributore[abbonamenti_distributore]
      abbonato -- codCitta --> citta
      abbonato -- codCitta --> distributore
      citta -- codCitta --> distributore
  
```

Figura 13 - Esempio di domanda con linguaggio SQL

In questo caso (figura 13) occorre fornire agli studenti l'immagine di un diagramma E/R di un Data Base, del quale l'insegnante deve avere il file di comandi per la sua generazione e la popolazione con dati di prova. Su questo DB, si possono fare domande per svolgere comandi sia in DML che in QL, anche complessi. Come per i linguaggi tradizionali, con la domanda l'insegnante scrive anche la soluzione di riferimento, l'output di questa soluzione, che è assimilabile ad una tabella, verrà confrontato con l'output del comando scritto dallo studente.

In questo caso oltre al modello di soluzione, l'insegnante dovrà fornire anche nel "Support files" il file di comandi per la generazione e popolazione del DB su cui svolgere l'esercizio, tali comandi dovranno essere scritti in sqlite e trasformati in un file con estensione .db (Figura 14)

Support files

Support files ?

File

abbonamenti...

Figura 14 - File con estensione db

L'autoformazione – esercitazioni formative con CodeRunner

I quiz che si possono creare con domande di tipo CodeRunner possono essere somministrati come esercitazione con la supervisione di un insegnante (in presenza o a distanza), o come esercitazioni formative da svolgere autonomamente, o come verifiche.

Nel caso lo studente sia autonomo nello svolgere l'esercitazione, vale la pena che l'insegnante progetti anche dei feedback specifici per alcuni casi di prova che possono essere più difficili da capire, specialmente all'inizio del percorso di studio del linguaggio.

In una domanda di questo tipo (Figura 15), fatta in una delle prime esercitazioni somministrate agli studenti, occorrerà predisporre ben 7 casi di prova. Si dovranno inserire i due casi di "voto non valido" e i 5 casi per testare le 5 fasce di valutazione.

scrivi un programma che richieda in input un voto (con la virgola, prompt "inserisci il voto: ") ed espliciti il giudizio secondo la seguente tabella:

minore di 2 o maggiore di 10: voto non valido
 tra 2 (compreso) e 4.5 gravemente insufficiente
 tra 4.5 (compreso) e 6 insufficiente
 tra 6 (compreso) e 8 buono
 tra 8 (compreso) e 9 ottimo
 tra 9 (compreso) e 10 compreso eccellente

Figura 15 - Esempio di domanda

Per rendere evidenti allo studente quali debbano essere le prove che vanno "passate" perché l'esercizio sia considerato corretto, tutti i casi di prova andranno messi in "show" e magari qualcuno si può configurare come "use as example", in modo che lo studente possa conoscere questi casi di prova già quando legge la domanda, come in Figura 16.

The screenshot shows a configuration panel for a test case. It includes the following elements:

- Test case 1:** A large empty text area for the problem description.
- Standard Input:** A text box containing the value "1".
- Expected output:** A text box containing the prompt "inserisci il voto: voto non valido".
- Extra template data:** A large empty text area for additional data.
- Test properties:**
 - Use as example
 - Display: Show (dropdown menu)
 - Hide rest if fail
 - Mark: 1.000
 - Ordering: 10

Figura 16 - Esempio di caso di prova

Verifiche pratiche a correzione semiautomatica

Nel caso in cui il test debba essere considerato come verifica vera e propria, alcuni casi di prova, invece, potranno essere configurati come "hide" in modo da verificare se lo studente sia in grado di trovare autonomamente la soluzione che soddisfa la generalità dei casi.

Si può anche dare una penalizzazione ogni volta che viene richiesto un "verifica risposta" (Figura 17).

The screenshot shows the 'Marking' configuration section. It includes:

- Marking:** A section header with a help icon.
- All-or-nothing grading
- Penalty regime: 5, 10, 15, 20, ... (dropdown menu)

Figura 17 - Penalizzazione standard

La raccomandazione è quella di tenere bassa, ma non nulla, la penalizzazione come in figura 18, per favorire l'apprendimento dai propri errori, e parallelamente distinguere gli studenti che raggiungono il risultato corretto sbagliando di meno.

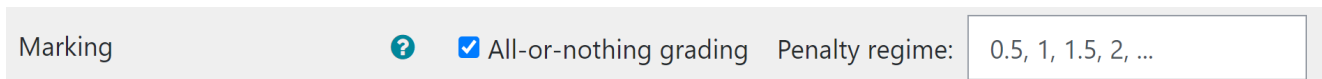


Figura 18 - Penalizzazione consigliata

Per evitare episodi di copiature nelle verifiche, anche utilizzando domande di tipo CodeRunner si può gestire nel test la scelta casuale di una domanda tra un insieme di domande che facciano parte di una stessa categoria o che abbiano uno stesso tag come in Figura 19.



Figura 19 - Esempio di quiz con domande casuali

Ad ogni domanda si può dare un peso diverso, esattamente come in un quiz con i tipi di domanda standard di Moodle, come nella parte a destra della Figura 19.

In Figura 20 troviamo un elenco di domande omogenee rispetto a certi criteri, alle quali si è assegnato lo stesso tag, il tag si attribuisce in fase di modifica della domanda (Figura 21).



Figura 20 - Domande aventi uno stesso tag

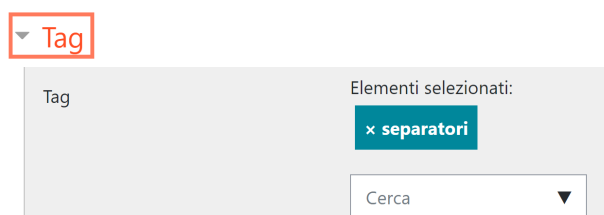


Figura 21 - Configurazione di un tag

Questo tipo di verifiche è estremamente comodo per mettere gli studenti nelle stesse identiche condizioni e anche per generare una verifica automatica molto sicura dal punto di vista delle copiature.

In un quiz di 4 domande, come in figura 19, dove ciascuna sia presa a caso in gruppo di 4 (stesso tag) vengono generate potenzialmente 44 (256) verifiche diverse.

Utilizzare una verifica di questo tipo è anche vantaggioso per quanto riguarda la correzione, anche se non si può dire che sia del tutto a correzione automatica.

Occorre sempre riguardare almeno le risposte che hanno ottenuto un punteggio nullo, perché in alcuni casi, come si vede in Figura 22, alla soluzione proposta manca proprio poco per essere considerata giusta, allora vale la pena di assegnare un punteggio parziale.

Domanda 2
Risposta errata
Punteggio ottenuto 0,00 su 1,50
Contrassegna domanda

Scrivi un programma che riceva in input (senza prompt) un testo in un'unica variabile composto da un elenco di informazioni separate da "/" (i due slash separano gli elementi della lista, dopo i due slash può esserci o no uno spazio) e restituisca in output una lista nella quale in ogni elemento ci sia un'informazione senza gli spazi.
visualizzare la lista nella shell senza formattazioni
ad esempio il tuo testo di input potrebbe essere il seguente:
Gatto//Gatto//CANE //coco rita

Answer: (penalty regime: 5, 10, 15, 20, ... %)

```
1 testo = input()
2 lista = []
3 line=testo.split("/")
4 print(line)
```

	Input	Expected	Got	
✘	pere// banane// cipolle// sed ano	['pere', 'banane', 'cipolle', 'sedano']	['pere', 'banane', 'cipolle', 'sedano']	✘
✘	Mario//Carla// Elena//Gianni	['Mario', 'Carla', 'Elena', 'Gianni']	['Mario', 'Carla', 'Elena', 'Gianni']	✘

Hide differences

Figura 22 - Esempio di risposta solo parzialmente errata

Conclusioni

Come conclusione si potrebbe dire che l'utilizzo della piattaforma Moodle, potenziata con questa estensione, in laboratorio o a distanza, contribuisce a promuovere negli studenti la ricerca autonoma e aiuta a sviluppare le capacità di osservazione e il pensiero critico. Quindi permette di realizzare nel proprio laboratorio di informatica una palestra per il pensiero computazionale, coerente con l'applicazione delle teoria dell'apprendimento del costruzionismo. Parallelamente stimola la realizzazione di percorsi concreti, che possono offrire agli studenti una rappresentazione visiva dei processi sottesi ai loro ragionamenti.



Giuliana Barberis

giuliana.barberis@gmail.com

Liceo Scientifico M. Curie di Pinerolo

Nata a Torino il 26 novembre 1965, diploma di Maturità Scientifica e laurea del vecchio ordinamento in Matematica ad indirizzo generale (Unito).

Dal 1991 al 2001 impiegata nella direzione sistemi informativi e poi nella direzione commerciale Italia dell'azienda GFT SPA come programmatrice in cobol, analista tecnico, analista funzionale, capoprogetto in un progetto di "Production Planning", collaboratrice dell'ufficio informatico per la direzione Commerciale Italia.

Dal 2002 docente di Informatica in Istituti Statali di Secondo Grado, fino al 2011 ITC, dal 2011 presso il Liceo Scientifico M. Curie di Pinerolo opzione Scienze Applicate.