# TEST AUTOMATICI SU AUTOMI A STATI FINITI E MACCHINE DI TURING CON CODERUNNER

**Adolfo Casagrande, Luciano Dereani, Eva Pantanali**

ISIS A. Malignani (UD)
*{adolfo.casagrande, luciano.dereani, eva.pantanali}@malignani.ud.it*

***COMUNICAZIONE***

***ARGOMENTO**: Valutazione automatica*

## Abstract

La valutazione automatica nel contesto educativo rappresenta una sfida continua, soprattutto per materie complesse come la teoria degli automi e delle macchine di Turing. In questo intervento, esploreremo l'utilizzo di CodeRunner, un potente strumento di Moodle, per creare test che si autocorreggono sulla realizzazione di automi a stati finiti e macchine di Turing. Abbiamo sviluppato due nuove tipologie di domande specifiche: una per gli automi a stati finiti e una per le macchine di Turing. Queste domande utilizzano per la validazione delle risposte dei template personalizzati scritti in python con l'ausilio della libreria Python AST (Abstract Syntax Tree) e per il disegno dei grafi, l'interfaccia grafica GraphUI già presente in CodeRunner.

L'integrazione di queste tecnologie consente non solo di automatizzare la correzione dei test, ma anche di offrire agli studenti un feedback immediato e dettagliato sulle loro soluzioni. Presenteremo i dettagli tecnici dell'implementazione, mostrando come i template sono stati progettati per verificare automaticamente la correttezza degli automi e delle macchine di Turing disegnate dagli studenti.

Inoltre, la stessa tecnica di personalizzazione delle domande è stata applicata per creare test di programmazione in Python che richiedono l'uso di particolari costrutti sintattici. Ad esempio, possiamo imporre l'utilizzo di un ciclo for o while, o richiedere una soluzione di tipo ricorsivo.

L'intervento si propone di dimostrare come l'adozione di strumenti avanzati come CodeRunner, opportunamente personalizzato, possa migliorare l'insegnamento e la valutazione delle discipline informatiche, rendendo il processo di apprendimento più interattivo, efficiente e personalizzato.

**Keywords**: Valutazione automatica, CodeRunner, didattica della programmazione in Python, automi a stati finiti, Macchine di Turing

## 1 INTRODUZIONE

Questo articolo si basa sull'esperienza maturata nell'insegnamento di Informatica nelle classi quinte del Liceo opzione Scienze Applicate e nasce dall'esigenza di creare dei test automatici per argomenti in cui tradizionalmente, per la natura stessa dei problemi, era necessaria una correzione manuale.

I test che sono stati oggetto di "automatizzazione" sono esercizi per la realizzazione di automi a stati finiti e macchine di Turing, dove le risposte degli studenti sono dei grafi, e per esercizi di programmazione in Python nei quali si vieta l'utilizzo di particolari costrutti sintattici.

# 2 REQUISITI

## 2.1 Plug-in CodeRunner

Requisito essenziale è l'installazione del plug-in CodeRunner [1]. CodeRunner è un plugin open-source per Moodle progettato per facilitare la creazione di test di programmazione automatizzati, offrendo agli insegnanti la possibilità di formulare domande in cui gli studenti devono scrivere e sottoporre codice che viene eseguito e valutato automaticamente.

CodeRunner esegue il codice degli studenti in un ambiente isolato (sandbox), riducendo il rischio che codice malevolo possa danneggiare il sistema o accedere a informazioni sensibili

Si ricorda che CodeRunner utilizza un server chiamato Jobe per eseguire il codice in un ambiente isolato. Questo server può essere installato localmente o configurato su un server remoto. Si rimanda alle istruzioni di installazione del plug-in per ulteriori dettagli https://coderunner.org.nz/.

## 2.2 Competenze di programmazione

Nell'eventualità di voler personalizzare i nuovi template proposti è necessario conoscere il linguaggio Python e l'ambiente di sviluppo di CodeRunner per il passaggio dei parametri. I diversi chatbot di I.A. possono comunque essere d'aiuto nella generazione del codice sorgente.

# 3 PROBLEMA

La tipologia di esercizi che si intende trasformare in test a correzione automatica prevede che lo studente realizzi un grafo come risposta ad un quesito; il grafo deve essere analizzato e in funzione dei casi di test predisposti soddisfare i requisiti richiesti dalla domanda.

Ad esempio, se viene richiesto di realizzare un automa a stati finiti che riconosca due cifre consecutive a partire da una sequenza di cifre binarie i test automatici devono prevedere i diversi casi di test per validare in automatico la risposta simulando il comportamento dell'automa e confrontando l'output ottenuto con l'input fornito.

In CodeRunner, la casella di testo per la risposta supporta componenti UI JavaScript plug-in, tra cui GraphUI e, se si sceglie come tipologia di domanda *directed_graph,* viene attivata un'interfaccia utente per l'inserimento di un grafo.

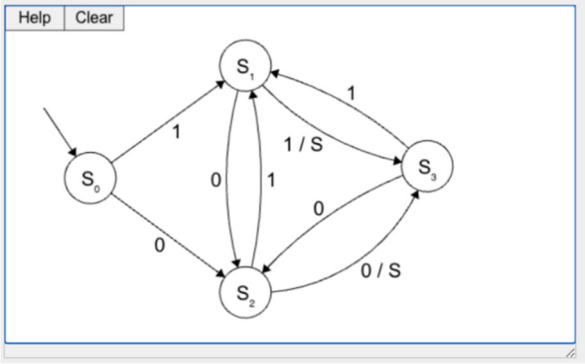


**Figura 22 - GraphUI per il disegno di un grafo**

Il codice testuale corrispondente alla rappresentazione grafica è visibile utilizzando i tasti Ctrl-Alt-m (Control-Option-m per MacOS) e viene di seguito riportato a solo scopo informativo:

{"edgeGeometry":[{"lineAngleAdjust":0,"parallelPart":0.5,"perpendicularPart":0},{"lineAngleAdjust":0,"parallelPart":0.5697567976276796,"perpendicularPart":20.80637995795019},{"lineAngleAdjust":0,"parallelPart":0.5,"perpendicularPart":0},{"lineAngleAdjust":0,"parallelPart":0.4375,"perpendicularPart":19},{"lineAngleAdjust":0,"parallelPart":0.5,"perpendicularPart":16},{"deltaX":-47,"deltaY":-70.96875},{"lineAngleAdjust":0,"parallelPart":0.5064996662671545,"perpendic

```
ularPart":38.928616866725605},{"lineAngleAdjust":0,"parallelPart":0.5,"perp
endicularPart":16},{"lineAngleAdjust":0,"parallelPart":0.5,"perpendicularPa
rt":16}],"nodeGeometry":[[86.25,174.7734375],[243.25,60.7734375],[427.25,16
2.7734375],[243.25,290.7734375]],"nodes":[["S_0",false],["S_1",false],["S_3
",false],["S_2",false]],"edges":[[0,1,"1"],[1,2,"1                          /
S"],[0,3,"0"],[1,3,"0"],[3,1,"1"],[-1,0,""],[3,2,"0                         /
S"],[2,1,"1"],[2,3,"0"]]}
```


L'help invece fornisce indicazione sulle funzioni disponibile nell'editor:

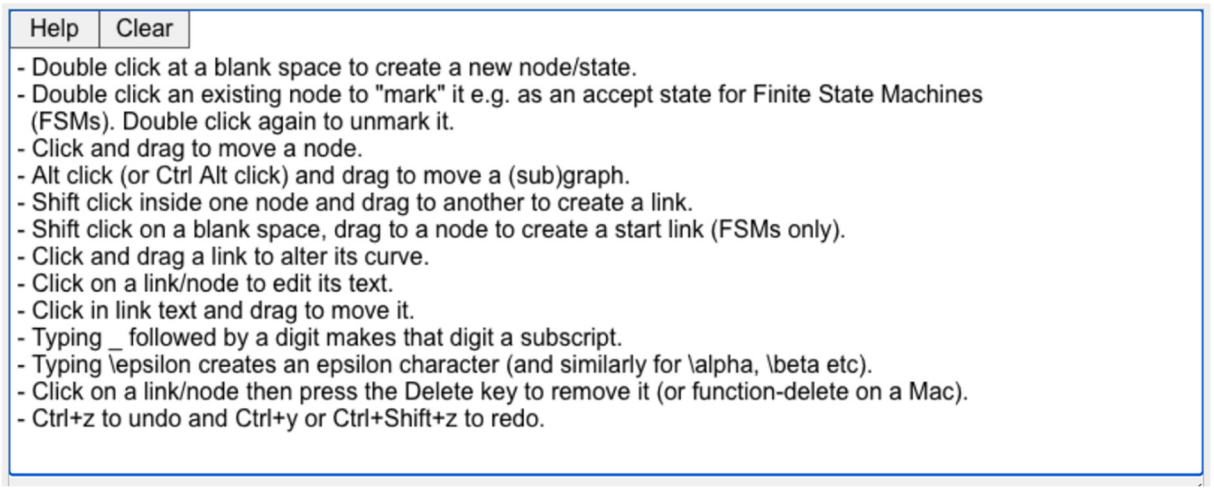


**Figura 23 - Comandi editor**

La valutazione della correttezza della risposta deve essere personalizzata scrivendo il codice Python che controlla la struttura del grafo nel campo Customitation → Template e che modifica quindi il comportamento standard della domanda *directed_graph*.

Al fine di non replicare lo stesso codice per tutti gli esercizi di uno stesso tipo è possibile definire un prototipo di domanda, a cui viene assegnato un nome, che contenga il codice di controllo personalizzato.

In questo modo la scrittura di un test si riduce a scegliere il tipo di domanda associato al prototipo corretto senza dover intervenire scrivendo ogni volta il programma personalizzato di test.

## 3.1 Creazione Prototipo di domanda

Indicazioni operativa per creare un prototipo di domanda personalizzato.

- Creare una nuova domanda appartenente alla categoria CR_CUSTOM_PROTOTYPES

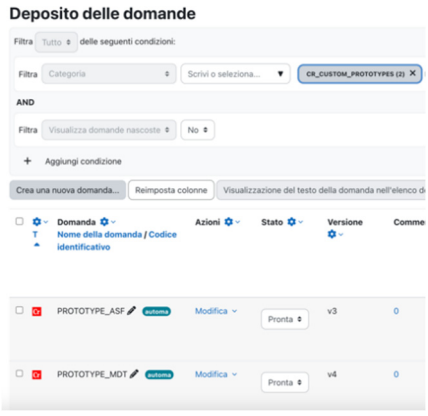


**Figura 24 - Nuova domanda**

- Attivare il checkbox *Customisation* della sezione *CodeRunner question type*

- Personalizzare il codice di controllo impostando il campo *Template* nella sezione *Customisation*:

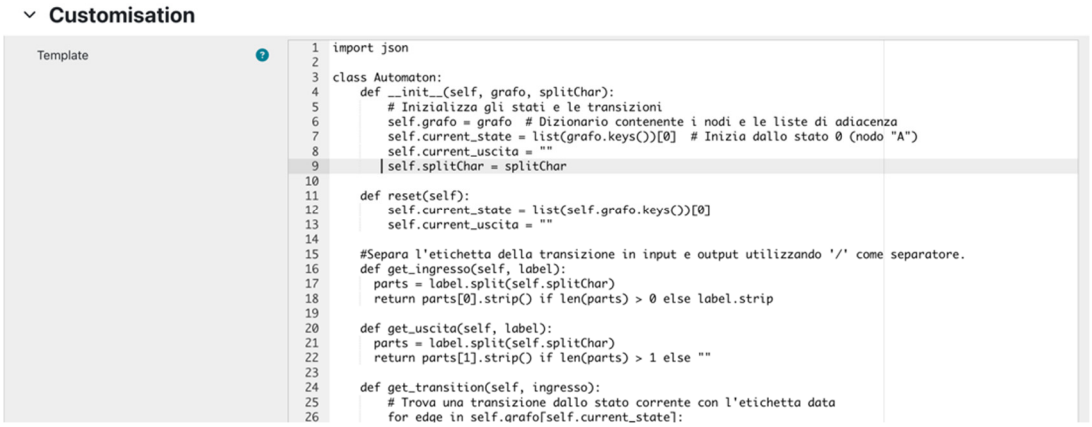


**Figura 25 - Programma Python di validazione**

- Impostare i campi nella sezione Advanced Customisation:
  - *Is prototype?* a **Yes** (user defined)
  - *Question type* al **nome** che si vuole assegnare a questo tipo di domanda

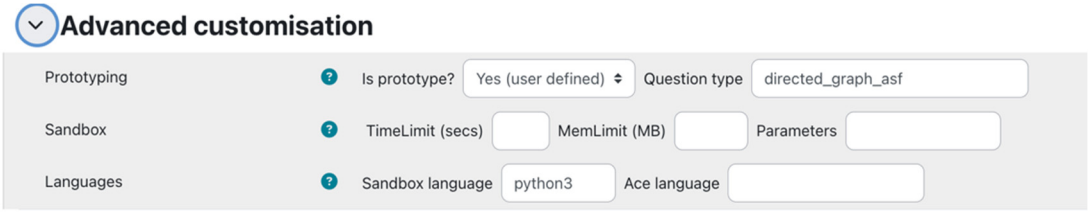


**Figura 26 - Opzioni prototipo**

- Impostare nella sezione *Generale* il *Nome della domanda* (si consiglia di utilizzare come prefisso PROTOTYPE in modo da distinguerle dalle domande normali) e il *Testo della domanda* (che costituirà una nota descrittiva visibile nella sezione *Question type details* in fase di inserimento della domanda, utile per dare delle indicazioni sulla tipologia di domanda, eventuali parametri di configurazione e sul formato della risposta).

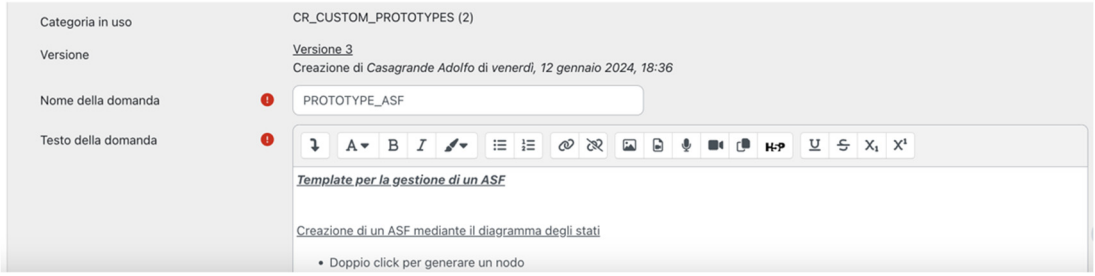


**Figura 27 - Nome e descrizione prototipo**

Salvata la *domanda prototipo* sarà ora possibile inserire domande appartenenti a questa tipologia in modo semplice come per tutte le altre domande di Moodle scegliendola dal campo *Question type*.

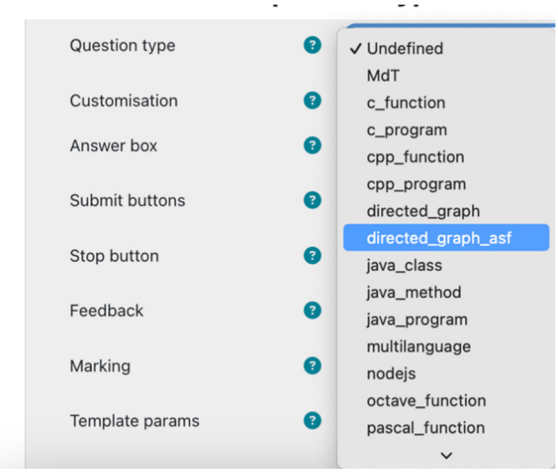


**Figura 28 - Elenco tipo domande disponibili**

## 3.2 Prototipo Automa a Stati Finiti

Questo tipo di domanda (disponibile se si importa PROTOTYPES_ASF [2]) consente di realizzare esercizi sugli Automi a Stati Finiti e di verificarne la correttezza. Il prototipo non fa parte dei prototipi forniti di default da coderunner ma è stato costruito a partire dal prototipo *directed_graph*.

Il prototipo *direct_graph* permette di realizzare automi mediante la classica rappresentazione grafica del diagramma degli stati (vedi Figura 1), fornendo una codifica testuale che può essere successivamente elaborata.

Il nuovo prototipo creato (*directed_graph_asf*) ha lo scopo di analizzare la codifica testuale dell'automa, fornita dal prototipo *directed_graph*, e di verificarne la correttezza in base all'input dato e all'output previsto nei test.


**Figura 29 - Tipo domanda Automi a Stati Finiti**

Il prototipo si aspetta che nel grafo che descrive l'automa le etichette degli archi indichino l'input atteso e il relativo output prodotto, separati da un carattere separatore (di default la barra /).

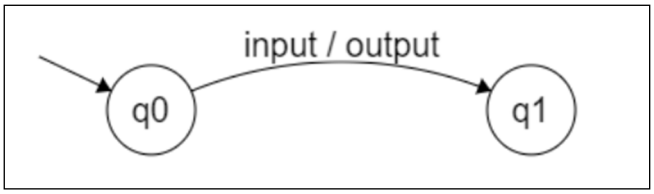


**Figura 30: Formato utilizzato per descrivere un ASF**

Di seguito un esempio.

*Biglietti da distributore automatico*

*Si vuole progettare un automa a stati finiti che controlli il funzionamento di un distributore automatico di biglietti da 2€; il distributore accetta soltanto monete da 50 centesimi di euro, 1€ o 2€, inserite una alla volta, senza restituire alcun resto ma utilizzandolo per l'emissione del biglietto successivo.*

Le etichette degli archi prevedono di specificare i valori di ingresso e di uscita separati da una barra (es. "0.5 / -" significa che, se viene inserita una moneta da 50 centesimi, non viene emesso il biglietto, mentre "2 / B" significa che inserendo una moneta da 2€ viene emesso il biglietto).

L'automa che risolve il problema può essere descritto dal seguente diagramma:

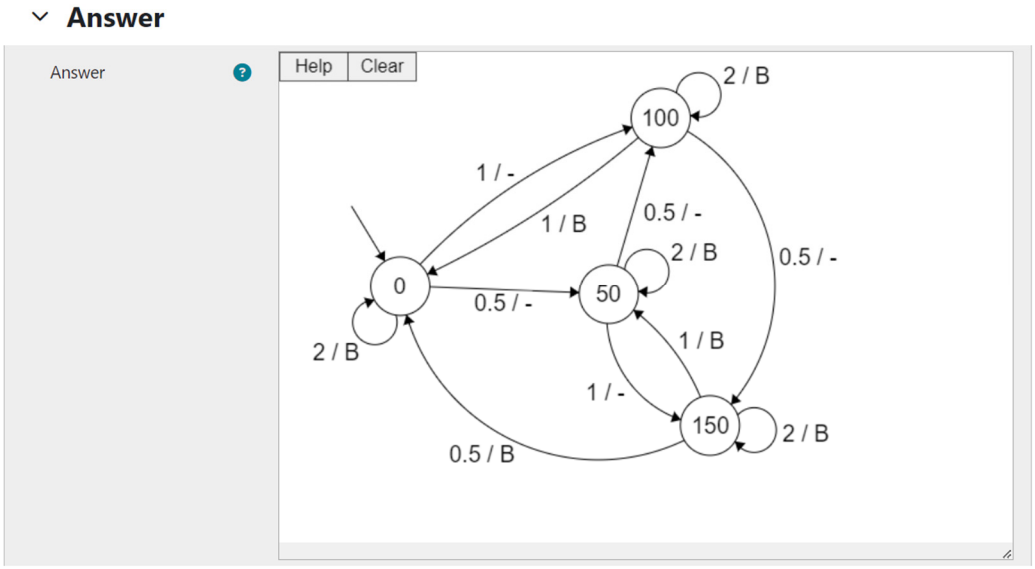


**Figura 31: ASF risolutivo**

Nei test forniti per verificare la correttezza della soluzione implementata dallo studente è necessario specificare la sequenza degli ingressi e la corrispondente sequenza delle uscite.

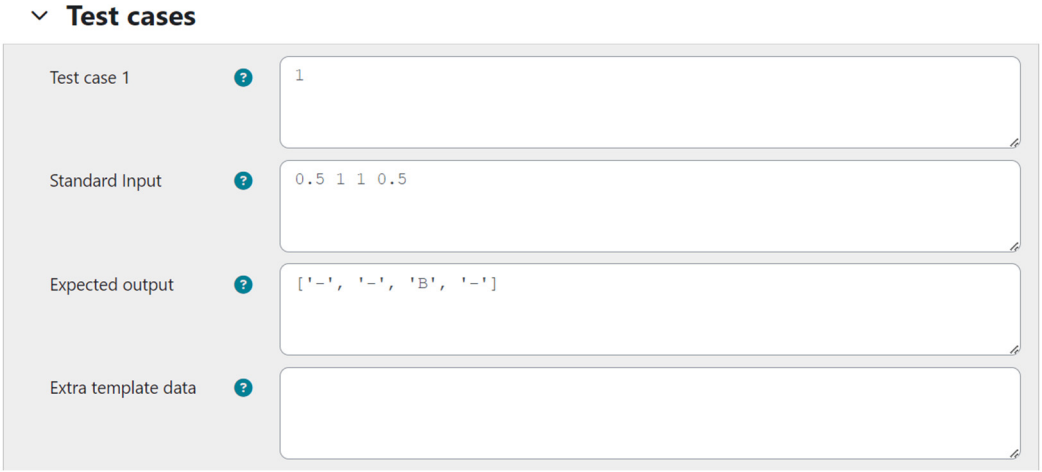


**Figura 32: Sequenze di input e output**

Nel test in Figura 10 si può vedere che di fronte all'inserimento della prima moneta da 50 centesimi (input 0.5) non viene emesso il biglietto (-); dopo aver inserito una seconda moneta da 1 euro l'importo accumulato risulta di 1,5 euro, non ancora sufficiente per emettere il biglietto (-); con la terza moneta da 1 euro si raggiunte la quota di 2,5 euro richiesta e quindi l'output dell'automa evidenzierà che è possibile emettere il biglietto (B), azzerando l'importo accumulato e lasciando a disposizione un resto di 50 centesimi; con la quarta moneta da 50 centesimi si raggiunge quota di 1 euro, non sufficiente per emettere il biglietto (-); in mancanza di ulteriori monete inserite l'automa non produce ulteriore output.

È possibile eventualmente cambiare il carattere separatore (di default la barra "/") specificando nella sezione "Extra template data" il carattere da utilizzare.


**Figura 33: Variazione del carattere separatore input/output**

## 3.3 Prototipo Macchina di Turing

Questo tipo di domanda (disponibile se si importa PROTOTYPES_MDT [2]) consente di realizzare esercizi sulle Macchine di Turing e di verificarne la correttezza. Il prototipo non fa parte dei prototipi forniti di default da coderunner ma anche questo è stato costruito a partire dal prototipo *directed_graph*.

Il nuovo prototipo creato (*MdT*) ha lo scopo di analizzare il programma rappresentato da sequenze di istruzioni in forma di quintuple e di verificarne la correttezza in base all'input dato e all'output previsto nei test.

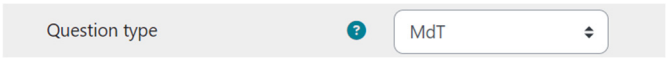


**Figura 34: Tipo domanda Macchina di Turing**

Il prototipo accetta una sequenza di istruzioni del tipo:

```
qp, I, U, qs, S
```

**Figura 35: Formato delle istruzioni di una MdT**

la cui interpretazione è la seguente: se ci si trova nello stato *qp* e si legge in ingresso *I*, si produce l'output *U*, ci si sposta nello stato *qs*, e si muove la testina di lettura/scrittura in base al valore di *S*, che può essere < (sposta di una casella a sinistra), > (sposta di una casella a destra), - (nessun spostamento).

La casella vuota del nastro viene indicata attraverso il simbolo _ (underscore).

Di seguito un esempio.

### *Controllo Bit di parità*

*Si vuole progettare una MdT che, dato un nastro iniziale contenente una sequenza binaria, termini la sua esecuzione aggiungendo alla sequenza data il bit di parità, che sarà pari a 0 se il numero di 1 della sequenza è pari, 1 se è dispari.*

La macchina che risolve il problema può essere descritta dal seguente programma:

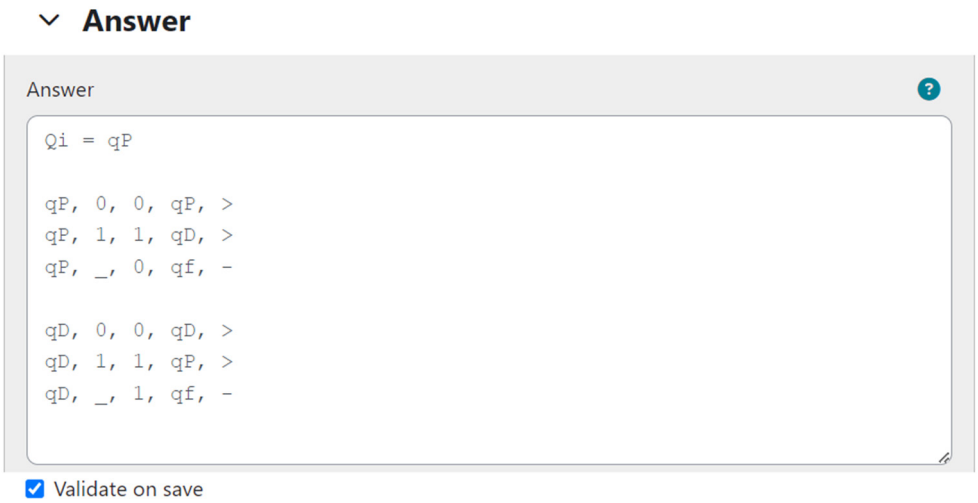


**Figura 36: MdT risolutiva**

È possibile aggiungere alcune istruzioni iniziali di configurazione della macchina che permettono:

- di specificare lo stato iniziale, con un'istruzione del tipo *Qi = q0* (se non espresso lo stato iniziale è lo stato 0)

- di specificare gli stati finali, con un'istruzione del tipo *Qf = [qf1, qf2, …, qfn]*
- di specificare la posizione iniziale della testina di lettura/scrittura, con un'istruzione del tipo *Ti = p*: il valore 0 per p indica la posizione del carattere più a sinistra presente nel nastro, mentre il valore -1 indica la posizione del carattere più a destra (se non espressa la posizione iniziale è sul carattere più a sinistra).

Nei test forniti per verificare la correttezza della soluzione implementata dallo studente è necessario specificare la sequenza degli ingressi e la corrispondente sequenza delle uscite.

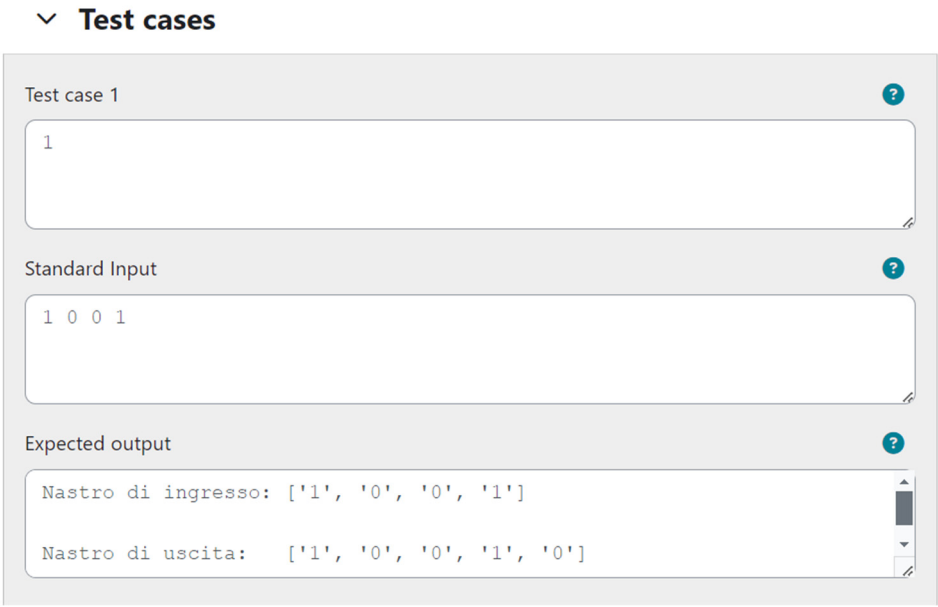


**Figura 37: Sequenze di input e output**

Nel test in Figura 16 si può vedere che di fronte a un nastro che contiene un numero pari di 1 è stato aggiunto in coda il valore 0 (se il numero di 1 fosse stato dispari il valore aggiunto sarebbe stato l'1).

## 3.4 Prototipo Programmazione in Python con vincoli

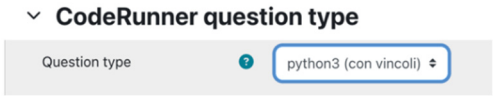


**Figura 38 - Tipo domanda programmazione in Python con vincoli**

Questo tipo di domanda (disponibile se si importa PROTOTYPES_PY_W_BANNED [2]) consente di realizzare esercizi di programmazione in Python nei quali si impone il vincolo di non usare alcune parole chiave corrispondenti a costrutti sintattici del linguaggio.

Lo scopo è quello di creare degli esercizi con correzione automatica nei quali lo studente deve risolvere il problema usando forzatamente le strutture sintattiche scelte dal docente.

Di seguito alcuni esempi.

### *A. Programmi ricorsivi*

Per imporre l'utilizzo della ricorsione nella soluzione dell'esercizio è sufficiente "bandire" l'uso di *while* e *for* impostando nella sezione Global extra il campo *banned*.

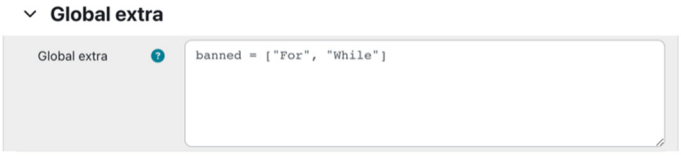


**Figura 39 - Impostazione parole chiave non utilizzabili**

In questo modo se lo studente risolve l'esercizio utilizzando un algoritmo iterativo il sistema di correzione automatico segnala l'errore.

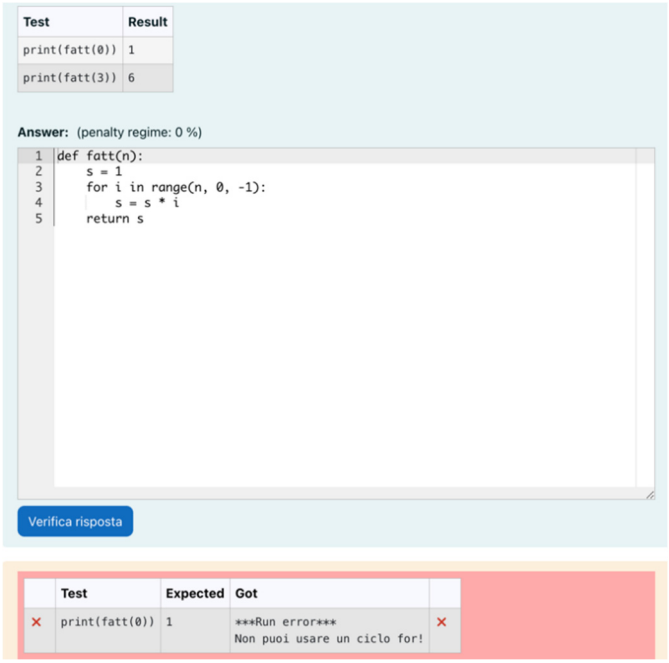


**Figura 40 - Esempio di verifica risposta errata**

## *B. Programmi iterativi con while / for*

Per imparare ad utilizzare i costrutti iterativi spesso vengono dati agli studenti degli esercizi in cui si fornisce un programma che fa uso di un ciclo *for* e si chiede di produrre un esercizio equivalente con l'uso del *while* (o viceversa).

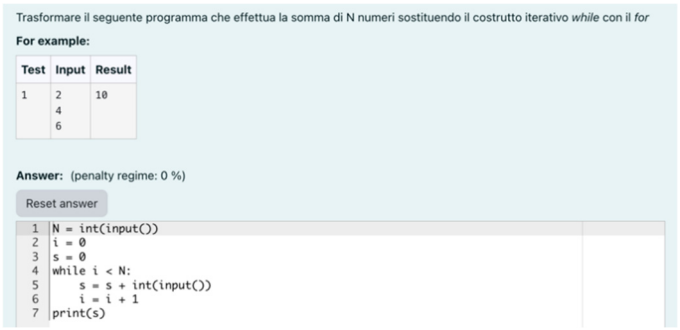


**Figura 41 - Anteprima esercizio**

In questo caso si imposta nella sezione Global extra il campo *banned* come nella figura seguente.

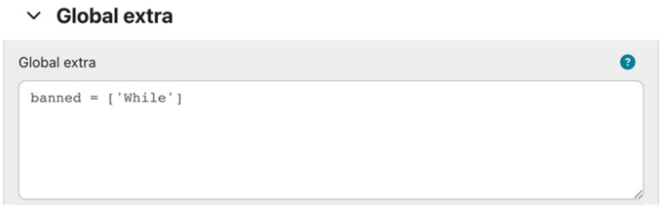


**Figura 42 - Vincolo non utilizzo while**

# 4 CONCLUSIONI

La personalizzazione dei template del plug-in CodeRunner consente di automatizzare il processo di apprendimento e valutazione per argomenti complessi come gli automi a stati finiti e le macchine di Turing, migliorandone l'efficacia e offrendo agli studenti uno strumento interattivo con feedback immediato e dettagliato sulle loro soluzioni, senza la necessità della presenza costante di un docente.

Inoltre, questa personalizzazione fornisce spunti per la creazione di test mirati su competenze specifiche di programmazione ed anche su argomenti non strettamente legati alla programmazione.

Per chi non volesse cimentarsi nella programmazione può utilizzare i prototipi di domanda forniti in allegato a questo articolo [2] (usando l'importazione domande di Moodle), da usare così come sono o come base da modificare a proprio piacimento.

## Riferimenti bibliografici

[1] CodeRunner, https://coderunner.org.nz/

[2] Export Prototipi Domande in *Formato Moodle XML,* https://drive.google.com/file/d/14D4Kn8z1HDLGahgdHDw-ahyweY-mDTRo