

BRICKS | TEMA

Test automatici su Automi a Stati Finiti e Macchine di Turing con CodeRunner

a cura di:

Adolfo Casagrande, Luciano Dereani,
Eva Pantanali



Valutazione automatica, CodeRunner

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.

Requisiti

Plug-in Code Runner

Requisito essenziale è l'installazione del plug-in CodeRunner.

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/>.

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.

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.

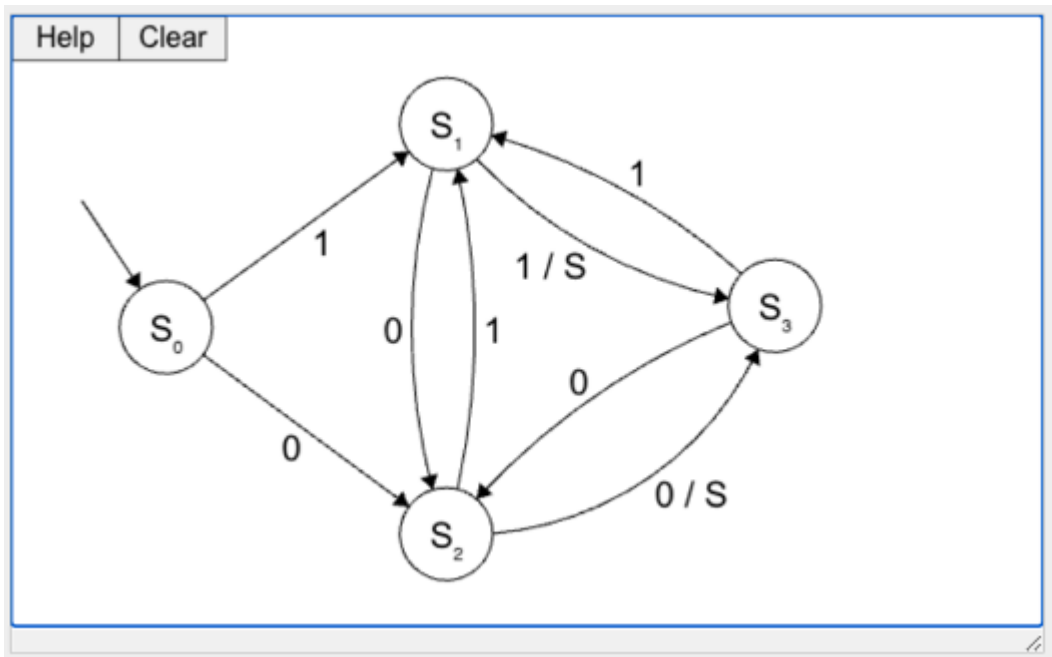


Figura 1 - GraphUI per il disegno di un grafo

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,
      "perpendicularPart": 38.928616866725605
    },
    {
      "lineAngleAdjust": 0,
      "parallelPart": 0.5,
      "perpendicularPart": 16
    },
    {
      "lineAngleAdjust": 0,
      "parallelPart": 0.5,
      "perpendicularPart": 16
    }
  ],
  "nodeGeometry": [
    [86.25, 174.7734375],
    [243.25, 60.7734375],
    [427.25, 162.7734375],
    [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 disponibili nell'editor:

Help
Clear

- Double click at a blank space to create a new node/state.
- Double click an existing node to "mark" it e.g. as an accept state for Finite State Machines (FSMs). Double click again to unmark it.
- Click and drag to move a node.
- Alt click (or Ctrl Alt click) and drag to move a (sub)graph.
- Shift click inside one node and drag to another to create a link.
- Shift click on a blank space, drag to a node to create a start link (FSMs only).
- Click and drag a link to alter its curve.
- Click on a link/node to edit its text.
- Click in link text and drag to move it.
- Typing `_` followed by a digit makes that digit a subscript.
- Typing `\epsilon` creates an epsilon character (and similarly for `\alpha`, `\beta` etc).
- Click on a link/node then press the Delete key to remove it (or function-delete on a Mac).
- Ctrl+z to undo and Ctrl+y or Ctrl+Shift+z to redo.

Figura 2 - comandi editor

La valutazione della correttezza della risposta deve essere personalizzata scrivendo il codice Python che controlla la struttura del grafo nel campo Customisation 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.

Creazione Prototipo di domanda

Indicazioni operativa per creare un prototipo di domanda personalizzato.

- Creare una nuova domanda appartenente alla categoria CR_CUSTOM_PROTOTYPES

Deposito delle domande

Filtra Tutto delle seguenti condizioni:

Filtra Categoria Scrivi o seleziona... CR_CUSTOM_PROTOTYPES (2) X

AND

Filtra Visualizza domande nascoste No

+ Aggiungi condizione

Crea una nuova domanda... Reimposta colonne Visualizzazione del testo della domanda nell'elenco di

☐	Domanda	Azioni	Stato	Versione	Comme
	Nome della domanda / Codice identificativo				
☐	PROTOTYPE_ASF automa	Modifica	Pronta	v3	0
☐	PROTOTYPE_MDT automa	Modifica	Pronta	v4	0

Figura 3 - Nuova domanda

- Attivare il checkbox *Customisation* della sezione *CodeRunner* question type
- Personalizzare il codice di controllo impostando il campo *Template* nella sezione *Customisation*:

✓ Customisation

```

Template
1 import json
2
3 class Automaton:
4     def __init__(self, grafo, splitChar):
5         # Inizializza gli stati e le transizioni
6         self.grafo = grafo # Dizionario contenente i nodi e le liste di adiacenza
7         self.current_state = list(grafo.keys())[0] # Inizia dallo stato 0 (nodo "A")
8         self.current_uscita = ""
9         self.splitChar = splitChar
10
11     def reset(self):
12         self.current_state = list(self.grafo.keys())[0]
13         self.current_uscita = ""
14
15     #Separa l'etichetta della transizione in input e output utilizzando '/' come separatore.
16     def get_ingresso(self, label):
17         parts = label.split(self.splitChar)
18         return parts[0].strip() if len(parts) > 0 else label.strip
19
20     def get_uscita(self, label):
21         parts = label.split(self.splitChar)
22         return parts[1].strip() if len(parts) > 1 else ""
23
24     def get_transition(self, ingresso):
25         # Trova una transizione dallo stato corrente con l'etichetta data
26         for edge in self.grafo[self.current_state]:

```

Figura 4 - 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

Advanced customisation

Prototyping	 Is prototype? Yes (user defined) Question type directed_graph_asf
Sandbox	 TimeLimit (secs) MemLimit (MB) Parameters
Languages	 Sandbox language python3 Ace language

Figura 5 - 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).

Categoria in uso	CR_CUSTOM_PROTOTYPES (2)
Versione	<u>Versione 3</u>
	Creazione di Casagrande Adolfo di venerdì, 12 gennaio 2024, 18:36
Nome della domanda	PROTOTYPE_ASF
Testo della domanda	<u>Template per la gestione di un ASF</u> Creazione di un ASF mediante il diagramma degli stati Doppio click per generare un nodo

Figura 6 - 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.

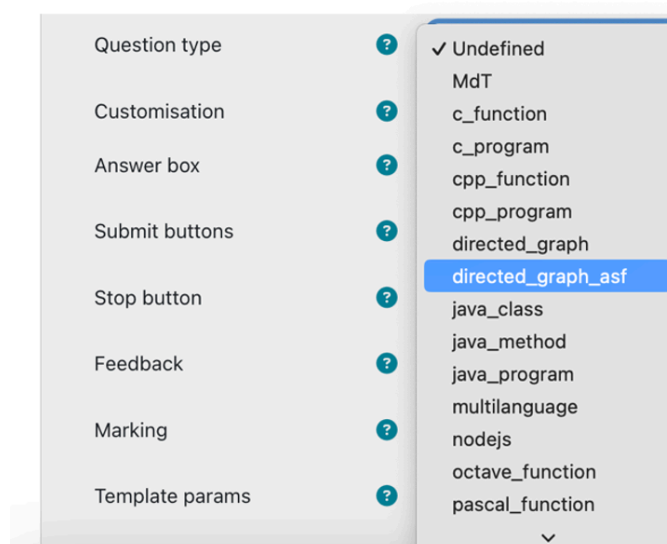


Figura 7 - elenco tipo domande disponibili

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.

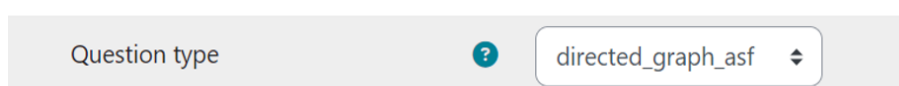


Figura 8 - Tipo domanda Automi a Stati Finiti

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 /).

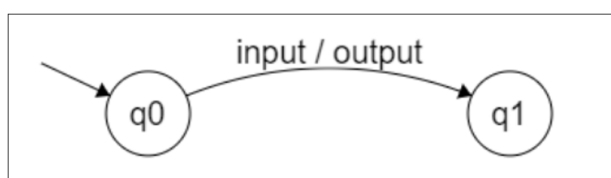


Figura 9 – Formato utilizzato per descrivere un ASF

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:

Answer

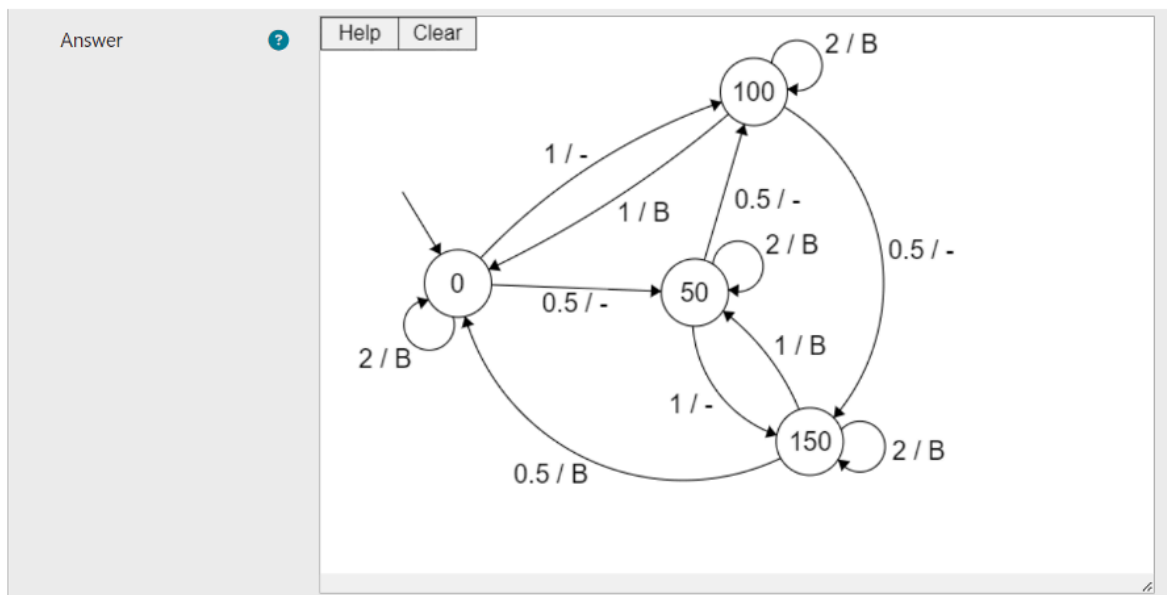


Figura 10 – ASF risolutivo

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.

Test cases

Test case 1	1
Standard Input	0.5 1 1 0.5
Expected output	['-', '-', 'B', '-']
Extra template data	

Figura 11 – 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 raggiunge 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 12 – Variazione del carattere separatore input/output

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 13 – Tipo domanda Macchina di Turing

Il prototipo accetta una sequenza di istruzioni del tipo:

`qp, I, U, qs, S`

Figura 14 – 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:

▼ Answer

Answer

$Q_i = q_P$

 $q_P, 0, 0, q_P, >$
 $q_P, 1, 1, q_D, >$
 $q_P, _, 0, q_f, -$

 $q_D, 0, 0, q_D, >$
 $q_D, 1, 1, q_P, >$
 $q_D, _, 1, q_f, -$

☒ Validate on save

Figura 15 – MdT risolutiva

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

- di specificare lo stato iniziale, con un'istruzione del tipo $Q_i = q_0$ (se non espresso lo stato iniziale è lo stato 0)
- di specificare gli stati finali, con un'istruzione del tipo $Q_f = [qf_1, qf_2, \dots, qf_n]$
- di specificare la posizione iniziale della testina di lettura/scrittura, con un'istruzione del tipo

$T_i = p$: 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.

▼ Test cases

Test case 1

1

Standard Input

1 0 0 1

Expected output

Nastro di ingresso: ['1', '0', '0', '1']

 Nastro di uscita: ['1', '0', '0', '1', '0']

Figura 16 – 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).

Prototipo Programmazione in Python con vincoli

▼ CodeRunner question type

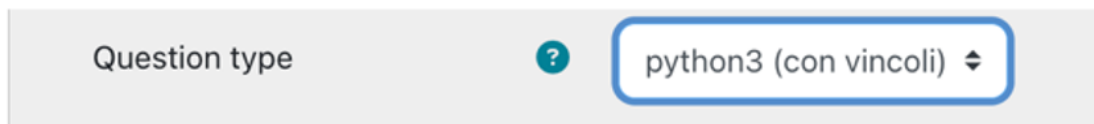


Figura 17 - Tipo domanda programmazione in Python con vincoli

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.

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`.

▼ Global extra

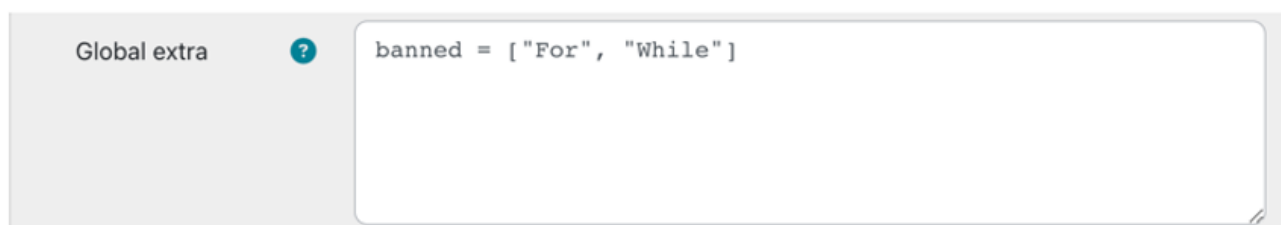


Figura 18 - impostazione parole chiave non utilizzabili

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

Test	Result
print(fatt(0))	1
print(fatt(3))	6

Answer: (penalty regime: 0 %)

```

1 def fatt(n):
2     s = 1
3     for i in range(n, 0, -1):
4         s = s * i
5     return s
    
```

Verifica risposta

Test	Expected	Got
✗ print(fatt(0))	1	***Run error*** Non puoi usare un ciclo for!

Figura 19 - esempio di verifica risposta errata

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).

Trasformare il seguente programma che effettua la somma di N numeri sostituendo il costrutto iterativo *while* con il *for*

For example:

Test	Input	Result
1	2 4 6	10

Answer: (penalty regime: 0 %)

Reset answer

```

1 N = int(input())
2 i = 0
3 s = 0
4 while i < N:
5     s = s + int(input())
6     i = i + 1
7 print(s)
    
```

Figura 20 - anteprima esercizio

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

Global extra

```
Global extra
banned = ['While']
```

Figura 21 - vincolo non utilizzo while

Conclusioni

La personalizzazione dei template del plug-in CodeRunner¹ 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 articolo² (usando l'importazione domande di Moodle), da usare così come sono o come base da modificare a proprio piacimento.



Adolfo Casagrande

adolfo.casagrande@malignani.ud.it

I.S.I.S. A. Malignani di Udine

Docente di Informatica presso l'ISIS A. Malignani di Udine

¹ CodeRunner, <https://coderunner.org.nz/>

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



Luciano Dereani

luciano.dereani@malignani.ud.it

I.S.I.S. A. Malignani di Udine

Docente di Informatica presso l'ISIS A. Malignani di Udine, webmaster e formatore per docenti su tematiche legate all'utilizzo del digitale nella didattica. Ha partecipato a numerosi progetti in ambito TEAL (Technology-Enhanced Active Learning) e vanta oltre 10 anni di esperienza nello sviluppo di software gestionali.



Eva Pantanali

eva.pantanali@malignani.ud.it

I.S.I.S. A. Malignani di Udine

Docente di Informatica presso l'ISIS A. Malignani di Udine, Funzione Strumentale per il Benessere e formatore per docenti su tematiche legate all'utilizzo del digitale nella didattica. Ha partecipato a numerosi progetti in ambito TEAL (Technology-Enhanced Active Learning) e vanta oltre vent'anni di esperienza nella progettazione e nello sviluppo web, specializzata nell'ottimizzazione per diversi dispositivi e nella creazione di soluzioni accessibili, con l'obiettivo di garantire un'esperienza utente inclusiva e di alta qualità. Ha maturato una consolidata esperienza nella gestione di team di lavoro, coordinando risorse e progetti complessi e multidisciplinari a livello internazionale. Inoltre, è esperta nell'ottimizzazione dei processi produttivi legati alle piattaforme digitali.