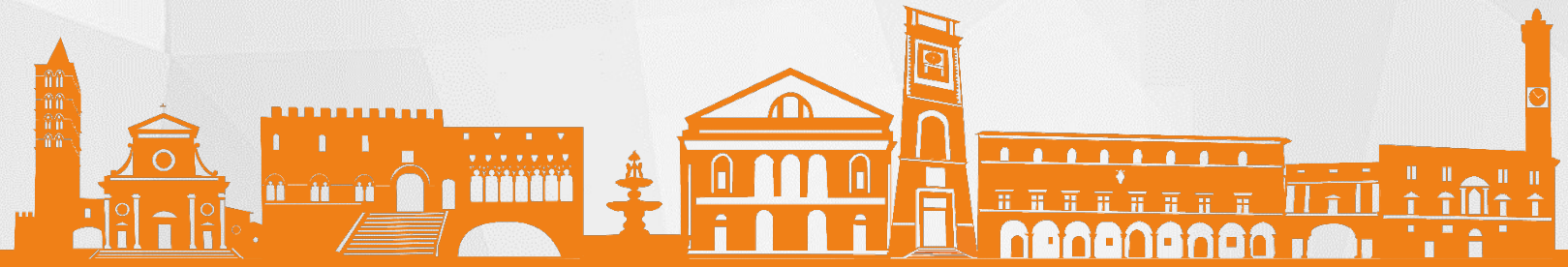
TEST AUTOMATICI SU AUTOMI A STATI FINITI E MACCHINE DI TURING CON CODERUNNER

Adolfo Casagrande
ISIS. A. Malignani
adolfo.casagrande@malignani.ud.it

Luciano Dereani
ISIS. A. Malignani
luciano.dereani@malignani.ud.it



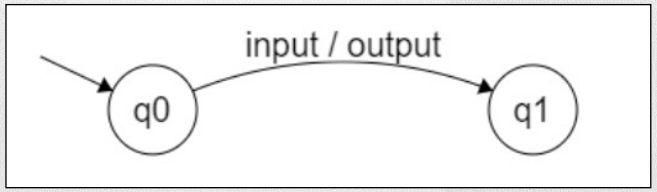
Introduzione

- Esigenza didattica classi 5^ Liceo Scienze Applicate
- Argomenti: ***automi a stati finiti*** e ***macchine di turing***
- **Editor** per lo svolgimento degli esercizi (grafico)
- Strumento di **correzione automatica**

Problema – Automi a stati finiti

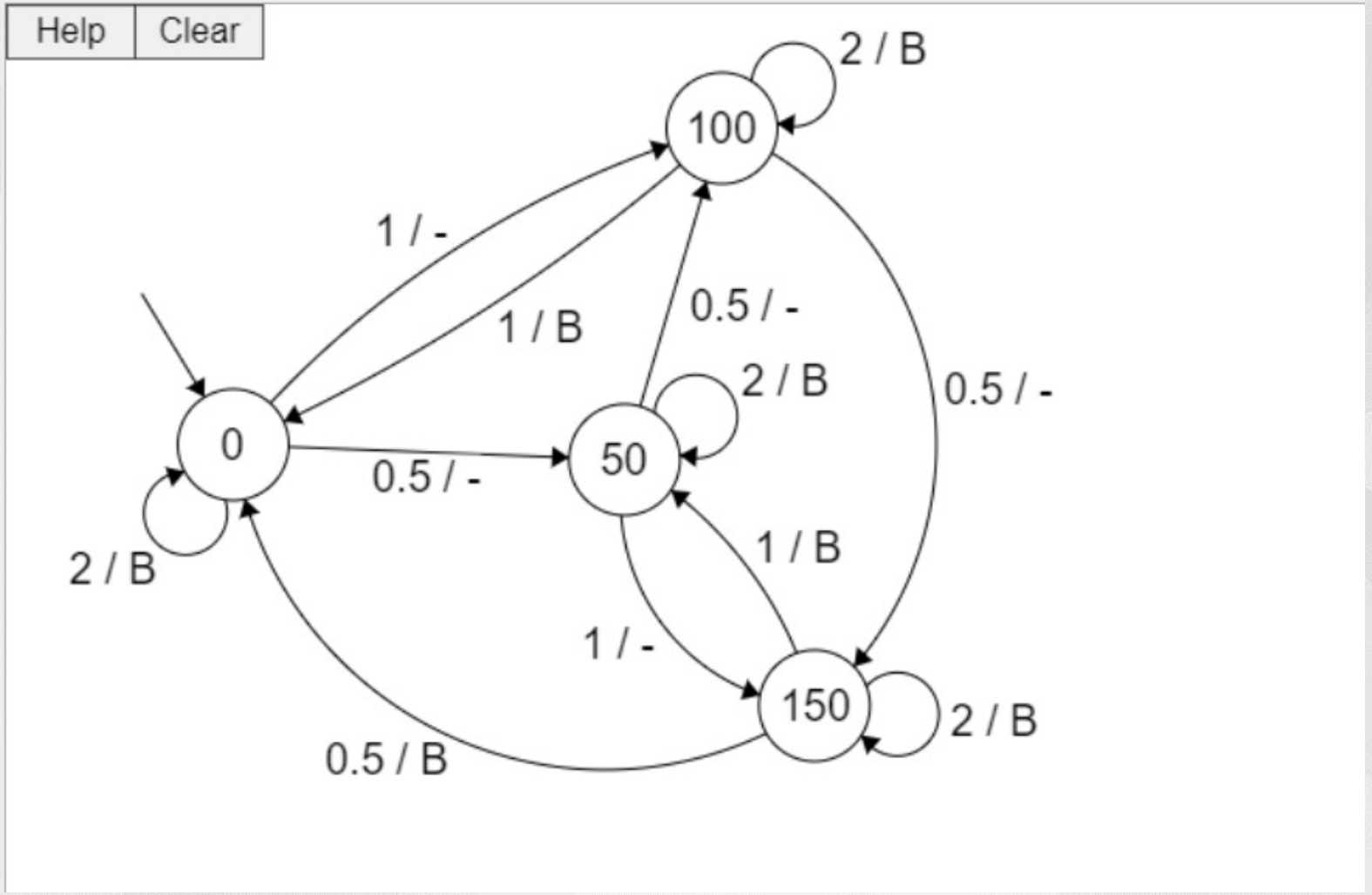
Biglietti da distributore automatico

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



Automa di Mealy

Soluzione – Distributore biglietti



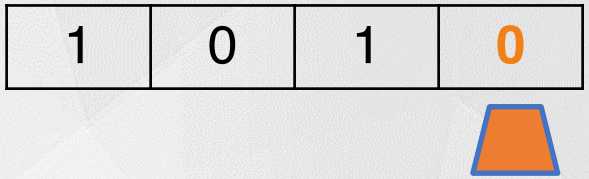
Problema – Macchina di Turing

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.

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

Formato istruzioni MdT



Soluzione – Bit di parità

```
Qi = qP

qP, 0, 0, qP, >
qP, 1, 1, qD, >
qP, _, 0, qf, -

qD, 0, 0, qD, >
qD, 1, 1, qP, >
qD, _, 1, qf, -
```

Problema – Utilizzo strutture di controllo

Programmi ricorsivi

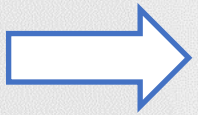
Realizzare un programma in python che calcoli il **fattoriale** di un numero utilizzando la **ricorsione** (e **non l'iterazione**).

```
fatt(n) = 1   se n <= 0

fatt(n) = n * fatt(n-1)  se n > 0
```

Problema – For o While

Programmi iterativi con for / while

Dato il seguente programma scritto utilizzando la struttura di controllo **for**, scrivere un programma equivalente che utilizzi il **while…**

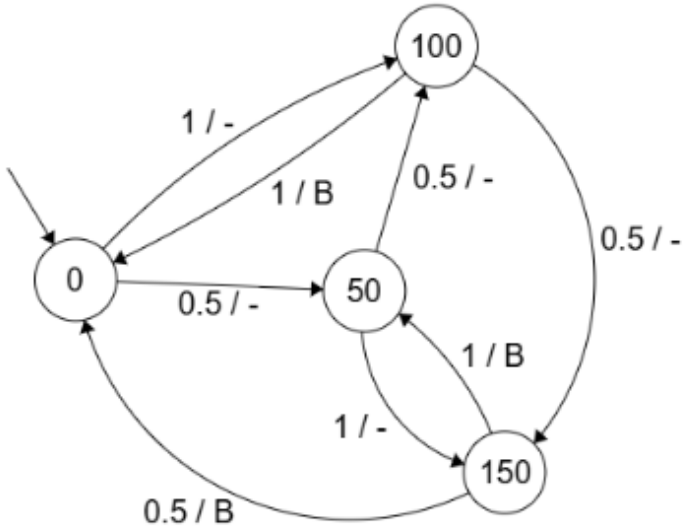
for  **while** (o viceversa)

Soluzione

- plugin Code-Runner di Moodle
- **personalizzazione tipologia domande**
- Ulteriori potenzialità: validazione automatica nell'uso di strutture di controllo in python

Esempio di validazione automatica - 1

Help | Clear



Verifica risposta

	Test	Input	Expected	Got	
✓	1	0.5 0.5 1 0.5	['-', '-', 'B', '-']	['-', '-', 'B', '-']	✓
✗	2	2 1 0.5 2 1 0.5 0.5 0.5	['B', '-', '-', 'B', 'B', '-', '-', 'B']	[] STOP 0	✗
✗	3	1 1 0.5 1 0.5 0.5 0.5 2 2 1 0.5 2 1 0.5	['-', 'B', '-', '-', 'B', '-', '-', 'B', 'B', 'B', '-', 'B', '-', 'B']	['-', 'B', '-', '-', 'B', '-', '-'] STOP 100	✗

Esempio di validazione automatica - 2

```
Qi = qP

qP, 0, 0, qP, >
qP, 1, 1, qD, >
qP, _, 0, qf, -

qD, 0, 0, qD, >
qD, 1, 1, qP, >
qD, _, 0, qf, -
```

Verifica risposta

	Input	Expected	Got
✓	1 0 0 1	Nastro di ingresso: ['1', '0', '0', '1'] Nastro di uscita: ['1', '0', '0', '1', '0']	Nastro di ingresso: ['1', '0', '0', '1'] Nastro di uscita: ['1', '0', '0', '1', '0']
✗	0 1 1 0 1	Nastro di ingresso: ['0', '1', '1', '0', '1'] Nastro di uscita: ['0', '1', '1', '0', '1', '1']	Nastro di ingresso: ['0', '1', '1', '0', '1'] Nastro di uscita: ['0', '1', '1', '0', '1', '0']
✗	1 0 0 1 0 1 1 1 0	Nastro di ingresso: ['1', '0', '0', '1', '0', '1', '1', '1', '0'] Nastro di uscita: ['1', '0', '0', '1', '0', '1', '1', '1', '0', '1']	Nastro di ingresso: ['1', '0', '0', '1', '0', '1', '1', '1', '0'] Nastro di uscita: ['1', '0', '0', '1', '0', '1', '1', '1', '0', '0']
✓	0	Nastro di ingresso: ['0'] Nastro di uscita: ['0', '0']	Nastro di ingresso: ['0'] Nastro di uscita: ['0', '0']

Soluzione – Fattoriale ricorsivo

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!	✗

Answer: (penalty regime: 0 %)

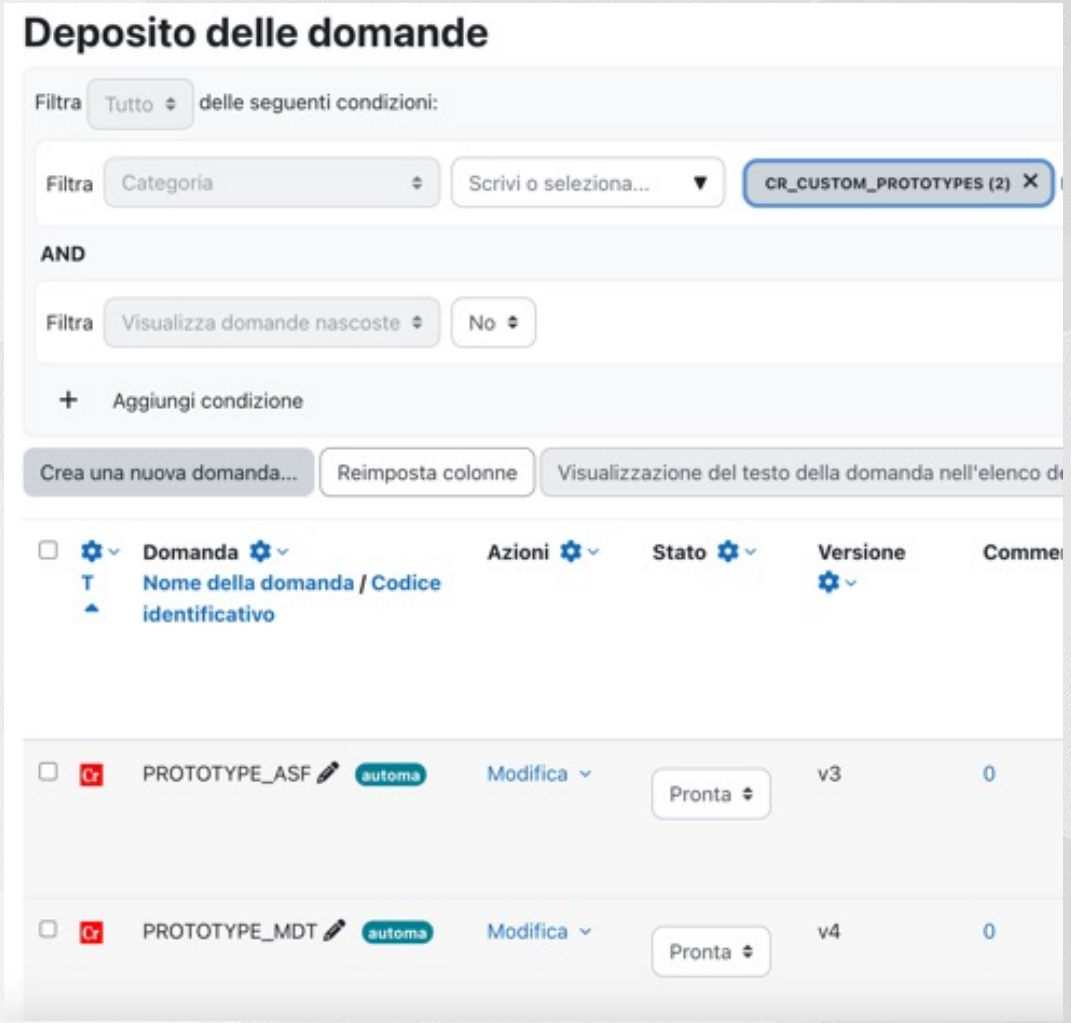
```
1  def fatt(n):
2      """ fattoriale ricorsivo """
3      if (n == 0):
4          return 1
5
6      if (n > 0):
7          return n * fatt(n-1)
```

	Test	Expected	Got	
✓	print(fatt(0))	1	1	✓
✓	print(fatt(1))	1	1	✓
✓	print(fatt(3))	6	6	✓
✓	print(fatt(5))	120	120	✓
✓	print(fatt(8))	40320	40320	✓

Passed all tests! ✓

Come creare un prototipo – passo 1

Creare una nuova domanda
appartenente alla categoria
CR_CUSTOM_PROTOTYPES



Come creare un prototipo – passo 2

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

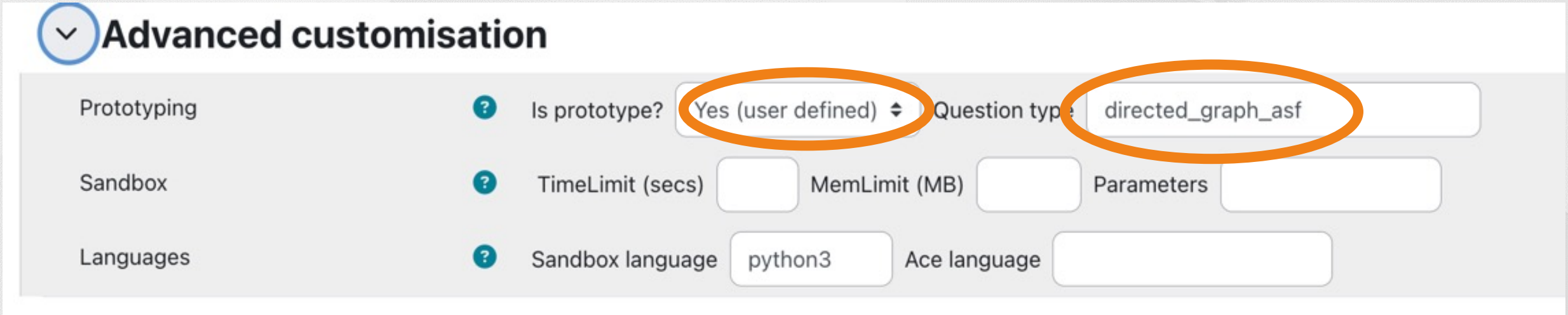
PYTHON

```python
import json

class Automaton:
    def __init__(self, grafo, splitChar):
        # Inizializza gli stati e le transizioni
        self.grafo = grafo  # Dizionario contenente i nodi e le liste di adiacenza
        self.current_state = list(grafo.keys())[0]  # Inizia dallo stato 0 (nodo "A")
        self.current_uscita = ""
        self.splitChar = splitChar

    def reset(self):
        self.current_state = list(self.grafo.keys())[0]
        self.current_uscita = ""

    #Separa l'etichetta della transizione in input e output utilizzando '/' come separatore.
    def get_ingresso(self, label):
      parts = label.split(self.splitChar)
      return parts[0].strip() if len(parts) > 0 else label.strip
```

Come creare un prototipo – passo 3

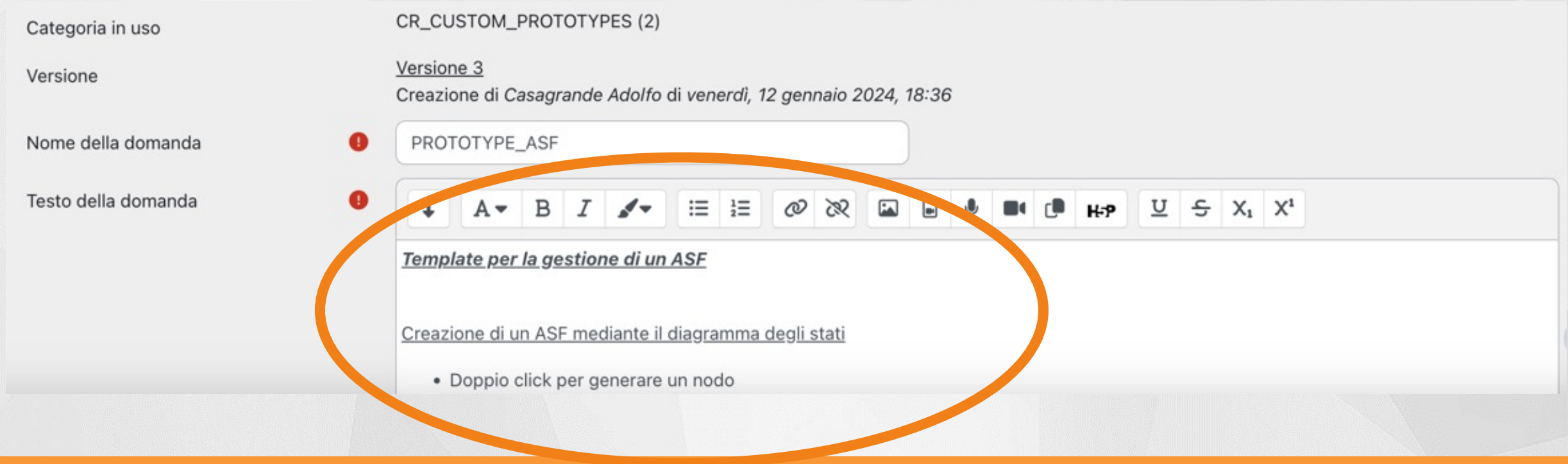
Impostare i campi nella sezione Advanced Customisation:

- **Is prototype?** a Yes (user defined)
- Question type al **nome** che si vuole assegnare a questo tipo di domanda



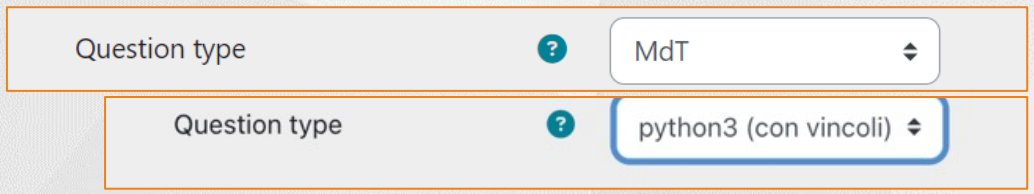
Come creare un prototipo – passo 4

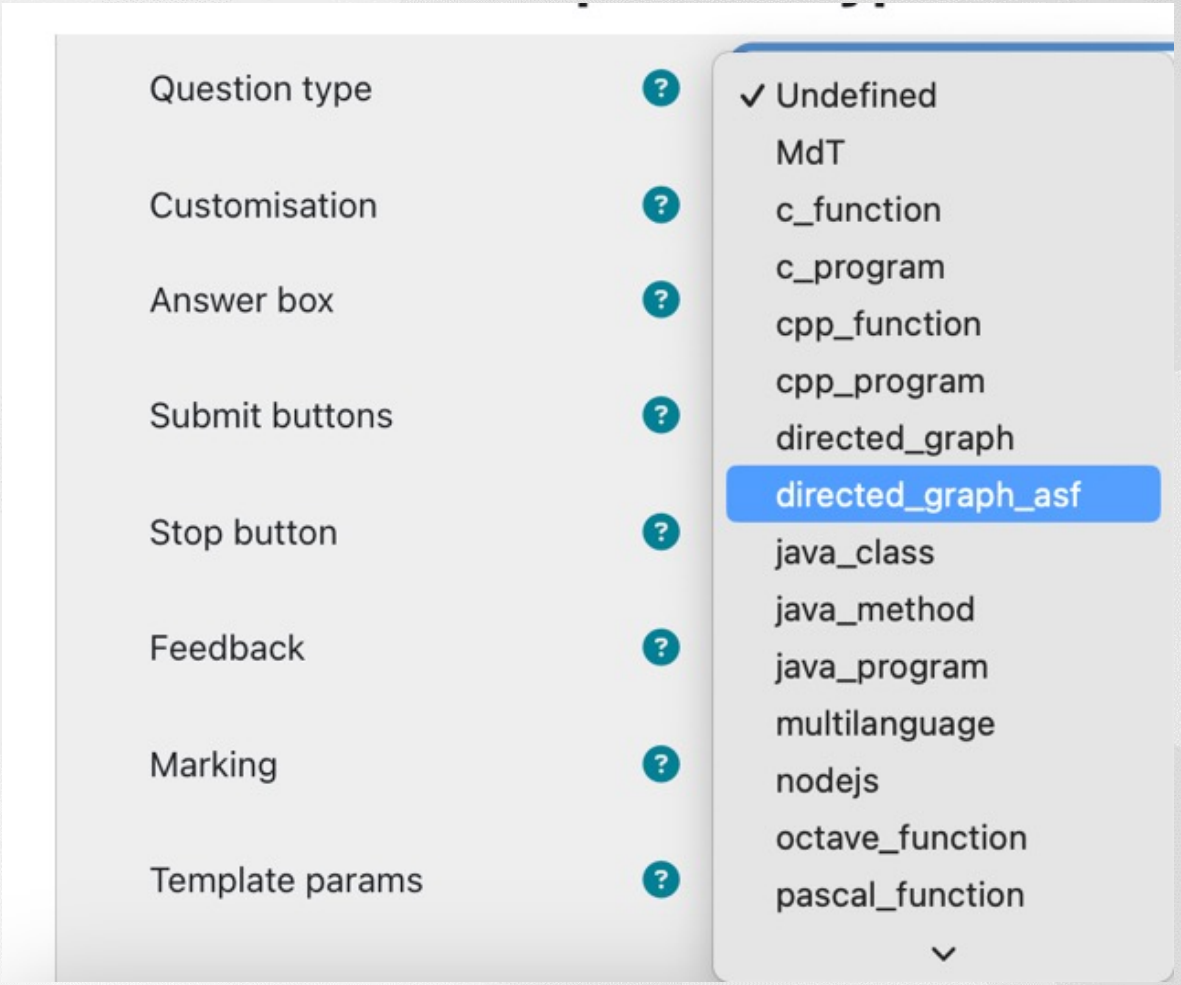
Impostare nella sezione Generale la **descrizione** del prototipo che funge da aiuto in fase di creazione del test



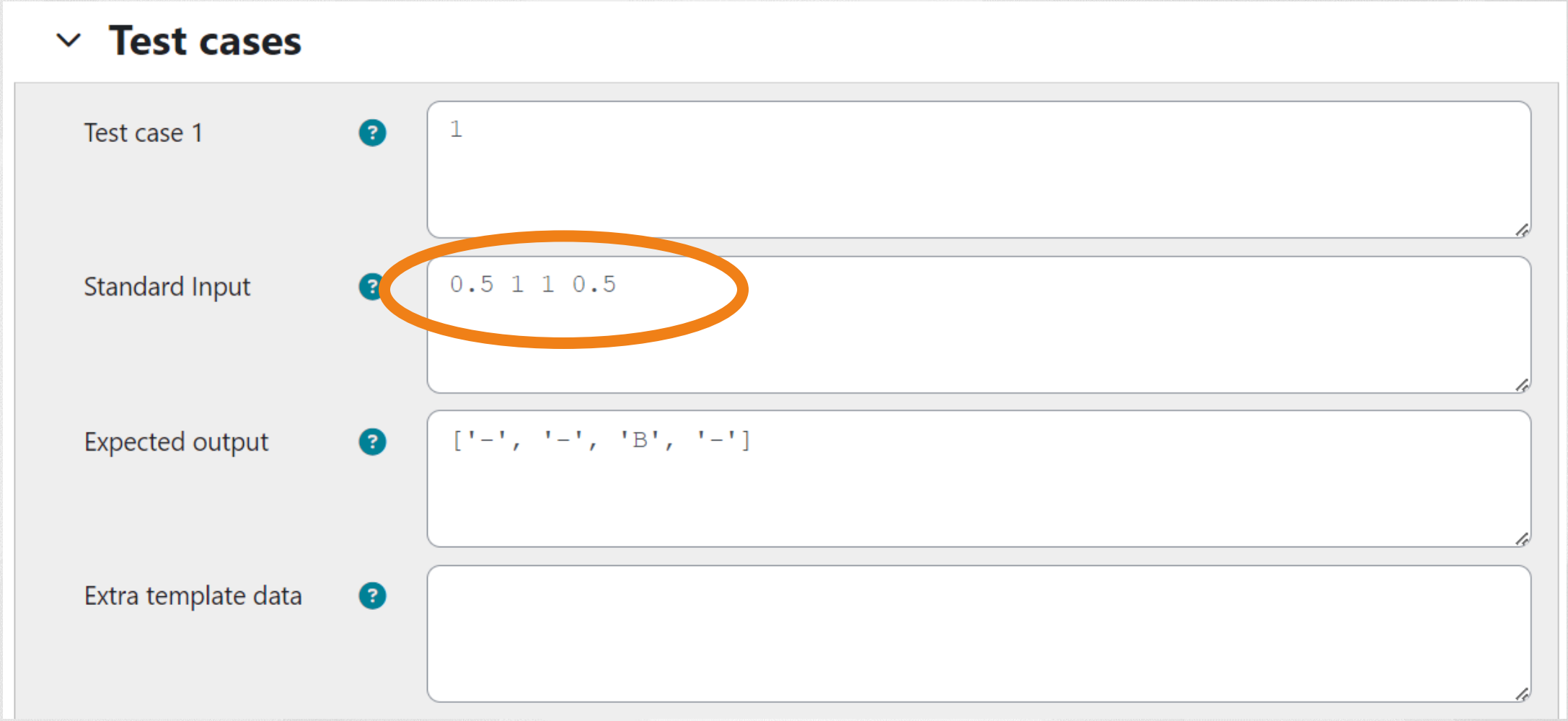
Creazione del TEST

- Selezione del prototipo
- Testo del problema
- Inserimento della soluzione
- Specifica dei casi di test



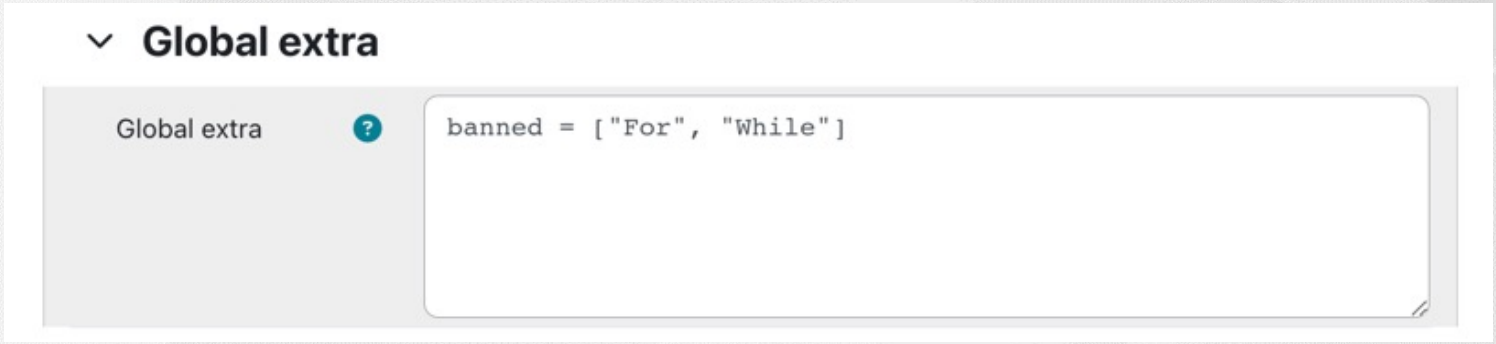


Esempio casi di test – automa a stati finiti



Creazione domande – Funzioni ricorsive

- Specifica dei vincoli (*global extra*)



Requisiti

- Plug in CodeRunner [https://coderunner.org.nz/]

- Prototipi: ***directed_graph_asf***, ***MdT***, ***python3 (con vincoli)***

  [**Download** e **import in moodle**]

- Competenze di programmazione

  [necessarie SOLO se si vuole personalizzare la tipologia dei test]

Conclusioni

La personalizzazione di CodeRunner permette di automatizzare l'apprendimento e la valutazione di argomenti complessi come:

- Automi a stati finiti
- Macchine di Turing

Vantaggi:

- Per docenti: **maggiore efficacia** nell'insegnamento grazie all'uso di test automatici
- Per studenti: strumento interattivo con **feedback immediato** e dettagliato

Grazie per l'attenzione