Programozási feladatok futtatása és ellenőrzése Moodle alatt

Python

Az anyag a Szeghalmy Sz., Fazekas A., Moodle használata a programozás oktatásban, (MoodleMoot 2019, Debreceni Egyetem, Informatikai Kar, június 24-25.) átalakított változata.

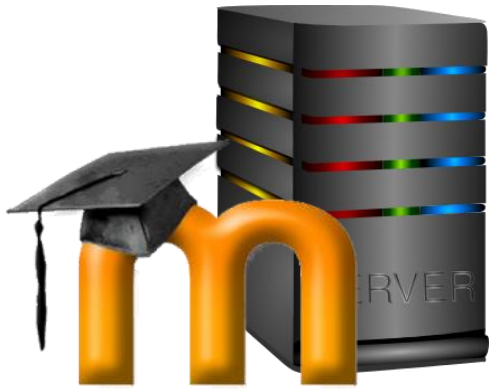
Tartalom

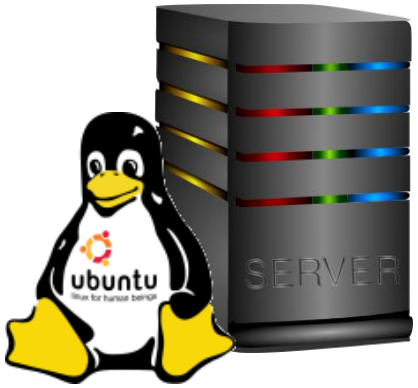
- A kérdéstípus elérése
- Egy szkript készítési feladat
- Egy függvényírási feladat
- A beadott válasz szövegének ellenőrzése („Old meg **while** ciklussal”)
- Kiértékelő sablon módosítása: közelítő egyezés ellenőrzése
- Kiértékelő sablon módosítása: tesztesetek generálása és kiértékelés

Megjegyzés

- Az első két típus elég az induláshoz, ha kerüli a valós számot tartalmazó végeredményt
- A létrehozott feladatok ugyanúgy felhasználhatók tesztekben, mint pl. egy feleletválasztós kérdés.

A háttér





Moodle szerver

CodeRunner plug-in

A forráskódok futtatására szolgáló szerver

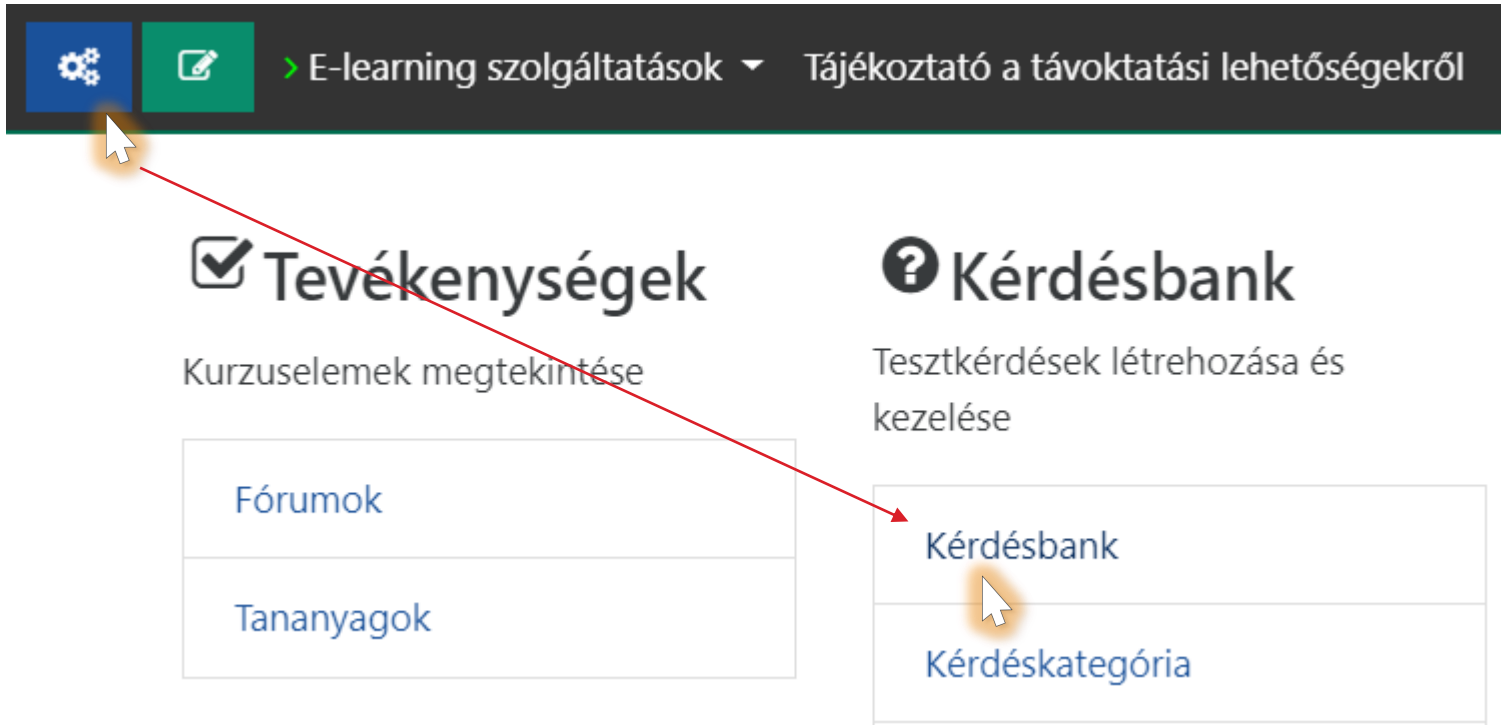
Ubuntu 18.04.1 LTS

job engine

Jelenleg: Python 3.6.9
numpy: 1.18.2

Kérdés létrehozása

- Kérdéseket létrehozni egy adott kurzuson belül tud, válassza ki a megfelelő kurzust.
- A kurzusadminisztrációnál kattintson a kérdésbank gombra.

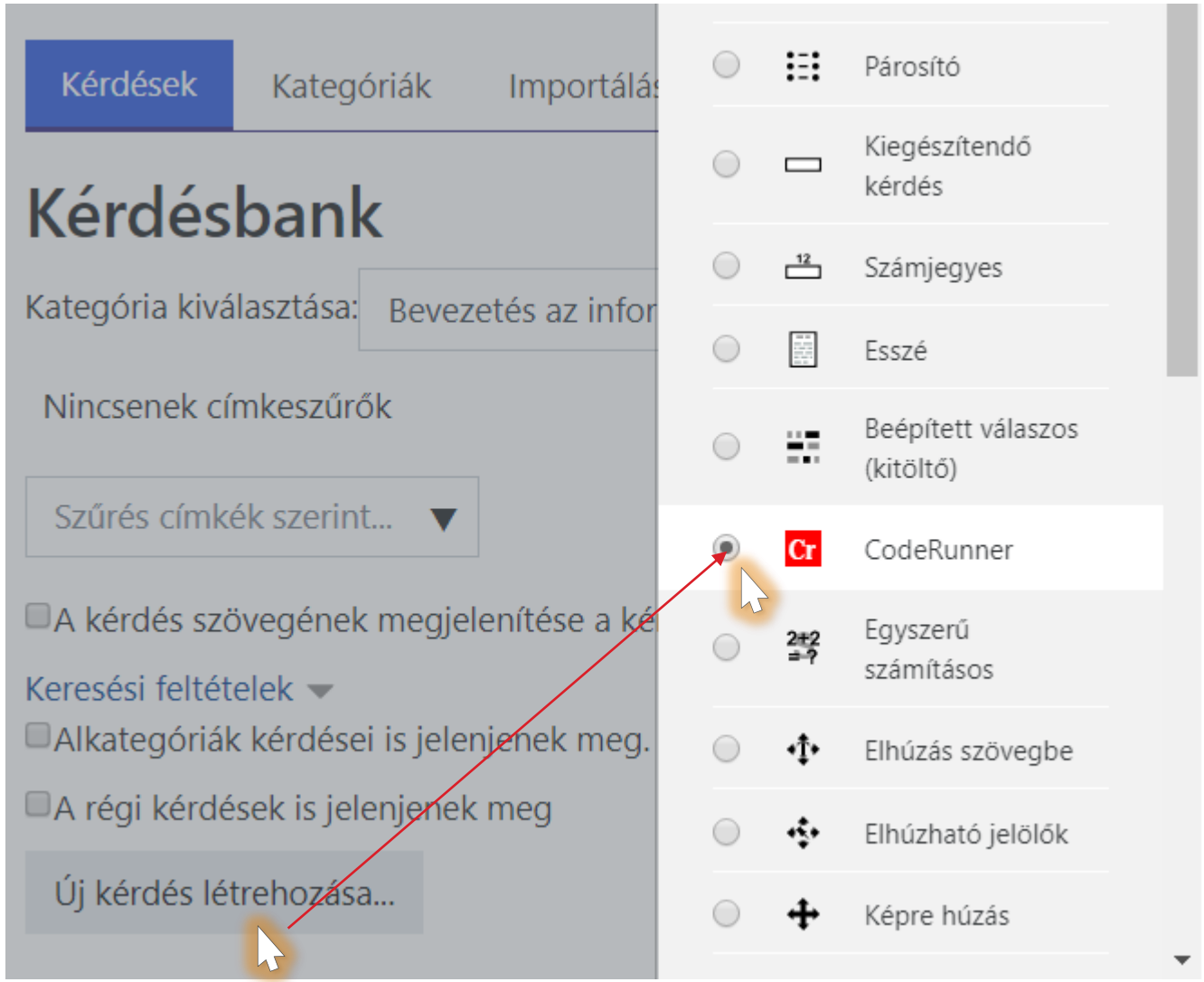


Tipp:

A kérdések rendszerezése érdekében hozzon létre kategóriákat.

(Ez utólag is megtehető, a kérdések mozgathatók a kategóriák között is majd.)

Kérdés létrehozása

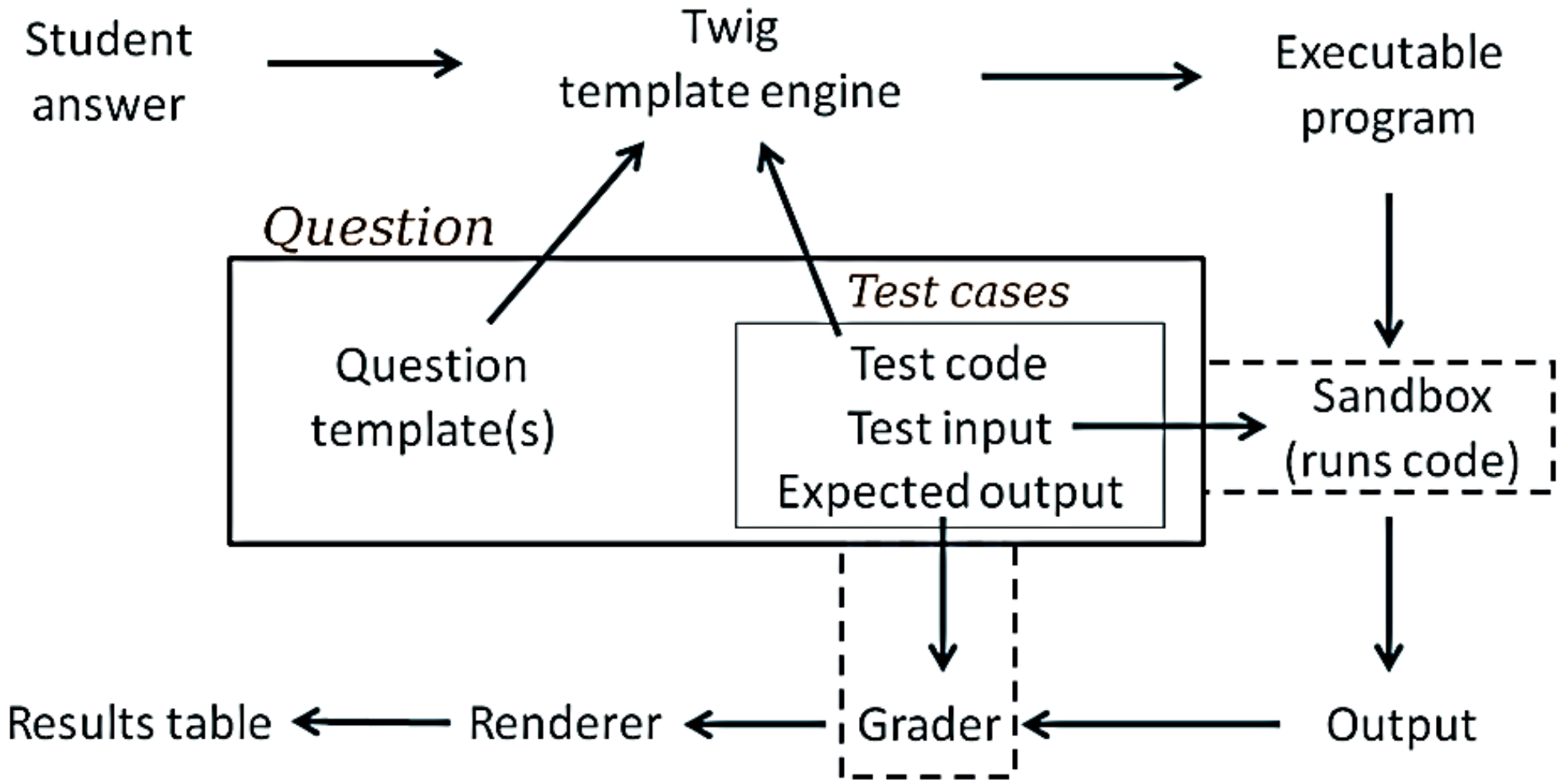


- Kattintson az új kérdés létrehozására
- Válassza a CodeRunner kérdéstípust
- Adja hozzá a kérdést

A feladat alkotóelemei

- Beállítások
  - pl.: programnyelv, kiértékelés módja, kimenet megjelenítése
  - meghatározza a kérdéshez tartozó alapértelmezett sablont
- A kérdés
- A válasz
- Tesztesetek
  - Input adatok
  - Tesztelő kódrészek
  - Elvárt kimenet
- A sablon

Futtatás és kiértékelés



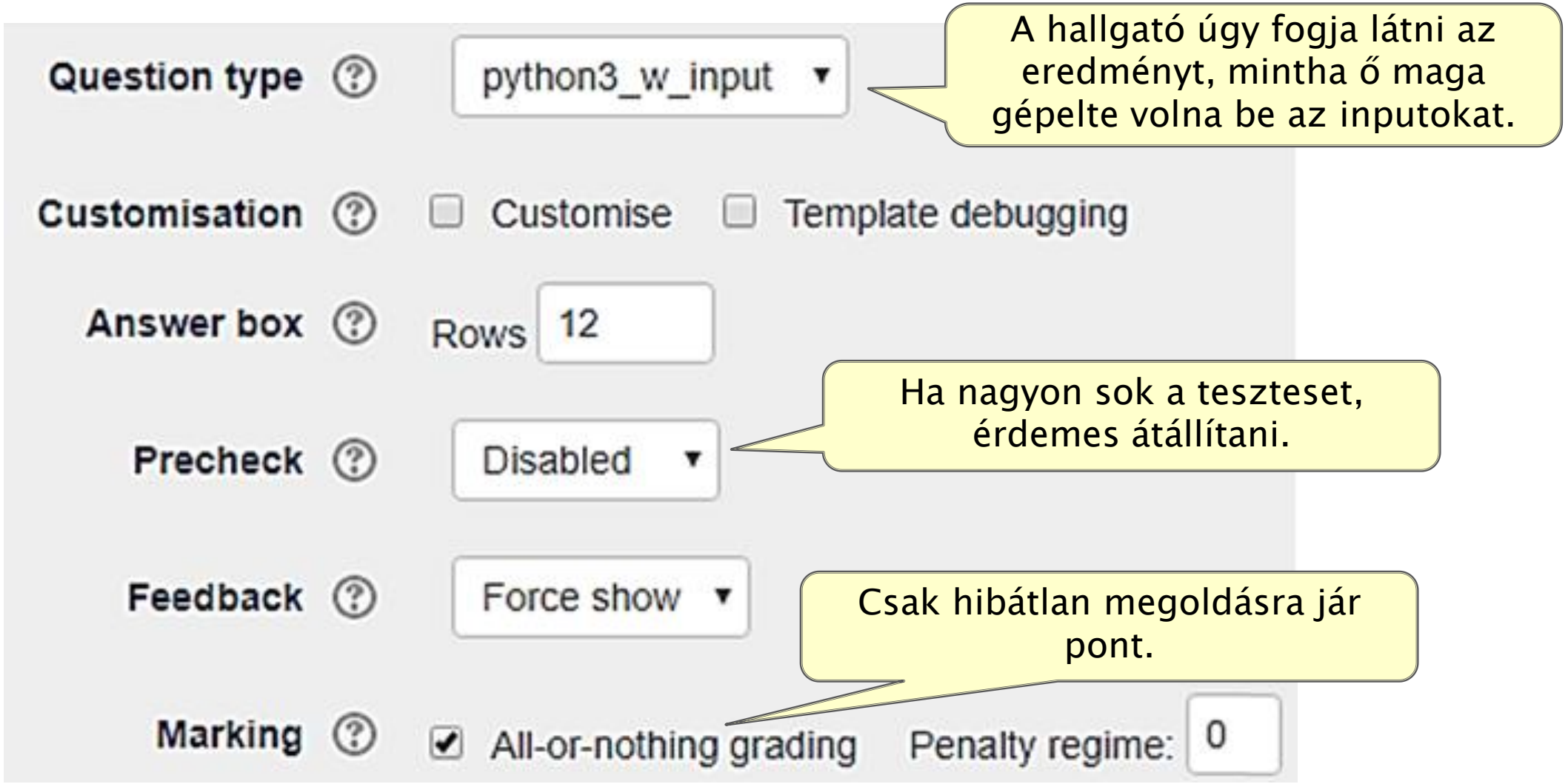
http://coderunner.org.nz/pluginfile.php/145/mod_page/content/2/coderunnerarchitecture.png

1. példa: Python szkript

- Az input-output kezelése a feladat része

- Sablon: alapértelmezett
- Kiértékelés: Pontos egyezés szerint (alapértelmezés)
- Csak tökéletes megoldásra jár pont

Beállítások



A kérdés szövege

Előre formázot

Írj programot, mely meghatározza az alábbi képlet eredményét és megjeleníti azt:

$$S_n = \frac{n(n+1)}{2}$$

Az *n* értékét (pozitív egész szám) a felhasználótól kérd be.

A beolvasásnál megjelenítendő szöveg:

```
Adj meg egy pozitív egész számot:<szóköz>
```

Az eredmény megjelenítésénél megjelenítendő szöveg:

```
Sn=<eredmény>
```

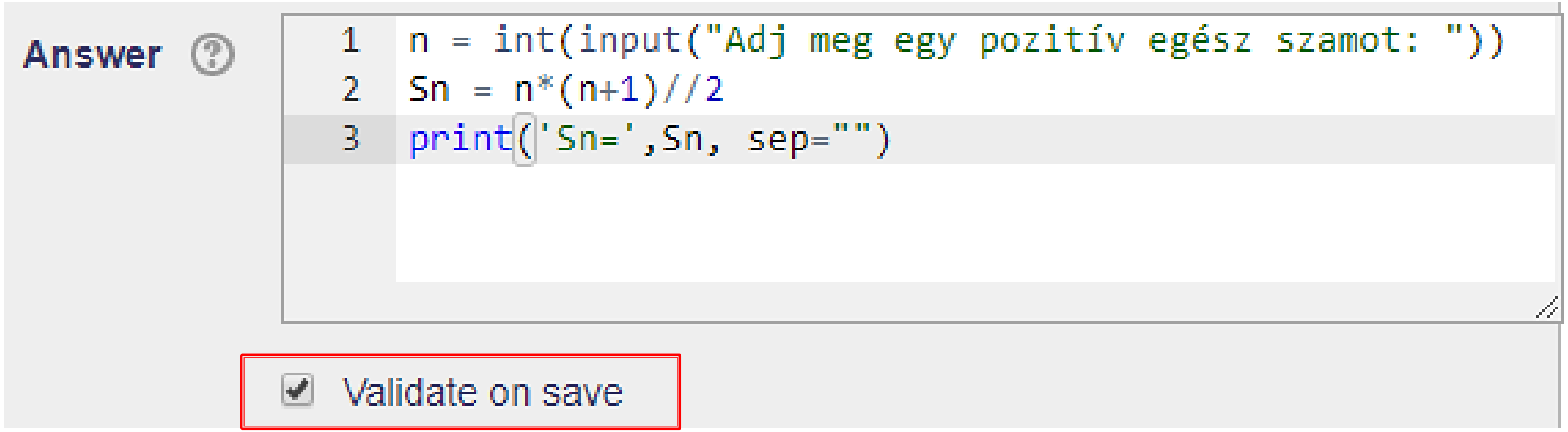
Az eredmény egész számként (tizedesjegyek nélkül) jelenjen meg.

Legyenek pontos előírások az üzenetek formázására.

A tesztesetek megadása

A helyes válasz megadása

- Ha megadjuk:
  - automatikusan generálható lesz az elvárt válasz a tesztekhez
  - a diákok a teszt beállításától függően láthatják ezt a megoldást



Az elvárt válaszok generáltatása

- Feladat mentése (a hiányzó válaszok miatt, sikertelen lesz)
- A felugró táblázatban kattintsunk a << jelekre
- Mentsük el. Kész a feladat.

Failed 5 test(s)

Test	Expected	Got
Test case 1	Adj meg egy pozitív egész szamot: 3 Sn=6	Adj meg egy pozitív egész szamot: 3 Sn=6 <<
Test case 2		Adj meg egy pozitív egész szamot: 10 Sn=55 <<

Click on the << button to replace the expected output of this testcase with actual output.
For more detailed information, save the question with 'Validate on save' unchecked and test manually

A hallgató által látott feladatkiírás

Írj programot, mely meghatározza az alábbi képlet eredményét és megjeleníti azt:

$$S_n = \frac{n(n+1)}{2}$$

Az *n* értékét (pozitív egész szám) a felhasználótól kérd be.

A beolvasásnál megjelenítendő szöveg:

```
Adj meg egy pozitív egész számot:<szóköz>
```

Az eredmény megjelenítésénél megjelenítendő szöveg:

```
Sn=<eredmény>
```

Az eredmény egész számként (tizedesjegyek nélkül) jelenjen meg.

For example:

Input	Result
3	Adj meg egy pozitív egész számot: 3 Sn=6

A feladat szövege

A példaként megjelölt tesztesetekből automatikusan generált táblázat

Egy megoldási kísérlet

- A hallgató a válasz mezőbe beírhatja a megoldását
- Az Ellenőrzés gombra kattintva kérheti a kiértékelést

Answer: (penalty regime: 0 %)

```
1  n = int(input("Adj meg egy pozitív egész számot: "))
2  sn = 2*n
3  print("Sn=", sn, sep='')
```

Ellenőrzés

A visszajelzés

	Input	Expected	Got	
✓	3	Adj meg egy pozitív egész számot: 3 Sn=6	Adj meg egy pozitív egész számot: 3 Sn=6	✓
✗	10	Adj meg egy pozitív egész számot: 10 Sn=55	Adj meg egy pozitív egész számot: 10 Sn=20	✗

Some hidden test cases failed, too.

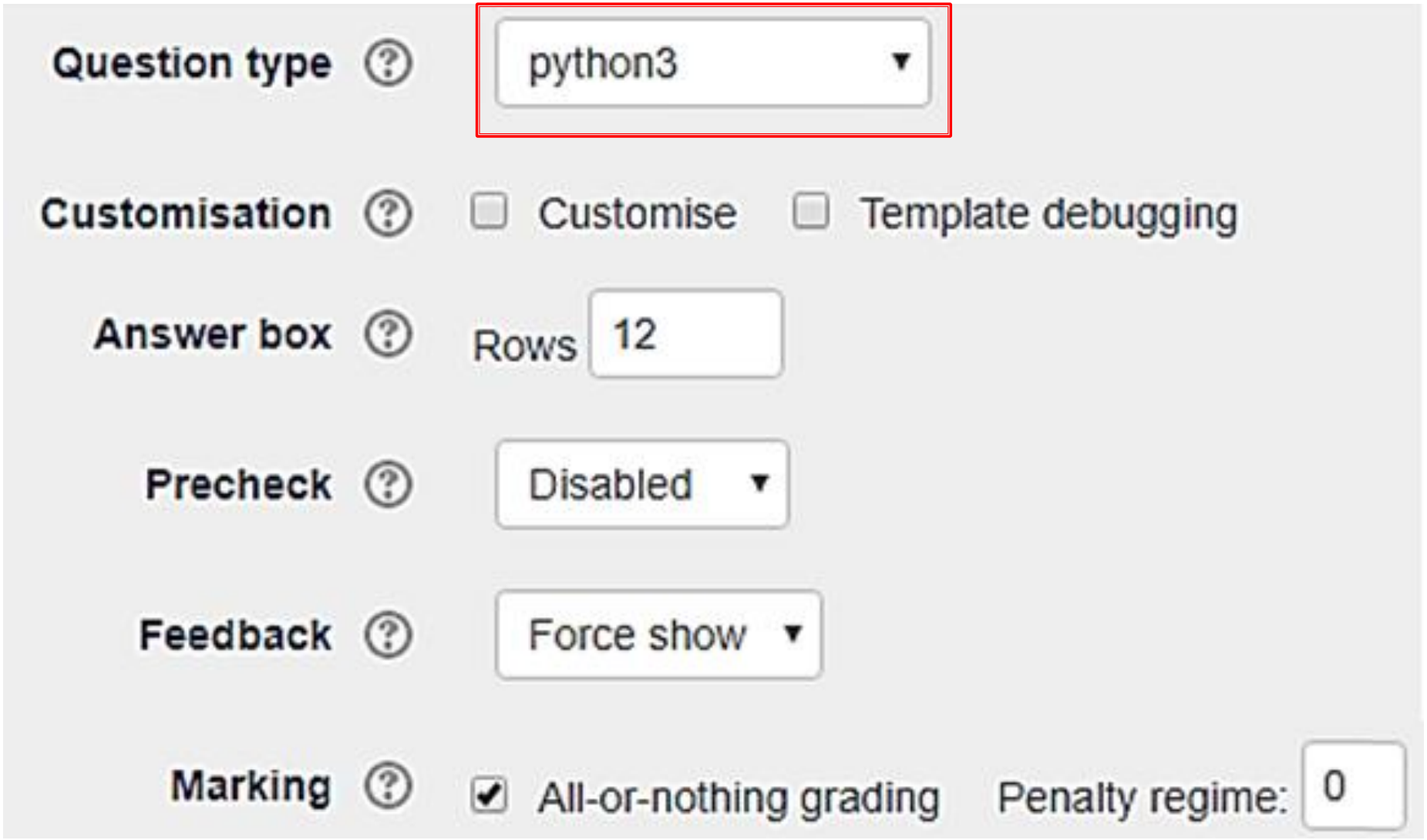
Your code must pass all tests to earn any marks. Try again.

Show differences

értesül arról, hogy legalább egy rejtett tesztesetre is hibás a válasza

kiemelheti az elvárt és a kapott kimenet közti különbséget

2. példa: Python függvény



A kérdés szövege

Bekezdés

Írj függvényt, mely egy egész számot vár paraméterként (*n*) és visszaadja az alábbi képlet eredményét egész számként:

$$S_n = \frac{n(n+1)}{2}$$

Útvonal: p

Függvény: a tesztesetek megadása

Test case 1

```
n = 3
r = sn(n)
print(r)
```

Csak az *n* értékét kell átírni a különböző tesztesetek előállításához.

Standard Input

Nem vár el adatbekérést a feladat, ezért ez üresen marad.

Expected output

A korábban látott módon generáltathatjuk, ha nem akarjuk kézzel megadni.

A válasz és a kiegészítendő válasz

Answer

```
1  def sn(n):
2      # az eredmény int típusú lesz
3      return (n*(n+1))//2
4
```

Validate on save

Answer box preload

```
1  def sn(n):
2      # a függvény int típusú számmal térjen vissza
```

A válasz egy részét előre megadható.

A hallgató által látott visszajelzés

- A beadott megoldás:

```
def sn(n):
    return n*(n+1)/2
```

- Kiértékelés módja:"Exact match"

Megj.: A "Nearly exact match" is ugyanezt eredményezné.

	Test	Expected	Got	
✗	n = 3 r = sn(n) print(r)	6	6.0	✗
✗	n = 5 r = sn(n) print(r)	15	15.0	✗
✗	n = 42 r = sn(n) print(r)	903	903.0	✗

Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Show differences

Mikor változtassunk a sablonon?

- Válasz mező tartalmának elemzésére
  - „Old meg while ciklussal...”,
  - „A megoldás során ne használd a math modult…”
- A kiértékelés megváltoztatására
  - részpontok osztása
  - **közelítő megoldások elfogadása abs(exp – got)<eps**
- Tesztestek generálására
  - véletlen értékekkel való tesztelés
  - …

A legegyszerűbb sablonok

A válasz mező tartalma szövegszerűen behelyettesítődik

Tesztesetenkénti

```
{{ STUDENT_ANSWER }}

{{ TEST.testcode }}
```

Egy konkrét teszteset kerül be a programba

Kombinált

```
{{ STUDENT_ANSWER }}

SEPARATOR = "#<ab@17943918#@>#"

{% for TEST in TESTCASES %}
{{ TEST.testcode }}
{% if not loop.last %}
print(SEPARATOR)
{% endif %}
{% endfor %}
```

Az összes teszteset bekerül a programba

A generált programok

A válasz

```
def sn(n):
    return n*(n+1)/2
```

Run 12

```
def sn(n):
...
```

Run 2

Run 1

```
def sn(n):
    return n*(n+1)/2

n = 3
r = sn(n)
print(r)
```

Tesztesetenként egy-egy program

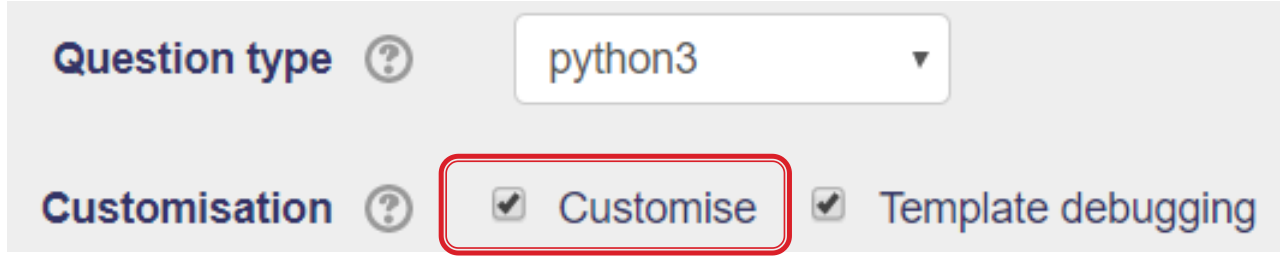
Run 1

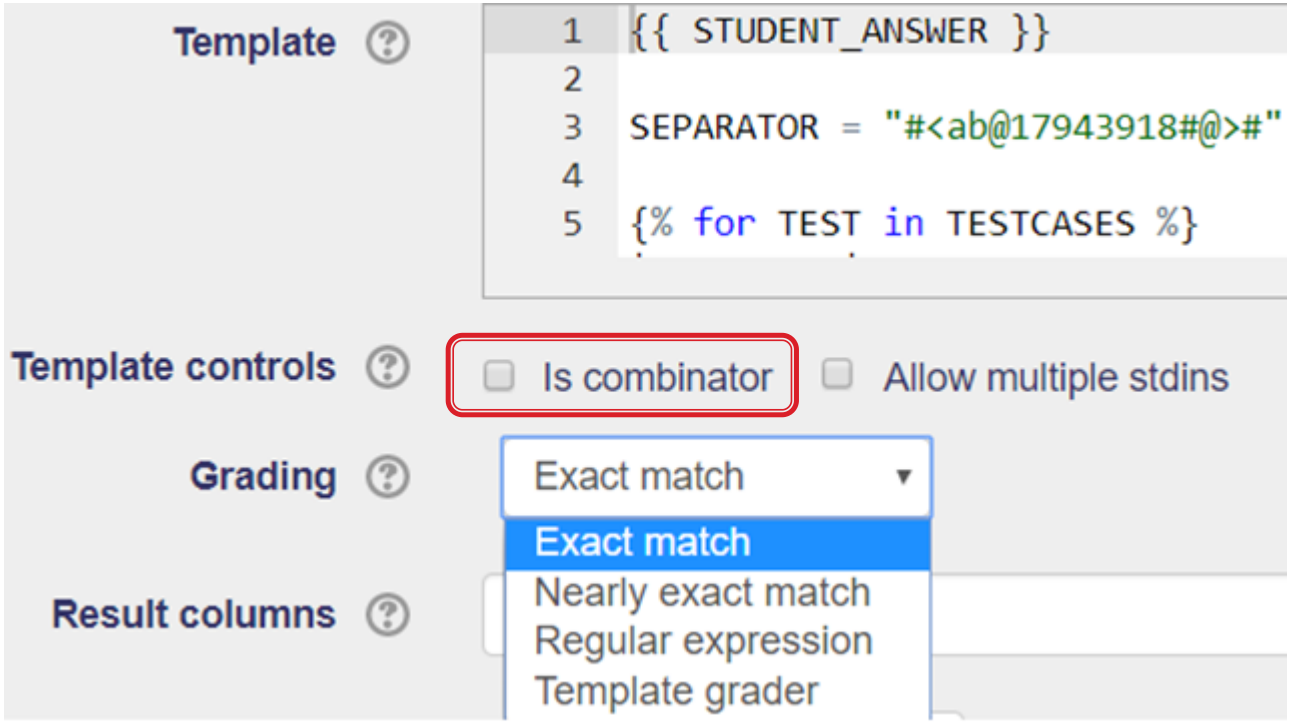
```
def sn(n):
    return n*(n+1)/2

SEPARATOR = "#<ab@17943918#@>#"

n = 3
r = sn(n)
print(r)
print(SEPARATOR)
n = 12
r = sn(n)
print(r)
print(SEPARATOR)
...
```

Egyetlen program

3. példa: „Old meg while ciklussal...”

- A kérdés típusnál jelöljük be a "Customise" mezőt
- Tesztesetenkénti sablon
- Pontos egyezés szerinti értékelés
- A szokott módon adjuk meg a
  - feladatleírást
  - teszteseteket
  - válasz mezők tartalmát





A beállítások és az új sablon

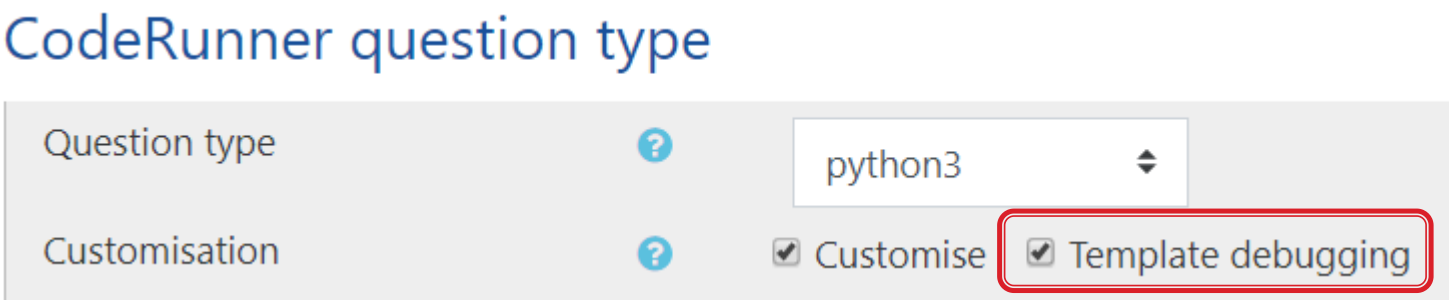
- Írjuk felül az eredeti template-et:

```
__student_answer__ = """{{ STUDENT_ANSWER | e('py') }}"""

if " while " not in __student_answer__:
    print("A feladatot while ciklussal kell megoldanod!")
    exit(0)
```

Írjuk felül az eredeti template-et:

```
{{ STUDENT_ANSWER }}
{{ TEST.testcode }}
```

- Debug céllal kérhetjük a sablon alapján generált kód megtekintését:



A visszajelzés

A beadott megoldás:

```
def sn(n):
    return n*(n+1)//2
```

	Test	Expected	Got	
✗	n = 3 r = sn(n) print(r)	6	A feladatot while ciklussal kell megoldanod!	✗
✗	n = 5 r = sn(n) print(r)	15	A feladatot while ciklussal kell megoldanod!	✗
✗	n = 42 r = sn(n) print(r)	903	A feladatot while ciklussal kell megoldanod!	✗

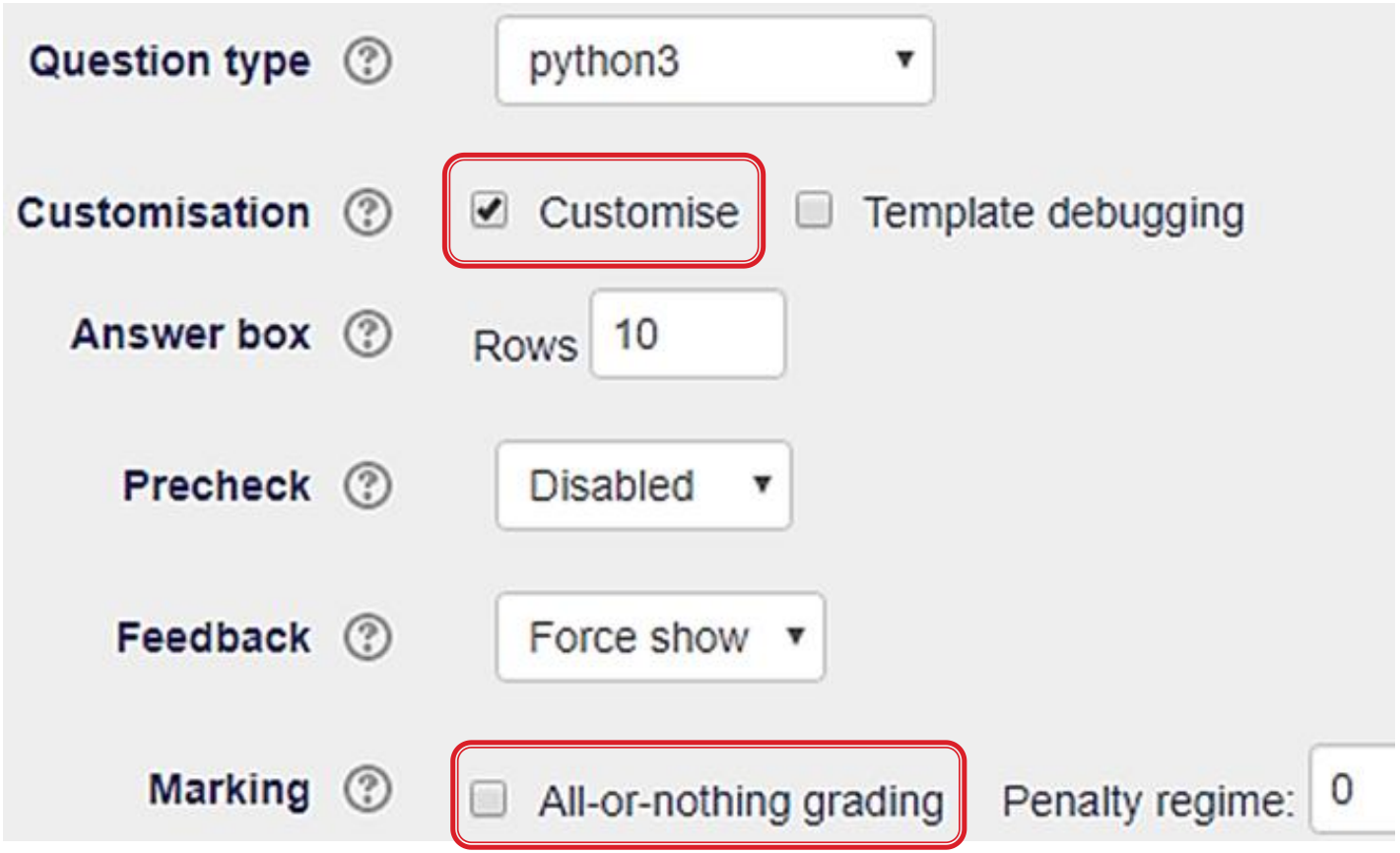
Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Show differences

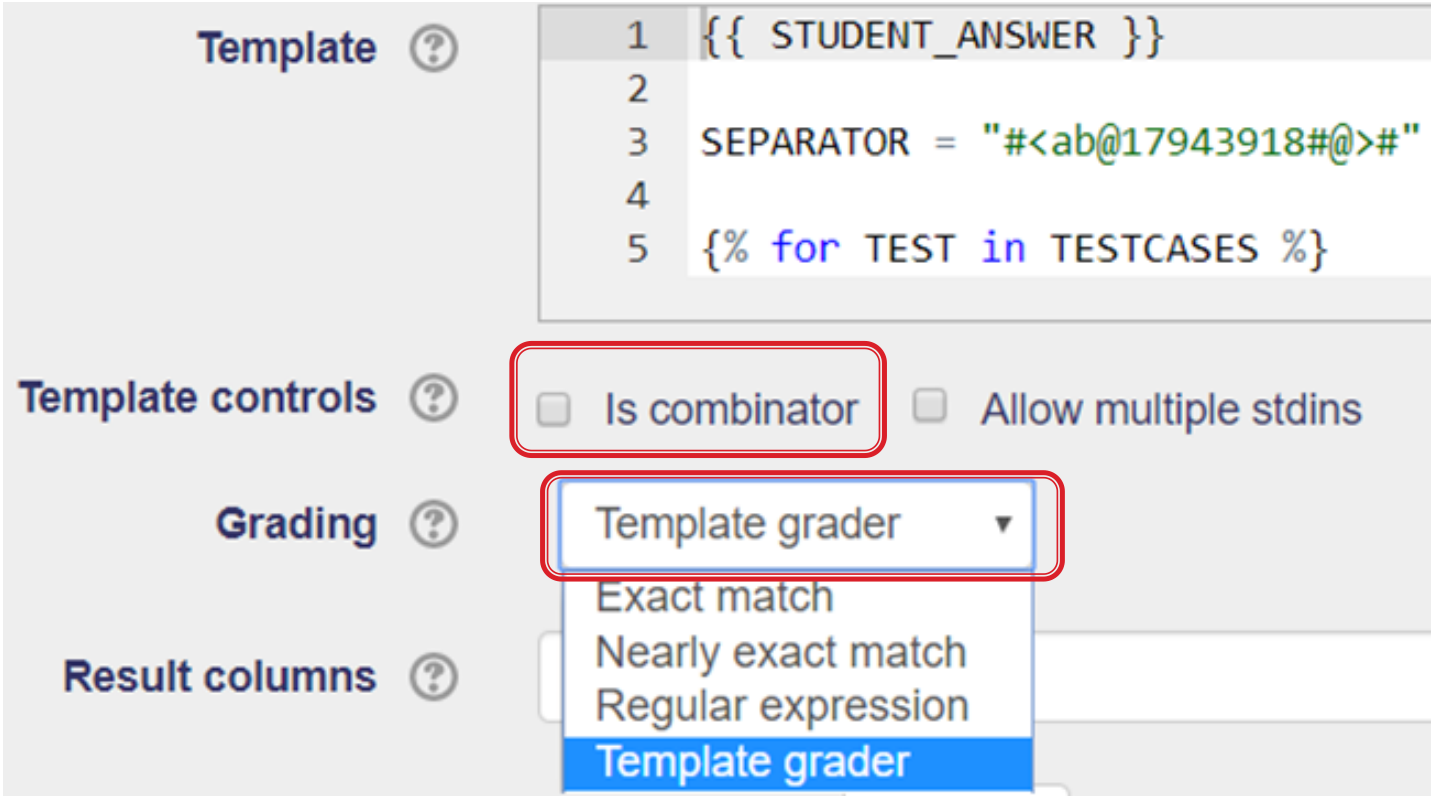
4. példa: részpontok osztása

- A kérdés típusnál jelöljük be a "Customise" mezőt
- Vegyük ki a pipát az "All-or-nothing" grading elől

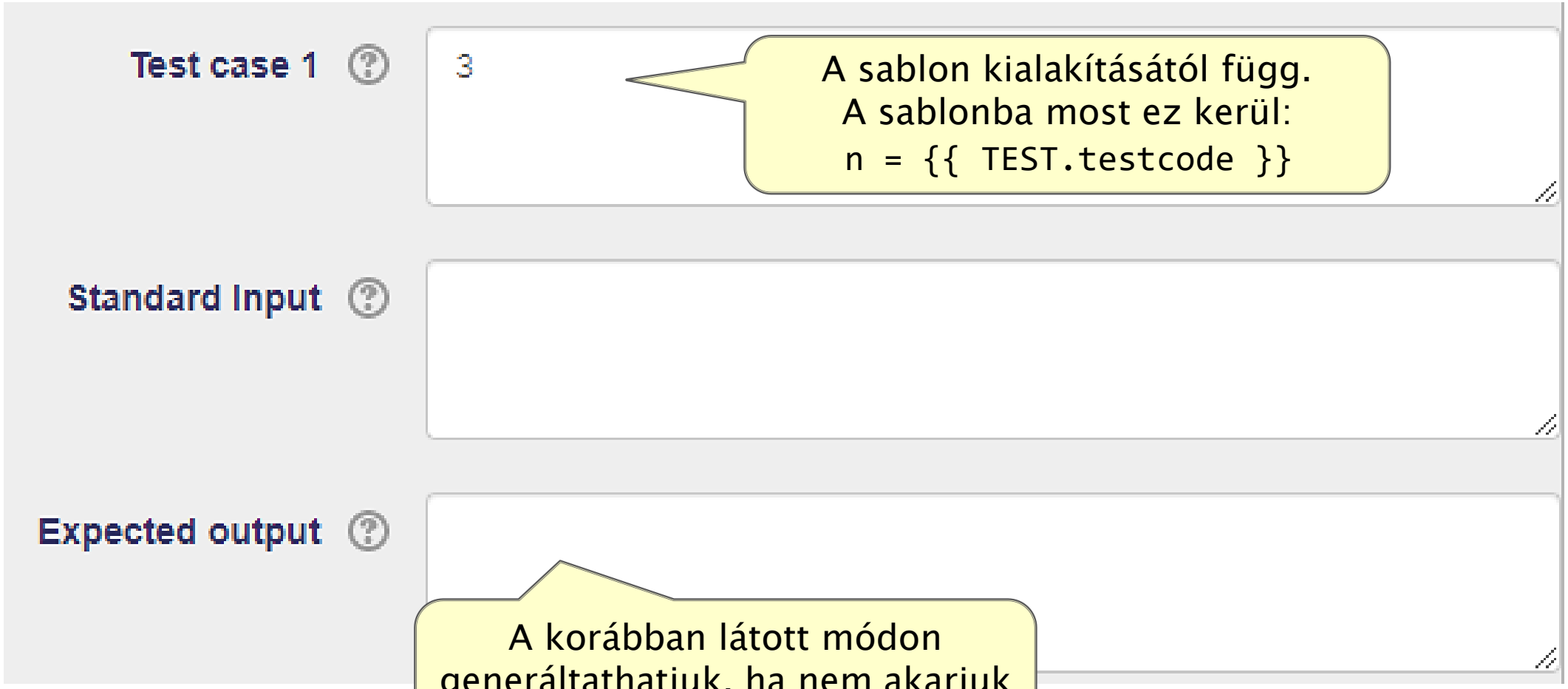


A sablon beállítása

- Tesztesetenkénti sablont fogunk használni
  - IsCombinator: False
- Az általunk megadott sablon végzi majd a pontozást:
  - Template grader



A tesztesetek megadása



A pontozást végző sablon

- Meghívja a hallgató által írt függvényt
- Összeveti az eredményt és az elvárt eredményt
- Kiírja a visszajelzést tartalmazó táblázat egy sorát a standard outputra.

- A táblázat szokásos oszlopai:
  - Iscorrect: helyes, helytelen, részben helyes
  - Input: a tesztadat
  - Got: a diák megoldásának eredménye
  - Expected: az elvárt eredmény
  - Fraction: a tesztesre kapott pont

A sablon

A saját sablon a diák számára nehezen értelmezhető hibákhoz vezethet. (pl.: a diák függvénye nem ad vissza értéket, vagy rossz típusú értéket ad)

Elkerülése: try-except

```
import json

{{ STUDENT_ANSWER }}

n = {{ TEST.testcode }}
got = sn(n)
exp = {{ TEST.expected }}

mark = 0
if got == exp and type(got) is int:
    mark = 1
elif abs(got - exp) < 0.00001:
    mark = 0.8

print(json.dumps({'iscorrect' : mark, 'input': n, 'got': str(got),
                                    'expected': str(exp), 'fraction': mark}))
```

A saját pontozónk

Az aktuális tesztesetre adott pontszám

1: helyes
0: helytelen
0<mark<1: részben helyes

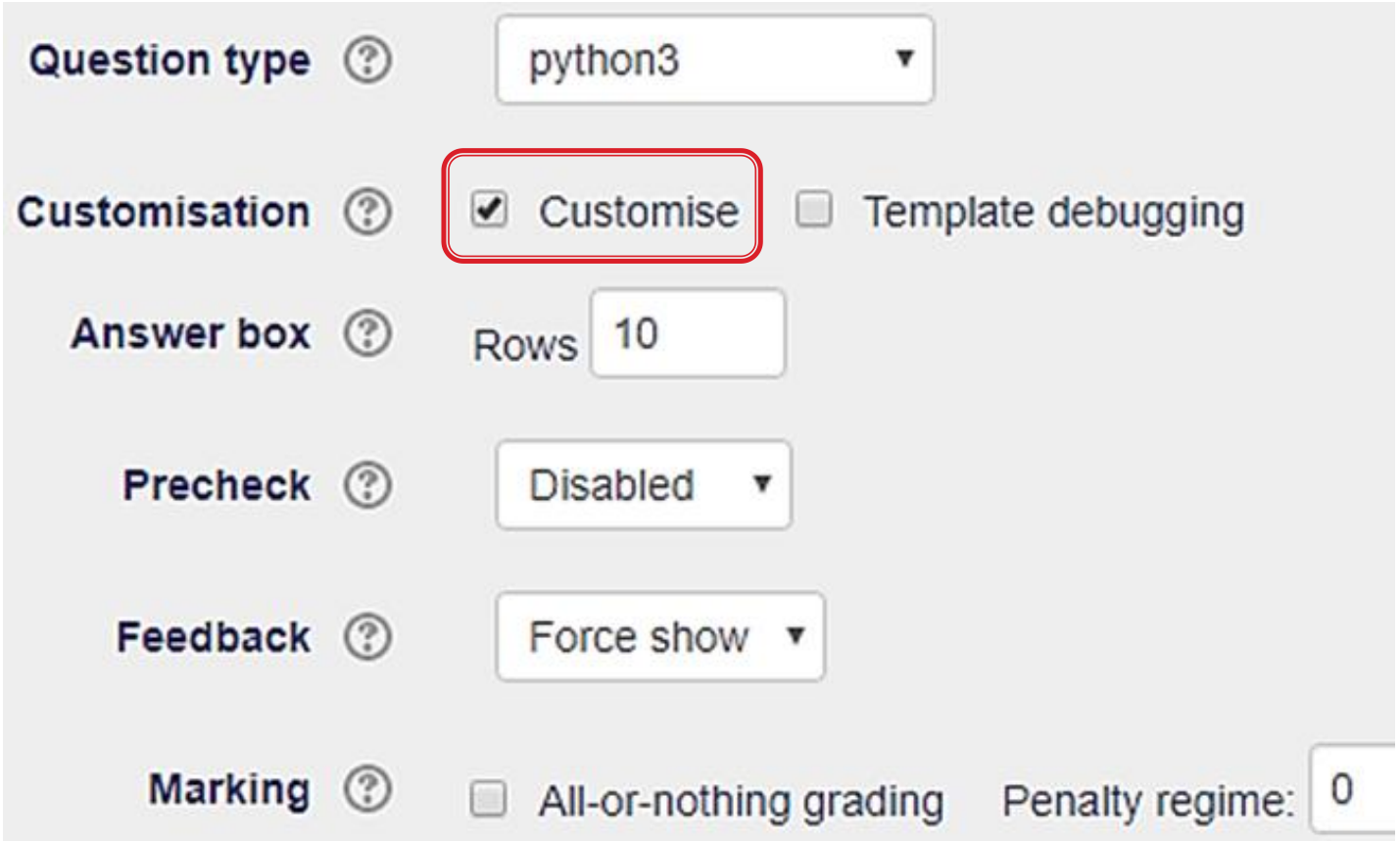
A visszajelzés

- A feladat szövege kikötötte, hogy a válasz legyen egész típusú
- Csak részpont jár

	Input	Expected	Got	Mark	
✓	3	6	6.0	0.8	✓
✓	5	15	15.0	0.8	✓
✓	42	903	903.0	0.8	✓
✓	2	3	3.0	0.8	✓

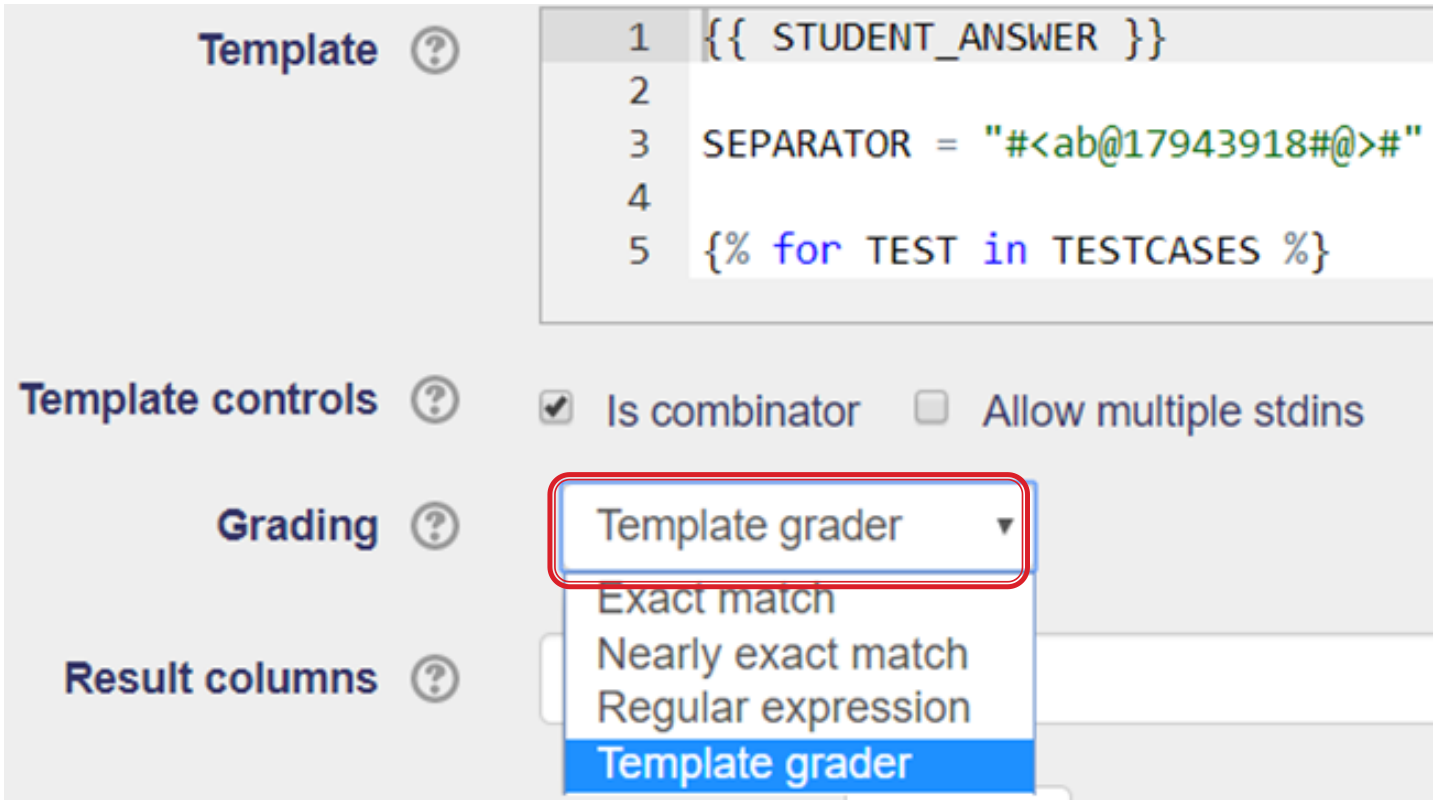
5. példa: Random teszt, saját kiértékelő

- A kérdés típusnál jelöljük be a "Customise" mezőt
- Adjuk meg a feladatleírást
- Töltsük ki a "preload answer" és "answer" mezőket, ha akarjuk
- Teszteseteken NE adjunk meg



A sablon beállításai

- Az általunk megadott sablon végzi majd a pontozást:
  - Template grader
- A sablont majd le kell cserélnünk



A sablon

- Létrehozza a visszajelzést tartalmazó táblázat fejlécét
- Generálja a tesztesetet
- Minden tesztesetre:
  - kiértékeli a hallgató megoldását
  - kiszámolja a helyes eredményt
  - összeveti az eredményt és az elvárt eredményt
  - hozzáfűzi az adott tesztesetre vonatkozó sort a táblázathoz
- Kiírja az **összeredményt** és **a visszajelző táblázatot** a standard outputra

A sablon

```
import json
import random

header = ['iscorrect', 'got', 'expected', 'input']
rows = [header]

{{ STUDENT_ANSWER }}

NUM_TEST = 100
mark = 0
for i in range(NUM_TEST):
    n = random.randint(0, 1000)
    exp = n*(n+1)//2
    got = sn(n)
    iscorrect = got == exp
    if iscorrect: mark += 1
    row = [iscorrect, str(got), str(exp), 'n: '+str(n)]
    rows.append(row)

print(json.dumps({'fraction': mark / NUM_TEST, 'testresults': rows}))
```

a táblázat fejlécének létrehozása

A kapott és az elvárt eredmény hasonlítása

A pontozás

a tesztesetre vonatkozó sor

a visszajelző táblázat kiírása