PYTHON IN CLOUD (ANGOL NYELVŰ)

2. forduló

CAMBRIDGE MOBILE TELEMATICS

A kategória támogatója: Cambridge Mobile Telematics (CMT)

Ismertető a feladathoz

Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Badge-ket** a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/40:00 **Elért pontszám:** 0/30

1. feladat 0/5 pont

The meaning of life

We have a Class of Life:

```
class Life:
    meaning: str

    def __init__(self):
        self.meaning = str(random.randint(0, 99))

    def get_meaning_of_life(self):
        if int(self.meaning) < 0 or int(self.meaning) > 99:
            raise ReachedEndOfTheGalaxy(f'Fatal: {self.meaning}')
        return self.meaning

    def get_special_meaning_of_life(self, name: str = 'Douglas'):
        if name.lower() == 'douglas':
            return '42'
        else:
            self.get_meaning_of_life()

class ReachedEndOfTheGalaxy(Exception):
    pass
```

You have to select all of the valid code snippets to pass the following test case.

```
class TestLife(TestCase):
    def test_i(self):
        life = Life()

        # pass the valid snippets here
```

Válaszok

- [x]
```
meaning = life.get_special_meaning_of_life()
        assert meaning == '42'
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
meaning = life.get_meaning_of_life()
        assert meaning in range(0, 99)
```

- [ ]
```
meaning = life.get_special_meaning_of_life('Bob')
```

```
        assert int(meaning) in range(0, 99)
```

- [x]
```
    meaning = life.get_special_meaning_of_life('Arnold')
        assert meaning == None
```
Ez a válasz helyes, de nem jelölted meg.

- [x]
```
    life.meaning = '101'
        with self.assertRaises(ReachedEndOfTheGalaxy) as e:
            meaning = int(life.get_meaning_of_life())
        assert e.exception.args[0] == 'Fatal: 101'
```
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

life.**get_meaning_of_life**() will return **str**, first have to cast it to int

life.**get_special_meaning_of_life**('Bob') will return **None**, because the return statement is missing in the else block of the function

2. feladat 0/10 pont

Exceptions

At CMT we have a class to request users via HTTP. The skeleton of a simplified implementation looks like this:

```
@dataclass(frozen=True)
class Config:
 backend_url: str = os.environ.get("BACKEND_URL") or "http://localhost"

class User:
 _url: str

 def __init__(self, url: str) -> None:
   self._url = url

 def _get(self, path: str, params: Optional[dict] = None) -> Any:
   try:
     response = requests.get(f"{self._url}/{path}", params=params)
     response.raise_for_status()
     return response.json()
   except requests.exceptions.HTTPError:
     raise UserError("Request got a HTTPError")
   except requests.exceptions.Timeout:
     raise UserError("Request got a Timeout")
   except requests.exceptions.SSLError:
     raise UserError("Request got a SSLError")

 def get_user_list(self) -> List[Dict[any, any]]:
   return self._get("users")

class UserError(Exception):
 pass
```

We created a skeleton test case, where we are looking for valid asserts. The Test case skeleton looks like this:

```
@mock.patch("user.requests.get")
   def test_i(self, get_mock):
       get_mock.return_value = {'users': ['John', 'Bruce', 'Arnold', 'Clark']}
       get_mock.side_effect = [
           requests.exceptions.Timeout(),
           requests.exceptions.HTTPError(),
           requests.exceptions.SSLError(),
       ]
       errors = []
       with self.assertRaises(UserError) as exc:
           User(Config().backend_url).get_user_list()
       errors.append(exc)
       with self.assertRaises(UserError) as exc:
           User(Config().backend_url).get_user_list()
       errors.append(exc)
       get_mock.reset_mock()
       with self.assertRaises(UserError) as exc:
           User(Config().backend_url).get_user_list()

       # valid asserts
```

Select the valid asserts from the following list.

Válaszok

- [ ]
```
    assert type(errors[0].exception) == Timeout
```

- [x]
```
assert 'SSL' in exc.exception.args[0]
```
Ez a válasz helyes, de nem jelölted meg.

- [x]
```
assert type(errors[1].exception) != HTTPError
```
Ez a válasz helyes, de nem jelölted meg.

- [x]
```
assert type(errors[0].exception) != Timeout
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
assert type(errors[2].exception) == UserError
```

- [x]
```
assert type(errors[0].exception) == UserError
```
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

i.) simply not true

ii.) that's true, latest exception in exc note: reset_mock do **not** delete the side_effects

iii.) true

iv.) true (opposite of the i.))

v.) out of range, error length only 2, the last exec is not appended

vi.) true

3. feladat 0/15 pont

Recursion 1

Here we have two enigmatic but likely important functions, f and g. We can get the outcomes of f(x) for small x, but are having trouble with larger ones with the current code.

```
def f(x):
    if x < 2:
        return 1
    return f(x - 1) + g(x + 1)

def g(x):
    if x < 2:
        return 1
    return x + f(x - 2)
```

Specifically what should f(1000) return?

Válasz

A helyes válasz:

26787715179656683023710626226500045264035120292638340186093759709258776278123403062329959470392396453189866822938828670629678632142307851089961443936746437009836417

Magyarázat

```
# Example solution (there are many ways to do this)
# fmemo = {}
# def f(x):
#     if x < 2:
#         return 1
#     if x not in fmemo:
#         fmemo[x] = f(x-1) + g(x+1)
#     return fmemo[x]
# gmemo = {}
# def g(x):
#     if x < 2:
#         return 1
#     if x not in gmemo:
#         gmemo[x] = x + f(x-2)
#     return gmemo[x]
# for i in range(1000):
#
# using the memo to save us from the depth limit (and having to change it with sys)
#     f(i)
# print(f(1000))
```

Megjelenés

Világos