PYTHON BACKEND FEJLESZTÉS

2. forduló

CAMBRIDGE MOBILE TELEMATICS

A kategória támogatója: Cambridge Mobile Telematics (TrueMotion)

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Fontos információk

Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat

A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.

Olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a versenyszabályzatban találsz!

2nd round - Testing, SQLAlchemy

Felhasznált idő: 01:49/15:00 **Elért pontszám:** 0/6

1. feladat 0/3 pont

Given the SystemUnderTest and a Validator, select which TestValidatorN class constitutes a stub, a mock, a dummy or a spy.

```
from abc import ABC, abstractclassmethod

class Validator(ABC):

    @abstractclassmethod
```

```
    def validate(name: str) -> bool:
        pass
class SystemUnderTest():
    def __init__(self, validator: Validator) -> None:
        self._validator = validator

    def publicApiDoesNotValidate(self) -> str:
        return 'OK'
    def publicApiDoesValidate(self, input_str: str) -> str:
        return 'OK' if self._validator.validate(input_str) else 'NOT OK'
class A(Validator):
    def validate(self, name: str) -> bool:
        raise Exception('not yet implemented')

class B(Validator):
    def validate(self, name: str) -> bool:
        return True
class C(Validator):
    def __init__(self, doesValidate: bool = True) -> None:
        super().__init__()
        self.doesValidate = doesValidate
        self.wasValidationCalled = False
    def validate(self, name: str) -> bool:
        self.wasValidationCalled = True
        return self.doesValidate
class D(Validator):
    def __init__(self, doesValidate: bool = True, validationMaxCalls = 1) -> None:
        super().__init__()
        self.doesValidate = doesValidate
```

```
        self.validationMaxCalls = validationMaxCalls

        self.wasValidationCalled = 0

    def validate(self, name: str) -> bool:

        self.wasValidationCalled += 1

        assert self.wasValidationCalled <= self.validationMaxCalls, "Validation was called more than i

        return self.doesValidate
```

Please put class names A, B, C and D in the order of Dummy, Mock, Spy, Stub. The answer should be like D, C, B, A.

Válaszok

A helyes válasz:
A, D, C, B
ADCB
A D C B
A,D,C,B
A, D C B

Magyarázat

Stub — used for providing the tested code with "indirect input".

Mock object — used for verifying "indirect output" of the tested code, by first defining the expectations before the tested code is executed.

Spy — used for verifying "indirect output" of the tested code, by asserting the expectations afterwards, without having defined the expectations before the tested code is executed. It helps in recording information about the indirect object created.

Dummy object — used when a parameter is needed for the tested method but without actually needing to use the parameter.

2. feladat 0/3 pont

Create a trip SQLAlchemy model, with an ID, LAT and LON coordinates.

Válasz

```
    class Trip(Base):
```

```
    __tablename__ = 'trips'

    id = Column(String)

    lat = Column(Float)

    lon = Column(Float)
```

```
class  Trip(Base):

    table = 'trips'

    id = Column(String)

    lat = Column(Float)

    lon = Column(Float)
```

```
class Trip(Base):

    table = 'trips'

    id: str

    lat: float

    lon: float
```

```
@alchemy('trips')

class Trip:

    id: String

    lat: Double

    lon: Double
```

Magyarázat

Table name must be defined as __tablename__

Column values must be instances of sqlachemy column objects

There is no @alchemy decorator

© 2022 Human Priority Kft.

KÉSZÍTETTE

Megjelenés

Világos