TESZTAUTOMATIZÁLÁS

2. forduló

<epam>

A kategória támogatója: EPAM

RENDELKEZÉSRE ÁLLÓ IDŐ:

30:00

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!

Felhasznált idő: 01:32/30:00 **Elért pontszám:** 0/20

1. feladat 0/2 pont

Milyen metódusra van szükség a ___ helyén, hogy csak a páros számokat írassuk ki?

```
const numbers = [1,2,3,4,5,6,7,8,9,10];
numbers.____(num => num % 2 === 0).forEach(num => console.log(num));
```

Válasz

- ◯ filter
- ◯ map

- some
- any

Magyarázat

map - a tömb elemeinek a módosítására használjuk.

some - arra használandó, hogy megnézzük, van-e a tömbben olyan elem, amit keresünk a predikátum alapján.

any - nincs ilyen metódus.

2. feladat 0/2 pont

Mi történik ha lefuttatjuk az alábbi kódot:

```
function run() {
  {
   var hello = "hello"
   let world = "world";
   console.log(hello, world);
  }
  console.log(hello, world);
}
run();
```

Válasz

- Kiírja kétszer egymás után, hogy "helloworld".
- Kiírja egyszer, hogy "helloworld", majd a futás leáll az alábbi hibával: ReferenceError: hello is not defined
- Kiírja egyszer, hogy "helloworld", majd a futás leáll az alábbi hibával: ReferenceError: world is not defined
- Kiírja kétszer egy sorba, hogy "helloworld".

Magyarázat

Az egyik változó let kulcsszóval, a másik pedig var kulcsszóval van deklarálva. A fő különbség a kettő között a 'scope' megtartása. Mivel a let figyelembe veszi, hogy a "run" metódusban nyitottunk egy új scope-ot, ezért a scope-on kívül már nem él a változónk. A var ezzel ellentétben nem veszi figyelembe a scope-okat, így azon kívül is elérhető, emiatt a referencia hiba csak a "world" változóra vonatkozik.

3. feladat 0/2 pont

Adott az alábbi mappaszerkezet a következő fájlokkal:

```
- features
 - search.feature
 - steps
  - steps.py
```

a search.feature tartalma:

```
Feature: Verification of the search functionality
  Scenario: The search functionality should find the proper results
    Given the Search page is opened
    When the 'test' text is added to the search bar
    And the search is executed
    Then the list should contain the 'test' text
```

a steps.py fájl **üres**.

A Behave (v1.2.6) Python eszközt felhasználva, ha lefuttatjuk a teszteket (alapértelmezett beállításokkal), az alábbiak közül melyik eredményt kapjuk a lépésekre?

Válasz

- 4 steps passed, 0 failed, 0 skipped, 0 undefined
- 0 steps passed, 0 failed, 0 skipped, 4 undefined
- 0 steps passed, 4 failed, 0 skipped, 0 undefined
- 0 steps passed, 0 failed, 4 skipped, 0 undefined

Magyarázat

Mivel a feature fájlban használt lépéseknek nincsen definíciójuk, így undefined-nak jelenik meg.

4. feladat 0/2 pont

Adott az alábbi Python kód a 'test.py' fájlban:

```
import unittest

class TestString(unittest.TestCase):
  def test_upper(self):
    self.assertEqual('test upper'.upper(), 'TEST UPPER')
```

```
  def test_isupper(self):
    self.assertTrue('TEST'.isupper())
    self.assertFalse('Test'.isupper())

if __name__ == '__main__':
   unittest.main()
```

A 'unittest' modult használva, hogyan tudjuk lefuttatni csak a 'test_upper' függvényt Python (v3.8) esetén?

Válasz

- `python -m unittest test/TestString/test_upper`
- `python -m unittest test.TestString.test_upper`
- `python -m unittest TestString.test_upper`
- `python -m test/TestString/test_upper`

Magyarázat

A 'unittest' modult használva úgy tudunk kifejezetten egy teszt függvényt futtatni, ha annak megadjuk az elérési útját úgy mint: fájlnév.osztálynév.függvénynév

5. feladat 0/2 pont

10 db egész számot szeretnénk az alábbi osztályban lévő listába gyűjteni. 1-től 10-ig, sorrendben.

Tekintsük az alábbi osztályban a for ciklust (Java 11):

```
public class MyCode {
  private List<Integer> numbers = new ArrayList<>();

  public void collectNumbers(){
   for (int i = 1; i < 11; i++) {
          numbers.add(i);
    }
 }
}
```

A felsoroltak közül mely kódrészletekkel lehetne kiváltani magát a for ciklust?

Ugyanazt a végeredményt elérve.

Válaszok

- [ ] `Stream.of(1, 11).forEach(numbers::add);`
- [x] `IntStream.range(1, 11).forEach(numbers::add);`
- [x] `numbers = IntStream.range(1, 11).boxed().collect(Collectors.toList());`
- [ ] `numbers.stream().collect(Collectors.summingInt(Integer::intValue));`

Magyarázat

IntStreammel összegyűjthetjük a megadott input paraméterek közti értékeket. Az első input paraméter inclusive, az utolsó exclusive.

A Stream.of() viszont csak a fixen megadott paramétereken iterál végig.

Az utolsó válaszlehetőség: a numbers listában már benne lévő értékeken iterál végig és adja össze őket.

6. feladat 0/2 pont

Tegyük fel, hogy Selenium WebDriverrel futtatunk automatizált UI teszteket, Java 8-as verzió használatával.

Tekintsük az alábbi kódrészletet:

```
@Component
public class WebDriverFactory {

    private WebDriver webDriver;

        public WebDriver getWebDriver(final String browserName) {
    if (Objects.isNull(webDriver)) {
      switch (browserName) {
      case "chrome":
        webDriver = new ChromeDriver();
        break;
      case "firefox":
        webDriver = new FirefoxDriver();
```

```
          break;
        default:
          throw new RuntimeException("Unsupported driver for browser=" + browserName);
        }
      }

      return webDriver;
    }

   public void closeWebDriver(){
      webDriver.quit();
      webDriver = null;
    }
  }

  public class Hooks {

    private WebDriver webDriver;

   @Autowired
    private WebDriverFactory factory;

    @Before
    public void beforeScenario() {
      webDriver = webdriverFactory.getWebDriver("chrome");
    }

    @After
    public void afterScenario() {
     webDriverFactory.closeWebDriver();
    }
  }
```

Mi történik, ha ezt a kódrészletet hozzáadjuk a Selenium-os tesztekhez, és azokat futtatjuk?

Válasz

- Minden UI scenario előtt létrejön egy chromedriver instance, amit a tesztek majd fel tudnak használni.
  Minden scenario után bezárul a chromedriver
- Minden UI scenario előtt létrejön egy chrome és egy firefox driver instance, amit a tesztek majd fel tudnak használni.
  Minden scenario után bezárul mindkettő driver.
- A tesztek futása előtt egyszer létrejön egy chromedriver instance, amit a tesztek majd fel tudnak használni.
  Amikor az összes teszt végzett, bezárul a chromedriver.
- A kód nem fut le, ugyanis az Object.isNull nem használható 9-esnél régebbi Java verzióval.

Magyarázat

A Hook osztályban @Before és @After annotációkat használunk. Ezek minden scenario előtt és után is futnak.

A WebDriver setupolásnál switch-et használunk. Csak egy ágba fogunk belefutni. A metódus hívásnál "chrome" stringet adtunk meg, így chromedrivert kapunk.

Az Objects.isNull Java 7-es verziótól kezdve érhető el.

7. feladat 0/2 pont

Az alábbi HTTP metódusok közül a HTTP Protocol ajánlása alapján melyik használható egy már meglévő entitás részleges módosítására?

Válasz

- ◯ PATCH
- ◯ POST
- ◯ PUT
- ◯ UPDATE

Magyarázat

A felsorolt metódusok közül a HTTP Protocol ajánlása alapján a PATCH metódus használandó egy meglévő entitás részleges módosítására; a POST új entitás hozzáadására; a PUT a teljes entitás lecserélésére; UPDATE metódus pedig nincs.

8. feladat 0/4 pont

Melyik reguláris kifejezés használható a következő minta teljes illesztésére: **Dcccc.** ?

Ahol a **D** egy számjegy és ezzel a számmal kezdődik a string, **c** pedig egy tetszőleges kis betű, amelyből 4 van egymás után (**c** karakter lehet különböző kis betű) a string-ben, és egy **.** karakterrel végződik. Tehát összesen 6 karakter hosszú.

"3ssss." illeszkedik, de a "3ssss.a" már nem.

Válasz

- ◯ `^[1-9][a-z]{4}\.`
- ◯ `^\d[A-Z]{4}\.$`
- ◯ `^\d\w{4}\.`

- [ ] `^\d\w{4}\.$`

Magyarázat

Az első és harmadik reguláris kifejezés a záró $ karakter hiánya miatt rögtön kiesik. Ezekre hosszabb kifejezések rész stringjei is illeszkednek.

A második reguláris kifejezés pedig 4db nagy betűt vár a kis betűk helyett. Ezért nem lesz jó ez sem.

A negyedik válasz a helyes, mivel az egy számmal kezdődő és négy kis betűvel folytatódó stringet vár ami után egy . karakter zárja a stringet.

9. feladat 0/2 pont

Az alábbi állítások közül melyik igaz?

Válasz

- [ ] Tesztelés magasabb szinten költségesebb, viszont az ezen a szinten lévő tesztek fenntartásának kisebb költsége van.
- [ ] Az alacsonyabb szinten lévő tesztek futási ideje nagyobb, az itt talált hibák javításának költsége alacsonyabb.
- [ ] Az alacsonyabb szinten lévő tesztek futási ideje kevesebb, az itt talált hibák javításának költsége magas.
- [ ] Tesztelés magasabb szinten költségesebb és az itt talált hibák javításának költsége általában magasabb.

Magyarázat

Magasabb tesztszinten nő a tesztek költsége, futási ideje, a talált hibák javításának költsége és a tesztek fenntartásának költsége is, ezért a megadottak közül csak a negyedik válasz helyes teljesen.