TESZTAUTOMATIZÁLÁS

3. forduló

<epam>

A kategória támogatója: EPAM

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

Felhasznált idő: 00:00/30:00 **Elért pontszám:** 0/65

1. feladat 0/5 pont

Találd meg a **hamis** állítást a REST-ről!

Válasz

- ○ A REST HTTP headereket használ, hogy metaadatot továbbítson.
- ○ A REST állapotmentes kommunikációt biztosít.
- ○ A RESTben az URL azt mondja meg, hol található meg egy resource(erőforrás), szóval az URL önmaga nem egy resource.
- ● A REST csak XML és JSON formátumot támogat.
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A REST egy architekturális koncepció, amelynek a középpontjában az "erőforrások" állnak, ezekre az erőforrásokra a "referencia" egy URL, ami megmondja, hogy "hol" található a resource(erőforrás). Az egyik alappillére az, hogy a kommunikáció

állapotmentes, vagyis a kérést fogadó szerver nem tárol adatot a kliensről. A kommunikációban a HTTP headerjeit használja, hogy metaadatokat tároljon.

A REST nem ad semmilyen megkötést az erőforrás reprezentációjával kapcsolatban, így az lehet XML, JSON vagy akár bármilyen más formátumú is.

2. feladat 0/10 pont

A következő megoldások közül melyekkel tudjuk kiprintelni a "Hello, world!" szöveget megfordítva?

Válaszok

- [x]
```
print(''.join(reversed("Hello, world!")))
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
print(str(reversed("Hello, world!")))
```

- [x]
```
print(*"Hello, world!"[::-1], sep='')
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
print(reversed("Hello, world!"))
```

Magyarázat

A reversed önmagában egy iterátort ad vissza, amiből közvetlenül nem nyerhető ki a tartalma, csak ha végig iterálunk rajta. Emiatt a közvetlen printelés és az str konverziós printelés nem az elvárt eredményt adja.

A string osztály join metódusa pont erre való, összefűzi az argumentumaként megadott iterable objektum elemeit a megadott stringgel, esetünkben egy üres stringgel, így a kívánt megoldást kapjuk.

A python string egyben egy immutable lista is, így a slicing értelmezett rajta, ezért a reversedhez hasonlóan a [::-1] slicinggal is meg tudjuk fordítani egy string tartalmát. Az eredmény azonban egy lista, amit a print függvény megfelelő paraméterezésével ugyanúgy a kívánt eredménynek megfelelően tudunk kiírni, ezért ez a megoldás is jó.

3. feladat 0/10 pont

Az alábbi Generikust használó kódok közül melyek helyesek?

Válaszok

- [ ]
```
class <T> MyName {
    T obj;
        ...
}
```

- [x]
```
public <T extends Number> void myName(T element)
{
        ...
}
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
public void myName(List<? extend Number> numbers) {
    ...
}
```

- [x]
```
public void myName(List<? super Number> numbers) {
    ...
}
```
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Az osztályra tett típusparamétert az osztálynév után kell feltüntetni: class MyName<T>

A típusparamétert lehet kiterjeszteni az "extends" és a "super" kulcsszóval.

A '?' karaktert használhatjuk helyettesítő karakterként a generikusoknál.

4. feladat 0/10 pont

Egy autó riasztórendszernek teszteléséhez a követelmények a következők:

Adott az aktivált riasztórendszer (AKTÍV)

Amikor vagy megemelik az autót (MEGEMELT), vagy elfordítják a kulcsot a zárban (KULCS)

Akkor a riasztó riaszt

A kód logikája a következőképpen néz ki:

```
IF ((MEGEMELT OR  KULCS) AND AKTÍV) THEN
        Riassz!
ELSE
        Ne riassz!
ENDIF
```

Teszt bemeneti értékek:

1. MEGEMELT + KULCS + AKTÍV

2. MEGEMELT + KULCS + nem AKTÍV

3. MEGEMELT + nem KULCS + AKTÍV

4. MEGEMELT + nem KULCS + nem AKTÍV

5. nem MEGEMELT + KULCS + AKTÍV

6. nem MEGEMELT + KULCS + nem AKTÍV

7. nem MEGEMELT + nem KULCS + AKTÍV

8. nem MEGEMELT + nem KULCS + nem AKTÍV

Feltételezve, hogy nincs rövidzár-kiértékelés, melyik tesztbemeneti értékkészlet szükséges a teljes MC/DC lefedettséghez?

Válasz

- ○ 1, 3, 8
- ○ 2, 6, 8
- ● 3, 4, 5, 7
  Ez a válasz helyes, de nem jelölted meg.
- ○ 1, 5, 7, 8

Magyarázat

1, 3, 8 és 2, 6, 8 helytelen. Három független atomi feltétel esetén négy teszt szükséges a teljes MC/DC lefedettség eléréséhez

3, 4, 5, 7 helyes.

A 3. és 7. eset azt mutatja, hogy a “MEGEMELT” függetlenül is befolyásolhatja a döntés eredményét.

Az 5. és 7. eset azt mutatja, hogy a “KULCS” függetlenül is befolyásolhatja a döntés eredményét.

A 3. és 4. eset azt mutatja, hogy az “AKTÍV” függetlenül is befolyásolhatja a döntés eredményét.

1, 5, 7, 8 helytelen. Az 1-es számú eset kombinálva a másik három bármelyikével (5, 7, 8) nem tudja kimutatni, hogy egyetlen feltétel függetlenül is befolyásolhatja a döntés eredményét.

5. feladat 0/15 pont

Adott a következő DOM struktúra. Mely CSS selector(okk)al tudjuk kiválasztani azt a "span"-t, amely "cg-field-choice-text" osztállyal rendelkezik, és nincs letiltva a hozzá tartozó checkbox?

```
<body>
        <div>
                <div>
                        <input name="Pear" type="checkbox" data-testid="filter-tree-item-checkbox" disabled="
                        <span class="cg-field-choice-box"></span>
                        <span class="cg-field-choice-text">Pear</span>
                </div>
                <div>
                        <input name="Plumb" type="checkbox" data-testid="filter-tree-item-checkbox" disabled=
                        <span class="cg-field-choice-box"></span>
                        <span class="cg-field-choice-text">Plumb</span>
                </div>
                <div>
                        <input name="Apple" type="checkbox" data-testid="filter-tree-item-checkbox">
                        <span class="cg-field-choice-box"></span>
                        <span class="cg-field-choice-text">Apple</span>
                </div>
                <div>
                        <input name="Apple" type="checkbox" data-testid="filter-tree-item-checkbox" disabled=
                        <span class="cg-field-choice-box"></span>
                        <span class="cg-field-choice-text">Apple</span>
                </div>
```

```
        </div>
</body>
```

Válasz

- span.cg-field-choice-text:not([disabled])
- input[data-testid='filter-tree-item-checkbox']:not([disabled]) > span.cg-field-choice-text
- input[data-testid='filter-tree-item-checkbox']:not([disabled]) ~ span.cg-field-choice-text
  Ez a válasz helyes, de nem jelölted meg.
- input[data-testid='filter-tree-item-checkbox']:not([disabled]) + span.cg-field-choice-text

Magyarázat

A span.cg-field-choice-text:not([disabled]) selector az összes cg-field-choice-text osztállyal rendelkező elemet megfogja, ezért helytelen.

A '>' jellel elválasztott selector akkor lenne megfelelő, ha a checkbox input a szülő eleme lenne a span-nak.

A '+' jellel elválasztott selector, akkor lenne megfelelő, ha közvetlen a checkbox elem után szereplne a span class="cg-field-choice-text" és nem ékelődne közzé a másik span class="cg-field-choice-box".

A '~' jellel elválasztott selector a helyes. A selector első része megfogja az egyetlen elérhető (nem letiltott elemet), majd a ~ jellel a vele egyszinten lévő, de utána következő elme(eke)t fogja meg.

6. feladat 0/15 pont

Egy gépjármű eladással foglalkozó cég szeretné kialakítani adatbázisát jól olvasható, rendszerezhető formátumban. Az adatokat már objektumként kapják, de nehezen olvasható formában:

```
const cars = {
    "daewoo/tosca/rendszam": "AAA-555",
    "daewoo/tosca/tipus": "1 generáció",
    "opel/corsa/rendszam": "CCC-333",
    "opel/corsa/tipus": "A"
};
```

Ezért szeretnék átalakítani a kapott objektumot a következő formátumra:

```
{
  daewoo: { tosca: { rendszam: 'AAA-555', tipus: '1 generáció' } },
  opel: { corsa: { rendszam: 'CCC-333', tipus: 'A' } }
}
```

Melyik kód segítségével tudják ezt elérni?

Válasz

-
```
const cars = { A kapott objektum. }
const newObject = {};

function createObject(obj, keyPath, value) {
    const lastKeyIndex = keyPath.length - 1;
    for (let i = 0; i < lastKeyIndex; ++i) {
```

```
        const key = keyPath[i];
        if (!(key in obj)) {
            obj[key] = {};
        }
        obj = obj[key];
    }
    obj[keyPath[lastKeyIndex]] = value;
}

for (const car in cars) {
    const array = car.split("/");
    createObject(newObject, array, cars[car]);
}
```

Ez a válasz helyes, de nem jelölted meg.

```
const cars = { A kapott objektum. };
const newObject = {};

for (const car in cars) {
    const array = car.split("/");
    newObject[array] = cars[car];
}
```

```
const cars = { A kapott objektum. };
const newObject = {};

function createObject(obj, keyPath, value) {
    const lastKeyIndex = keyPath.length - 1;
    for (let i = 0; i < lastKeyIndex; ++i) {
        const key = keyPath[i];
        if (!(key in obj)) {
            obj[key] = {};
        }
        obj = obj[key];
        obj[keyPath[lastKeyIndex]] = value;
    }
}

for (const car in cars) {
    const array = car.split("/");
    createObject(newObject, array, cars[car]);
}
```

```
const cars = { A kapott objektum. };
let newObject = {};

for (const car in cars) {
    const array = car.split("/");
    const lastKeyIndex = array.length - 1;
    for (let i = 0; i < lastKeyIndex; ++i) {
        const key = array[i];
        if (!(key in newObject)) {
            newObject[key] = {};
        }
        newObject = newObject[key];
    }
    newObject[array[lastKeyIndex]] = cars[car];
}
```

Magyarázat

A csak egy for ciklust tartalmazó kód, egy új objektumot fog készíteni melyben a kulcsokban nem" /" jelek lesznek hanem "," karakterek. Ez a válasz helytelen.

A két egymásba ágyazott for ciklust tartalmazó kód csak egy {"tipus" = "A"} objektumot hoz létre. Ez a válasz helytelen.

A két hasonló kódrészletet tartalmazó válaszok közül a helyes kód az amelyikben a value hozzáadás a for cikluson kivülre esik. Ez fogja eredményezni a kért objektumot. A másik kód amelyikben "value" érték hozzáadás a for cikluson belülre kerül egy ilyen objektumot eredményez

```
{
  daewoo: {
    rendszam: 'AAA-555',
    tosca: { rendszam: 'AAA-555', tipus: '1 generáció' },
    tipus: '1 generáció'
  },
  opel: {
    rendszam: 'CCC-333',
    corsa: { rendszam: 'CCC-333', tipus: 'A' },
    tipus: 'A'
  }
}
```