PYTHON BACKEND FEJLESZTÉS

6. forduló

CAMBRIDGE MOBILE TELEMATICS

A kategória támogatója: Cambridge Mobile Telematics (TrueMotion)

RENDELKEZÉSRE ÁLLÓ IDŐ:

25:00

Ismertető a feladathoz

Threading, Error handling

Felhasznált idő: 01:47/25:00 **Elért pontszám:** 0/16

1. feladat 0/8 pont

What is the maximum number of threads used in this example?

```
def do_nothing():
   time.sleep(5)
with ThreadPoolExecutor(max_workers=20) as executor:
   for i in range(100):
       executor.submit(do_nothing)
```

Válasz

- 1
- 2
- 20
- 21

- ◯ 22
- ◯ 100

Magyarázat

One main thread and at maximum 20 other threads created by the main thread.

2. feladat 0/8 pont

Find and fix the errors in the code below.

```
import sqlite3
class NotFoundError(Exception):
    pass
class NotAuthorizedError(Exception):
    pass
def blog_lst_to_json(item):
    return {
        'id': item[0],
        'published': item[1],
        'title': item[2],
        'content': item[3],
        'public': bool(item[4])
    }
def fetch_blog(id: str):
    try:
        # connect to the database
        con = sqlite3.connect('application.db')
        cur = con.cursor()
        # execute the query and fetch the data
        cur.execute(f"SELECT * FROM blogs where id=?", [id])
        result = cur.fetchone()
        # return the result or raise an error
        if result is None:
            raise NotFoundError(f'Unable to find blog with id {id}.')
        data = blog_lst_to_json(result)
        if not data['public']:
            raise NotAuthorizedError(f'Access denied to blog with id {id}.')
        # close the connection
        con.close()
        return data
    except sqlite3.OperationalError as e:
        print(e)
        raise NotFoundError(f'Unable to find blog with id {id}.')
    except Exception:
        return {}
```

Válaszok

```
def fetch_blog(id: str):
         data = blog_lst_to_json(result)
         if not data['public']:
             raise NotAuthorizedError(f'You are not allowed to access blog with id {id}.')
-
-        # close the connection
-        con.close()
-
+
     return data
     except sqlite3.OperationalError as e:
         print(e)
         raise NotFoundError(f'Unable to find blog with id {id}.')
     except Exception:
         return {}
-
+    finally:
+        # close the connection
+        con.close()
```

(Answer A)

```
def fetch_blog(id: str):
         data = blog_lst_to_json(result)
         if not data['public']:
             raise NotAuthorizedError(f'You are not allowed to access blog with id {id}.')
-
-        # close the connection
-        con.close()
-
+
     return data
     except sqlite3.OperationalError as e:
         print(e)
         raise NotFoundError(f'Unable to find blog with id {id}.')
-    except Exception:
-        return {}
-
+    finally:
+        # close the connection
+        con.close()
```

(Answer B)

```
def fetch_blog(id: str):
     try:
         # connect to the database
-        con = sqlite3.connect('application.db')
-        cur = con.cursor()
```

```
-
-        # execute the query and fetch the data
-        cur.execute("SELECT * FROM blogs where id=?", [id])
-        result = cur.fetchone()
-
-        # return the result or raise an error
-        if result is None:
-            raise NotFoundError(f'Unable to find blog with id {id}.')
-
-        data = blog_lst_to_json(result)
-        if not data['public']:
-            raise NotAuthorizedError(f'You are not allowed to access blog with id {id}.')
-
-        # close the connection
-        con.close()
-
-        return data
+        with sqlite3.connect('application.db') as con:
+            cur = con.cursor()
+
+            # execute the query and fetch the data
+            cur.execute("SELECT * FROM blogs where id=?", [id])
+            result = cur.fetchone()
+
+            # return the result or raise an error
+            if result is None:
+                raise NotFoundError(f'Unable to find blog with id {id}.')
+
+            data = blog_lst_to_json(result)
+            if not data['public']:
+                raise NotAuthorizedError(f'You are not allowed to access blog with id {id}.')
+
+            return data
     except sqlite3.OperationalError as e:
         print(e)
         raise NotFoundError(f'Unable to find blog with id {id}.')
-    except Exception:
-        return {}
-
+    finally:
+        # close the connection
+        con.close()
```

(Answer C)

```
def fetch_blog(id: str):
    try:
        # connect to the database
-        con = sqlite3.connect('application.db')
-        cur = con.cursor()
-
-        # execute the query and fetch the data
```

```
-         cur.execute("SELECT * FROM blogs where id=?", [id])
-         result = cur.fetchone()
-
-         # return the result or raise an error
-         if result is None:
-             raise NotFoundError(f'Unable to find blog with id {id}.')
-
-         data = blog_lst_to_json(result)
-         if not data['public']:
-             raise NotAuthorizedError(f'You are not allowed to access blog with id {id}.')
```

(Answer D)

Magyarázat

Please be aware of the + and - notations!

A: A hides the exception

D: *NotFoundError* is hidden because of the global exception handler

Update (2021.12.07.): Answer C is considered to be correct, as well: variable "con" *will be defined* and multiple close() method calls is nop in case of sqlite3 driver. The code will run without problem.