LU09.A09: Unittests auflisten

Erstelle ein JSON-Array mit allen Unittests in einem Python-Projekt.

Aufgabe

Das Skript durchsucht ein Projekt nach Dateien, die Unittests enthalten. Die Dateinamen müssen `test_` oder `_test` enthalten. Zum Beispiel:

- **test_**main.py
- main**_test**.py

Innerhalb dieser Dateien sucht das Skript nach Funktionen, deren Bezeichner mit `test_` beginnt. Die Bezeichner dieser Funktionen werden in eine Liste geschrieben. Schlussendlich wird diese Liste sortiert und als JSON-Array ausgegeben.

Beispiel

```
[
    "test_figure_init_empty",
    "test_figure_add_shape",
    "test_figure_add_shape_multiple",
    "test_figure_take_shape",
    "test_figure_remove_shape_by_title",
    "test_figure_remove_shape_by_index",
    "test_figure_total_area_scale1",
    "test_figure_total_area_scale3",
    "test_figureshape_init_empty",
    "test_figureshape_set_figure",
    "test_figureshape_set_shape",
    "test_main"
]
```

Wieso Python und Bash?

Grundsätzlich könnte man die Aufgabe ausschliesslich mit Python lösen oder als reines Bash-Skript schreiben. Jede Sprache hat jedoch ihre Vorteile, die wir hier ausnutzen wollen:

- Dateien durchsuchen: Hier spielt Bash seine Stärke aus
- Arbeiten mit Collections: Dies lässt sich leichter in Python umsetzen.

Hinweise

Das Python-Skript kann nicht unter Windows ausgeführt werden, da wir Bash-Befehle nutzen.

Um auf das Windows-Laufwerk C zuzugreifen, verwendest du im WSL den Pfad `/mnt/c`.

Aufruf

1. Öffne das Windows Subsystem for Linux (WSL).
2. Wechsel in den Ordner, in dem Ihr Projekt gespeichert ist. Zum Beispiel: `cd „/mnt/c/BZZ/Python/m122-lu08-a02-listtest-ghwalin“`
3. Starte das Python Skript mit `python3 main.py ORDNER_PFAD`. Ersetze `ORDNER_PFAD` durch den Pfad zum gewünschten Projekt.

Vorgehen

Wie immer lohnt es sich, das Projekt schrittweise anzugehen. Jede Funktion innerhalb des Programms kann einzeln realisiert und getestet werden.

Da wir auf einer reinen Linux-Shell keinen Debugger haben, behelfen wir uns mit `print()`-Befehlen um die Zwischenresultate anzuzeigen.

1. Argument prüfen / übernehmen

In der Funktion `main()` prüfen wir das Argument „Pfad zum gewünschten Projekt“.

- Falls kein Argument übergeben wurde, beendet das Programm mit einer Fehlermeldung
- Wechsle in das angegebene Verzeichnis. Nutze dazu den Python-Befehl `os.chdir()`. [1)]
- Falls das Verzeichnis nicht existiert, wird das Skript mit einer Fehlermeldung beendet.

2. execute_bash

Die Funktion `execute_bash` soll die verschiedenen Bash-Befehle ausführen. Falls ein Fehler auftritt, wird eine Meldung ausgegeben und das Skript beendet.

- Parameter: Bash-Kommando (String)
- Return: Resultat-Objekt

3. find_test_modules

Diese Funktion sucht mit Hilfe eines **Bash**-Befehls im aktuellen Ordner nach allen Dateien, die PyTests enthalten können. Als Returnwert liefert die Funktion eine Liste aller Dateinamen.

Diese Dateien können anhand ihres Dateinamens erkannt werden:

- `test_?????.py`
- `?????_test.py`

- Ermittle zunächst den Bash-Befehl, um die gewünschten Dateien in einem Ordner zu finden.
- Teste den Befehl auf der Kommandozeile des WSL.

4. find_test_functions

Die Funktion durchsucht alle Dateien in einer Liste nach PyTest-Funktionen. Es sollen alle Zeilen gefunden werden, die `def test_` enthalten. \\Zum Beispiel:

- def test_main():
- def test_something(capsys):
- …

Die gefundenen Zeilen werden an `sanitize_function_name` übergeben. Speichere die „gereinigten“ Funktionsnamen in einer Liste

- Argument: Eine Liste mit Dateinamen
- Returnwert: Eine Liste mit allen PyTest-Funktionen.\\Zum Beispiel:
  - test_main
  - test_something

5. sanitize_function_name

Die Namen der Testfunktionen soll in dieser Funktion „gereinigt“ werden.

- Argument: Eine Codezeile mit einer Test-Funktion, z.B. `def test_something(capsys): # Test some things`
- Returnwert: Der Name der Test-Funktion, z.B. `test_something`

Es werden alle sonstigen Angaben in dieser Zeile entfernt:

- Das Schlüsselwort `def`.
- Leerzeichen am Anfang und am Ende.
- Die Liste mit den Parametern.
- Der Doppelpunkt
- Allfällige Zeilenkommentare

6. output_json

Die Funktion erzeugt ein JSON-Array mit den Namen aller Testfunktionen. Dieses JSON-Array wird in der Konsole ausgegeben.

- Argument: Liste mit Namen von Testfunktionen.
- Returnwert: None

Bonus-Auftrag

In der Vorlage ist bereits eine Funktion `select_project()` vorhanden. Diese öffnet ein Fenster, in dem der Benutzer das gewünschte Projekt auswählen kann.

Ergänze den Ablauf in `main()`, dass diese Funktion aufgerufen wird, falls kein Argument übergeben wurde.

Die Funktion `select_project` nutze ein Linux-Paket, dass in WSL nicht standardmässig installiert ist. Um es zu installieren, benötigst du die Befehle:

```
sudo apt update
sudo apt install zenity
```

M122-LU09

1)
Man könnte auch den Bash-Befehl `cd` verwenden, dieser ändert das Verzeichnis aber nur temporär. Dann müssten wir bei jedem Bash-Befehl das Verzeichnis explizit angeben.