Zadanie5. Napisanie programu oraz testów jednostkowych

a. Napisanie klasy o nazwie Pracownik(), której parametrami początkowymi będzie Imię, Nazwisko, wynagrodzenie.
   i. Klasa powinna zawierać 3 funkcje:
      1. Email - imie.nazwisko@testemail.com
      2. Pełna nazwa – Imię Nazwisko
      3. Podwyżka – wynagrodzenie * stala, gdzie stala = 2

Np. danymi inicjującymi są
Imie = Jan
Nazwisko = Kowalski
Wynagrodzenie = 2000

Funkcja email zwraca Jan.Kowalski@testemail.com
Funkcja Nazwa zwraca Jan Kowalski
Funkcja Podwyzka zwraca 4000

TestCase obsługuje również metodę setUp, która pomaga tworzyć zasoby na podstawie testu. Metoda setUp jest pomocna, gdy mamy wspólny zestaw kodu, który chcemy uruchomić przed każdym z testów. setUp pozwala umieścić cały ten kod przygotowawczy w jednym miejscu, zamiast powtarzać go w kółko dla każdego testu.

TestCase obsługuje odpowiednik metody setUp o nazwie tearDown. tearDown jest przydatne, jeśli na przykład musimy wyczyścić połączenia z bazą danych lub modyfikacje wprowadzone w systemie plików po zakończeniu każdego testu.

Przykład:

```
import unittest

class FishTank:
  def __init__(self):
    self.has_water = False

  def fill_with_water(self):
    self.has_water = True

class TestFishTank(unittest.TestCase):
  def setUp(self):
    self.fish_tank = FishTank()

  def test_fish_tank_empty_by_default(self):
    self.assertFalse(self.fish_tank.has_water)

  def test_fish_tank_can_be_filled(self):
```

```
        self.fish_tank.fill_with_water()
        self.assertTrue(self.fish_tank.has_water)
```

FishTank.has_water jest początkowo ustawiona na False, ale można ją ustawić na True, wywołując FishTank.fill_with_water (). Podklasa TestCase TestFishTank definiuje metodę o nazwie setUp, która tworzy instancję nowej instancji FishTank i przypisuje ją do self.fish_tank.

Ponieważ setUp jest uruchamiany przed każdą indywidualną metodą testową, tworzona jest nowa instancja FishTank zarówno dla test_fish_tank_empty_by_default, jak i test_fish_tank_can_be_filled. test_fish_tank_empty_by_default sprawdza, czy has_water zaczyna się jako fałsz. test_fish_tank_can_be_filled sprawdza, czy parametr has_water jest ustawiony na True po wywołaniu funkcji fill_with_water ().

Zadanie6.

Napisanie funkcji, której celem jest zwracanie informacji czy klub piłkarski posiada trenera.
Plus napisanie testów jednostkowych (zastosować setUP).

Zadanie7

Napisanie funkcji, która stworzy plik Kluby.txt i zapisane zostaną w nim Manchester City, Bayern (w metodzie setUP). Dodatkowo będzie inna klasa, która będzie kasowała ten plik (metoda tearDown). Plus napisanie testów jednostkowych

Zadanie8

Napisanie skryptu, który utworzy bazę danych z pewnymi danymi początkowymi, jeśli jej nie ma, wraz z kilkoma funkcjami, które pozwolą dodawać, usuwać, aktualizować wiersze. Skrypt nazwijmy artDB.py

1. Stworzenie tabeli KLUBY z kolumnami nazwa (text), trener (text), kraj (text), liczba_pilkarzy (int), najlepszy_zawodnik (text), sponsor_glowny (text)
2. Funkcje insert, delete wybrany klub, update wybrany klub, select wybrany klub
3. Napisanie testów jednostkowych (używamy setUp, tearDown)

Przykład:

```
import os
import sqlite3

def create_database():
    conn = sqlite3.connect("mydatabase.db")
    cursor = conn.cursor()
    # tworzenie tabeli
    cursor.execute("""CREATE TABLE albums
                        (title text, artist text)
                    """)
    # insert
    cursor.execute("INSERT INTO albums VALUES "
```

```
                "('ONE', 'U2')")
    # zapisanie danych do bazy
    conn.commit()
def select_all_albums(artist):
    conn = sqlite3.connect("mydatabase.db")
    cursor = conn.cursor()
    sql = "SELECT * FROM albums WHERE artist=?"
    cursor.execute(sql, [(artist)])
    result = cursor.fetchall()
    cursor.close()
    conn.close()
    return result
if __name__ == '__main__':

    if not os.path.exists("mydatabase.db"):
        create_database()
    print(select_all_albums('U2'))
```

Zadanie9 Napisanie testu jednostkowego poprawnego zalogowania się użytkownika.

Kroki:

1. Importujemy unittest, requests, json
2. Funkcja gl_url, która zwraca https://reqres.in
3. Funkcja test_valid_login()
    a. url sklada się z gl_url + /api/login/
    b. Dane do poprawnego zalogowania się:

```
{
   "email": "eve.holt@reqres.in",
   "password": "cityslicka"
}
```

RESPONSE

```
{

   "token": "QpwL5tke4Pnpja7X4"

}
```

    c. Zmienne
        i. response, gdzie zostanie wywołany post (requests.post)
        ii. token – json.loads(response.text)
    d. Test porównujący status_code poprawnego zalogowania się oraz token