Юдкін

Етап 2-3(а саме лише пункт створення Классів Бази данних та Таблиці,

і створення юніт тестів, тобто лише 1 етап)

Так виглядає класс Бд

Тестувати будемо заготовлену бд

class DB:  
 def \_\_init\_\_(self, db\_name):  
 self.name = db\_name  
  
 def create\_db(self):  
 link = f'C:/Users/Max/ITLab1/database/main/database/{self.name}.json'  
 data = {  
 "name": self.name,  
 "tables": {}  
 }  
 json\_object = json.dumps(data, indent=4)  
 with open(link, 'w+') as f:  
 f.write(json\_object)  
 print("The database was created")  
  
 def get\_db\_info\_json(self):  
 link = f'C:/Users/Max/ITLab1/database/main/database/{self.name}.json'  
 with open(link, 'r+') as f:  
 data = json.load(f)  
 db\_instance = data  
 return db\_instance  
  
 def set\_db\_info\_json(self, db\_instance):  
 link = f'C:/Users/Max/ITLab1/database/main/database/{self.name}.json'  
 with open(link, 'r+') as f:  
 json\_object = json.dumps(db\_instance, indent=4)  
 f.seek(0)  
 f.write(json\_object)  
 f.truncate()  
  
 def view\_db(self):  
 db\_instance = self.get\_db\_info\_json()  
 data = {  
 "tables": db\_instance["tables"],  
 "db\_name": self.name,  
 }  
 return data  
  
 def create\_table(self, table\_name, cols\_num):  
 db\_instance = self.get\_db\_info\_json()  
 new\_data = {table\_name: {}}  
 db\_instance["tables"].update(new\_data)  
 self.set\_db\_info\_json(db\_instance)  
 print("The table in " + self.name + " was created and have name " + table\_name)  
 cols\_value = ""  
 for j in range(int(cols\_num)):  
 cols\_value += str(j)  
 return cols\_value  
  
 def download(self):  
 db\_instance = self.get\_db\_info\_json()  
 response = HttpResponse(content\_type='application/json; charset=utf-8')  
 response['Content-Disposition'] = 'attachment; filename="db.json"'  
  
 t = loader.get\_template('main/download.json')  
 c = {  
 "db": db\_instance,  
 }  
  
 response.write(t.render(c))  
 return response  
  
 def edit\_db\_name(self, new\_db\_name):  
 link\_old = f'C:/Users/Max/ITLab1/database/main/database/{self.name}.json'  
 link\_new = f'C:/Users/Max/ITLab1/database/main/database/{new\_db\_name}.json'  
 os.rename(link\_old, link\_new)  
 self.name = new\_db\_name  
 db\_instance = self.get\_db\_info\_json()  
  
 db\_instance["name"] = new\_db\_name  
  
 def delete\_table(self, table\_name):  
 db\_instance = self.get\_db\_info\_json()  
 del db\_instance["tables"][f"{table\_name}"]  
 self.set\_db\_info\_json(db\_instance)

А ось так класс Таблиця

class Table:  
 def \_\_init\_\_(self, db\_obj, table\_name):  
 self.db\_obj = db\_obj  
 self.name = table\_name  
 self.db\_instance = self.db\_obj.get\_db\_info\_json()  
  
 def get\_cols\_type\_json(self):  
 return self.db\_instance["tables"][self.name]["cols\_type"]  
  
 def get\_cols\_name\_json(self):  
 return self.db\_instance["tables"][self.name]["cols\_name"]  
  
 def get\_rows\_json(self):  
 return self.db\_instance["tables"][self.name]["rows"]  
  
 def set\_rows\_json(self, row):  
 self.db\_instance["tables"][self.name]["rows"].append(row)  
 return self.db\_instance  
  
 def add\_row\_json(self, fields):  
 new\_data = {}  
 cols\_name = self.get\_cols\_name\_json()  
 for i in range(len(fields)):  
 new\_data[f"{cols\_name[i]}"] = fields[i]  
 table = self.get\_rows\_json()  
 try:  
 last\_row = table[-1]  
 last\_id = last\_row["id"]  
 except Exception:  
 last\_id = 0  
 new\_data["id"] = last\_id+1  
 new\_db\_instance = self.set\_rows\_json(new\_data)  
 self.db\_obj.set\_db\_info\_json(new\_db\_instance)  
  
 def create\_cols(self, types):  
 new\_data = {"cols\_type": types}  
  
 self.db\_instance["tables"][f"{self.name}"].update(new\_data)  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)  
  
 def create\_names(self, names):  
 if len(names) != len(set(names)):  
 print("ERROR: Some cols have same names")  
  
 del self.db\_instance["tables"][f"{self.name}"]  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)  
 return HttpResponseRedirect(f"/main/home/db/{self.db\_obj.name}")  
 else:  
 new\_data = {"cols\_name": names}  
 empty\_rows = {"rows": []}  
 self.db\_instance["tables"][f"{self.name}"].update(new\_data)  
 self.db\_instance["tables"][f"{self.name}"].update(empty\_rows)  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)  
 return HttpResponseRedirect(f"/main/home/db/{self.db\_obj.name}")  
  
 def view\_table(self):  
 rows = self.db\_instance["tables"][f"{self.name}"]["rows"]  
 cols\_name = self.db\_instance["tables"][f"{self.name}"]["cols\_name"]  
 data = {  
 "db\_name": self.db\_obj.name,  
 "table\_name": self.name,  
 "rows": rows,  
 "cols\_name": cols\_name,  
 }  
 return data  
  
 def delete\_row(self, row\_id):  
 rows = self.db\_instance["tables"][f"{self.name}"]["rows"]  
 i = 0  
 for row in rows:  
 if row['id'] == int(row\_id):  
 break  
 i += 1  
 print(f"Row with id:{i + 1} was deleted")  
 try:  
 del rows[i]  
 except IndexError:  
 print(f"Row with id:{row\_id + 1} does not exist")  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)  
  
 def edit\_row(self, row\_id, fields):  
 print("Fields")  
 print(fields)  
 rows = self.db\_instance["tables"][f"{self.name}"]["rows"]  
 i = 0  
 for row in rows:  
 if row['id'] == int(row\_id):  
 break  
 i += 1  
 try:  
 row\_to\_edit = rows[i]  
 j = 0  
 cols\_name = self.get\_cols\_name\_json()  
 for row\_field\_name in cols\_name:  
 row\_to\_edit[f"{row\_field\_name}"] = fields[j]  
 j += 1  
  
 except IndexError:  
 print(f"Row with ID:{row\_id+1} does not exist")  
 print("Row edited: ")  
 print(row\_to\_edit)  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)  
  
 def del\_same\_rows(self):  
 rows = self.db\_instance["tables"][f"{self.name}"]["rows"]  
 same\_rows\_list = set()  
 i = 0  
 for row in rows:  
 j = 0  
 for row2 in rows:  
 if row2['id'] == row['id']:  
 j += 1  
 continue  
 temp\_id = row2['id']  
 row2['id'] = row['id']  
 print(row)  
 print(row2)  
 if row == row2:  
 same\_rows\_list.add(i)  
 same\_rows\_list.add(j)  
  
 row2['id'] = temp\_id  
 j += 1  
 i += 1  
 print(same\_rows\_list)  
 for el in sorted(same\_rows\_list, reverse=True):  
 del rows[el]  
  
 self.db\_obj.set\_db\_info\_json(self.db\_instance)

Юніт тести виглядають ось так

class SimpleTest(unittest.TestCase):  
  
 def setUp(self):  
 # Every test needs access to the request factory.  
 self.factory = RequestFactory()  
  
 def test\_del\_same\_rows(self):  
 request = self.factory.delete('home/db/ForTestPurpose/table/Test/del\_same\_rows')  
 response = views.del\_same\_rows(request, "ForTestPurpose", "Test")  
 self.assertEqual(response.status\_code, 200)  
  
 def test\_delete\_row(self):  
 request = self.factory.delete('home/db/ForTestPurpose/table/Test/delete\_row/1')  
 response = views.delete\_row(request, "ForTestPurpose", "Test", "1")  
 self.assertEqual(response.status\_code, 200)  
  
 def test\_delete\_table(self):  
 request = self.factory.delete('home/db/ForTestPurpose/table/Test/delete\_table')  
 response = views.del\_same\_rows(request, "ForTestPurpose", "Test")  
 self.assertEqual(response.status\_code, 200)

Запускаємо тести

