hw\_num: 18

Опишете ORM‑модели и реализуете единый Python‑скрипт на основе PeeWee, который при запуске создаёт файл БД, создаёт таблицы, заполняет их тестовыми данными и выводит результаты в консоль.

hw\_theme:

  - sqlite

  - CREATE

  - INSERT

  - UPDATE

  - python

  - PeeWee

> **\*\*Технологии:\*\*** Python, SQLite, PeeWee ORM

**### 1. Описание моделей и полей**

Опишите модели PeeWee, соответствующие таблицам проекта «Фитнес клуб». Поле `id` не объявляйте явно — PeeWee автоматически создаёт первичные ключи `AutoField`. Для каждого поля укажите тип PeeWee и основные параметры.

Таблица методов и функций PeeWee, которые понадобятся в этом ДЗ:

|Метод / Функция|Описание|

|`DB.connect()`|Устанавливает соединение с базой данных.|

|`DB.create\_tables([Model1, Model2, …])`|Создаёт указанные модели (таблицы) в БД, если их ещё нет.|

|`DB.close()`|Закрывает соединение с базой после всех операций.|

|`Model.create(\*\*fields)`|Создаёт и сразу сохраняет в БД одну запись с переданными значениями полей.|

|`Model.insert\_many(list\_of\_dicts).execute()`|Пакетно вставляет несколько записей (полезно для создания сразу 2–3 мастеров или 3–4 услуг).|

|`Model.select()`|Начинает запрос на выборку всех записей из таблицы.|

|`Model.select().where(condition)`|Возвращает записи, удовлетворяющие условию (например, trainersselect().where(Trainers.phone=='123')`).|

|`Model.get(condition)`|Возвращает ровно одну запись по условию или бросает исключение, если ни одной или больше одной записи не найдено.|

|`Model.update(\*\*fields).where(condition).execute()`|Обновляет поля выбранных записей по условию.|

|`Model.delete().where(condition).execute()`|Удаляет записи по заданному условию.|

|`model\_instance.save()`|Сохраняет изменения в уже загруженном объекте модели (если нужны правки после `get()`).|

|`Model.select().join(OtherModel)`|Выполняет SQL‑JOIN с другой моделью, чтобы получить связанные через ForeignKeyField записи.|

|`ForeignKeyField(backref='…')`|При объявлении модели создаёт удобный атрибут в связанной модели (например, ` trainers\_workouts` для доступа ко всем записям этого тренера).|

|`Model.get\_or\_create(\*\*fields)`|Ищет запись по полям, если не найдена — создаёт её; возвращает кортеж `(instance, created\_flag)`.|

|`Model.count()`|Возвращает число записей, подходящих под параметры запроса.|

**### 2. Реализация скрипта**

- Подключите базу:

  ```python

  from peewee import SqliteDatabase

  DB = SqliteDatabase('BODY\_FIT.db')

  ```

- Зарегистрируйте существующие модели этой БД:

  ```python

  # Пример:

  # DB.connect()

  ```

- Добавьте тестовые данные через ORM:

  - не менее 2–3 тренеров;

  - не менее 3–4 тренировок;

  - не менее 3 записей на тренировки, каждая привязана к одному тренеру и ровно двум тренировкам.

- После вставки сделайте выборку и выведите результаты в консоль:

  ```python

  # Пример вывода:

  # for t in Trainers.select(): ...

> [!warning]

> **### Критерии проверки 👌**

> 1. Все модели описаны с корректными полями и параметрами, `id` не объявляется явно.

> 2. Реализованные связи «многие‑ко‑многим»

> 3. Скрипт при запуске создаёт или подключается к `BODY\_FIT.db`, создаёт таблицы и наполняет их тестовыми данными без ошибок.

> 4. В базе есть минимум 2–3 тренера, 3–4 тренировка и ≥3 записей, каждая содержит более 2 тренировок.

> 5. В консоли отображается список тренеров, тренировок и записей с указанием связанных тренировок.

> 6. Код соответствует PEP‑8, использует аннотации типов и методы PeeWee для всех операций.

---