

Санкт-Петербургский Национальный
Исследовательский Университет Информационных
технологий, механики и оптики

Контрольная работа №2

Расширение проекта. Взаимодействие с базой данных.

Выполнил: Фоминцев
Денис Русланович
Группа № 3123
Проверила: Казанова
Полина Петровна

Цель работы:

Постановка задачи

Продолжить работу над проектом, созданным в Контрольной №1 Требуется разработать программы согласно указанным ниже заданиям (на выбор, но можно реализовать свою тему). Реализация классов должна быть в отдельных модулях. Клиентский код должен импортировать вашу библиотеку и посредством объектов воспользоваться реализуемой в них функциональностью. В тексте использовать комментарии, поясняющие ваши действия и принятые решения. Объем задания (конкретные указания, что надо выполнить) указан ниже. Программное решение общего задания требуется представить в виде отдельных решений по каждому заданию, которые должны показать развитие общего проекта.

Задачи:

Задание 4. Исключения

- Реализуйте в своем проекте поддержку исключений.
- Создайте в своем проекте классы исключений и покажите их применение.
- Предложите в вашей процедуре (любой на ваш выбор) реализацию возбуждения исключения в случае нахождения некоего соответствия вместо того, чтобы возвращать флаг состояния, который должен интерпретироваться вызывающей программой, т.е. с помощью исключения обеспечить способ подачи сигнала, не возвращая значение.

Задание 5. Многопоточность и асинхронность

- Добавьте в свой проект реализацию вторичного потока для любого алгоритма.
- Подберите наиболее наглядный пример, обоснуйте свой выбор.
- Добавьте в свой проект реализацию асинхронного вызова для любого алгоритма.
- Подберите наиболее наглядный пример, обоснуйте свой выбор.

Задание 6. Реализация взаимодействия с базой данных.

- Реализовать хранение данных в базе данных.
- Реализовать взаимодействия с базой данных на основе ORM.
Выбор типа СУБД и технологии разработки на усмотрение разработчика – обосновать принятые решения.

Ход работы:

Задание №4

Для начала реализуем класс исключений (`IntFloatValueError(Exception)`). В тот момент, когда пользователю представляется возможность ввода суммы (`Int/Float`) у него есть возможность ошибаться сколько угодно, но при этом у него будет столько же шансов ввести верное число. До того, пока пользователь не напишет правильную сумму, программа не продолжит свою работу (см. рисунок 1).

```
63
64     class IntFloatValueError(Exception): # Исключения
65         def __init__(self, value):
66             self.value = value
67
68         def __str__(self):
69             return "{}" не является верным вводом, принимаются только положительные значения ' \
70                 'и значения с плавающей точкой'.format(self.value)
71
72     Inf = 0
73     array = ()
74     def begin(s):
75         global Inf
76         alf = '0123456789.'
77         t = s
78         # Inf = 0
79         count = 0
80         for i in range(len(t)):
81             if (t[i] in alf):
82                 count += 1
83                 if (count == len(t)):
84                     Inf = 1
85             else:
86                 print(IntFloatValueError(s))
87                 print('Попробуйте еще раз:')
88                 break
89         return Inf
90
91     while (Inf == 0):
92         print('Введите сумму, которую хотите вложить (Int/Float): ')
93         s = str(input())
94         begin(s)
95         Inf = 0
96         array += (s,)
```

Рисунок 1. Класс исключения (`IntFloatValueError(Exception)`)

Задание №5

Реализация многопоточности реализуется с помощью модуля `Thread`. В данном примере с помощью него реализован декоратор отсчета времени (см. рисунок 2).

```
4     from peewee import *
5     import sqlite3
6     from threading import *
7     from threading import Thread
8     import asyncio
9
10    t_start = time.perf_counter()
11
12    print('I этап работы программы. Идет запуск, подождите 5 секунд...')
13    def one(num): # Многопоточность
14        time.sleep(num)
15        print(round(time.perf_counter() - t_start) + 1, '...')
16
17    for i in range(5):
18        th = Thread(target=one, args=(i, ))
19        th.start()
20
21    th.join() # ждем завершения потоков
22    print()
23
```

Рисунок 2. Многопоточность

Реализация асинхронности реализуется с помощью модуля Asyncio. В данном примере с помощью него реализован декоратор вывода информации с базы данных, сохраненных чуть ранее (см. рисунок 3).

```
213     async def goodbye(): # Асинхронность
214         k = 0
215         for i in range(3):
216             k += 1
217             await asyncio.sleep(i)
218             print(k, '...')
219         print('Представлена информация из базы данных с помощью ORM:')
220
221     async def main():
222         print('Подождите окончания работы программы...')
223         await goodbye()
224         query = Bank.select().order_by(Bank.Number_of_contribution.desc())
225         art = query.dicts().execute()
226         for client in art:
227             print('client: ', *list(art))
228
229     asyncio.run(main())
230     cursor.close()
```

Рисунок 3. Асинхронность

Задание №6

Все данные, введенные пользователем, сохраняются в базу данных Sqlite. На рисунке 4 показано создание базы данных и основной таблицы.

```
24     try:
25         connection = SqliteDatabase('Kr.sqlite')
26         cursor = connection.cursor()
27         print('База данных успешно создана и подключена к Sqlite')
28         print()
29
30         sql = "DROP TABLE IF EXISTS bank"
31         cursor.execute(sql)
32
33         array = ()
34         arr = []
35         answer = 0
36         mnth = 0
37
38
39         cursor.execute("""CREATE TABLE IF NOT EXISTS bank(
40             Sum INT,
41             Number INT,
42             Result INT,
43             Month INT);
44             """)
45         connection.commit()
```

Рисунок 4. Создание таблицы базы данных

На рисунке 5 показан пример взаимодействия с таблицей данных с помощью ORM.

```
171     array += (round(answer, 2),) + (mth,)
172     arr.append(array)
173
174     cursor.executemany("INSERT INTO bank VALUES(?, ?, ?, ?);", arr)
175     connection.commit()
176
177     cursor.execute("SELECT * FROM bank")
178     all_results = cursor.fetchall()
179
180     print()
181     pr = 0
182     if (all_results[1] == 1):
183         pr = '5'
184     elif (all_results[1] == 2):
185         pr = '12'
186     else:
187         pr = '7'
188
189     print('Данные Sqlite таблицы сохранены. Номер вклада {}. Количество месяцев {}. Процент: {}%. Исходная сумма {}, Конечная сумма {}'.format(all
190
191
192     class BaseModel(Model):
193         class Meta:
194             database = connection
195
196
197     class Bank(BaseModel):
198         Number_of_contribution = AutoField(column_name='Number')
199         Initial_Summa = TextField(column_name='Sum', null=True)
200         Result_Summa = TextField(column_name='Result', null=True)
201         Month_Time = TextField(column_name='Month', null=True)
202
203         class Meta:
204             table_name = 'bank'
205
```

Рисунок 5. Взаимодействия с базой данных на основе ORM

Пример работы программы:

```
C:\Users\User\AppData\Local\Programs\Python\Python39\python.exe
I этап работы программы. Идет запуск, подождите 5 секунд...
1 ...
2 ...
3 ...
4 ...
5 ...

База данных успешно создана и подключена к Sqlite

Bank
    Калькулятор вкладов
Введите сумму, которую хотите вложить (Int/Float):
sefs
"sefs" не является верным вводом, принимаются только положительные значения и значения с плавающей точкой
Попробуйте еще раз:
Введите сумму, которую хотите вложить (Int/Float):
1
"1" не является верным вводом, принимаются только положительные значения и значения с плавающей точкой
Попробуйте еще раз:
Введите сумму, которую хотите вложить (Int/Float):
1211
Введите номер варианта вклада, который вас интересует (1-2)
Срочный вклад (1); Вклад с капитализацией процентов (2)
fsfes
Попробуйте ещё раз (1-2): 3
Попробуйте ещё раз (1-2): 1
Выбран срочный вклад, введите срок
12
Вклад размером: 1211. Процент: 5%. Время 12 месяц(-а/-ев)
Итоговая сумма: 1937.6

Данные Sqlite таблицы сохранены. Номер вклада 1. Количество месяцев 12. Процент: 5%. Исходная сумма 1211, Конечная сумма 1937.6

Подождите окончания работы программы...
1 ...
2 ...
3 ...
Представлена информация из базы данных с помощью ORM:
Client: {'Number_of_contribution': 1, 'Initial_Summa': '1211', 'Result_Summa': '1937.6', 'Month_Time': '12'}

Соединение с Sqlite закрыто
Время окончания: 32.9874019

Process returned 0 (0x0)      execution time : 33.171 s
Для продолжения нажмите любую клавишу . . .
```

Вывод:

В данной работе расширили наш проект, созданный в первой контрольной работе. Подключили базу данных.