
Разработка приложений баз данных на Python в примерах

АЛЕКСАНДР САЯПИН

Разработка приложений баз данных

Установка языка Python и необходимых библиотек

Для разработки мы используем широко распространенный язык Python.

Для того, чтобы его использовать, вам необходимо распаковать архив port_python в вашу рабочую папку. Это среда языка Python, которая может быть запущена независимо от того, установлен ли язык Python на вашем компьютере, и включает необходимые нам библиотеки.

Запуск среды разработки

После завершения установки в проводнике Windows войдите во вновь созданную папку, и, нажав Shift, щелкните правой кнопкой мыши на свободном месте в окне. Выберите пункт Открыть окно команд.

Откроется окно (рисунок 1).

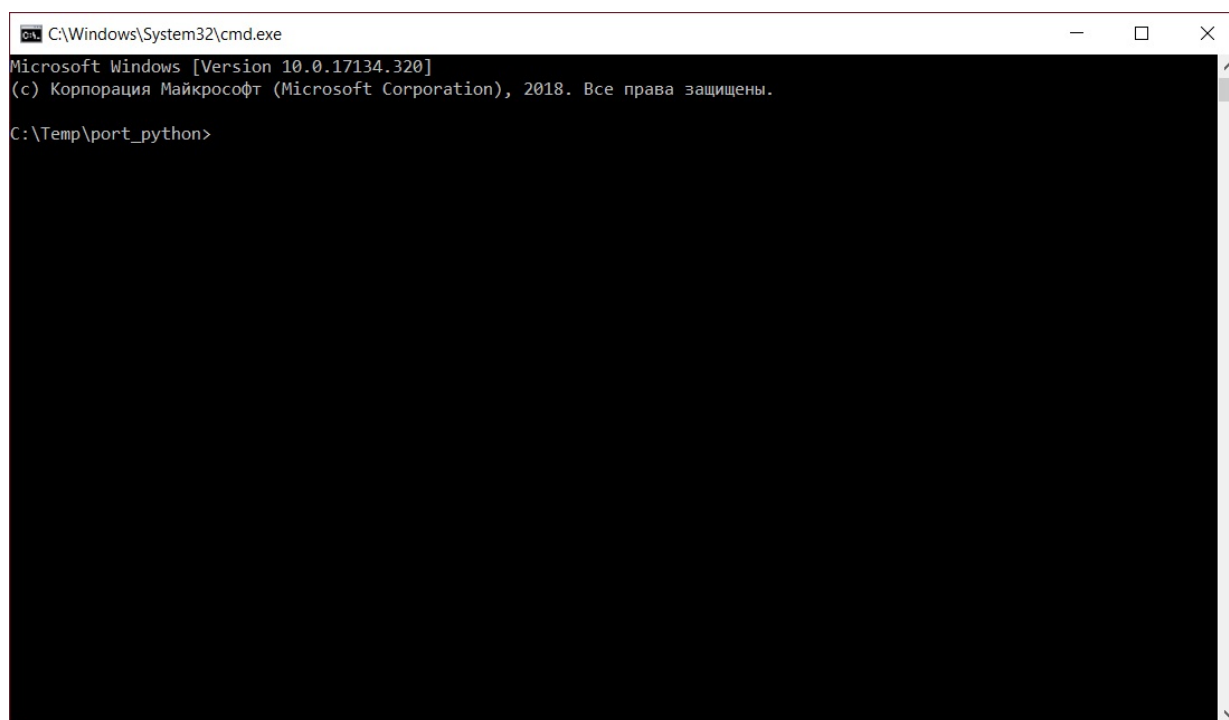


Figure 1: Среда разработки Python

Затем из рабочей папки запустите редактор текстовый редактор Notepad2. Для этого войдите в папку Notepad2 рабочей папки, и запустите файл Notepad2.exe. Откроется окно редактора (рисунок 2).

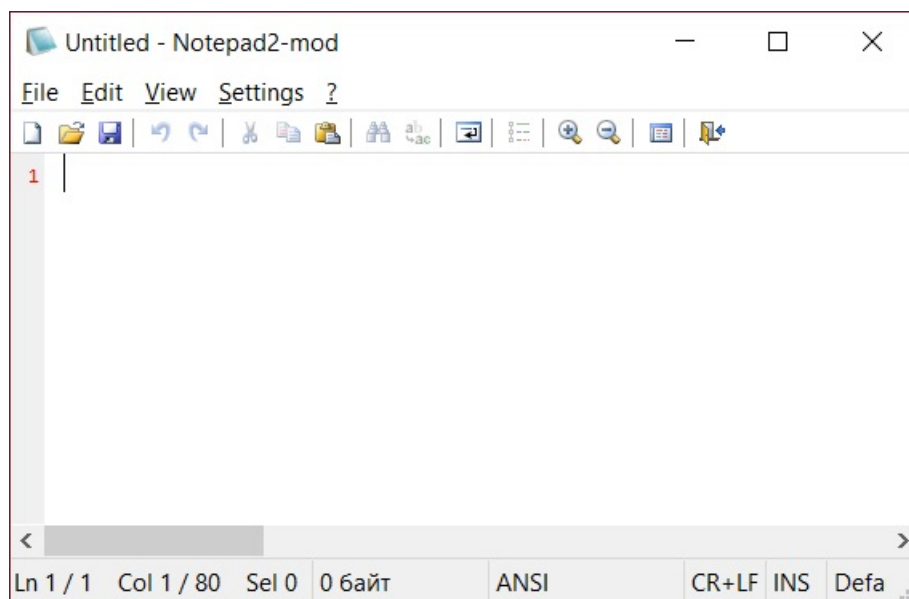


Figure 2: Notepad2

Здесь можно создавать и редактировать программы на языке Python.

Запуск первой программы

Давайте для начала создадим простейшую программу, которая подключается к вашему серверу базы данных.

Введите нижеследующий текст, соблюдая пробелы.

Текст программы:

```
1 import mysql.connector
2
3 mydb = mysql.connector.connect(host="127.0.0.1",
4                                port="3311",
5                                user="root",
6                                passwd="root")
7
8 print(mydb)
```

Разберем программу: Здесь мы подключаем библиотеку для работы с сервером БД MySQL

```
1 import mysql.connector
```

Здесь мы подключаемся к серверу БД, указывая сетевой адрес сервера (host) имя пользователя (user), и пароль (passwd)

```
1 mydb = mysql.connector.connect(host="127.0.0.1",
2                               port="3311",
3                               user="root",
4                               passwd="root")
```

Выводим на экран созданное подключение

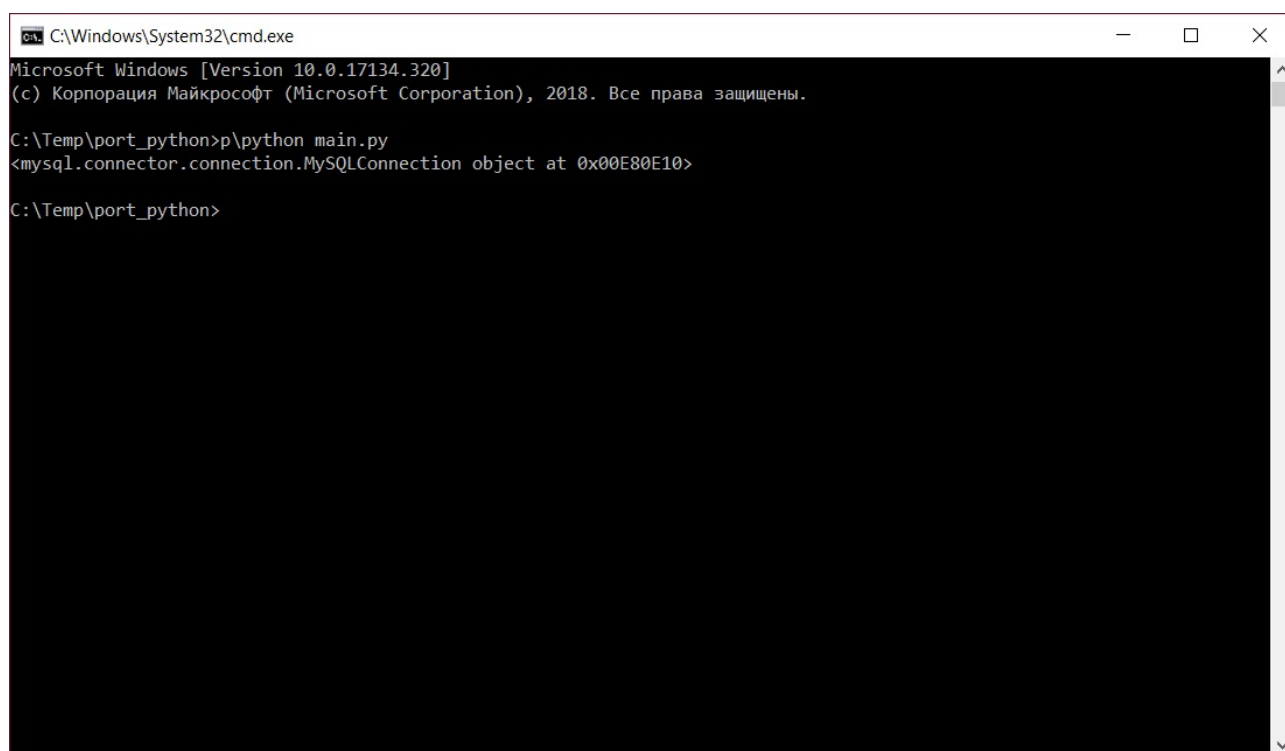
```
1 print(mydb)
```

Сохраните только что созданный файл, он нам пригодится как основа для дальнейшей работы. Для этого используйте пункт меню File/Save as.... Сохраните файл в папку port_python. Имя файла не должно содержать русских символов. Далее будем считать, что вы дали файлу имя main.py.

После сохранения файла переключитесь на окно командной строки (рисунок 1) и введите команду

```
1 p\python main.py
```

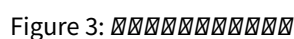
В результате программа выполнится, и вы увидите результат работы наподобие того, что приведено на рисунке 3.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.320]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Temp\port_python>p\python main.py
<mysql.connector.connection.MySQLConnection object at 0x00E80E10>

C:\Temp\port_python>
```

Figure 3: 

Учтите, что к этому моменту ваш сервер баз данных MySQL должен быть уже запущен.

Если ваш результат отличается, проверьте текст программы, а также параметры подключения, такие как хост, порт, имя пользователя и пароль.

Первый запрос

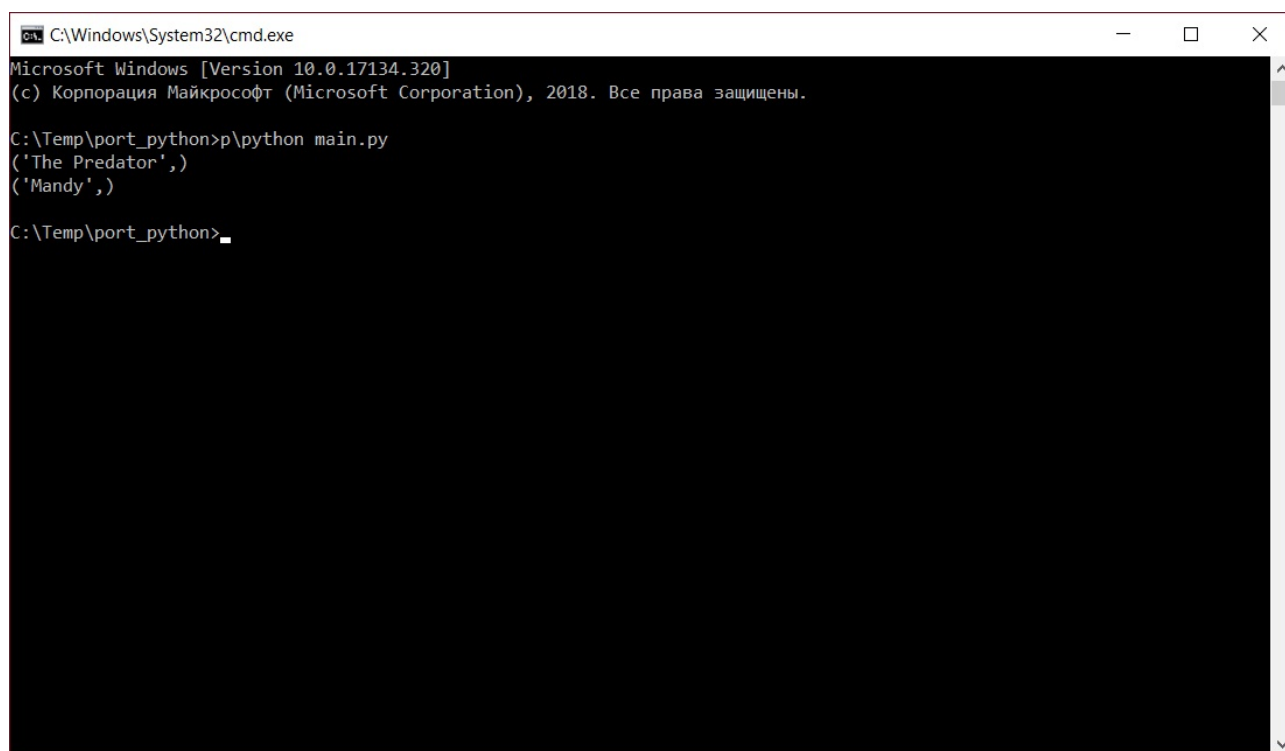
Давайте попробуем получить информацию из базы данных. Для этого доработаем нашу программу:

```
1 import mysql.connector
2
3 mydb = mysql.connector.connect(host="127.0.0.1",
4                               port="3311",
5                               user="root",
6                               passwd="root",
7                               database="films")
8
9 query = "SELECT Name FROM Film"
10
11 mycursor = mydb.cursor()
12
13 mycursor.execute(query)
14
15 for row in mycursor:
16     print(row)
```

В программу мы внесли следующие изменения:

1. В подключение к БД добавили новый параметр, `database="film"`, указав название базы данных, к которой мы подключаемся. Вам следует подставить вместо `film` название вашей базы данных.
2. Добавили новую переменную `query`, в которой хранится текст запроса. В данном случае запрос извлекает все названия фильмов (поле `Name`) из таблицы `Film`. Вам следует подставить названия своей таблицы и ваше название поля, в котором хранится название фильма.
3. Создали так называемый курсор (`mycursor`). Он предназначен для выполнения запроса и содержит результаты запроса, которые можно перебирать в цикле.
4. Выполняем запрос, который содержится в переменной `query` (`mycursor.execute(query)`)
5. В цикле (`for`), перебираем все строки в курсоре (`for row in cursor`), называя каждую из них `row`.
6. Печатаем каждую из строк (`print(row)`)



В результате на экран будут выведены названия фильмов, как показано на рисунке 4.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.320]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Temp\port_python>python main.py
('The Predator',)
('Mandy',)

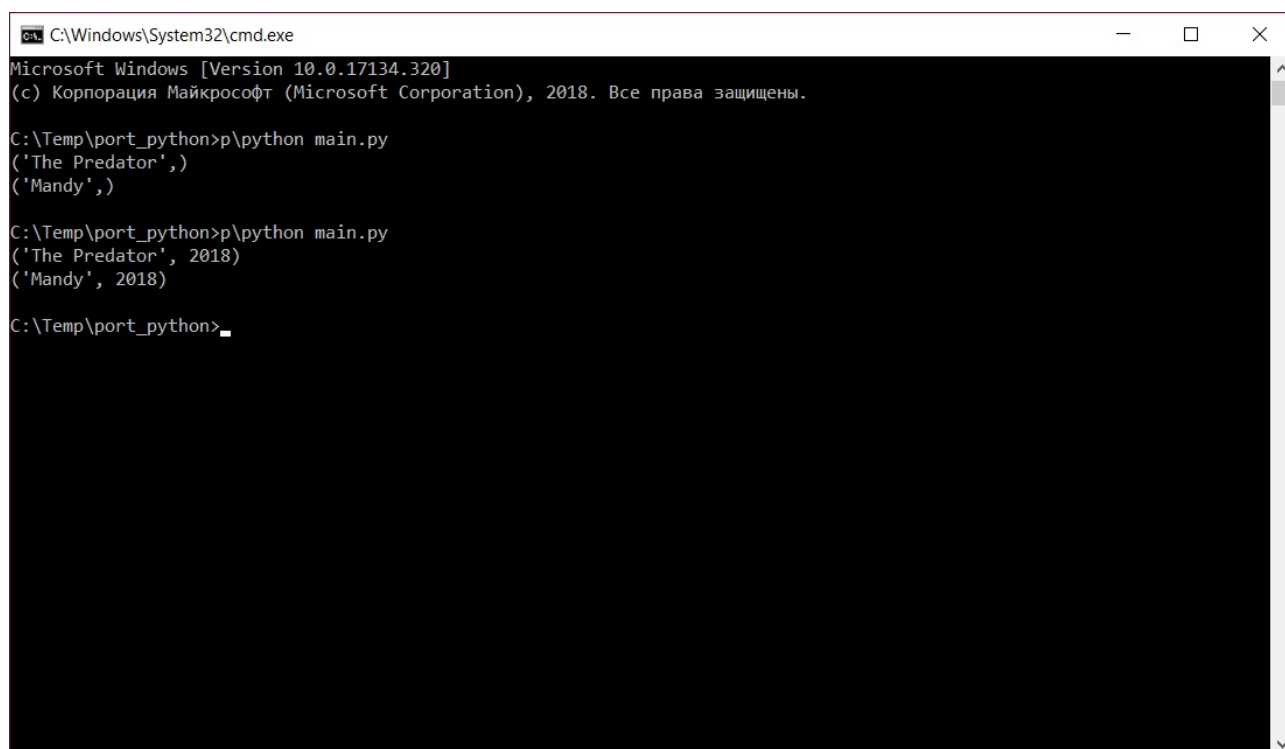
C:\Temp\port_python>
```

Figure 4:  

Изменим запрос так, чтобы выводились названия фильмов и их год выхода. Для этого в переменную query запишем такой запрос:

```
1 SELECT Name, Year FROM Film
```

Теперь результат запроса будет таким (рисунок 5).



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.320]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Temp\port_python>p\python main.py
('The Predator',)
('Mandy',)

C:\Temp\port_python>p\python main.py
('The Predator', 2018)
('Mandy', 2018)

C:\Temp\port_python>_
```

Figure 5: Изображение экрана командной строки Windows, выполняющей Python-скрипт.

Создание окна приложения

Работать в чисто тестовом режиме не всегда удобно, поэтому создадим текстовый экраный интерфейс. Для этого мы будем использовать библиотеку `npyscreen`.

Давайте выведем результат запроса в окно интерфейса. для этого нам нужно будет доработать нашу программу:

```
1 import mysql.connector
2 import npyscreen
3
4 class App(npyscreen.StandardApp):
5     def onStart(self):
6         self.addForm("MAIN", MainForm, name="Программа
           работысбазойданныхфильмов")
7
8 class MainForm(npyscreen.ActionFormMinimal):
9     def create(self):
10         self.grid = self.add(npyscreen.GridColTitles)
11         self.grid.values= []
12
13         mydb = mysql.connector.connect(host="127.0.0.1",
```

```
14         port="3311",
15         user="root",
16         passwd="root",
17         database="films")
18
19     query = "SELECT Name, Year FROM Film"
20
21     mycursor = mydb.cursor()
22
23     mycursor.execute(query)
24
25     for row in mycursor:
26         grid_row = []
27         for cell in row:
28             grid_row.append(cell)
29         self.grid.values.append(grid_row)
30
31     def on_ok(self):
32         self.parentApp.setNextForm(None)
33
34
35 my_app = App()
36 my_app.run()
```

Запуск программы приведет к появлению окна, показанного на рисунке 6.

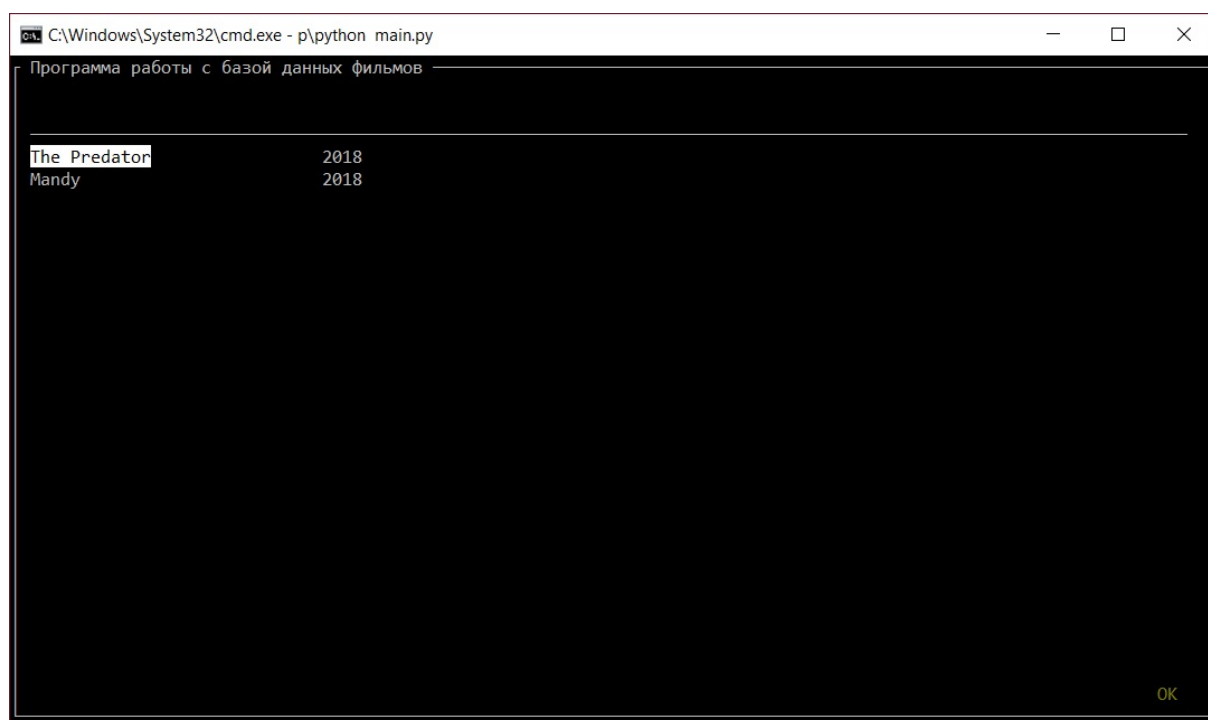


Figure 6: Скриншот программы, работающей с базой данных фильмов

Перемещаться по таблице можно, используя клавиши управления курсором. Нажав клавишу Tab, можно переместиться на окрannую кнопку Ok. Нажав ее, вы закроете окно программы.

Рассмотрим изменения в программе:

добавляем в программу библиотеку для работы с экранным интерфейсом

```
1 import npyscreen
```

создаем приложение для работы нашей программы, класс приложения называется App.

При создании приложения (метод onStart) мы добавляем к нашему приложению (self.addForm()) основную ("MAIN") экранную форму программы (MainForm, она описана далее) с заголовком (name="Программа работы с базой данных фильмов").

```
1 class App(npyscreen.StandardApp):
2     def onStart(self):
3         self.addForm("MAIN", MainForm, name="Программа
           работысбазойданныхфильмов")
```

Описываем экранную форму (указываем, что она наследуется от формы npyscreen.ActionFormMinimal) и называется MainForm

```
1 class MainForm(npyscreen.ActionFormMinimal):
```

и что при ее создании

```
1 def create(self):
```

на форму нужно добавить таблицу (npyscreen.GridColTitles), в нашей форме она называется grid

```
1 self.grid = self.add(npyscreen.GridColTitles)
```

и что в таблице не будет ни строк, ни столбцов (пока)

```
1 self.grid.values= []
```

Далее мы снова обращаемся к нашей базе данных, как мы это делали в прошлой программе

```
1 mydb = mysql.connector.connect(host="127.0.0.1",
2                                 port="3311",
3                                 user="root",
4                                 passwd="root",
5                                 database="films")
6
7 query = "SELECT Name, Year FROM Film"
8
9 mycursor = mydb.cursor()
10
11 mycursor.execute(query)
```

а затем заполняем таблицу

```
1 for row in mycursor:
2     grid_row = []
3     for cell in row:
4         grid_row.append(cell)
5     self.grid.values.append(grid_row)
```

перебирая сначала строки, и создавая пустую строку

```
1 for row in mycursor:
2     grid_row = []
```

а затем перебирая элементы строки, добавляя их в качестве ячеек в строку

```
1 for cell in row:
2     grid_row.append(cell)
```

и добавляя строку целиком

```
1 self.grid.values.append(grid_row)
```

Создаем функцию `on_ok` (это имя является стандартным)

```
1 def on_ok(self):
2     self.parentApp.setNextForm(None)
```

которая будет вызываться при нажатии экранной кнопки `Ok`. В данном случае функция заставляет приложение в целом (`self.parentApp`) переключиться на следующую форму (`setNextForm`), которой не существует (`None`), в результате чего наша программа закончит работу.

Последними строками программы мы создаем наше приложение, называя его `my_app`

```
1 my_app = App()
```

и запускаем его

```
1 my_app.run()
```

Запросы с параметрами

Текстовый параметр

Теперь изменим программу так, чтобы можно было задавать какой-либо параметр в запросе, например, год выхода фильма. Теперь программ будет выглядеть так:

```
1 import mysql.connector
2 import npyscreen
3
4 class App(npyscreen.StandardApp):
5     def onStart(self):
6         self.addForm("MAIN", MainForm, name="Программа
           работысбазойданныхфильмов ")
7
8 class MainForm(npyscreen.ActionForm):
9     def create(self):
10         self.year = self.add(npyscreen.TitleText, name="Год выходафильма ",
           value="2018")
11         self.grid = self.add(npyscreen.GridColTitles)
12
13         self.mydb = mysql.connector.connect(host="127.0.0.1",
14                                             port="3311",
15                                             user="root",
```

```
16         passwd="root",
17         database="films")
18
19     self.mycursor = self.mydb.cursor()
20
21     def on_ok(self):
22         query = "SELECT Name, Year FROM Film WHERE Year={0}".format(self.
23             year.value)
24
25         self.mycursor.execute(query)
26
27         self.grid.values= []
28         for row in self.mycursor:
29             grid_row = []
30             for cell in row:
31                 grid_row.append(cell)
32             self.grid.values.append(grid_row)
33
34     def on_cancel(self):
35         self.parentApp.setNextForm(None)
36
37 if __name__ == '__main__':
38     my_app = App()
39     my_app.run()
```

Мы изменили тип оконной формы, теперь она называется `npyscreen.ActionForm` в строке

```
1 class MainForm(npyscreen.ActionForm):
```

Такая форма содержит две кнопки, Ok и Cancel.

Мы немного изменили код, вызываемый при создании формы:

```
1 def create(self):
2     self.year = self.add(npyscreen.TitleText, name="Год выходафильма ",
3         value="2018")
4     self.grid = self.add(npyscreen.GridColTitles)
5
6     self.mydb = mysql.connector.connect(host="127.0.0.1",
7         port="3311",
8         user="root",
9         passwd="root",
10        database="films")
11
12     self.mycursor = self.mydb.cursor()
```

Добавлено поле для ввода номера года

```
1 self.year = self.add(npyscreen.TitleText, name="Год выхода фильма ",
    value="2018")
```

Называться поле на экране будет Год выхода фильма, а его значение по умолчанию будет 2018.

Теперь база данных mydb тоже является частью формы (self.mydb), и будет существовать столько, сколько существует форма (а в прошлой версии программы сразу после создания формы и заполнения таблицы ссылка на базу данных переставала существовать). Курсор так же будет существовать столько же, сколько и форма.

Часть программы, извлекающая данные из БД и заполняющая таблицу, перенесена в обработчик события нажатия кнопки Ok (то есть, теперь запрос к базе данных будет выполняться только при нажатии кнопки Ok):

```
1 def on_ok(self):
2     query = "SELECT Name, Year FROM Film WHERE Year={0}".format(self.year.
        value)
3
4     self.mycursor.execute(query)
5
6     self.grid.values= []
7     for row in self.mycursor:
8         grid_row = []
9         for cell in row:
10             grid_row.append(cell)
11         self.grid.values.append(grid_row)
```

Обратите внимание, строка с запросом изменилась:

```
1 query = "SELECT Name, Year FROM Film WHERE Year={0}".format(self.year.
    value)
```

Функция format подставляет в строку указанные значения, в данном случае значение года, в место, где располагается последовательность {0}.

При нажатии кнопки Cancel форма будет закрыта:

```
1 def on_cancel(self):
2     self.parentApp.setNextForm(None)
```

При запуске программы будет показано окно (рисунок 7)

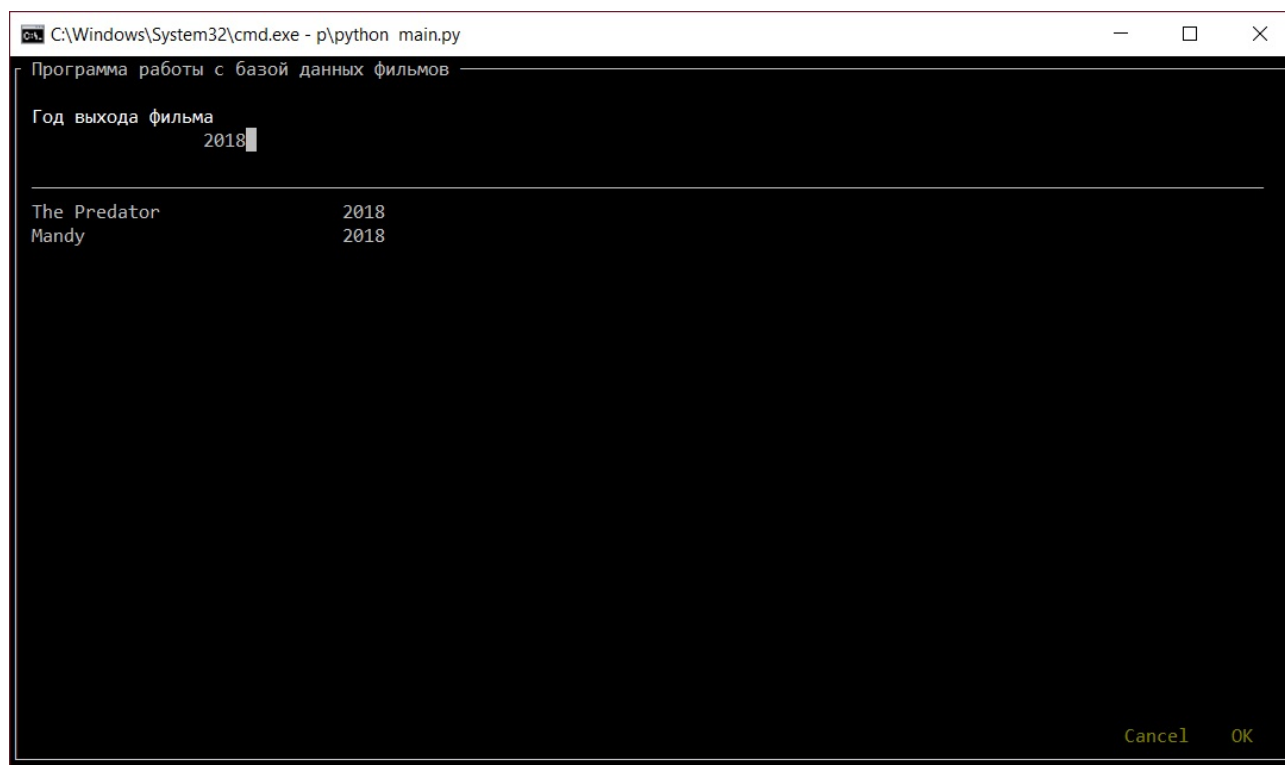


Figure 8: Скриншот программы работы с базой данных фильмов

Во многих случаях удобнее не вводить текст для поиска, а выбирать из таблицы базы данных. Например, если вам необходимо найти фильмы, где играет определенный актер, логично выбрать нужного актера из списка.

Для этого есть такой элемент интерфейса, как выпадающий список. Рассмотрим пример такой программы:

```

1 import mysql.connector
2 import npyscreen
3
4 class App(npyscreen.StandardApp):
5     def onStart(self):
6         self.addForm("MAIN", MainForm, name="Программа
           работысбазойданныхфильмов")
7
8 class MainForm(npyscreen.ActionForm):
9     def create(self):
10         self.genre = self.add(npyscreen.TitleCombo, name="Жанр")
11         self.grid = self.add(npyscreen.GridColTitles)
12
13         self.mydb = mysql.connector.connect(host="127.0.0.1",
14                                             port="3311",

```

```
15         user="root",
16         passwd="root",
17         database="films")
18
19     self.mycursor = self.mydb.cursor()
20     query = "SELECT Name FROM Genre"
21
22     self.mycursor.execute(query)
23
24     self.genre.values = []
25     self.genre_names = []
26     for row in self.mycursor:
27         self.genre.values.append(row[0])
28         self.genre_names.append(row[0])
29
30     def on_ok(self):
31         genre_name = self.genre_names[self.genre.value]
32         query = "SELECT idGenre, Name FROM Genre WHERE Name='{0}'".format(
33             genre_name)
34
35         self.mycursor.execute(query)
36
37         self.grid.values= []
38         for row in self.mycursor:
39             grid_row = []
40             for cell in row:
41                 grid_row.append(cell)
42             self.grid.values.append(grid_row)
43
44     def on_cancel(self):
45         self.parentApp.setNextForm(None)
46
47     if __name__ == '__main__':
48         my_app = App()
49         my_app.run()
```

В данном случае добавился мы заменили текстовое поле ввода на выпадающий список:

```
1         self.genre = self.add(npyscreen.TitleCombo, name="Жанр")
```

и добавили новый участок кода в конец функции def create(self):

```
1         query = "SELECT Name FROM Genre"
2
```



```
3         self.mycursor.execute(query)
4
5         self.genre.values = []
6         self.genre_names = []
7         for row in self.mycursor:
8             self.genre.values.append(row[0])
9             self.genre_names.append(row[0])
```

Здесь мы в переменную `query` записываем текст запроса, который извлекает список названий жанров из таблицы `Genre` и выполняем его.

После этого очищаем список жанров в выпадающем списке

```
1         self.genre.values = []
```

и создаем список названий жанров (мы его будем использовать далее)

```
1         self.genre_names = []
```

Далее мы в цикле перебираем названия жанров и добавляем их как в варианты выбора в выпадающем списке

```
1             self.genre.values.append(row[0])
```

так и в список названий жанров

```
1             self.genre_names.append(row[0])
```

В функции, вызываемой по нажатию кнопки `Ok`, мы изменили запрос:

```
1         genre_name = self.genre_names[self.genre.value]
2         query = "SELECT idGenre, Name FROM Genre WHERE Name='{0}'".format(
                    genre_name)
```

Далее мы, как обычно, заполняем таблицу на экране.