

Лекция 2. **Представления** и **маршрутизация** – ключевые элементы Django

Часть 1



Вопросы, которые будут рассмотрены в лекции:

- Что такое маршрутизатор (диспетчер) адресов?
- Как работает маршрутизатор (очередность рассмотрения адресов)?
- Что такое представление?
- Как генерируется ответ на запрос пользователя?
- Функции `path()` и `re_path()` для задания адреса. В чем различие?
- Регулярные выражения. Синтаксис, примеры, ссылки.
- Передача параметров в функции представления.
- Определение параметров по умолчанию.
- Определение параметров через функцию `path()`.
- Определение параметров по умолчанию в функции `path()`.
- Параметры строки запроса
- Обработка исключений при запросах к серверу
- Редиректы 301, 302



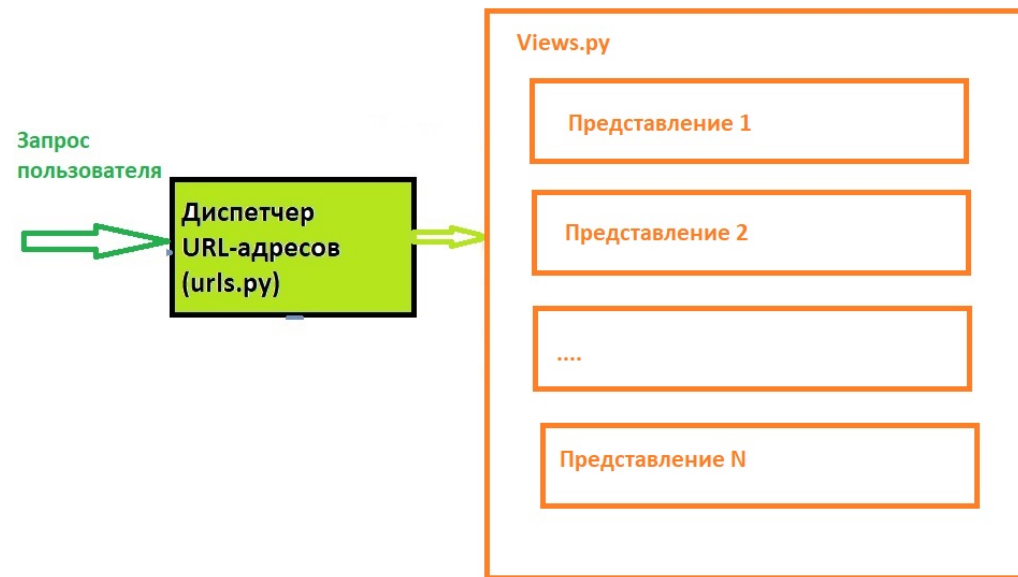
Маршрутизатор (urls.py)

От пользователя приходит запрос. Например:

<http://127.0.0.1.8000>

<http://127.0.0.1.8000/region/2> region/<число>

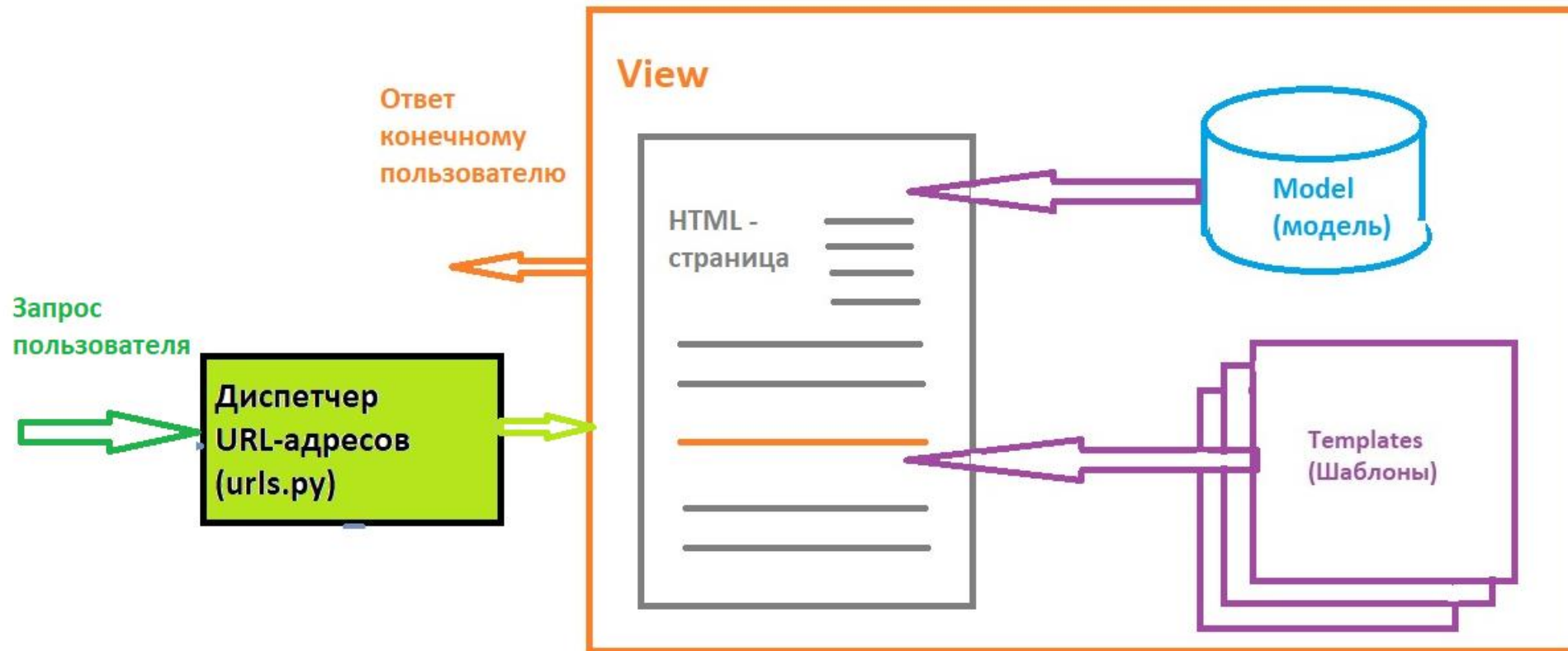
<http://127.0.0.1.8000/region/sibir> region/<слаг(строка)>



В маршрутизаторе фиксируется тип адреса и в списке шаблонов ищется первое совпадение по шаблону. Если найдено, то активизируется соответствующее представление.

Если ни одно совпадение не найдено, генерируется ошибка 404.





Такое разделение на данные, шаблоны и представления представляет паттерн MTV. Удобство: каждый блок разрабатывается отдельно

MTV : Model - Template - View
паттерн, который использует фреймворк Django



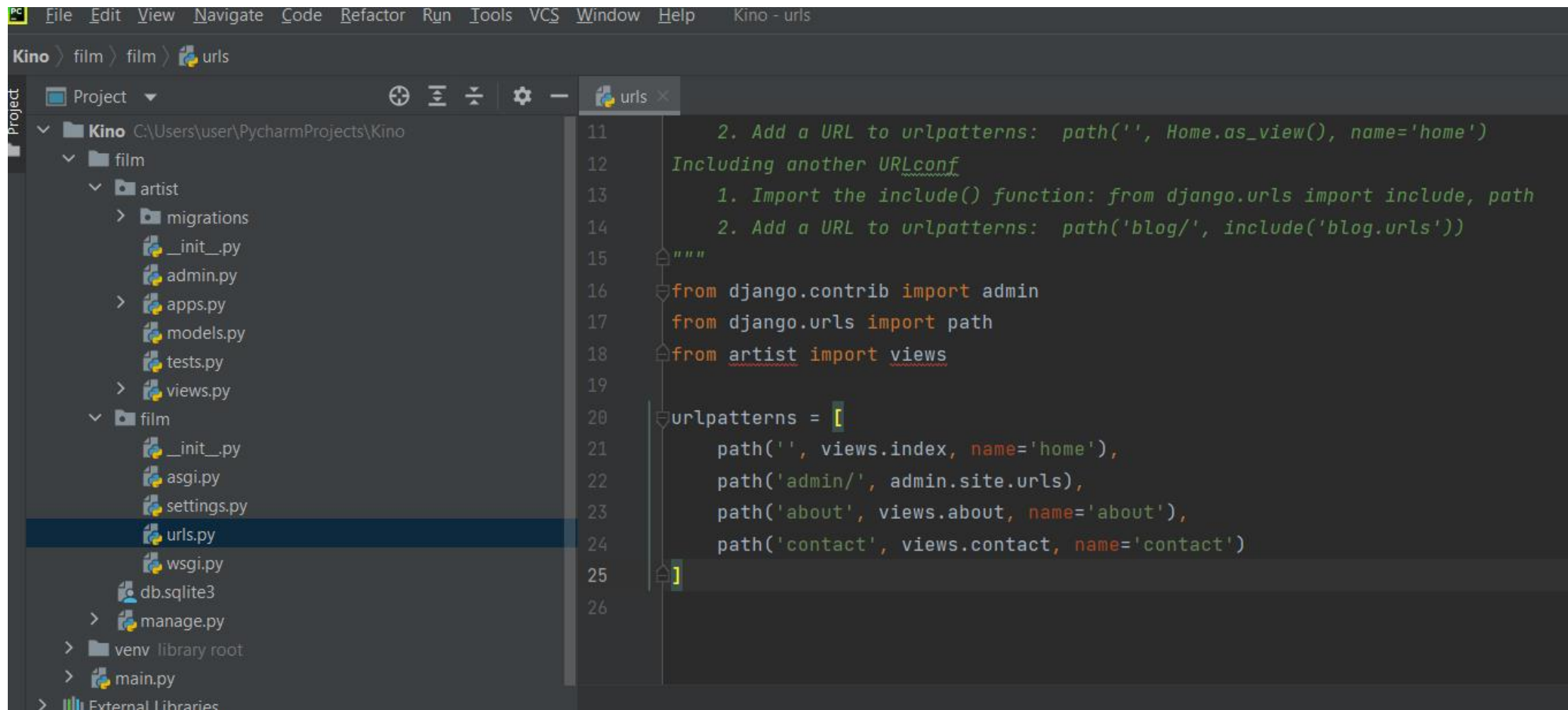
маршрутизатор - файл urls.py:

Маршрутам, содержащимся в файле urls.py, в файле представлений views.py. сопоставлены функции. Переменная `urlpatterns` файла urls.py содержит эти сопоставления:

```
from django.contrib import admin
from django.urls import path
from firstapp import views

urlpatterns = [
    path('', views.index),
    path('about', views.about),
    path('contact', views.contact),
    path('admin/', admin.site.urls),
]
```





The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the current file is 'Kino - urls'. The breadcrumb navigation shows the path: Kino > film > film > urls. The left sidebar displays the project structure for 'Kino' at 'C:\Users\user\PycharmProjects\Kino'. The 'film' directory is expanded, showing subdirectories 'artist' and 'film'. The 'film' subdirectory is further expanded, listing files: __init__.py, asgi.py, settings.py, urls.py (which is selected and highlighted), and wsgi.py. Other files visible in the project are db.sqlite3, manage.py, and main.py. The main editor window shows the content of 'urls.py' with line numbers 11 through 26. The code includes comments in green and Python code in orange and black. The code defines a list of URL patterns for the 'film' app, including paths for home, admin, about, and contact views.

```
11     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12     Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15     """
16     from django.contrib import admin
17     from django.urls import path
18     from artist import views
19
20     urlpatterns = [
21         path('', views.index, name='home'),
22         path('admin/', admin.site.urls),
23         path('about', views.about, name='about'),
24         path('contact', views.contact, name='contact')
25     ]
26
```



Представления (файл `views.py`)

В Django за обработку запросов пользователя отвечают представления (views, размещаются в файле `views.py`).

В представлениях реализованы функции, которые принимают данные запроса в виде объекта **request** и генерируют ответ пользователю (**response**) в виде HTML страниц.

Представления в Django можно реализовывать как в виде функций, так и в виде классов.



файл views.py:

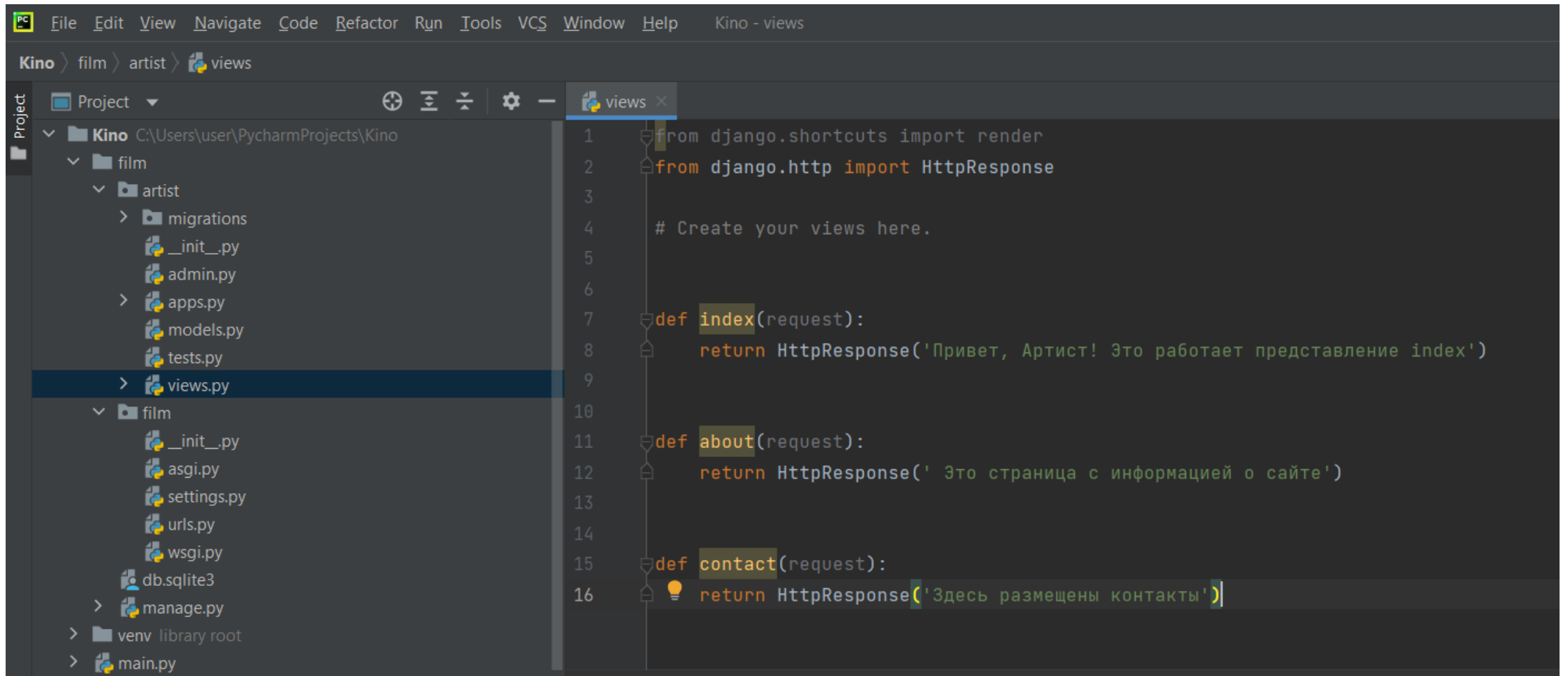
```
from django.http import HttpResponseRedirect

def index(request):
    """
    :param request:
    :return:
    """
    return HttpResponseRedirect("<h2>Главная</h2>")

def about(request):
    """
    :param request:
    :return:
    """
    return HttpResponseRedirect("<h2>О сайте</h2>")

def contact(request):
    """
    :param request:
    :return:
    """
    return HttpResponseRedirect("<h2>Контакты</h2>")
```





Запустим сервер: `python manage.py runserver`

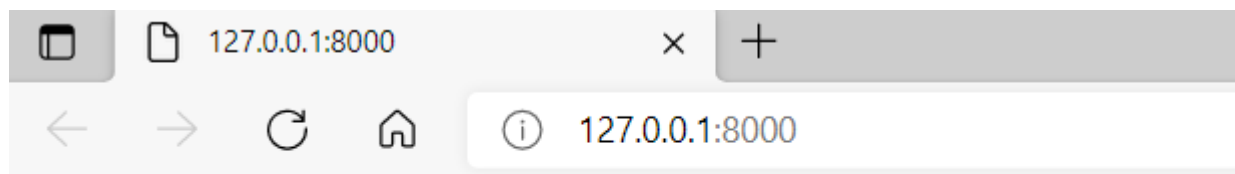
В браузере перейдем по адресу <http://127.0.0.1:8000>

Попадаем на главную страницу сайта.

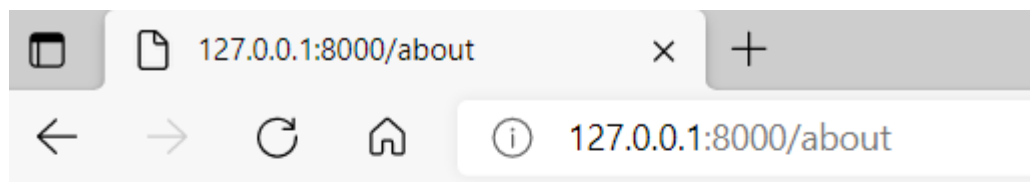
По адресу <http://127.0.0.1:8000/about> попадем на страницу **О сайте**.

По адресу <http://127.0.0.1:8000/contact> попадем на страницу **Контакты**.

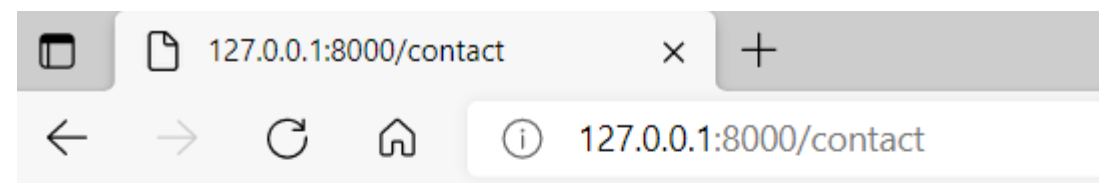




Привет, Артист! Это работает представление index



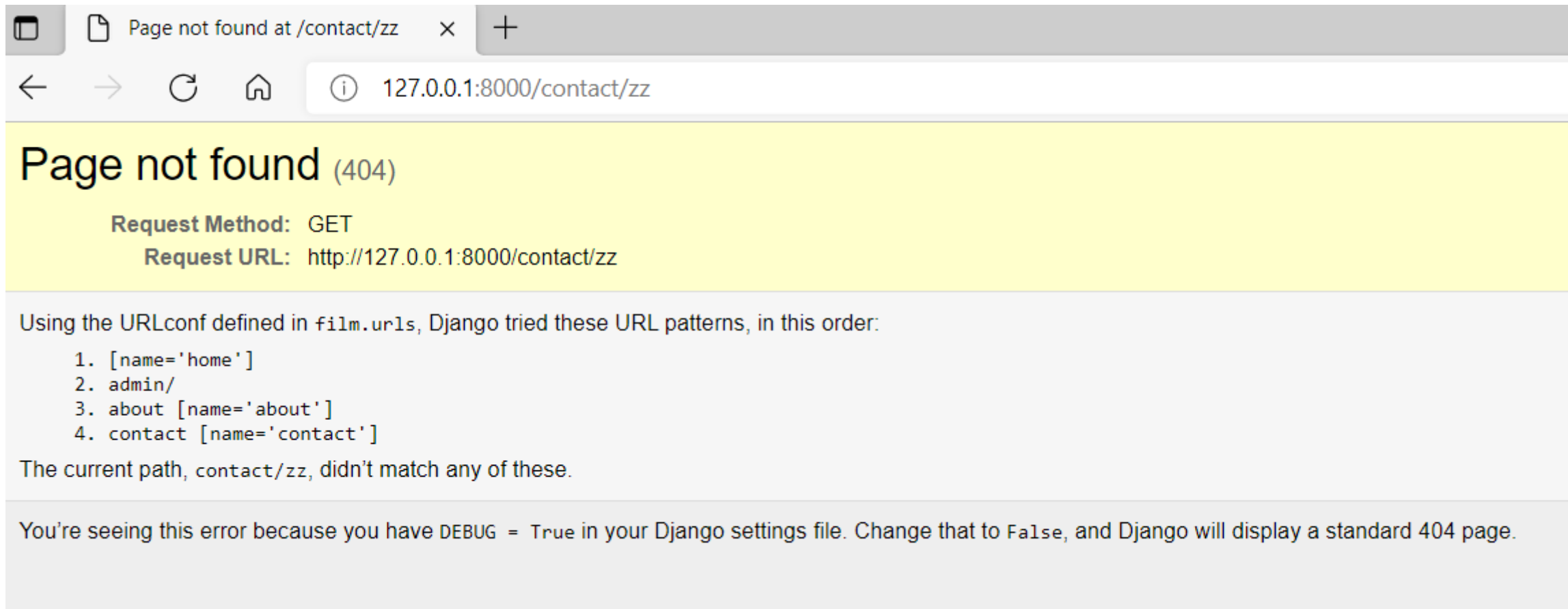
Это страница с информацией о сайте



Здесь размещены контакты



Попытка набрать иной адрес приведет к ошибке



Функции `path()` и `re_path()`

При использовании функции `path()` запрошенный путь должен абсолютно точно соответствовать указанному в маршруте адресу URL. Иначе есть вероятность возникновения ошибки.

В пакете `django.urls` имеется альтернативная функция `re_path()`. Функция `re_path()` позволяет задавать адреса URL посредством регулярных выражений.



Функция re_path и регулярные выражения

```
urlpatterns = [  
    path('', views.index) ,  
    re_path(r'^about', views.about) ,  
    re_path(r'^contact', views.contact) ,  
]
```

Выражение `^about` указывает на то, что адрес может не точно соответствовать, а только начинаться с about.



File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino - urls

Kino > film > film > urls

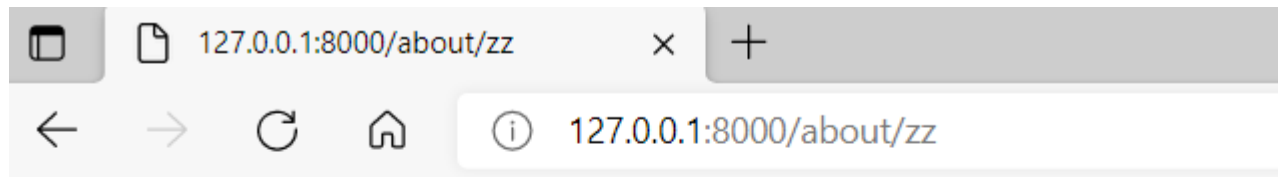
Project

- Project
 - Kino C:\Users\user\PycharmProjects\Kino
 - film
 - artist
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py
 - film
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - db.sqlite3
 - manage.py
 - venv library root
 - main.py
 - External Libraries

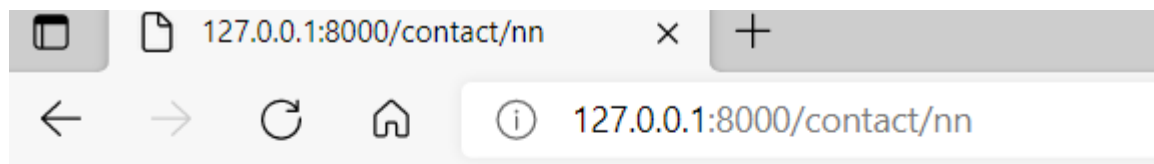
urls

```
11      2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12      Including another URLconf
13      1. Import the include() function: from django.urls import include, path
14      2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15      """
16      from django.contrib import admin
17      from django.urls import path, re_path
18      from artist import views
19
20      urlpatterns = [
21          path('', views.index, name='home'),
22          path('admin/', admin.site.urls),
23          re_path(r'^about', views.about, name='about'),
24          re_path(r'^contact', views.contact, name='contact')
25      ]
26
```

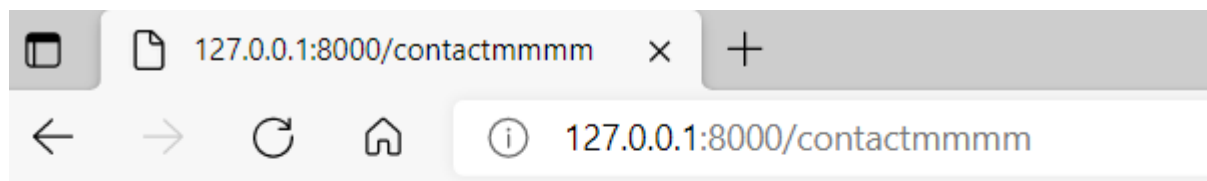




Это страница с информацией о сайте



Здесь размещены контакты



Здесь размещены контакты

```
def index(request):  
    return HttpResponse('Привет, Артист! Это работает представление index')  
  
def about(request):  
    return HttpResponse(' Это страница с информацией о сайте')  
  
def contact(request):  
    return HttpResponse('Здесь размещены контакты')
```



Регулярные выражения

Регулярные выражения задают шаблоны, которые используются для поиска соответствующего фрагмента текста и сопоставления символов.

Например:

- для определения нужного формата (телефонного номера или email-адреса;
- для разбивки строк на подстроки;
- для поиска, замены и извлечения символов;
- для быстрого выполнения нетривиальных операций.



Синтаксис регулярных выражений

Спец. символ	Зачем нужен
.	Задаёт один произвольный символ
[]	Заменяет символ из квадратных скобок
-	Задаёт один символ, которого не должно быть в скобках
[^]	Задаёт один символ из не содержащихся в квадратных скобках
^	Обозначает начало последовательности
\$	Обозначает окончание строки
*	Обозначает произвольное число повторений одного символа
?	Обозначает строго одно повторение символа
+	Обозначает один символ, который повторяется несколько раз
	Логическое ИЛИ . Либо выражение до, либо выражение после символа
\	Экранирование. Для использования метасимволов в качестве обычных
()	Группирует символы внутри
{ }	Указывается число повторений предыдущего символа



дополнительные конструкции, которые позволяют сокращать регулярные выражения:

- **\d** — соответствует любой одной цифре и заменяет собой выражение **[0-9]**;
- **\D** — исключает все цифры и заменяет **[^0-9]**;
- **\w** — заменяет любую цифру, букву, а также знак нижнего подчёркивания;
- **\W** — любой символ кроме латиницы, цифр или нижнего подчёркивания;
- **\s** — соответствует любому пробельному символу;
- **\S** — описывает любой непробельный символ.



Регулярные выражения в Python

В Python при работе с регулярными выражениями импортируется модуль `re`: `import re`

Наиболее популярные методы модуля `re` :

- `re.match()` - ищет по заданному шаблону в начале строки
- `re.search()` - ищет не только в начале строки
- `re.findall()` - возвращает список всех найденных совпадений
- `re.split()` - разделяет строку по заданному шаблону
- `re.sub()` - ищет шаблон в строке и заменяет его на подстроку
- `re.compile()` - собирает регулярное выражение в отдельный объект, который потом может использоваться для поиска



Примеры регулярных выражений

Пример 1. Проверить формат телефонных номеров в списке.

Номер должен быть длиной 10 знаков и начинаться с 8 или 9.

```
import re
```

```
li = ['9999999999', '999999-999', '99999x9999']
```

```
for val in li:
```

```
    if re.match(r'[8-9]{1}[0-9]{9}', val) and len(val) == 10:
```

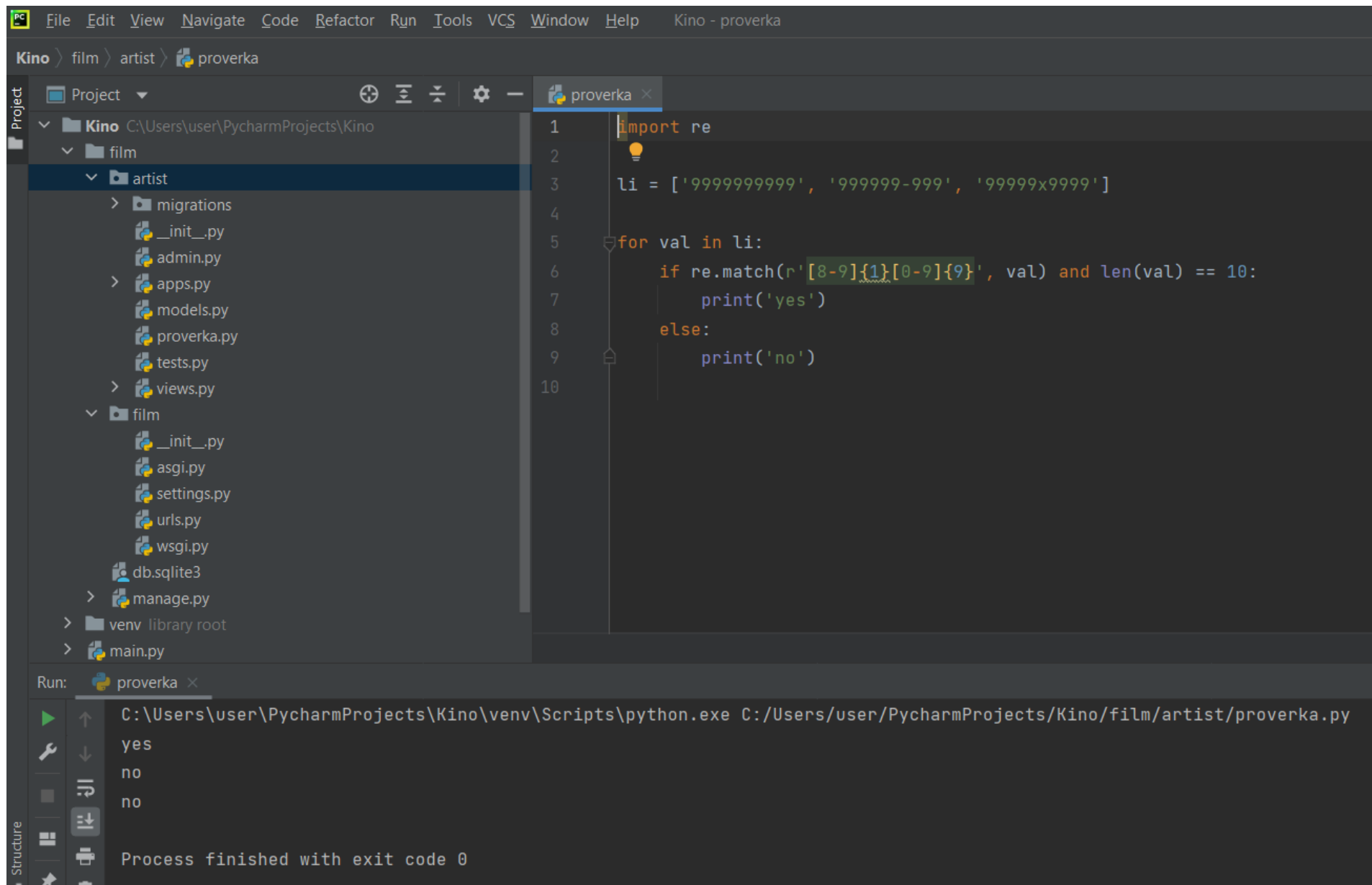
```
        print('yes')
```

```
    else:
```

```
        print('no')
```

Результат: yes no no





Пример 2

#Извлечь дату из строки

#Применим \d чтобы извлечь цифры:

```
result = re.findall(r'\d{2}-\d{2}-\d{4}', 'Ivan 12345 10-07-2012, Andrey 33-256 18-12-2014, Mike 65785 19-05-2008')  
print(result)
```

#Применим скобки, чтобы извлечь только год:

```
result = re.findall(r'\d{2}-\d{2}-(\d{4})', 'Ivan 12345 10-07-2012, Andrey 33-256 18-12-2014, Mike 65785 19-05-2008')  
print(result)
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino - proverka

Kino > film > artist > proverka

Project

- Project
- Kino C:\Users\user\PycharmProjects\Kino
 - film
 - artist
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - proverka.py
 - tests.py
 - views.py
 - film
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - db.sqlite3
 - manage.py
 - venv library root
 - main.py

```
1 import re
2
3 li = ['9999999999', '999999-999', '99999x9999']
4
5 for val in li:
6     if re.match(r'[8-9]{1}[0-9]{9}', val) and len(val) == 10:
7         print('yes')
8     else:
9         print('no')
10
11 result = re.findall(r'\d{2}-\d{2}-\d{4}', 'Ivan 12345 10-07-2012, Andrey 33-256 18-12-2014, Mike 65785 19-05-2008')
12 print(result)
13
14 result = re.findall(r'\d{2}-\d{2}-(\d{4})', 'Ivan 12345 10-07-2012, Andrey 33-256 18-12-2014, Mike 65785 19-05-2008')
15 print(result)
16
17
```

Run: proverka

C:\Users\user\PycharmProjects\Kino\venv\Scripts\python.exe C:/Users/user/PycharmProjects/Kino/film/artist/proverka.py

yes
no
no

['10-07-2012', '18-12-2014', '19-05-2008']
['2012', '2014', '2008']

Process finished with exit code 0



Варианты возможных сопоставлений шаблонов адресов и запросов:

Адрес	Запрос
<code>r'^\$'</code>	<code>http://127.0.0.1/</code> (корень сайта)
<code>r'^about'</code>	<code>http://127.0.0.1/about/</code> или <code>http://127.0.0.1/about/contact</code>
<code>r'^about/contact'</code>	<code>http://127.0.0.1/about/contact</code>
<code>r'^products/\d+/'</code>	<code>http://127.0.0.1/products/23/</code> или <code>http://127.0.0.1/products/6459/abc</code> Но не соответствует запросу <code>http://127.0.0.1/products/abc/</code>
<code>r'^products/\D+/'</code>	<code>http://127.0.0.1/products/abc/</code> или <code>http://127.0.0.1/products/abc/123</code> Не соответствует запросу <code>http://127.0.0.1/products/123/</code> или <code>http://127.0.0.1/products/123/abc</code>
<code>r'^products/phones tablets/'</code>	<code>http://127.0.0.1/products/phones/1</code> или <code>http://127.0.0.1/products/tablets/</code> Не соответствует запросу <code>http://127.0.0.1/products/clothes/</code>



<code>r'^products/phones tablets/'</code>	<p><code>http://127.0.0.1/products/phones/1</code> или <code>http://127.0.0.1/products/tablets/</code></p> <p>Не соответствует запросу <code>http://127.0.0.1/products/clothes/</code></p>
<code>r'^products/\w+'</code>	<p><code>http://127.0.0.1/products/abc/</code> или <code>http://127.0.0.1/products/123/</code></p> <p>Не соответствует запросу <code>http://127.0.0.1/products/abc-123</code></p>
<code>r'^products/[-\w]+/'</code>	<code>http://127.0.0.1/products/abc-123</code>
<code>r'^products/[A-Z]{2}/'</code>	<p><code>http://127.0.0.1/products/RU</code></p> <p>Не соответствует запросам <code>http://127.0.0.1/products/Ru</code> или <code>http://127.0.0.1/products/RUS</code></p>



Подробнее о регулярных выражениях можно посмотреть по ссылкам :

<https://docs.python.org/3.8/library/re.html> - документация по регулярным выражениям в Python 3

<https://tproger.ru/translations/regular-expression-python/>
практическое применение регулярных выражений



Конец части 1. Продолжение следует..

