

Фреймворки Flask и FastAPI





Знакомство и содержание курса



Алексей Петренко

Freelance developer

- 💥 Python-developer, Team Leader
- 💥 Опыт программирования более 20 лет
- 💥 Разрабатывал IT-решения для Министерства обороны РФ
- 💥 Преподаватель GeekBrains с 2018 года



План курса

- 1 Знакомство с Flask
- 2 Погружение во Flask
- 3 Дополнительные возможности Flask

- 4 Введение в многозадачность
- 5 Знакомство с FastAPI
- 6 Дополнительные возможности FastAPI

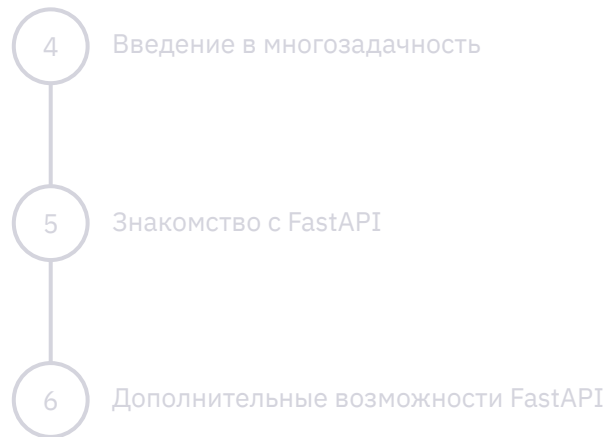
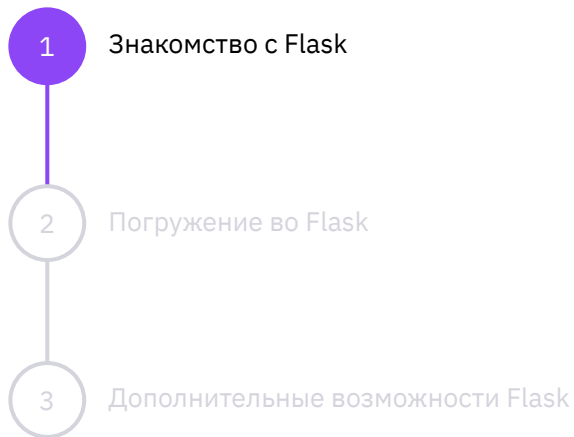
Знакомство с Flask

Урок 1





План курса











Содержание урока



Что будет на уроке сегодня

-  Узнаем о фреймворке Flask
-  Разберёмся в его установке и настройке для первого запуска
-  Изучим работу функций представлений - view
-  Узнаем о способах передачи html кода от сервера клиенту
-  Изучим работу с шаблонизатором Jinja
-  Разберёмся с наследованием шаблонов



Что такое Flask

Что такое Flask?

Flask — это микрофреймворк для Python, созданный в 2010 году разработчиком по имени Армин Ронахер. Приставка «микро» говорит о том, что Flask действительно маленький. У него в комплекте нет ни набора инструментов, ни библиотек, которыми славятся другие популярные фреймворки. Но он создан с потенциалом для расширения. Во фреймворке есть набор базовых возможностей, а расширения отвечают за все остальное. «Чистый» Flask не умеет подключаться к базе данных, проверять данные формы и так далее. Для добавления этих функций нужно использовать расширения. Это помогает использовать только те из них, которые на самом деле нужны.

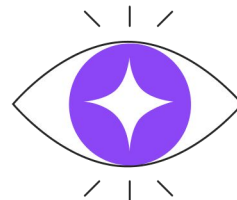




Hello world!



Установка Flask и обязательный компонентов



Создаём виртуальное окружение и запускаем `pip install Flask`

- Werkzeug — набор инструментов WSGI, стандартного интерфейса Python для развёртывания веб-приложений и взаимодействия между ними и различными серверами разработки. Отвечает за роутинг, обработку запросов и ответов, а также предоставляет такие возможности, как debugger и reloader.
- Jinja2 — движок шаблонов и одновременно современный язык шаблонов для Python, созданный по образцу шаблонов Django. Он быстр, широко используется и безопасен, благодаря дополнительной среде отрисовки шаблона в песочнице.
- Click — фреймворк для написания приложений командной строки. Он предоставляет консольную команду flask и позволяет добавлять пользовательские команды управления.
- MarkupSafe поставляется с Jinja. MarkupSafe исключает ненадежный ввод при рендеринге шаблонов, чтобы избежать атак путем внедрения нежелательного кода.
- itsDangerous - дополнение, которое подписывает данные для обеспечения их целостности. Он используется для защиты cookie файлов в сеансе Flask.



Первый проект



Первое веб-приложение

Рассмотрим код в IDE



Первый запуск

```
flask --app  
lesson_1/project run
```



Первая оптимизация

Создаём wsgi.py



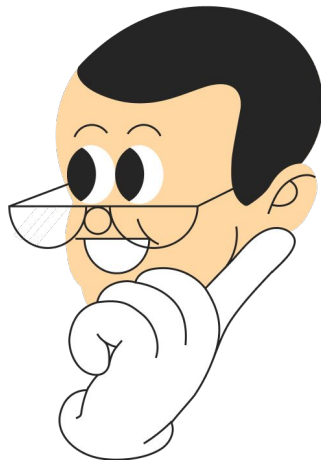
Устройство view функций



Устройство view функций

Разберём подробнее как работают функции представления на примерах:

```
@app.route('/Николай/')  
def nike():  
    return 'Привет, Николай!'
```



Множественное декорирование

Одна функция-представление может быть декорирована несколькими декораторами

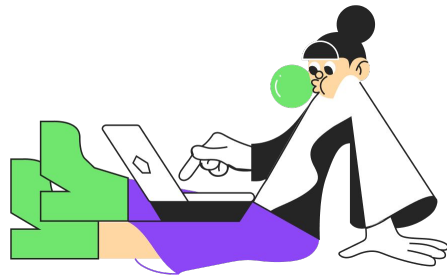
```
@app.route('/Фёдор/')  
@app.route('/Fedor/')  
@app.route('/Федя/')  
def fedor():  
    return 'Привет, Феодор!'
```



Прописываем логику обработки URL

Типы переменных при передаче в функцию

- string - (по умолчанию) принимает текст без слеша
- int - принимает позитивные целые числа
- float - принимает позитивные числа с плавающей точкой
- path - как string, но принимает слеша
- uuid - принимает строки UUID (универсальный уникальный идентификатор)





Выводим HTML



Многостраничный текст с тегами

```
@app.route('/text/')  
def text():  
    return """<p>Вот не думал, не гадал,<br>Программистом взял и  
стал.<br>Хитрый знает он язык,<br>Он к другому не привык.</p>"""
```





Рендеринг HTML файла

```
from flask import Flask
from flask import render_template
```

```
app = Flask(__name__)
```

```
@app.route('/index/')
def html_index():
    return render_template('index.html')
```





Шаблонизатор Jinja



Пробрасываем контекст из представления в шаблон

Передать ключевые аргументы в шаблон - стандартная практика для Flask и Jinja

```
@app.route('/index/')
def index():
    context = {
        'title': 'Личный блог',
        'name': 'Харитон',
    }
    return render_template('index.html', **context)
```



Вывод контекста в шаблоне

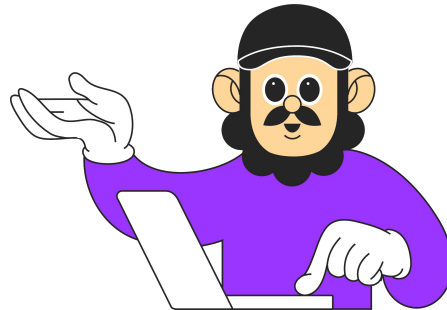
Помимо простой передачи {{ переменной }} можно использовать ветвления и циклы в шаблонах

Условный оператор:

```
{% if логика %}  
    значение  
{% elif логика %}  
    значение  
{% else %}  
    значение  
{% endif %}
```

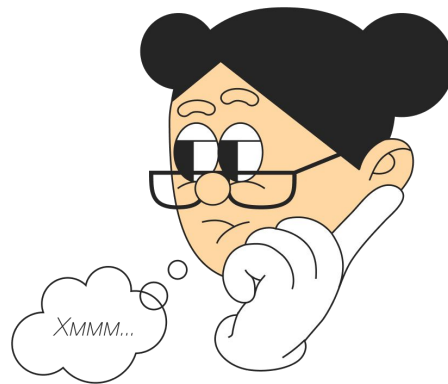
Цикл:

```
{% for item in item_list %}  
    {{ item }}  
{% endfor %}
```



Вывод сложных структур в цикле

```
{% for user in users %}
    <h2>{{ user.name }}</h2>
    <p>{{ user.mail }}</p>
    <p>{{ user.phone }}</p>
{% endfor %}
```





Наследование шаблонов



Базовый и дочерние шаблоны

Дочерние шаблоны расширяют содержимое базового шаблона

Базовый шаблон

```
...  
{% block content %}  
    Страница не заполнена  
{% endblock %}  
...
```

Дочерний шаблон







```
{% extends 'base.html' %}  
  
{% block content %}  
    <h1 class="display-2">Страница заполнена</h1>  
{% endblock %}
```



Итоги занятия



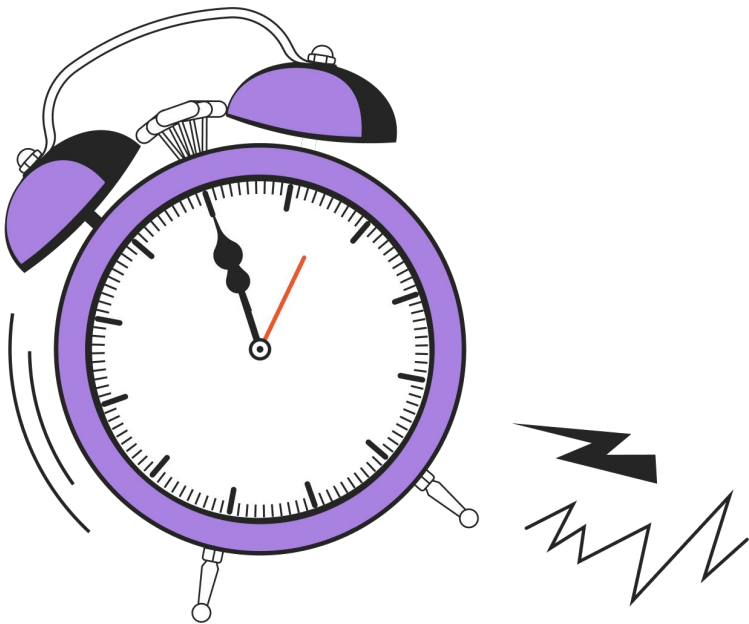
На этой лекции мы

-  Узнали о фреймворке Flask
-  Разобрались в установке и настройке Flask для первого запуска
-  Изучили работу функций представлений - view
-  Узнали о способах передачи html кода от сервера клиенту
-  Изучили работу с шаблонизатором Jinja
-  Разобрались с наследованием шаблонов



Задание

1. Для закрепления материалов лекции попробуйте самостоятельно набрать и запустить демонстрируемые примеры.





Спасибо за внимание