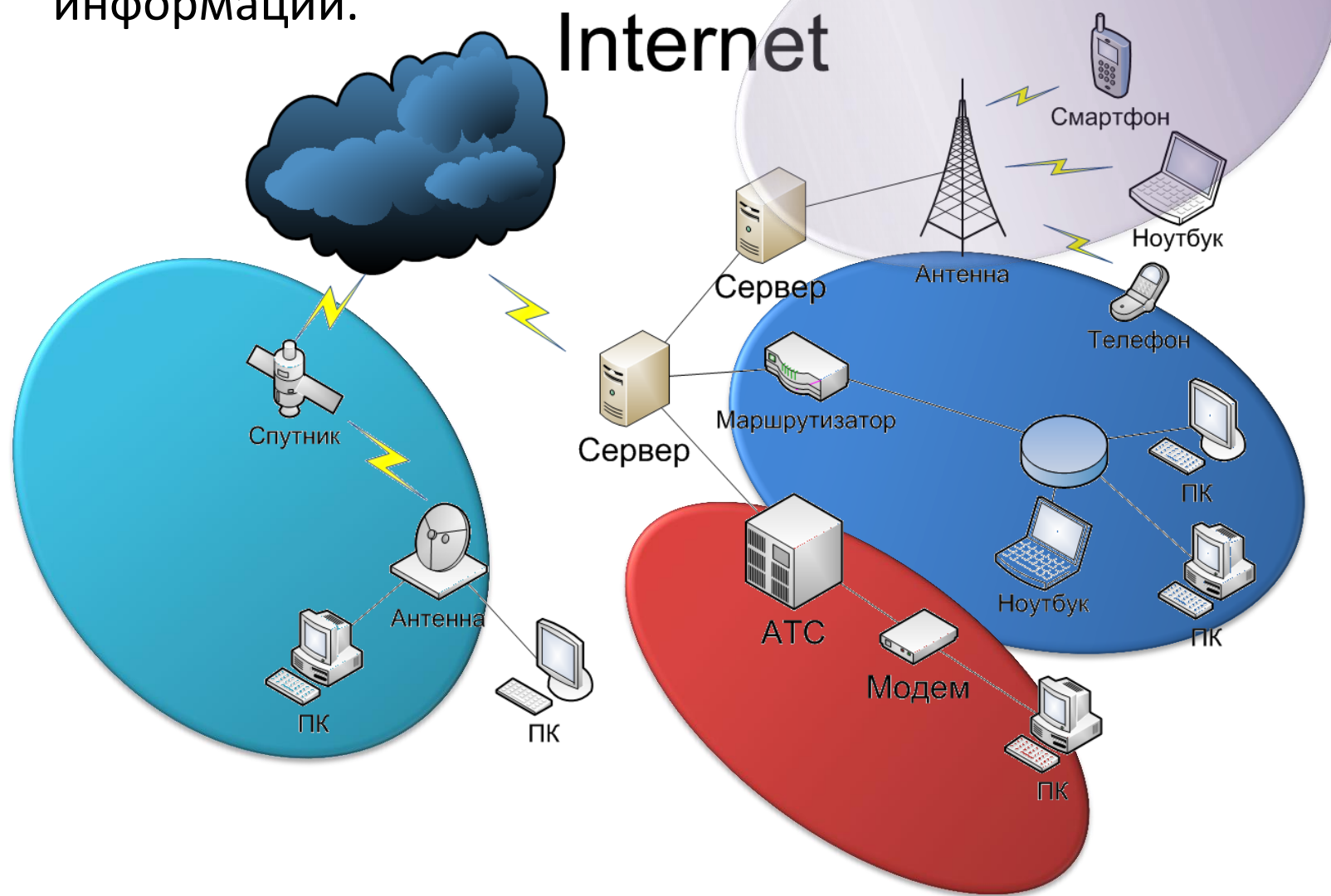


# Python и WEB

Емельянов А. А.  
login-const@mail.ru

# Что такое интернет?

- Интернёт — всемирная система объединённых компьютерных сетей для хранения и передачи информации.



# Жизнь одного запроса...

- Работа в интернете сводится к простому алгоритму действий:
  - Вводим в адресную строку браузера требуемый адрес.
  - Маршрутизатор ведет запрос к серверу, на котором хранится запрашиваемая информация (сайт).
  - DNS-сервер преобразует путь в IP адрес.
  - Мы скачиваем или изучаем нужный файл с компьютера, находящегося в любом конце света.

Путь: ПК → Роутер → DNS-сервер → Сервер сайта → DNS-сервер → Роутер → ПК

# Протокол обмена данными http

- **HTTP** — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Аббревиатура HTTP расшифровывается как *HyperText Transfer Protocol*, «протокол передачи гипертекста». В соответствии со спецификацией [OSI](#), HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации [RFC 2616](#).

# Ответы сервера (коды состояния http)

- **Код состояния HTTP** — часть первой строки ответа сервера при запросах по протоколу [HTTP](#). Он представляет собой целое число из трёх десятичных цифр. Первая цифра указывает на **класс состояния**. За [кодом ответа](#) обычно следует отвлечённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.
- [1xx: Informational](#) (информационные)
- [2xx: Success](#) (успешно):
  - [200 OK](#) («хорошо»)
  - [201 Created](#) («создано»)
  - [202 Accepted](#) («принято»)
- [3xx: Redirection](#) (перенаправление)
- [4xx: Client Error](#) (ошибка клиента):
  - [400 Bad Request](#) («плохой, неверный запрос»);
  - [404 Not Found](#) («не найдено»)
- [5xx: Server Error](#) (ошибка сервера):
  - [500 Internal Server Error](#) («внутренняя ошибка сервера»)
  - [501 Not Implemented](#) («не реализовано»)
  - [502 Bad Gateway](#) («плохой, ошибочный шлюз»)
  - [504 Gateway Timeout](#) («шлюз не отвечает»)

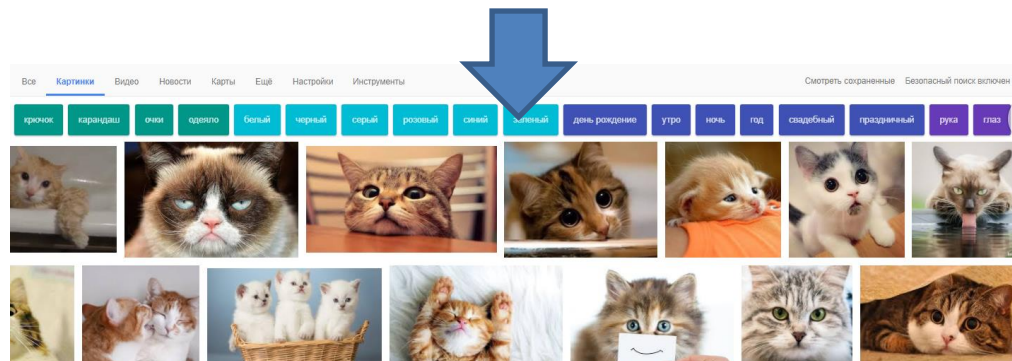
# Метод get

- Используется для запроса содержимого указанного ресурса.
- Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:– GET:

[https://www.google.ru/search?q=%D0%BA%D0%BE%D1%82%D0%B8%D0%BA%D0%B8&newwindow=1&safe=active&source=Inms&tbm=isch&sa=X&ved=0ahUKEwiAwPWYqqzXAhXiQZoKHWBbC3YQ\\_AUICigB&biw=1777&bih=854](https://www.google.ru/search?q=%D0%BA%D0%BE%D1%82%D0%B8%D0%BA%D0%B8&newwindow=1&safe=active&source=Inms&tbm=isch&sa=X&ved=0ahUKEwiAwPWYqqzXAhXiQZoKHWBbC3YQ_AUICigB&biw=1777&bih=854)



[https://www.google.ru/search?q=котики&newwindow=1&safe=active&source=Inms&tbm=isch&sa=X&ved=0ahUKEwiAwPWYqqzXAhXiQZoKHWBbC3YQ\\_AUICigB&biw=1777&bih=854](https://www.google.ru/search?q=котики&newwindow=1&safe=active&source=Inms&tbm=isch&sa=X&ved=0ahUKEwiAwPWYqqzXAhXiQZoKHWBbC3YQ_AUICigB&biw=1777&bih=854)



# Метод post

- Применяется для передачи пользовательских данных заданному ресурсу. При этом передаваемые данные включаются в тело запроса. С помощью метода POST обычно загружаются файлы на сервер.
- В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты.
- При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием [URI](#) нового ресурса в заголовке Location.
- Сообщение ответа сервера на выполнение метода POST не кэшируется.

# Запрос к google в python

```
from requests import get, post
import urllib
import urllib.parse
from bs4 import BeautifulSoup
from IPython.core.display import display, HTML

def google(query):
    template = 'https://www.google.ru/search?{'
    url = template.format(urllib.parse.urlencode({'q' : query}))
    req = get(url)

    assert req.status_code == 200, 'request failed'
    soup = BeautifulSoup(req.text, "lxml")

    for i, li in enumerate(soup.findAll('h3', attrs={'class' : 'r'})):
        link = li.find('a')
        if link:
            link['href'] = 'http://www.google.com' + link['href']
            link = '<i>{}.</i> {}'.format(i, link)
            display(HTML(str(link)))

google('Hello, World!')
```



# Запрос к google в python (результат)

- 
0. [Hello, world! — Википедия](#)
  1. [Hello, world! — Википедия](#)
  2. ["Hello, World!" program - Wikipedia](#)
  3. [Копипаста:Hello, world! — Lurkmore](#)
  4. [HelloWorld](#)
  5. [HELLOWORLD.RU](#)
  6. [HelloWorld \(@Hello\\_WorldLive\) | Twitter](#)
  7. [Hello World, Episode 9: Inside Russia's Creepy, Innovative Internet](#)
  8. [Картинки по запросу Hello, World!](#)
  9. [Hello World – Создаем первую программу \(Руководство по ...](#)
  10. [HelloWorld tickets | Genting Arena, Birmingham](#)

# Метод get в python

```
query = ""
template = 'http://localhost:80/'
url = template.format(urllib.parse.urlencode({'q' : query}))
req = get(url)

assert req.status_code == 200, 'request failed'
soup = BeautifulSoup(req.text, "lxml")

display(HTML(str(soup.findAll("table", attrs={"class": "images_table"})[0])))
```



*youtube.com*

ПОДБОРКА ЛУЧШИХ МОМЕНТОВ  
2016 СМЕШНЫЕ КОТЫ!  
СМЕШНЫЕ **КОТИКИ** НЕЛЕПЫЕ ...  
1280 X 720; 28 КБ - jpg



*elle.ru*

Мимишные ролики с котами  
940 X 627; 211 КБ - jpeg



*ctclove.ru*

**КОТИКИ**, КОШКИ, КОТЫ -  
Love  
632 X 355; 31 КБ - jpg

# Метод post в python

```
In [5]: def login(username, password):  
        req = post("http://localhost:80/login", data={"username": username, "password": password})  
        display(HTML(str(req.text)))  
  
        login("king_menin", "pass")
```

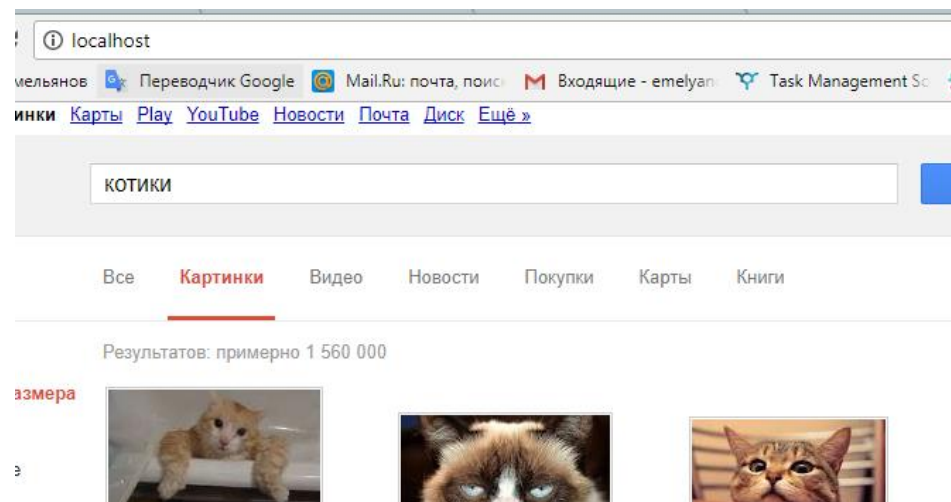
Your login information was correct.

# Свой сервер на python (bottle)

```
from bottle import request, run, route
import urllib
import urllib.parse
import requests
```

```
LOGIN = "king_menin"
PASS = "pass"
```

```
@route('/')
def main():
    query = "q=котики&tbm=isch"
    template = 'https://www.google.ru/search?{}'.format(query)
    url = template.format(urllib.parse.urlencode({'q': query}))
    req = requests.get(url)
    return req.text
```



# Свой сервер на python (обрабатываем get и post)

```
@route('/login')
def login():
    return ...
    <form action="/login" method="post">
        Username: <input name="username" type="text" />
        Password: <input name="password" type="password" />
        <input value="Login" type="submit" />
    </form>
    ...

def check_login(username, password):
    return username == LOGIN and password == PASS

@route('/login', method='POST')
def do_login():
    username = request.forms.get('username')
    password = request.forms.get('password')
    if check_login(username, password):
        return "<p>Your login information was correct.</p>"
    else:
        return "<p>Login failed.</p>"
```



← → ↻ ⓘ localhost/login

📧 Антон Емельянов 🌐 Переводчик Google 📧 Mail.Ru: почта, поиск 📧 Входящие - emelyan

Username:  Password:



← → ↻ ⓘ localhost/login

📧 Антон Емельянов 🌐 Переводчик Google

Your login information was correct.

# Свой сервер на python (а что «сзади»?)

```
In [*]: run(host='localhost', port=80, debug=True)
```

```
Bottle v0.12.13 server starting up (using WSGIRefServer())...
```

```
Listening on http://localhost:80/
```

```
Hit Ctrl-C to quit.
```

```
127.0.0.1 - - [07/Nov/2017 14:29:59] "GET / HTTP/1.1" 200 44530
```

```
C:\ProgramData\Anaconda3\lib\re.py:301: DeprecationWarning: Flags not at the start of the expression ((?m)[urbURB]?(?:' (truncated)
```

```
    p = sre_compile.compile(pattern, flags)
```

```
C:\ProgramData\Anaconda3\lib\sre_parse.py:763: DeprecationWarning: Flags not at the start of the expression \{\{\{((?:((?m)[urbURB (truncated)
```

```
    p = _parse_sub(source, state, sub_verbose)
```

```
127.0.0.1 - - [07/Nov/2017 14:29:59] "GET /textinputassistant/tia.png HTTP/1.1" 404 767
```

```
127.0.0.1 - - [07/Nov/2017 14:29:59] "GET /xjs/_/js/k=xjs.hp.en_US.GLzu9bBYBAQ.O/m=sb_he,d/rt=j/d=1/t=zcms/rs=ACT90oH86F_ziFXid d-SVD1n0xXHECPcpw HTTP/1.1" 404 931
```

```
127.0.0.1 - - [07/Nov/2017 14:30:00] "GET /images/nav_logo229.png HTTP/1.1" 404 759
```

```
127.0.0.1 - - [07/Nov/2017 14:30:00] "GET /xjs/_/js/k=xjs.hp.en_US.GLzu9bBYBAQ.O/m=sb_he,d/rt=j/d=1/t=zcms/rs=ACT90oH86F_ziFXid d-SVD1n0xXHECPcpw HTTP/1.1" 404 931
```

```
127.0.0.1 - - [07/Nov/2017 14:30:00] "GET /client_204?&atyp=i&biw=1600&bih=769&ei=d5kBWtnf04jI6ASQuqfIBQ HTTP/1.1" 404 802
```

```
127.0.0.1 - - [07/Nov/2017 14:33:26] "GET /login HTTP/1.1" 200 240
```

```
127.0.0.1 - - [07/Nov/2017 14:34:28] "POST /login HTTP/1.1" 200 42
```

**FLASK**

- **Model-View-Controller (MVC)**
  - схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.
- **Модель (Model)**
  - предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
- **Представление (View)**
  - отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- **Контроллер (Controller)**
  - интерпретирует действия пользователя, оповещая модель о необходимости изменений.



# Необходимые модули для работы

- flask
- flask\_wtf
- flask\_login

# Пример о flask

- [Тут](#)

# Домашнее задание 8

- Целью этого задания является знакомство с WEB программирование в python.
- **Deadline (получение полных баллов): 10.05.2018**
- **Адрес:** login-const@mail.ru
- Задание состоит эмуляторе посылки сообщений трех метаклассов
- **Текс условия доступен по [ссылке](#).**

**СПАСИБО ЗА ВНИМАНИЕ**