

Основы программирования на Python

Семинар 1

Тёма Стрельцов

О курсе

- Преподаватель -- Тёма Стрельцов (@in_chainz | astreltsov@hse.ru)
- Ассистент -- Саша Безуглый (@Bzgly)
- Семинары будут в формате live coding -- разбираем детально ту или иную тему, вместе решаем задачи и пишем код
- Консультации -- по договоренности с Сашей или мной (лучше с Сашей)
- Любые вопросы безо всякого стеснения пишите в чат -- с радостью ответим :) мемы в чатике тоже приветствуются
- Среда разработки -- на ваш выбор (VsCode/PyCharm/Jupyter/Colab). На семинарах будем пользоваться юпитером, инструкцию по установке выложим на гитхаб

О курсе

- Формула оценки:

$$O_{\text{final}} = 0.1 \cdot O_{\text{quizzes}} + 0.4 \cdot O_{\text{homeworks}} + 0.2 \cdot O_{\text{midterm}} + 0.3 \cdot O_{\text{project}}$$

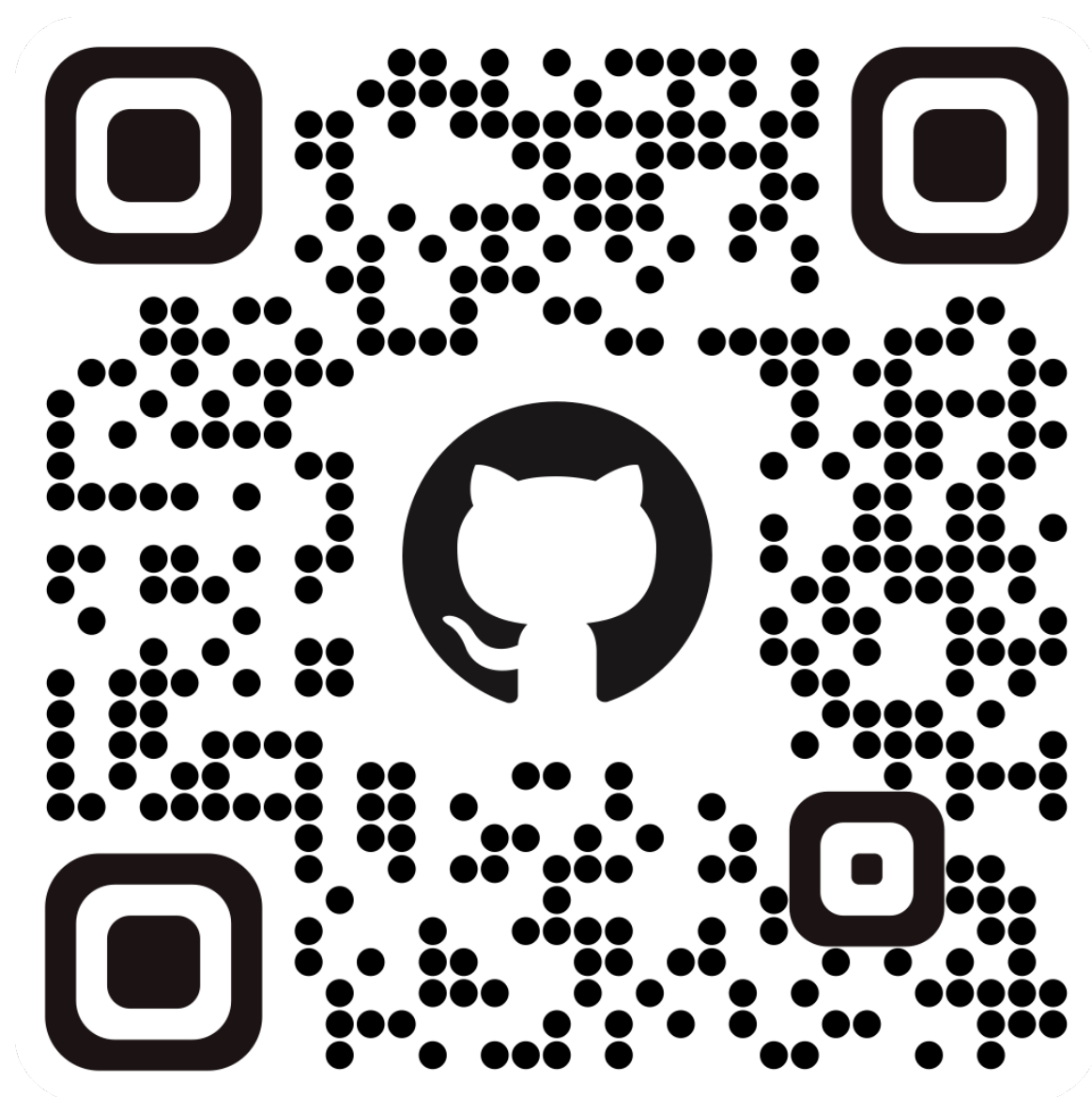
- Округление O_{final} арифметическое
- Самостоялки -- на семинарах, на понимание предыдущей темы ("пятиминутки")
- Домашки -- контесты или задания с ручной проверкой (проверяет ассистент), оценка в формуле -- среднее арифметическое за все дз, (почти) после каждого семинара будем стараться выкладывать контест на тему занятия
- Контрольная -- после примерно 6 семинаров, скорее всего, на осенней сессии
- Проект -- вместо экзамена, будет индивидуальный или групповой (2 человека)

О чуть более неприятном

- За списывание жестко баним, давайте не доводить до записок в УО
- Обсуждать решения домашек можно, но заимствование кода не допускается (отсматриваем как автопроверкой, так и глазами дополнительно). При этом копировать код из открытых источников можно (не забудьте ссылку в комментарии дать)
- Обнуляем как списавшему, так и давшему списать, без разбирательств, кто у кого
- Если что-то не получается, лучше обратитесь к ассистенту/преподавателю, никто вас без помощи не бросит!



Полезные ресурсы



Github курса



Чатик



Канал

Почему Python?

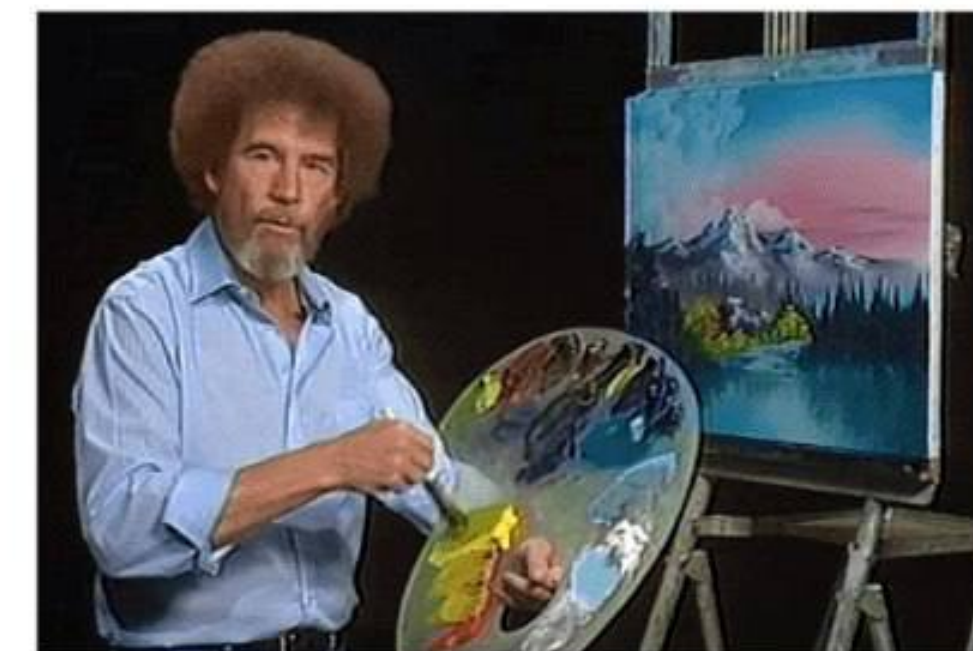
- Очень лаконичный и простой для освоения язык
- Универсален: на нем пишут веб-сервисы, работают с данными, варят всякий ML, делают ботов и тд
- Хороший набор встроенных библиотек (и еще больше в PyPi)
- Достаточно понятно интерпретируемые ошибки (привет C++ и JavaScript)



Сбор данных

RegExp
 $\wedge\{1,8\}$

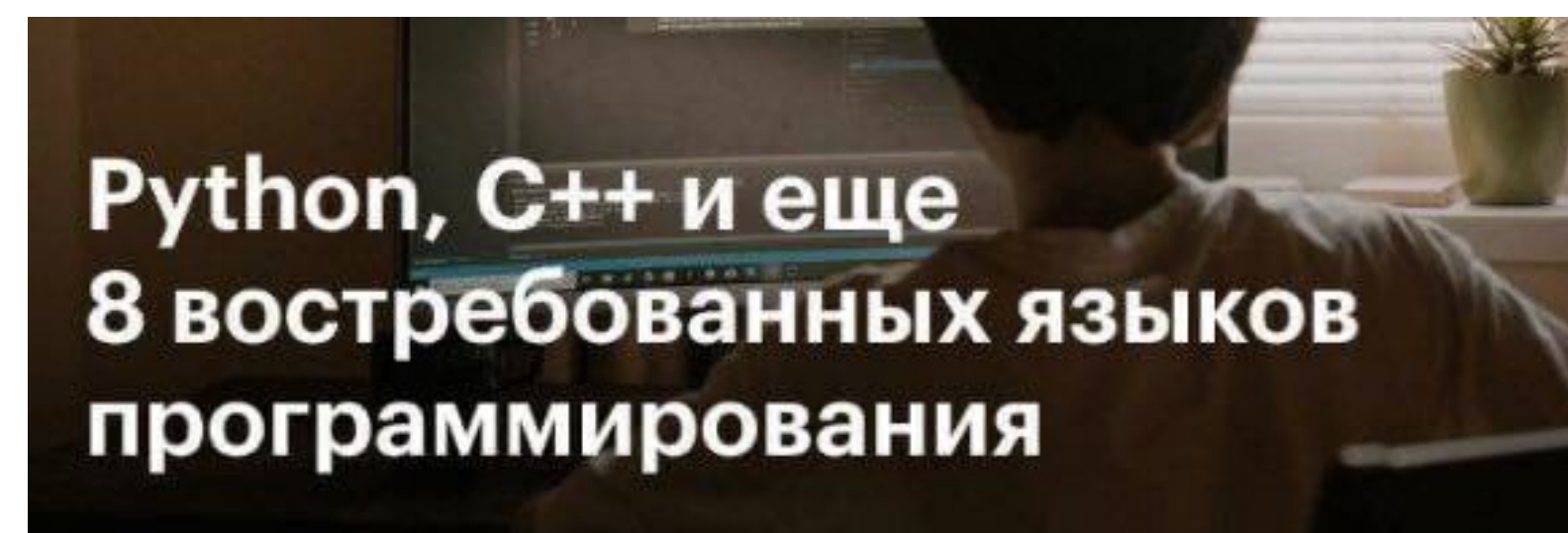
Natural
Language
Processing



Визуализация

Что будем изучать на этом курсе

- Ввод-вывод, типы данных, арифметика
- Условные конструкции, циклы
- Коллекции, строки
- Функции, рекурсия, лямбды, декораторы
- Итераторы, генераторы, `itertools/more_itertools/functools`
- ООП (Аж 2 семинара, надеюсь, даже про метаклассы поговорим)
- Также планируем семинар на свободную тему (предлагайте!)

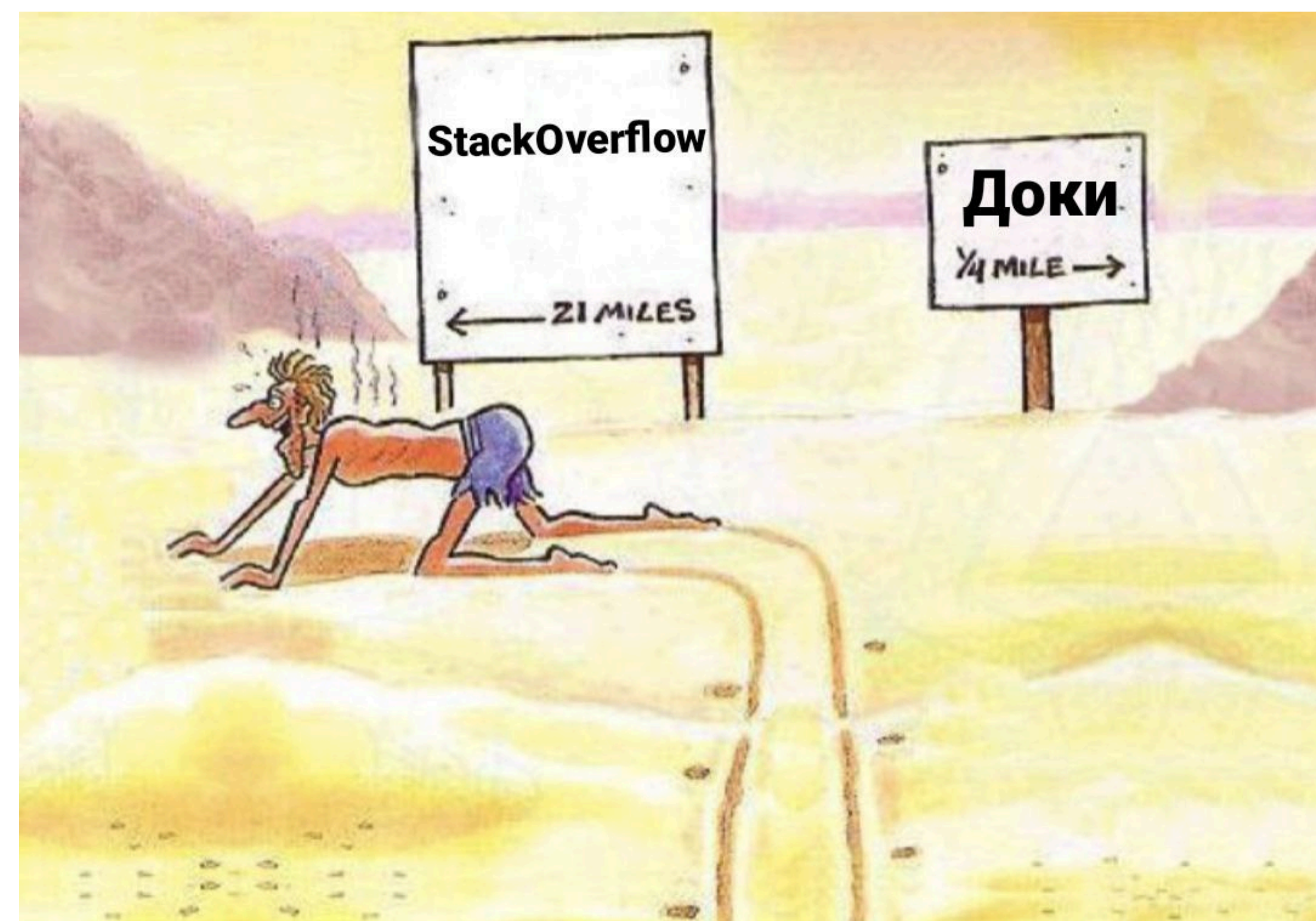


Самые востребованные языки для программистов:

3. C++
2. Python.
1. Грузинский:

Что еще полезно помнить

- Старайтесь писать читабельный код, вам же потом будет проще, особенно когда будете делать проект (PER-8). И помните, что simple is better than complex, but complex is better than complicated (Zen of Python)
- Перед тем, как спросить нас, почему ваше решение не проходит какие-то автотесты, попробуйте сами потестировать его на каких-то данных и краевых случаях
- Не бойтесь делать ошибки, в программировании это нормально
- А также старайтесь как можно больше гуглить по-английски (например, очень полезно искать какую-то информацию на stackoverflow.com) и читать документацию






Zen of Python

```
~ 17s < system ^ 15:14:27
> ipython
Python 3.9.13 (main, May 24 2022, 21:13:51)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.1.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Вопросы?

за сколько становятся мидлом  **Найти**  

Поиск [Картинки](#) [Видео](#) [Карты](#) [Товары](#) [Новости](#) [Переводчик](#) [Все](#)

Добавлены результаты по запросу «за сколько становятся **быдлом**». [Отменить](#)