



Язык Python

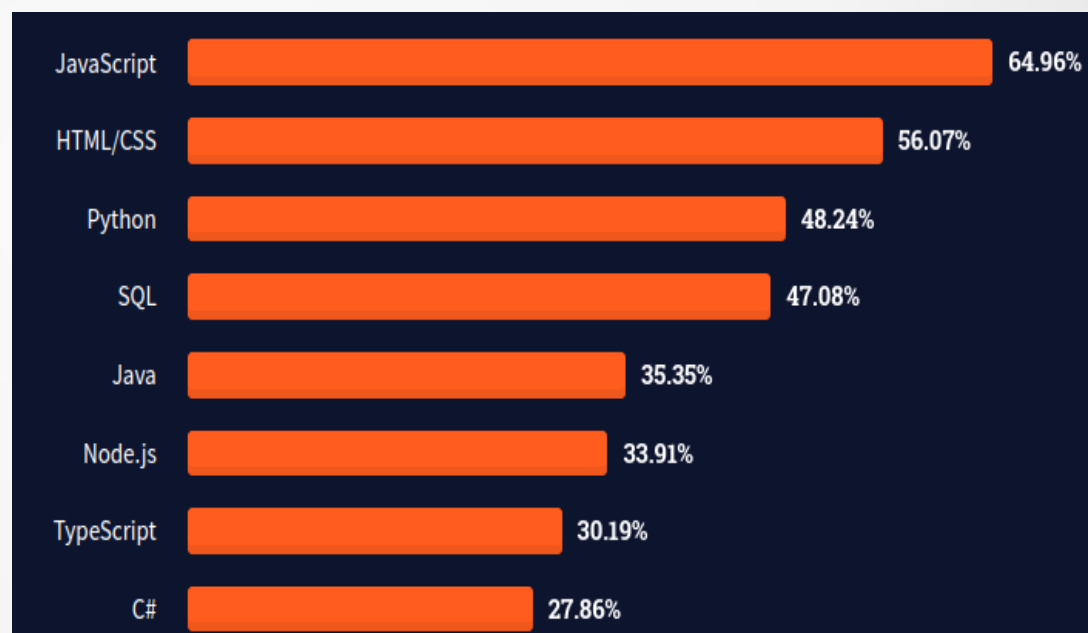
Евгений Борисов

Python. Особенности и возможности

Рейтинг языков программирования TIOBE

Nov 2022	Programming Language	
1		Python
2		C
3		Java
4		C++
5		C#

Рейтинг технологий программирования
StackOverflow



Python. Особенности и возможности

Проект с историей, существует уже более 30 лет.

Поддержка современных технологий в библиотеках.

Открытое сообщество — доступен всем, над разработкой работают энтузиасты со всего мира.

Универсален — подходит почти для любых решений в области программирования.

Мультиплатформенный — есть реализации почти для всех операционных систем и аппаратных платформ.

Python. Особенности и возможности

Язык программирования общего назначения,

Высокоуровневый,

Императивный, объектно-ориентированный,

Строгая динамическая типизация

Python. Особенности и возможности

Python — высокоуровневый язык программирования.

Высокоуровневый язык программирования — оптимизирован для удобства использования, применяются абстракции — структуры данных, набор вспомогательных функций и т.п.

Низкоуровневый язык — оптимизирован для эффективности выполнения, близок к машинному коду и его конструкциям (Assembler).

Python. Особенности и возможности

Python — императивный язык программирования.

Императивный язык - программа это строго упорядоченный список команд для выполнения.

Декларативный язык - программа это описания результата, который мы хотим получить (SQL)

Python. Особенности и возможности

Python - объектно-ориентированный язык программирования, поддерживает процедурный, структурный и функциональный стиль.

Парадигмы (стили) программирования:

Процедурная — программа строго упорядоченный список команд (Assembler, Shell)

Структурная — программа набор подпрограмм, выполняемый в определённом порядке.

Объектно-ориентированная — программа как набор деталей встроенных друг в друга образующих вместе единый механизм.

Функциональная — программа как суперпозиция математических функций.

Python. Особенности и возможности

Python — строго типизированный язык программирования с возможностью динамической типизации.

Строго типизированный язык - определён ограниченный список типов данных

Динамическая типизация - в процессе выполнения программы переменная может связываться с данными разных типов, объявляем переменную не указываем явно, какой тип данных в ней будет содержаться.

Статическая типизация - тип переменной объявляется явно и в процессе выполнения программы он не меняется.

Python. Особенности и возможности

Существуют реализации Python как интерпретатора, так и компилятора.



Интерпретация — программа оптимизируется и выполняется интерпретатором (специальной виртуальной машиной).

Недостаток — может выполняться медленно;

Достоинства — независимость от платформы, меньший размер;



Компиляция — программа преобразуется в машинный код (исполняемый файл), который выполняется аппаратной частью непосредственно.

Недостаток — ограниченная переносимость, ограничения на инструментарий языка;

Достоинства — можно добиться оптимального использования вычислительных ресурсов;



Python. Особенности и возможности

Python имеет очень много разнообразных библиотек и фреймворков

В стандартных (встроенных) библиотеках Python можно найти средства для работы с файлами протоколами Интернета, базами данных и др.

Индекс пакетов для Python

The Python Package Index (PyPI) is a repository of software for the Python programming language.

<http://pypi.org>

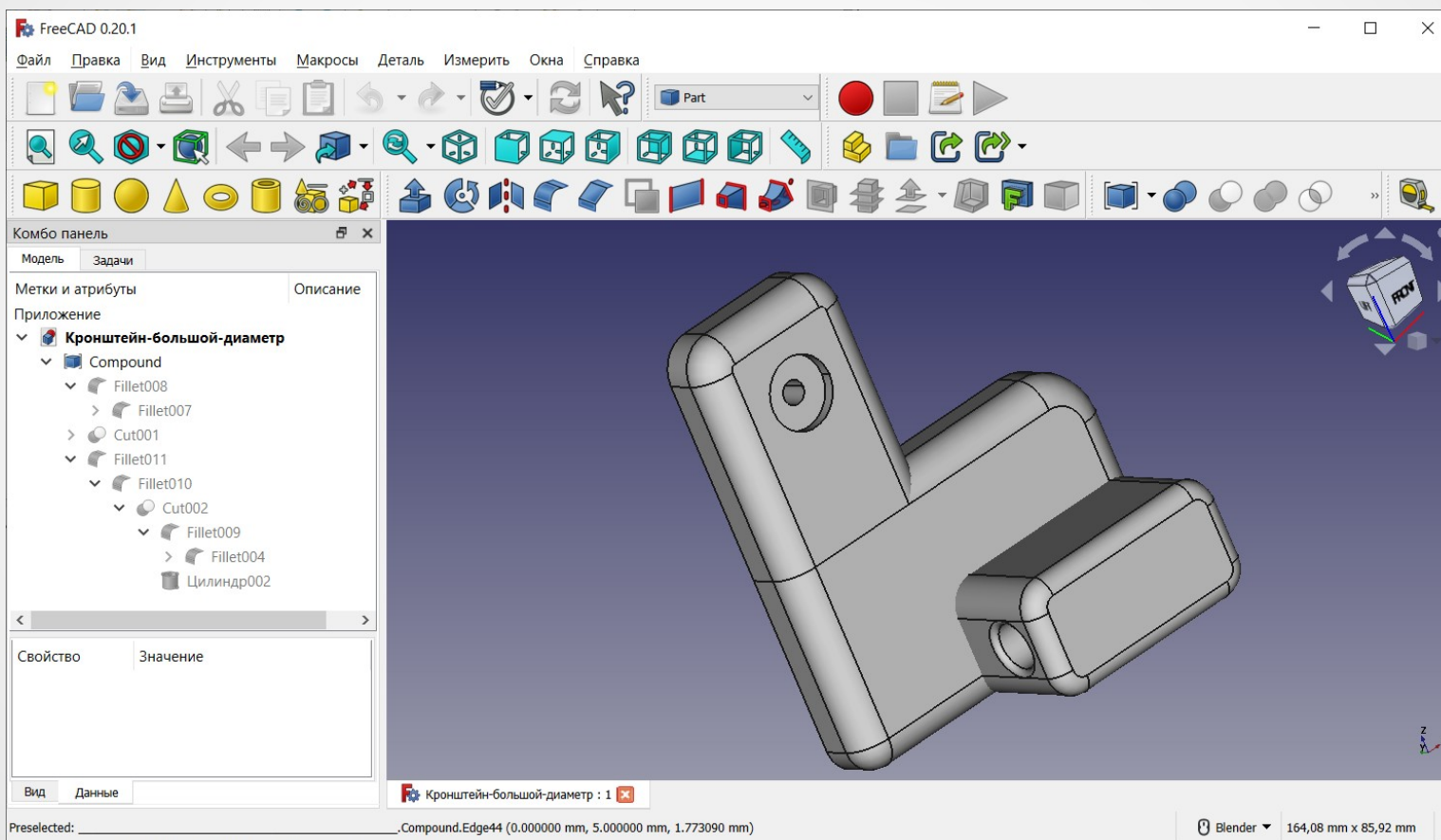
Python. Особенности и возможности

примеры приложений реализованных на Python - World of Tanks - <http://tanki.su>



Python. Особенности и возможности

примеры приложений реализованных на Python - FreeCad - <http://www.freecadweb.org>



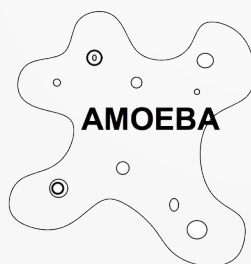
Python. Особенности и возможности



автор первой версии Python
- голландский программист Гвидо ван Россум

центр математики и информатики в Нидерландах,

изначально планировался как язык сценариев для системы Амоеба,
начало проекта в декабре 1989 года



Амоеба — открытая микроядерная
распределённая операционная система,
разработанная группой во главе с Эндрю
Таненбаумом в Амстердамском свободном
университете. <https://www.cs.vu.nl/pub/amoeba/>



Amoeba



Developed at:

- Vrije Universiteit (Amsterdam)
- (Free University)

In cooperation with:

- Centrum voor Wiskunde en Informatica (Amsterdam)
- (Center for Mathematics and Computer Science)
- Research began in 1980

Python. Особенности и возможности

20 февраля 1991 года был опубликован код языка Python через сеть Usenet.

Так появилась первая версия языка с номером 0.9.0

В языке уже присутствовали:

- основные типы данных (list, dict, str),
- поддержка модулей (пакеты подпрограмм),
- классы с наследованием,
- обработка исключений,

В январе 1994 года вышла Python 1.0

Python. Особенности и возможности



Гвидо назвал свой язык в честь комедийного телесериала «Летающий цирк Монти Пайтона»



логотип просуществовал до 2006 года
и изменился только потому,
что пользователи ассоциировали Python со змеями



Why Python?

Python. Особенности и возможности

29 июня 1994 года вышла статья Майкла Маклей из Национального института стандартов и технологий США (NIST) «Если бы Гвидо сбил автобус?»

<https://legacy.python.org/search/hypermail/python-1994q2/1040.html>

Публикация затронула проблему зависимости Python-сообщества от решений Гвидо ван Россума

в 1995 году была создана Python Software Foundation — некоммерческая организация, которая должна была отвечать за защиту и развитие языка Python.

Контроль за соблюдением порядка осуществляет совет руководителей — временный орган, который регулярно переизбирается и состоит из пяти человек,

Гвидо ван Россум получил шуточный титул великодушного пожизненного диктатора (Benevolent Dictator For Life).

В 2018 году Гвидо ван Россум отказался от титула BDFL, поддержал других кандидатов и сделал язык Python полностью независимой технологией, которая больше не нуждается в опеке основателя.

Python. Особенности и возможности

3 декабря 2008 года выходит Python 3.0

Большое количество нововведений не позволяло третьей и второй версиям быть полностью совместимыми.

До 2020 вторая и третья ветка развивались параллельно,

С конца 2020 поддержка второй ветки была официально завершена

Python. Особенности и возможности

Python Enhancement Proposals (PEPs)

<https://peps.python.org>

Предложения по улучшению Python - официальная документация языка.

Позиции из списка документов PEPs открыто обсуждаются сообществом Python.

PEP 8 Style Guide for Python Code / Руководство по оформлению кода.

PEP 13 Python Language Governance / Список руководителей проекта.

Python. Особенности и возможности

PEP20 - 19 правил по улучшению языка Питон от Тима Петерса,

1. Красивое лучше уродливого.
2. Явное лучше неявного.
3. Простое лучше сложного.
4. Сложное лучше запутанного.
5. Развёрнутое лучше вложенного.
6. Разреженное лучше плотного.
7. Читаемость имеет значение.
8. Особые случаи не настолько особые, чтобы нарушать правила.
9. При этом практичность важнее безупречности.
10. Ошибки не должны замалчиваться.
11. Если не замалчиваются явно.
12. Встретив двусмысленность, отбрось искушение угадать.
13. Должен существовать один - и, желательно, только один – очевидный способ сделать что-то.
14. Хотя этот способ поначалу может быть и не очевиден, если вы не голландец.
15. Сейчас лучше, чем никогда.
16. Хотя никогда часто лучше, чем *прямо* сейчас.
17. Если реализацию сложно объяснить – идея точно плоха.
18. Если реализацию легко объяснить – возможно, идея хороша.
19. Пространства имен – отличная штука! Будем использовать их чаще!

Python: дистрибутивы



CPython

Anaconda (Miniconda)



Python: IDE

IDLE

iPython / Jupyter

PyCharm

Visual Studio Code

Eclipse + PyDev

Vim

Apache Zeppelin



IP[y]:



Python: что почитать?

SoloLearn : Python

Python Help : Tutorial

Sebastian Raschka Python Machine Learning

Python: типы данных

Логические

Списки

Числовые

Множества

Строки

Словари

None

Python: тип данных логический

Boolean Type:

True

False

Python: типы данных числовые

Numeric Type:

int – целое число

7

float – число с плавающей точкой

7.5, 75e-1

Python: тип данных строки

Text Sequence Type

'привет'

"медвед"

**'''превед
Медвед'''**

Python: типы данных списки

Sequence Type:

list – список

[1, 2, 'a', [4,'a', 5,] ,]

tuple – кортеж

(1, 2, 'a',)

Python: типы данных множества

Set Types:

set – множество

set([1,2,2,3,4,2,3,4]) → {1,2,3,4}

frozenset – неизменяемое множество

Python: типы данных словарь

Mapping Types:

dict – словарь

{'a':1, 'b':2, 'zzz':7,}

Python: изменяемые типы данных

всё есть объекты

присваивание создаёт новый объект

immutable:

int float bool string tuple frozenset

mutable:

list dict set

Python: операции

**Операции с данными:
арифметические, логические,
строковые, битовые**

управление

ЦИКЛЫ

ВВОД / ВЫВОД

Python: операции с данными

присваивание, арифметика и сравнения

`a,b = 1,2`

`a,b = b,a`

`a = 10`

`a += 7`

`a / b`

`a // 3`

`a % 3`

`a - b`

`a + b`

`a * b`

`a**2`

`a<10`

`b<=7`

`a>2`

`a!=b`

`a==1`

Python: операции с данными

ЛОГИЧЕСКИЕ

```
a = True  
b = False
```

```
a or b  
a and b  
not b
```

Python: операции с данными

БИТОВЫЕ

`a = 255`

`b = 7`

`a^b`

`a&b`

`a|b`

`a>>3`

Python: операции с данными

строковые

`s = 'abc'`

`s*3 → 'abccabccabc'`

`s + 'dmr' → 'abccdmr'`

Python: операции управления

```
if not x:  
    print('x')  
elif y:  
    print('y')  
else:  
    print('z')
```

Python: цикл while

```
i=0
while i<5:
    print(i)
    i+=1
```

```
i=0
while i<5:
    i+=1
    if i<3:
        continue
    print(i)
```

```
i=0
while True:
    print(i)
    i+=1
    if i>5:
        break
```

Python: цикл for

```
for x in [1,2,3,4]:  
    print(x)
```

Python: списки (list)

```
s=[1,7,3,4,['a','b']]
```

```
s.append(9)
```

```
s=[1,5,3,4,]
```

```
s.insert(5,'a')
```

```
len(s)    sorted(s)
```

```
s.index(2)
```

```
s[2]  s[2:]  s[2:4]
```

```
2 in s
```

```
s = list(range(10))
```

```
s = [ i/2 for i in range(10) if i!=3 ]
```

Python: кортежи (tuple)

```
c = (1,2,3,5)
```


Python: словари (dict)

```
d = { 'a':1, 'b':44, 'c':45, 'cvc':-1, }
```

```
d['c']→ 45
```

```
d.keys()    d.values()
```

Python: множества (set)

```
s = set([1,2,3,1,3,4,5])
```

```
{1,2,3,4,5}
```

```
s[2] → error
```

операции: & | -

Python: менеджер контекстов (with)

```
with open('temp.txt','r') as f:  
    x = f.read()
```

```
with open('temp.txt','r') as f:  
    x = [ s for s in f.read().split('\n') if s ]
```

Python: функции

```
def myfunc(x,y=1):  
    print(x)  
    return x+1,y/2
```

```
a,b = myfunc(y=5,x=-1)
```

Python: итераторы

объект перечислитель

реализует навигацию по элементам другого объекта

выдаёт следующий элемент `__next__()`

если элементов больше нет
то бросает исключение

```
s='abcdef'
it_s = iter(s)
it_s.__next__()
for c in it_s:
    print(c)
```

```
s='abcdef'
for c in s:
    print(c)
```

Python: генераторы

генерирует последовательность

```
def ones(n):  
    while n > 0:  
        n -= 1  
        yield 1
```

```
for o in ones(4):  
    print(o)
```

Python: функциональное программирование

```
squares = map(lambda x: x * x, [0, 1, 2, 3, 4])
```

```
sum = reduce(lambda a, x: a + x, [0, 1, 2, 3, 4])
```

Python: OOP

```
class Animal:
```

```
    def __init__(self, name, color):  
        self.name = name  
        self.color = color
```

```
class Dog(Wolf):
```

```
    def bark(self):  
        super().bark()  
        print("Woof!")
```

```
    def __repr__(self):  
        return "Dog({})".format(self.name)
```

```
class Wolf(Animal):
```

```
    def bark(self):  
        print("Grr...!")
```


Python: ООП декораторы

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self._allowed = False
```

```
def calculate_area(self):
    return self.width * self.height
```

@classmethod

```
def new_square(cls, side_length):
    return cls(side_length, side_length)
```

@staticmethod

```
def square(a):
    return a**2
```

@property

```
def allowed(self):
    return self._allowed
```

@allowed.setter

```
def allowed(self, value):
    self._allowed = not(value)
```

```
sq = Rectangle.new_square(5)
```

```
print(sq.calculate_area())
```

```
# 25
```

```
sq.allowed=0
```

```
print(sq.allowed)
```

```
# True
```

```
print(Rectangle.square(4))
```

```
# 16
```

Python: модули

```
import numpy as np
```

```
help(np)
```

```
np.__name__
```

```
np.__version__
```

```
from numpy.random import rand
```

Python: менеджер пакетов pip

```
# pip search pep8
```

```
# pip install autopep8
```

```
# pip list
```

```
# pip uninstall autopep8
```

Python: утилиты

показывает места нарушения стиля
pep8 --first main.py

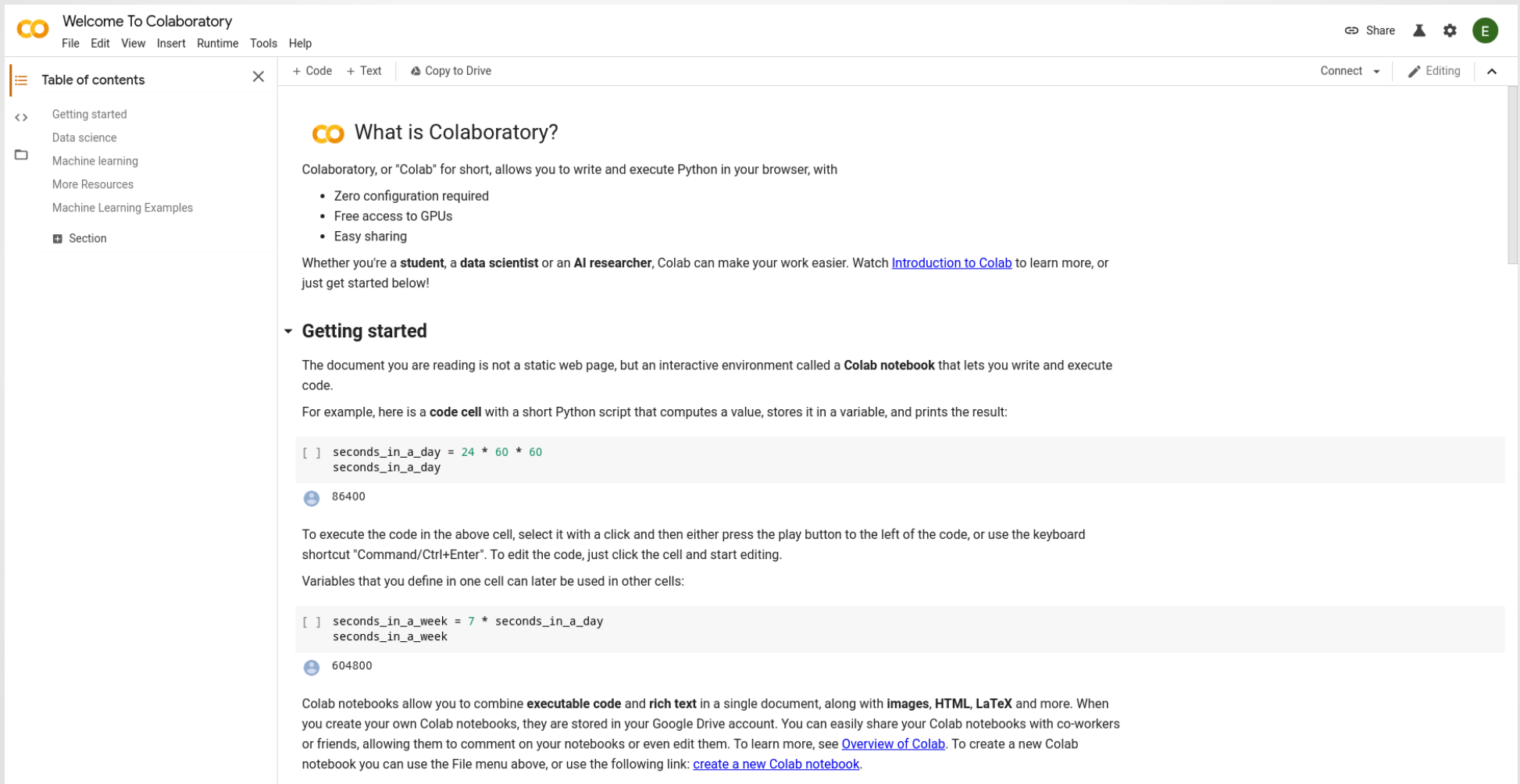
определяет и исправляет нарушения стиля
autopep8 ./ --recursive --in-place -a

форматирует комментарии
docformatter --in-place example.py

универсальная утилита приведения кода к PEP
pyformat

Python: Google Colab

<https://colab.research.google.com/>



The screenshot displays the Google Colaboratory web interface. At the top, there's a navigation bar with the Colab logo, 'Welcome To Colaboratory', and a menu (File, Edit, View, Insert, Runtime, Tools, Help). On the right of the bar are icons for sharing, user profile, settings, and a green 'E' icon. Below the bar, a sidebar on the left contains a 'Table of contents' and a list of links: '<> Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and 'Section'. The main content area is titled 'What is Colaboratory?' and features the Colab logo. It explains that Colaboratory, or 'Colab', allows writing and executing Python in a browser. A bulleted list highlights: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. A paragraph states that Colab can make work easier for students, data scientists, and AI researchers, with a link to 'Introduction to Colab'. A section titled 'Getting started' explains that the document is an interactive 'Colab notebook' and provides an example of a code cell. The code cell contains a Python script to calculate seconds in a day, showing the output '86400'. Below this, it explains how to execute code and edit cells. Another code cell shows calculating seconds in a week, with output '604800'. The page concludes by stating that Colab notebooks combine executable code, rich text, images, HTML, and LaTeX, and are stored in the user's Google Drive account. It provides a link to 'create a new Colab notebook'.

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Share User Settings E

Table of contents

- <> Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

+ Code + Text Copy to Drive

Connect Editing

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

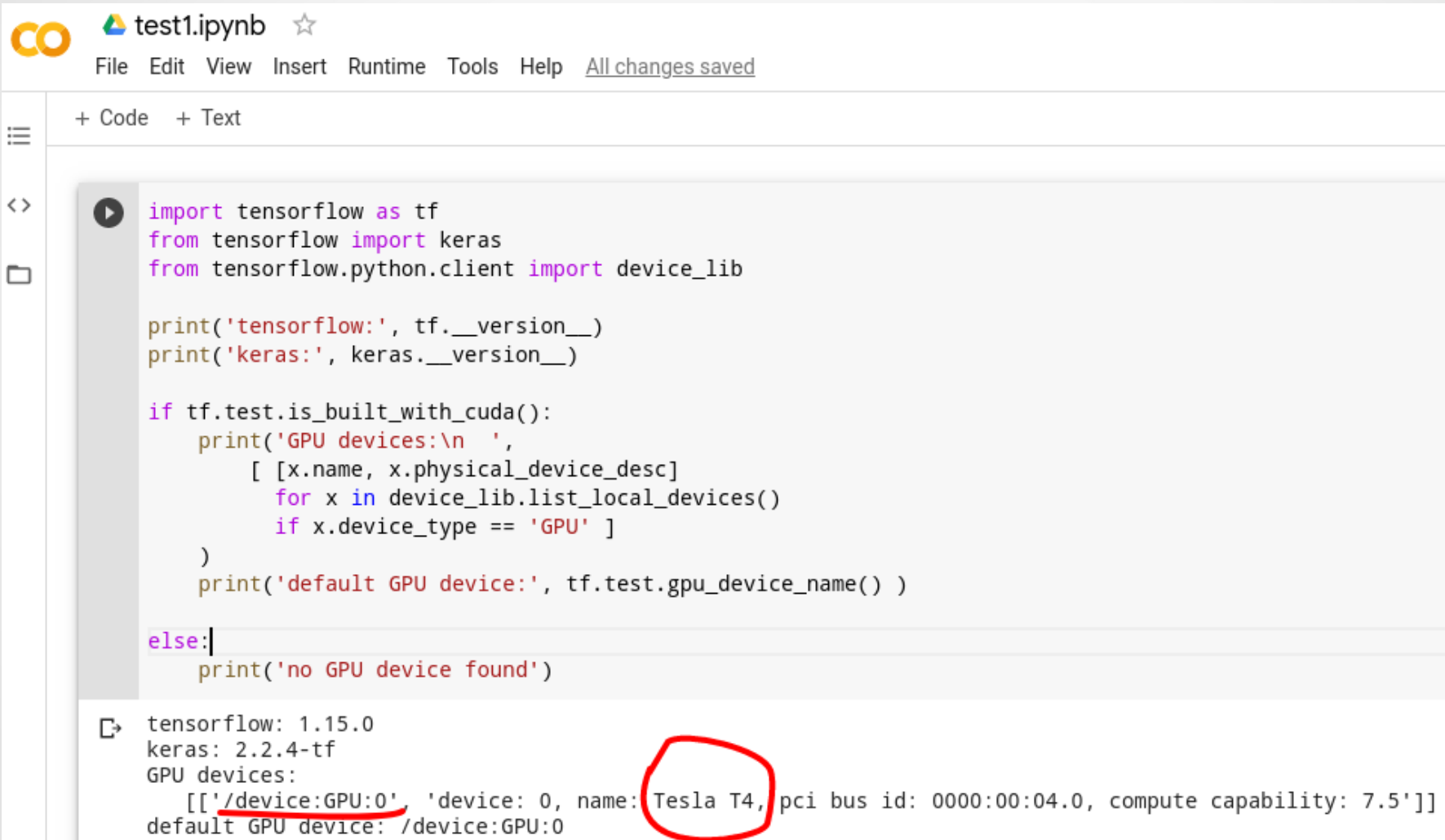
```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

604800

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see [Overview of Colab](#). To create a new Colab notebook you can use the File menu above, or use the following link: [create a new Colab notebook](#).

Python: Google Colab

<https://colab.research.google.com/>



The image shows a Google Colab notebook titled 'test1.ipynb'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', along with a status 'All changes saved'. Below the menu is a toolbar with '+ Code' and '+ Text' buttons. The main area contains a code cell with the following Python code:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.python.client import device_lib

print('tensorflow:', tf.__version__)
print('keras:', keras.__version__)

if tf.test.is_built_with_cuda():
    print('GPU devices:\n ',
          [ [x.name, x.physical_device_desc]
            for x in device_lib.list_local_devices()
            if x.device_type == 'GPU' ]
          )
    print('default GPU device:', tf.test.gpu_device_name() )
else:
    print('no GPU device found')
```

Below the code cell, the output is displayed:

```
tensorflow: 1.15.0
keras: 2.2.4-tf
GPU devices:
[['/device:GPU:0', 'device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5']]
default GPU device: /device:GPU:0
```

In the output, the string `['/device:GPU:0', 'device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5']` is highlighted with a red circle, and the word `Tesla` is also circled in red.

Python: Yandex DataSphere

<https://cloud.yandex.ru/blog/posts/2020/05/datasphere>

The screenshot displays the Yandex DataSphere IDE interface. On the left, a file explorer shows a directory with files: `catboost_info`, `train_data.npy`, `train_labels.npy`, and `yadisk.token`. The main editor window, titled `test.ml`, contains a Python script for training a CatBoost classifier. The script includes imports for `CatBoostClassifier` and `numpy`, followed by loading training data and labels, creating a `CatBoostClassifier` instance with specific parameters, and fitting the model. The output shows the learning rate set to `0.003512` and a progress log for 30 iterations, detailing learning, total, and remaining times.

```
[1]: from catboost import CatBoostClassifier
import numpy as np

[3]: train_data = np.load('train_data.npy')
train_labels = np.load('train_labels.npy')

[5]: model = CatBoostClassifier(iterations=1000,
                               task_type="CPU",
                               devices='0:1')

[6]: model.fit(train_data, train_labels, verbose=True)

Learning rate set to 0.003512
0:   learn: 0.6906569   total: 49.8ms   remaining: 49.7s
1:   learn: 0.6881785   total: 50.1ms   remaining: 25s
2:   learn: 0.6871994   total: 50.2ms   remaining: 16.7s
3:   learn: 0.6864504   total: 50.3ms   remaining: 12.5s
4:   learn: 0.6839919   total: 50.4ms   remaining: 10s
5:   learn: 0.6815451   total: 50.5ms   remaining: 8.37s
6:   learn: 0.6791099   total: 50.6ms   remaining: 7.18s
7:   learn: 0.6783759   total: 50.7ms   remaining: 6.29s
8:   learn: 0.6776453   total: 50.8ms   remaining: 5.59s
9:   learn: 0.6757266   total: 50.9ms   remaining: 5.04s
10:  learn: 0.6733133   total: 51ms     remaining: 4.58s
11:  learn: 0.6723679   total: 51.1ms   remaining: 4.21s
12:  learn: 0.6704690   total: 51.2ms   remaining: 3.89s
13:  learn: 0.6685772   total: 51.3ms   remaining: 3.61s
14:  learn: 0.6661971   total: 51.4ms   remaining: 3.38s
15:  learn: 0.6633391   total: 51.5ms   remaining: 3.17s
16:  learn: 0.6609833   total: 51.6ms   remaining: 2.98s
17:  learn: 0.6586417   total: 51.7ms   remaining: 2.82s
18:  learn: 0.6567942   total: 51.8ms   remaining: 2.68s
19:  learn: 0.6549535   total: 51.9ms   remaining: 2.54s
20:  learn: 0.6521563   total: 52ms     remaining: 2.42s
21:  learn: 0.6503330   total: 52.1ms   remaining: 2.32s
22:  learn: 0.6494321   total: 52.2ms   remaining: 2.22s
23:  learn: 0.6471437   total: 52.3ms   remaining: 2.13s
24:  learn: 0.6464625   total: 52.4ms   remaining: 2.04s
25:  learn: 0.6446604   total: 52.5ms   remaining: 1.97s
26:  learn: 0.6428651   total: 52.6ms   remaining: 1.9s
27:  learn: 0.6410765   total: 52.7ms   remaining: 1.83s
28:  learn: 0.6404061   total: 52.8ms   remaining: 1.77s
29:  learn: 0.6381544   total: 52.9ms   remaining: 1.71s
30:  learn: 0.6359178   total: 53ms     remaining: 1.66s
```

<https://habr.com/ru/article/565086/>

Python: упражнения numpy

<https://github.com/rougier/numpy-100>

100 numpy exercises

This is a collection of exercises that have been collected in the numpy mailing list, on stack overflow and in the numpy documentation. The goal of this collection is to offer a quick reference for both old and new users but also to provide a set of exercises for those who teach.

If you find an error or think you've a better way to solve some of them, feel free to open an issue at <https://github.com/rougier/numpy-100>.

File automatically generated. See the documentation to update questions/answers/hints programmatically.

Run the `initialize.py` module, then for each question you can query the answer or an hint with `hint(n)` or `answer(n)` for `n` question number.

```
In [ ]: %run initialise.py
```


Python: упражнения pandas

<https://github.com/ajcr/100-pandas-puzzles/>

100 pandas puzzles

Inspired by [100 Numpy exercises](#), here are 100* short puzzles for testing your knowledge of [pandas](#)' power.

Since pandas is a large library with many different specialist features and functions, these exercises focus mainly on the fundamentals of manipulating data (indexing, grouping, aggregating, cleaning), making use of the core DataFrame and Series objects.

Many of the exercises here are stright-forward in that the solutions require no more than a few lines of code (in pandas or NumPy... don't go using pure Python or Cython!). Choosing the right methods and following best practices is the underlying goal.

The exercises are loosely divided in sections. Each section has a difficulty rating; these ratings are subjective, of course, but should be seen as a rough guide as to how inventive the required solution is.

If you're just starting out with pandas and you are looking for some other resources, the official documentation is very extensive. In particular, some good places get a broader overview of pandas are...

- [10 minutes to pandas](#)
- [pandas basics](#)
- [tutorials](#)
- [cookbook and idioms](#)

Python: virtualenv

проблема: пакеты определённых версий могут быть несовместимы между собой

решение: виртуальные python-среды

позволяет работать с несколькими версиями python

держат одновременно несколько наборов пакетов разных версий

```
# pip install virtualenv
```

```
# mkdir /home/user/python3.8_env
```

```
# virtualenv -p python3.8 python3.8_env
```

```
# source /home/user/python3.8_env/bin/activate
```

Python: что почитать?

<http://www.sololearn.com/Course/Python/>

http://github.com/mechanoid5/ml_lectorium

Дейтел П., Дейтел Х. Python: Искусственный интеллект, большие данные и облачные вычисления (2020)