



# **Лекция 2: Язык Python**

Евгений Борисов

# Python: реализации языка



IronPython



**Cpython**

**Cythone**

**Jython (Java)**

**IronPython (.NET)**

**PyPy (compiler)**



# Python: дистрибутивы



**CPython**

**Anaconda (Miniconda)**



# Python: IDE

**iPython / Jupyter**

**PyCharm**

**Visual Studio Code**

**Eclipse + PyDev**

**Vim**

**Apache Zeppelin**



**IP[y]:**

 **VS Code**



 **eclipse**



# Python: про версии

**CPython — стандарт де-факто**

**2.7 vs 3.7**

**PEP (Python Enhancement Proposals)  
предложения по улучшению Python**

**PEP 8: руководство по написанию кода**

# **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: numpy

```
import numpy as np
```

```
x = np.random.rand(2,5)
```

```
y = np.random.rand(2,3)
```

```
x.T.dot(y)
```

```
x[:,2]
```

```
x[ 1, x[1,:]>0.5 ]
```

# 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: упражнение

ДАНО:

последовательность блоков переменной длины  
блоки состоят из слов  $\{0,1\}$  длины 8

0 1 0 0 1 0 1 0   1 1 1 1 0 0 1 0   0 0 1 1 1 0 1 1   0 1 0 1 0 1 0 0 ...

если первый символ блока 0  
то размер блока - 2 слова  
иначе размер блока - 1 слово

ЗАДАЧА:

определить размер последнего блока последовательности

# 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'. Another code cell calculates seconds in a week, showing the 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:

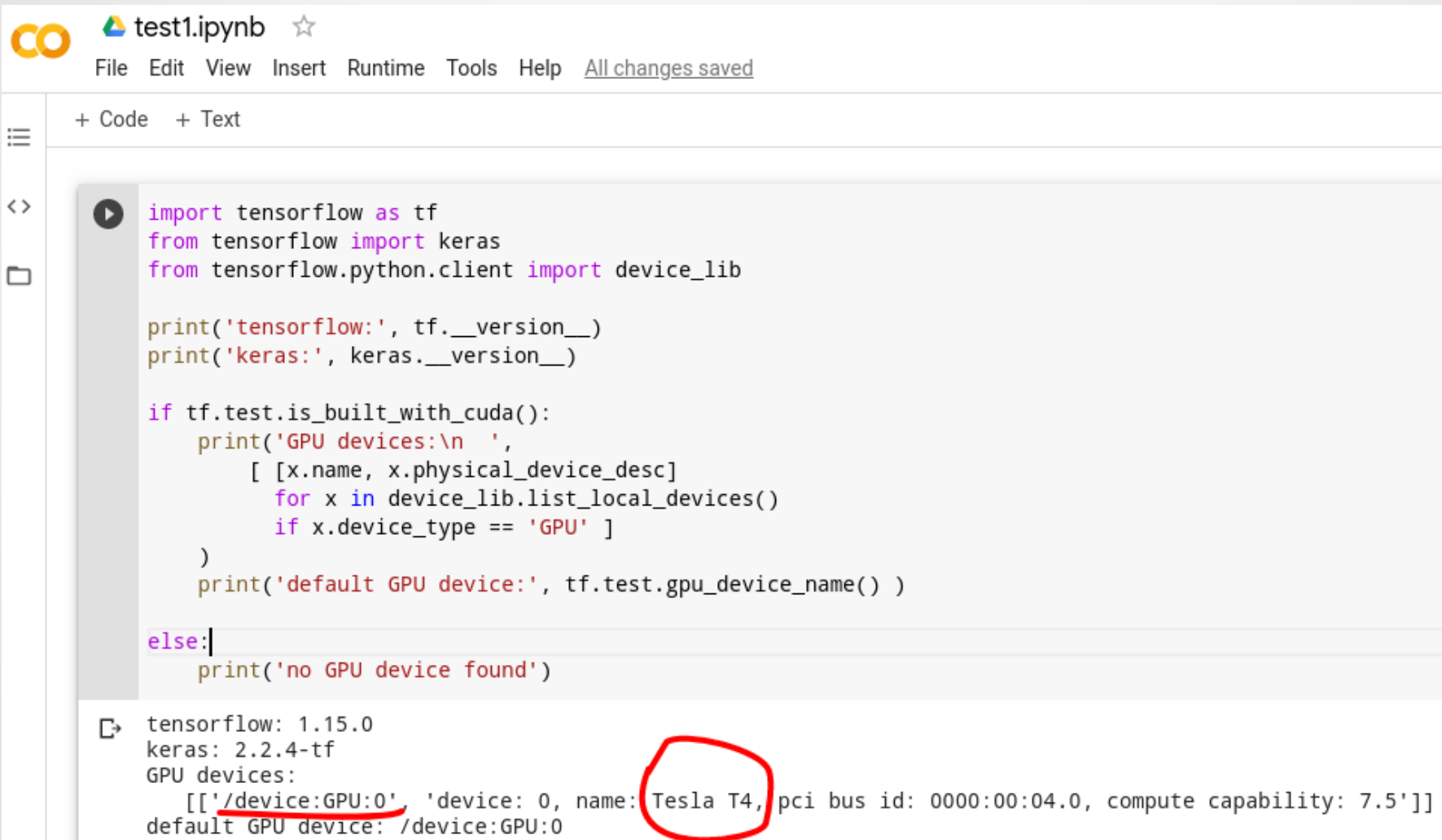
```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

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/>



```
test1.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

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')

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
```



# Python

**В сети появился курс по Python от Агентства нацбезопасности США**

<https://dev.by/news/v-seti-poyavilsya-kurs-po-python-ot-anb>

<https://nsa.sfo2.digitaloceanspaces.com/comp3321.pdf>

Doc ID: 6689691

(U)

## Instructor Notes

Updated about 2 years ago by [redacted] in [COMP 3321](#)

python fese

(U//FOUO) Instructor notes for COMP 3321.

Recommendations

### ~~UNCLASSIFIED//FOR OFFICIAL USE ONLY~~

(U) So, you're teaching the Python class. What have you gotten yourself into? You should probably take a few moments (or possibly a few days) to reconsider the life choices that have put you in this position.

### (U) Course Structure

(U) As mentioned in the [introduction](#), this course is designed for flexibility. When taught in a classroom setting, a single lesson or module can be covered in a session that lasts between 45 and 90 minutes, depending on the topics to be covered. The standard way to structure the course is as a full-time, two week block. During the first week, the ten lessons are covered with morning and afternoon lectures. During the second week, up to ten modules are covered in a similar manner, as needed or requested by the students in the class. (If the class needs are not known, take a vote). During the first few days of class, students should choose a project to work on. On the last day, students should report back on their progress and, if possible, demonstrate their work. Instructors should be available outside of lectures to assist students with exercises and projects.

(U) The two week block is not the only way of teaching the course. The material could be presented at a more leisurely pace, for instance during a weekly brown bag lunch that continues for several months. Alternatively, if students are already prepared (or willing to do some of the initial lessons in a self-study manner), a great deal can be accomplished in a two or three day workshop. For instance, if all students already have a basic knowledge of Python, they might well start with the lessons on [tooling](#) and [writing modules and packages](#), then move on to cover various modules of interest.

Approved for Release by NSA on 12-02-2019, FOIA Case # 108165

# Python: что почитать?

- Andrew Ng - Machine Learning
- Константин Воронцов - Машинное обучение
- Евгений Борисов - <http://mechanoid.kiev.ua>
- [http://github.com/mechanoid5/ml\\_lectorium](http://github.com/mechanoid5/ml_lectorium)

Python: почти последний слайд...



**Вопросы ?**

# Python: последний слайд...



IP[y]:

**практика ....**