

Научное программирование в Python

Лекция1. Основы языка программирования Python. Среды разработки. Базовые типы и конструкции языка

*Киреев В.С.,
к.т.н., доцент*

Москва, 2022

Гвидо ван Россум



Гвидо ван Россум известен как создатель языка Python. С момента создания языка Гвидо многократно награждался различными премиями. В 2001 году он получил награду за вклад в развитие свободного программного обеспечения от 'Free Software Foundation', а в мае 2003-го удостоился премии 'NLUUG Award'. В 2006 году ему присвоили звание Выдающегося Инженера.

Язык Python

Python («Пайтон», «Питон») - высокоуровневый, интерпретируемый, объектно-ориентированный, императивный, строго типизированный язык общего назначения, который имеет динамическую типизацию. Разрабатывался Гвидо ван Россумом в 1989 г.. Первая публикация Python состоялась в феврале 1991 года — это была версия 0.9.0. Вторая версия Python вышла в 2000 г., третья (современная) в 2008 г.

<https://www.python.org/>

Высокоуровневость Python

Низкоуровневые языки — языки, близкие к машинному коду или его конструкциям. Высокоуровневые языки разрабатываются для удобства использования и скорости написания программы. В них применяются абстракции — структуры данных, набор вспомогательных функций и так далее.

Объектно-ориентированность Python

В объектно-ориентированных языках основа это классы и экземпляры классов это равносильно типу и объекту этого типа. Выполнение условных задач или же просто работа программы строится на взаимодействии различных классов. Python также поддерживает и процедурное программирование это значит, что программу можно написать без единого класса. В основе функциональных языков лежит отличная от предыдущих вычислительная система, называемая лямбда-исчисление

Интерпретируемость Python

Языки делятся на интерпретируемые, такие как Python, JS, PHP, R, Ruby и компилируемые -C, C++, Pascal. В первом случае программа выполняется интерпретатором, во втором программа сначала преобразуется в понятные компьютеру исполняемые файлы.

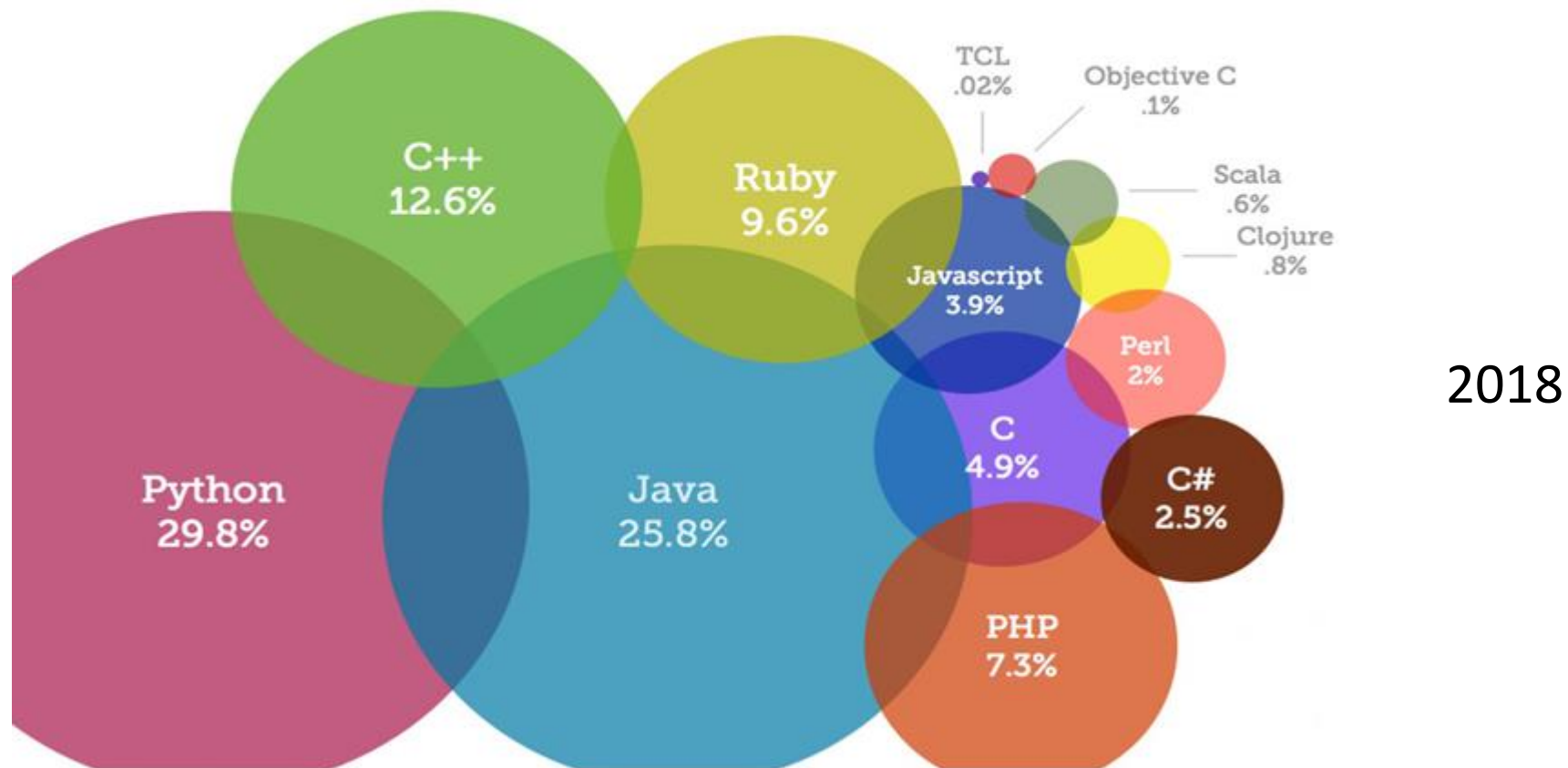
Императивность Python

Языки программирования также могут быть разделены на императивные и декларативные. В императивном языке программист будет указывать последовательность команд для выполнения. Декларативные языки требуют описания результата, который программист хочет получить в ходе выполнения запроса.

Динамическая типизация в Python

Динамическая типизация предполагает, что в процессе выполнения команды переменная может содержать объекты различных типов. Статическая типизация предполагает, что при задании переменной сразу указывается тип данных, который она может содержать. Python поддерживает динамическую типизацию.

Популярность Python



2018

<https://www.dotnetlanguages.net/most-popular-programming-languages/>

Популярность Python

Most popular languages on GitHub as of June 07, 2021

#	language	repos count
1	JSX	54711718
2	JavaScript	14240090
3	Java	9827505
4	Python	6909587
5	HTML	5367639
6	C#	2893002
7	C++	2569474
8	PHP	2385827
9	Jupyter Notebook	2239515
10	CSS	2039268
11	TypeScript	1986262
12	Ruby	1904129
13	C	1726646

<https://github.com/oprogramador/github-languages#all>

Python Enhancement Proposals. Python

- Пробелы - самый предпочтительный метод отступов.
- Табуляция должна использоваться только для поддержки кода, написанного с отступами с помощью табуляции.
- Python 3 запрещает смешивание табуляции и пробелов в отступах.
- Ограничьте длину строки максимум 79 символами.
- ...

<https://peps.python.org/pep-0008/>

Дзен языка Python

- Красивое лучше, чем уродливое.
- Простое лучше, чем сложное.
- Читаемость имеет значение.
- Явное лучше, чем неявное
- ...

PEP 20, Тим Питерс

<https://peps.python.org/pep-0020/>

Достоинства Python

- Расширяемость и гибкость.
- Интерпретируемость и кроссплатформенность.
- Стандартизированность.
- Open Source.
- Сильное комьюнити и конференции.
- Широта применения.
- Востребованность на рынке труда и поддержка гигантами IT-сферы.

Недостатки Python

- Python медленный.
- Python не особо адекватно распоряжается памятью.
- Python строго привязан к системным библиотеками. Отсюда возникают сложности при попытке использовать язык на новых программных платформах.

Применение Python

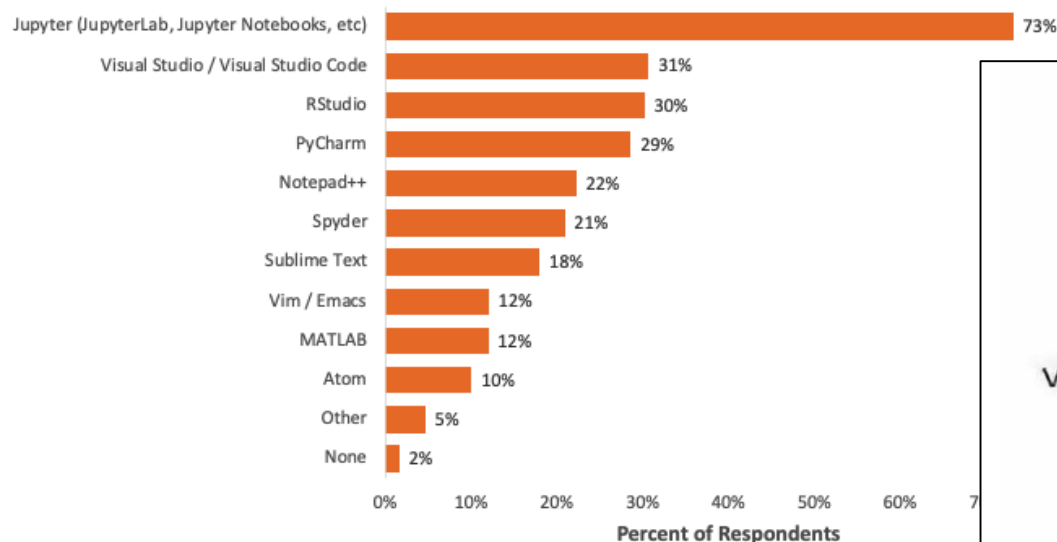
- Веб-разработка
- Data Science и Machine Learning
- Написание скриптов
- Автоматизация тестирования ПО
- Разработка мобильных приложений
- Системное администрирование
- Геймдев
- ...

Понятие IDE

IDE (Integrated Development Environment) – это интегрированная, единая среда разработки, которая используется разработчиками для создания различного программного обеспечения. IDE представляет собой комплекс из нескольких инструментов, а именно: текстового редактора, компилятора (или интерпретатора), средств автоматизации сборки отладчика.

Использование IDE

Which of the following integrated development environments (IDEs) do you use on a regular basis?



Note: Data are from the 2019 Kaggle ML and Data Science Survey. You can learn more about the study here: <https://www.kaggle.com/c/kaggle-survey-2019>.

A total of 19717 respondents completed the survey; the percentages in the graph are based on a total of 19717 respondents who were asked and who answered the question.



Copyright 2020 Business Over Broadway

Most Popular Python IDE, Editors

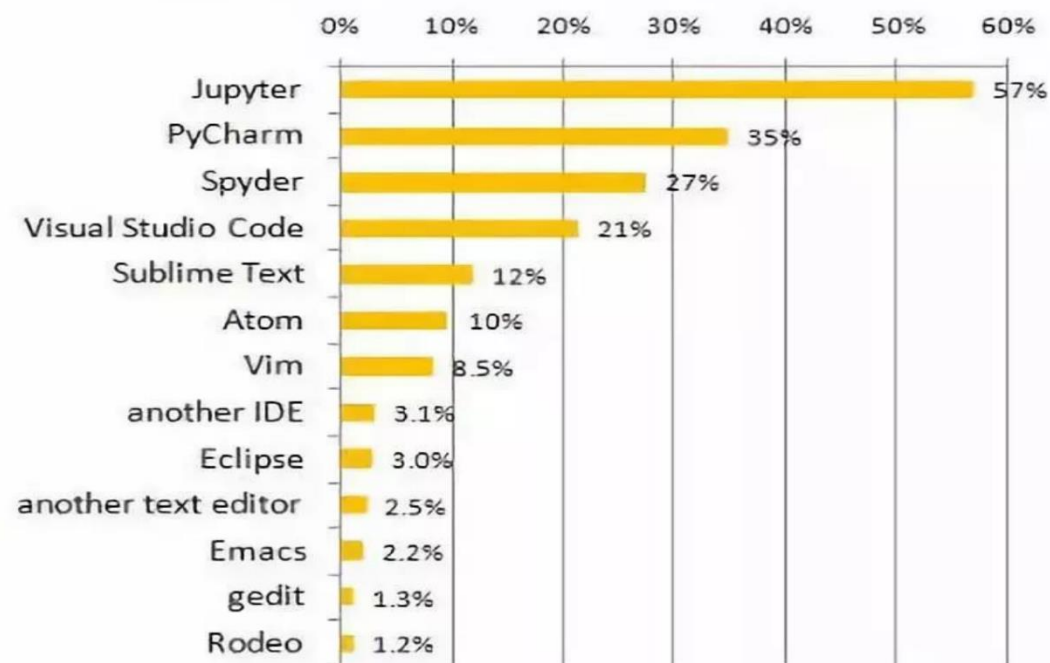



图1 最流行的Python IDEs

<https://www.kaggle.com/c/kaggle-survey-2019>

IDE для языка Python


Редакторы	Среды, интегрированные с библиотеками DS
<ul style="list-style-type: none">• IDLE • VS Code • Atom • Eclipse + PyDev 	<ul style="list-style-type: none">• Spyder • Anaconda 
IDE	Интерактивные ноутбуки
<ul style="list-style-type: none">• PyCharm • DataGrip 	<ul style="list-style-type: none">• Jupyter Notebook • Zeppelin 



Jupyter Notebook












Jupyter Notebook - это веб-приложение с открытым исходным кодом, которое позволяет создавать и обмениваться документами, содержащими живой код, уравнения, визуализации и повествовательный текст. Jupyter Notebook позволяет запускать клетки с кодом в произвольном порядке. Всего ячейки бывают трёх типов: `code` — код, `markdown` — текст с формулами на Latex (в одинарных или двойных $\$$), `raw` — неформатированный текст.

Jupyter Notebook














The screenshot displays the Jupyter Notebook interface. At the top, the header shows the Jupyter logo, the text "jupyter Untitled", a Python logo, and a "Logout" button. Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are buttons for "Trusted", a pencil icon, and "Python 3". A secondary toolbar contains icons for saving, adding cells, undo, redo, and other navigation functions. The main area features a code input cell labeled "In []:" which is currently empty. Below the code cell is a rich text editor toolbar with icons for text formatting (bold, italic, monospace), linking, unlinking, and other editing tools. The editor contains a LaTeX formula:
$$N(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$
. To the right of the code input, the rendered version of this formula is displayed. At the bottom of the interface, there are two buttons: "+ Код" (Code) and "+ Текст" (Text).

jupyter Untitled  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted  Python 3 

          Code 

In []:

$$N(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

+ Код + Текст

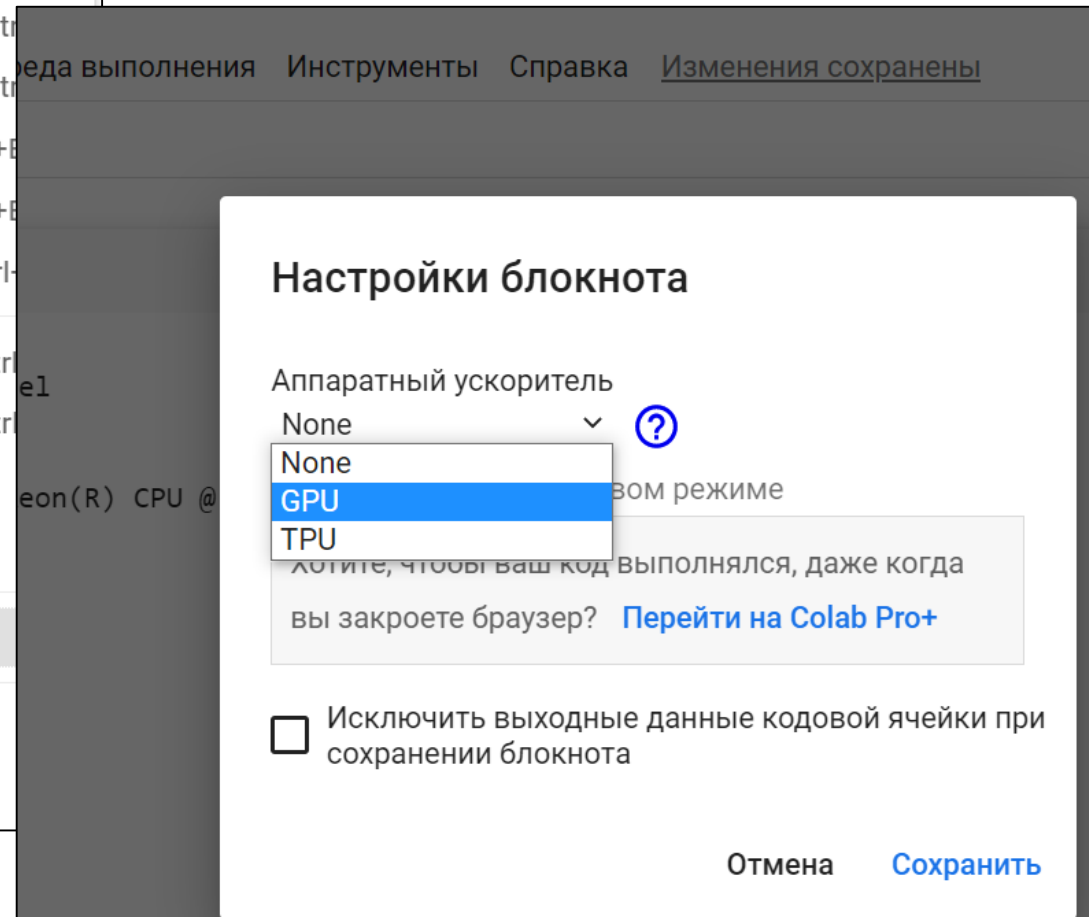
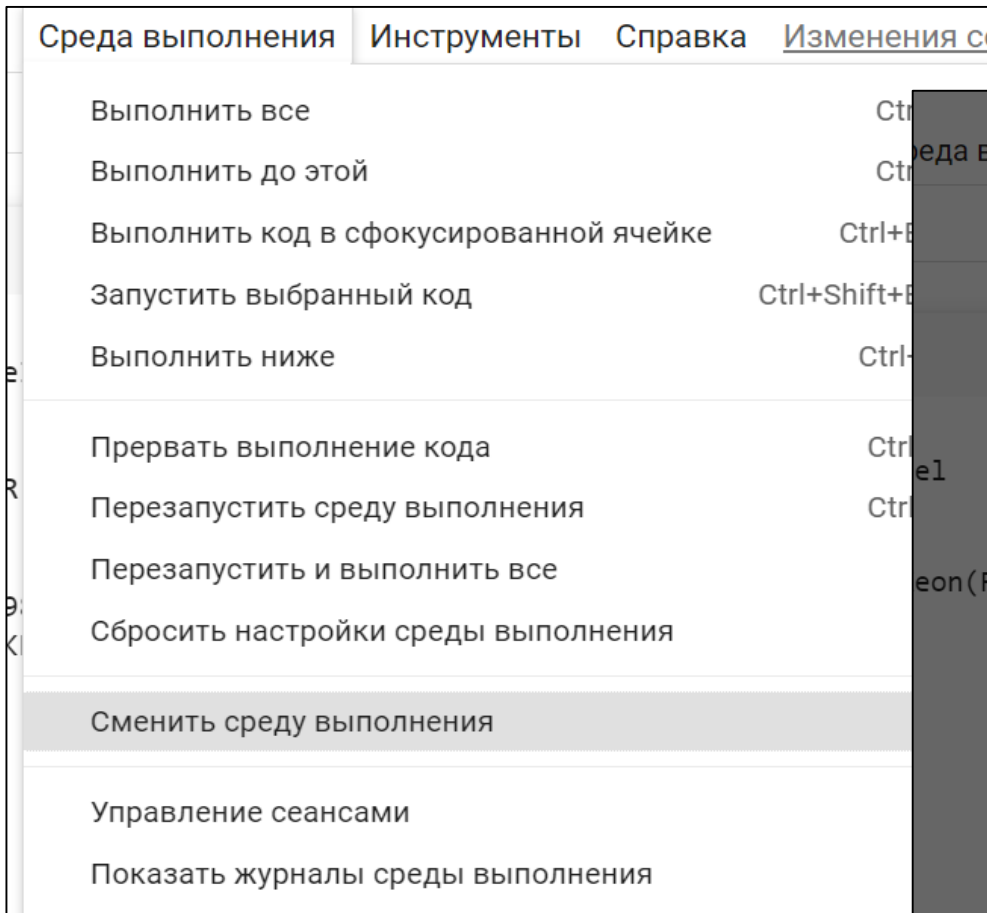
Google Colab

[Google Colaboratory](#) — бесплатная среда Jupyter Notebook, которая выполняется на облачных серверах Google и позволяет использовать аппаратное оборудование бэкенда, включая GPU и TPU. Таким образом можно работать со всеми возможностями Jupyter Notebook, не устанавливая его на локальной машине. Каждые 12 часов Colab будет принудительно восстанавливать занятые ресурсы.

Другие особенности Google Colab

- Создание/Загрузка/Совместное использование записных книжек
- Импорт/Сохранение записных книжек с/на Google Диск
- Импорт /публикация записных книжек из GitHub
- Импорт внешних наборов данных, например, из Kaggle
- Интеграция PyTorch, TensorFlow, Keras, OpenCV

Смена среды выполнения



Ресурсы Colab в среде с GPU


```
1 !nvidia-smi
```

Mon Apr 4 09:42:34 2022

```
+-----+
| NVIDIA-SMI 460.32.03      Driver Version: 460.32.03      CUDA Version: 11.2     |
+-----+-----+-----+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                       |                    |     MIG M.     |
+-----+-----+-----+-----+-----+
|    0   Tesla K80           Off      | 00000000:00:04.0 Off  |             0        |
| N/A   44C    P8      28W / 149W |      0MiB / 11441MiB |      0%      Default  |
|                                       |                    |             N/A     |
+-----+-----+-----+-----+-----+


+-----+
| Processes: |
| GPU   GI    CI          PID    Type   Process name                      GPU Memory |
|          ID    ID                                   |           Usage   |
+-----+-----+-----+-----+-----+
| No running processes found |
+-----+
```


Совместное редактирование ноутбука





Предоставьте доступ пользователям и группам


Совместный доступ не настроен



Скопируйте ссылку




<https://colab.research.google.com/drive/1HrUlj5dezut6goq0uJ...>Копировать ссылку



Доступно пользователям, у которых есть ссылка ▾
Просматривать могут все в Интернете, у кого есть эта ссылка.

Читатель ▾



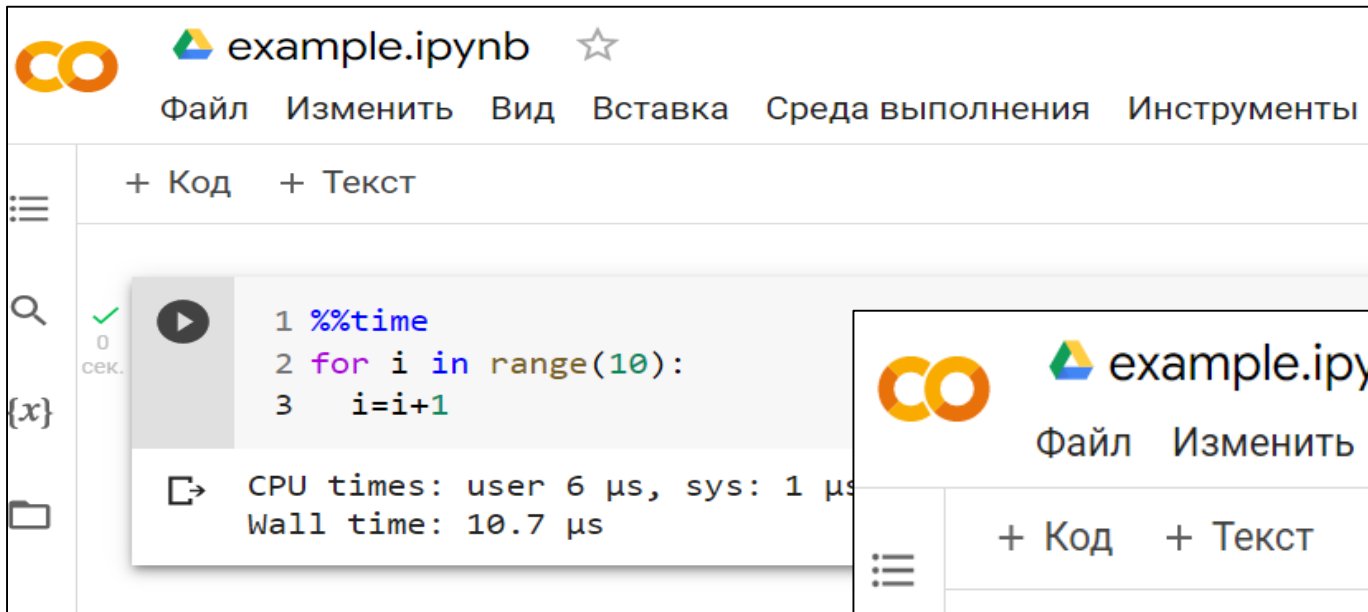
Комментарии и советы видны пользователям с правом на просмотр файла.

«Магия» Google Colab

«Магические» команды — это встроенные команды IPython, в том числе и Jupyter Notebook, разработка в котором ведется в ячейках. «Магические» команды могут быть двух видов:

- Выполняющиеся для одной строки — %
- Выполняющиеся для всей ячейки — %%.

Примеры работы «магических» команд

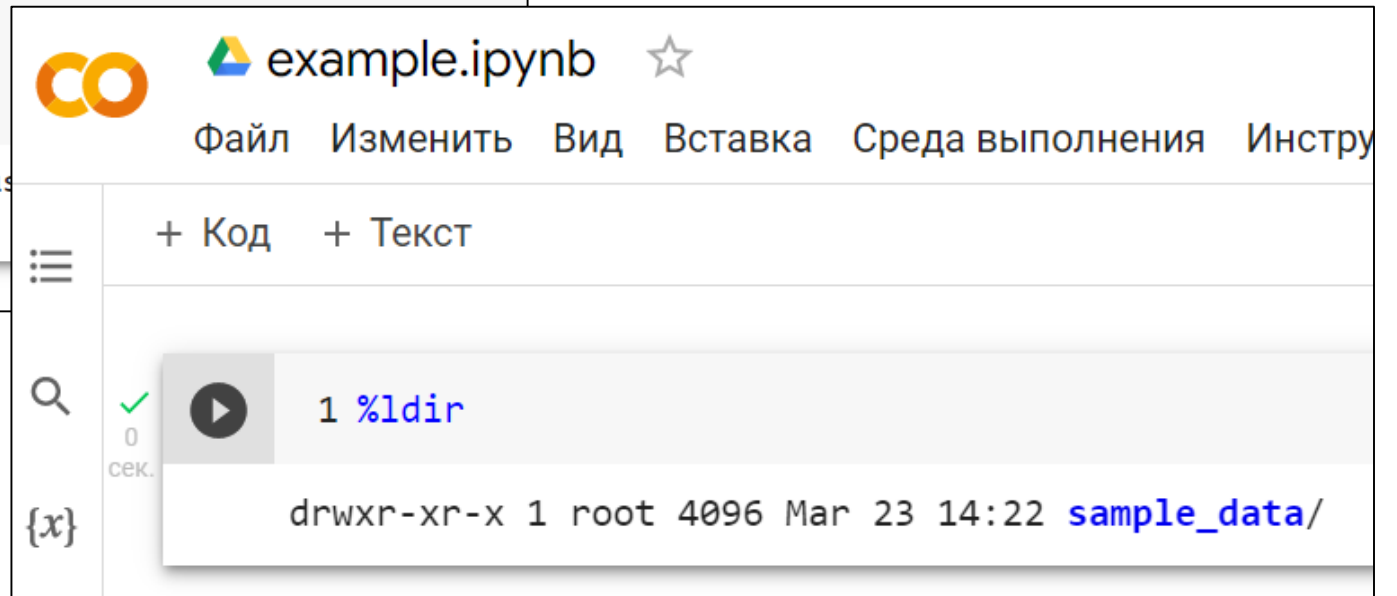


The screenshot shows the JupyterLab interface with a file named 'example.ipynb'. The menu bar includes 'Файл', 'Изменить', 'Вид', 'Вставка', 'Среда выполнения', and 'Инструменты'. The left sidebar has icons for a menu, search, and a variable '{x}'. The main area contains a code cell with a play button icon, a green checkmark, and a timer showing '0 сек.'. The code in the cell is:

```
1 %%time
2 for i in range(10):
3     i=i+1
```

Below the code, the execution output is displayed:

```
CPU times: user 6 µs, sys: 1 µs
Wall time: 10.7 µs
```



The screenshot shows the JupyterLab interface with a file named 'example.ipynb'. The menu bar includes 'Файл', 'Изменить', 'Вид', 'Вставка', 'Среда выполнения', and 'Инструменты'. The left sidebar has icons for a menu, search, and a variable '{x}'. The main area contains a code cell with a play button icon, a green checkmark, and a timer showing '0 сек.'. The code in the cell is:

```
1 %ldir
```

Below the code, the execution output is displayed:

```
drwxr-xr-x 1 root 4096 Mar 23 14:22 sample_data/
```

Полезные «магические» команды

- *%lsmagic* – показывает список всех доступных «магических» команд для строк и ячеек
- *%run* – выполнение скрипта *.py* в ноутбуке
- *%%timeit* – показывает время выполнения кода Python в ячейке
- *%history* – показывает историю выполненных команд
- *%lsmagic* – показывает список всех доступных «магических» команд для строк и ячеек
- *%%bash* – Можно вызывать Bash, например команда *mkdir TEST* сделает новую папку в текущей

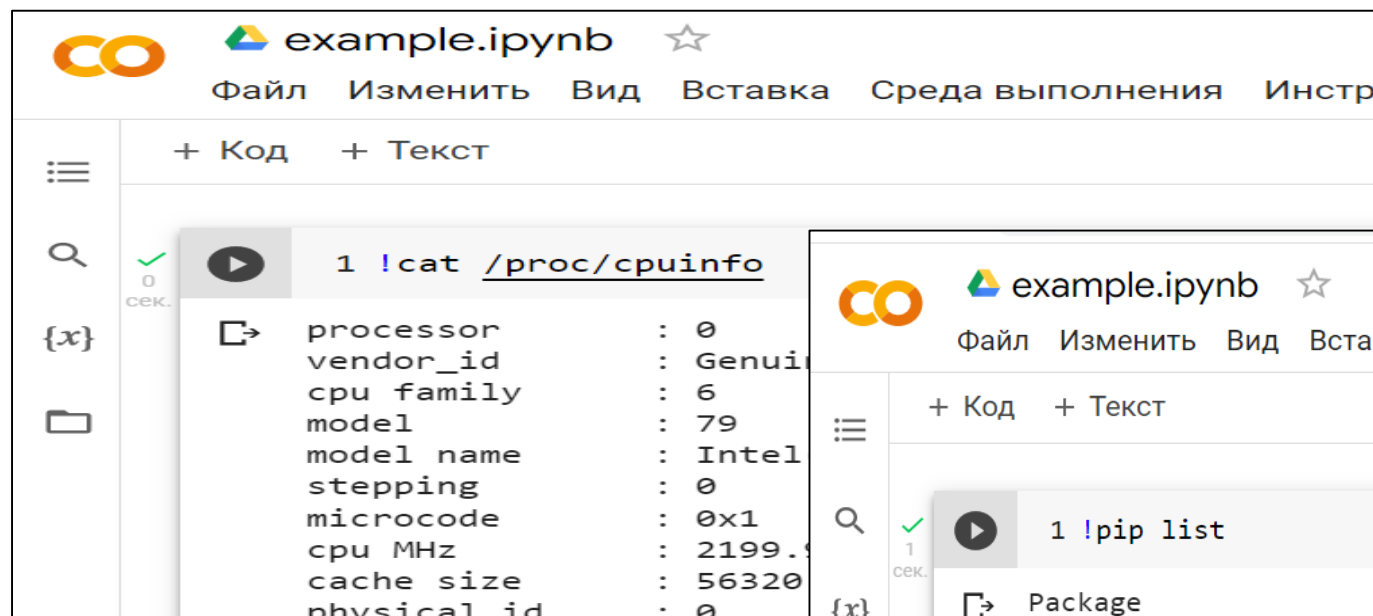
Bash в Google Colab

Bash — это универсальный инструмент для выполнения различных задач, и одновременно, это скриптовый язык программирования, позволяющий создавать сценарии для автоматизации различных операций. Для вызова Bash-команд в Colab нужно добавить «!» перед командой.

Полезные команды Bash

- `!ls` – вывести содержимое текущей папки
- `!cd` – смена текущей папки
- `!cat` – вывод содержимого файла
- `!echo` - выводиться строку текста в терминал
- `!wget` - скачать содержимое по ссылке

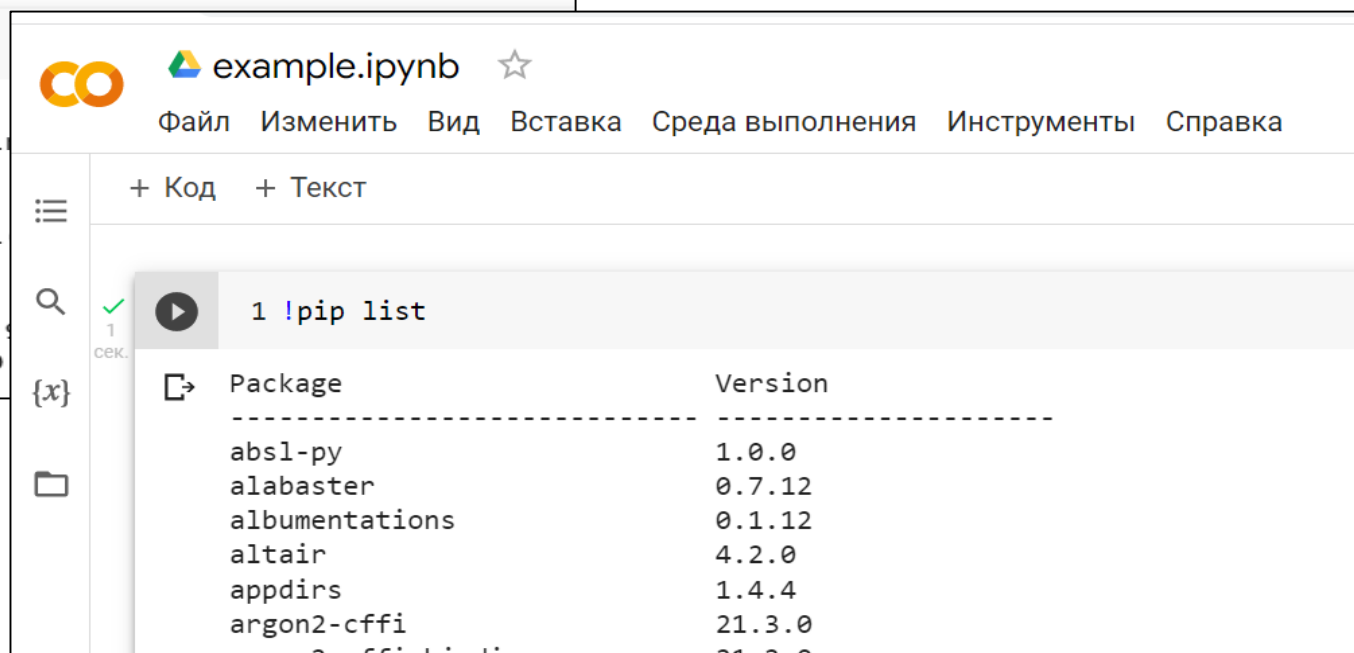
Примеры работы Bash в Google Colab



The screenshot shows the Google Colab interface with a file named 'example.ipynb'. The menu bar includes 'Файл', 'Изменить', 'Вид', 'Вставка', 'Среда выполнения', and 'Инструменты'. The left sidebar has icons for a menu, search, and file explorer. The main code cell contains the command `1 !cat /proc/cpuinfo`. The output is displayed as a table of CPU information.

```
1 !cat /proc/cpuinfo
```

processor	: 0
vendor_id	: GenuineIntel
cpu family	: 6
model	: 79
model name	: Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz
stepping	: 0
microcode	: 0x1
cpu MHz	: 2199.998
cache size	: 56320 KB
physical id	: 0



The screenshot shows the Google Colab interface with a file named 'example.ipynb'. The menu bar includes 'Файл', 'Изменить', 'Вид', 'Вставка', 'Среда выполнения', 'Инструменты', and 'Справка'. The left sidebar has icons for a menu, search, and file explorer. The main code cell contains the command `1 !pip list`. The output is displayed as a table of installed packages and their versions.

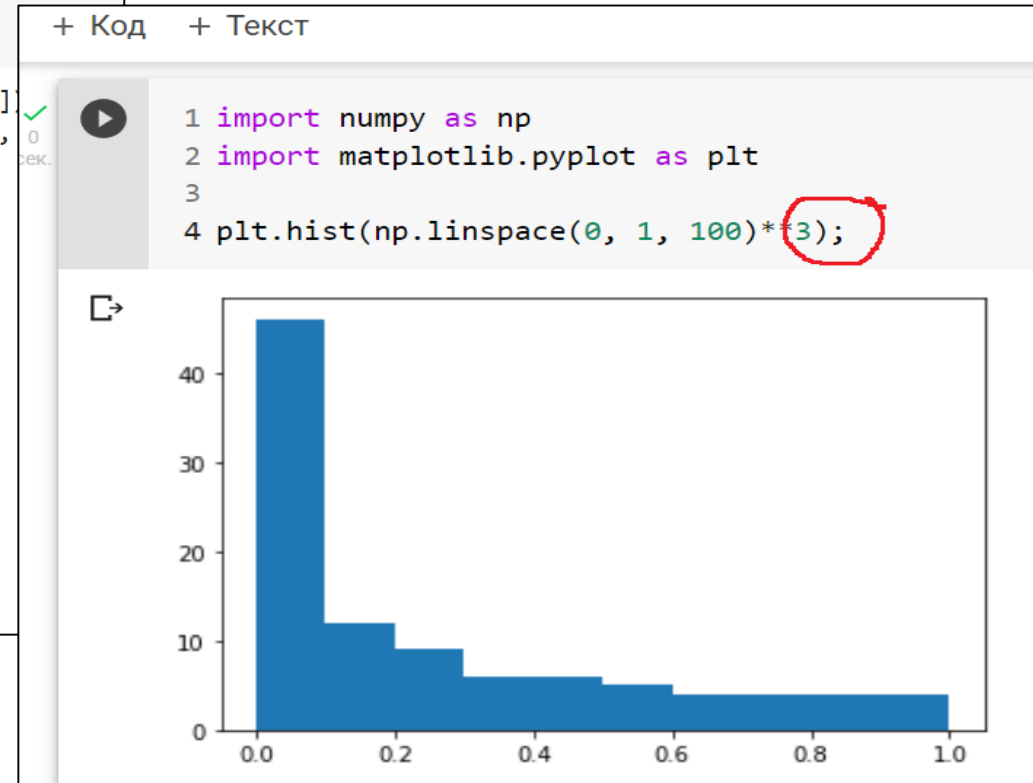
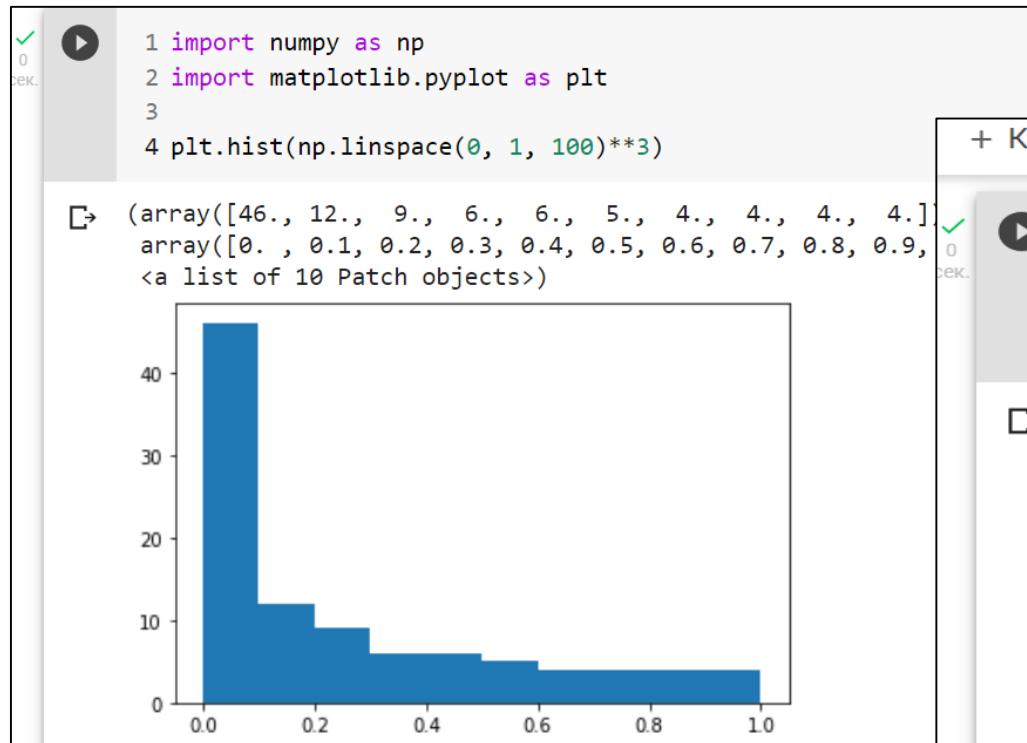
```
1 !pip list
```

Package	Version
abs1-py	1.0.0
alabaster	0.7.12
albumations	0.1.12
altair	4.2.0
appdirs	1.4.4
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0

Быстрые сочетания клавиш Google Colab

- *CTRL + ENTER* – запуск текущей ячейки
- *SHIFT + ENTER* - запуск текущей ячейки и переход к следующей
- *ALT + ENTER* – запуск текущей ячейки и вставка новой ячейки под текущей

Подавление вывода последней строки

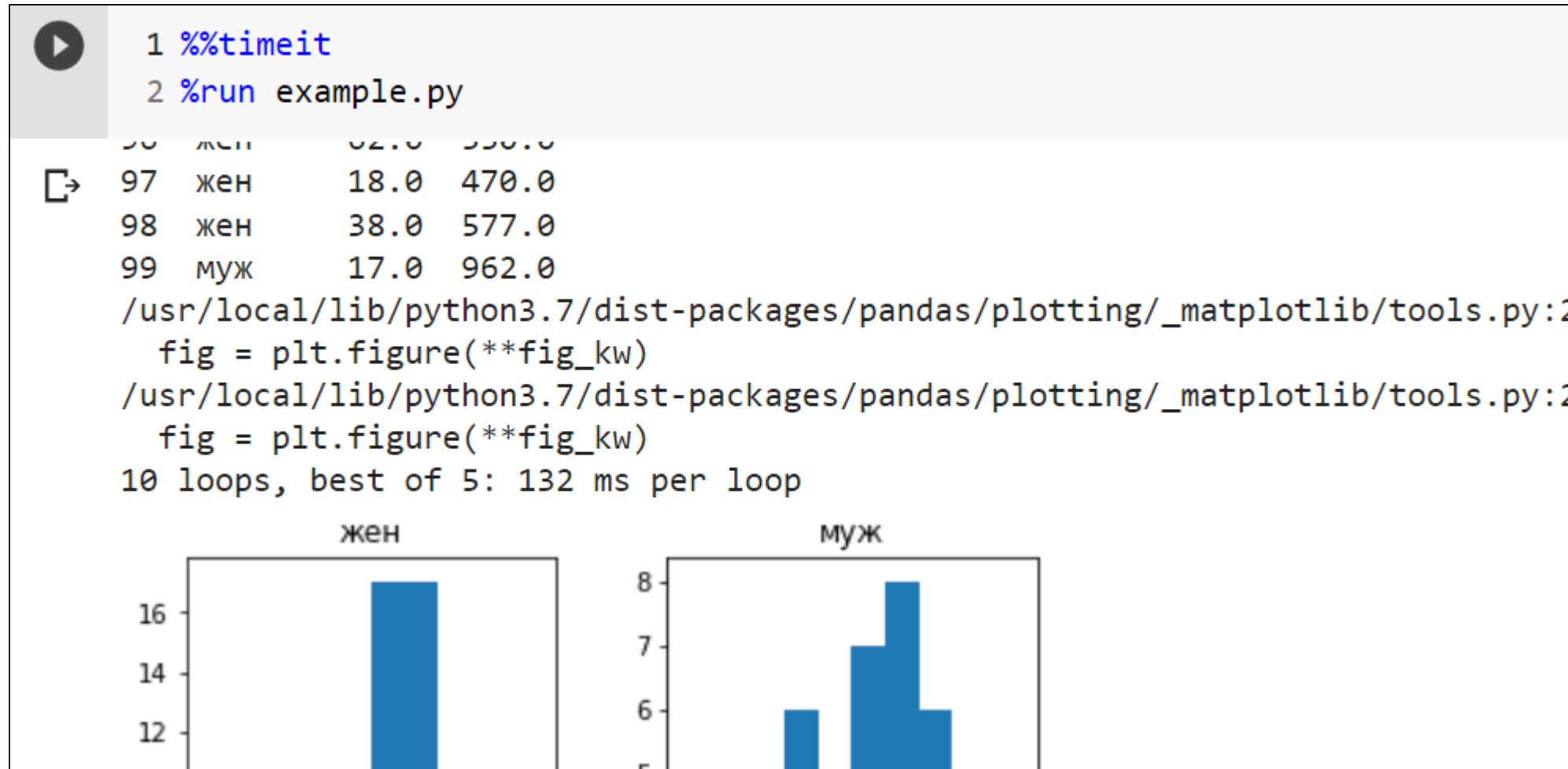


Пример кода (генерация данных)

```
1 %%writefile example.py
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 df=pd.DataFrame({'пол':np.random.choice(['муж','жен'],size=100,p=(0.4,0.6)),
7                 'возраст':np.random.choice(list(np.rint(np.random.normal(35,15,size=100)))+\
8                 list(np.rint(np.random.normal(40,20,size=100))),size=100),
9                 'доход':np.random.choice(list(np.rint(np.random.normal(500,100,size=100)))+\
10                 list(np.rint(np.random.normal(1000,200,size=100))),size=100)})
11 print('размерность датафрейма', df.shape)
12 print('первые 5 строк датафрейма',df.head(5))
13 print('последние 5 строк датафрейма',df.tail(5))
14
15 df['возраст'].hist(by=df['пол']);
```

Writing example.py

Пример кода (запуск скрипта «магической» командой)



Пример кода (использование ngrok)

```
1 !python -m pip install -q dash
2 !pip install -q flask flask-ngrok
```

```
█ 9.6 MB 8.3 MB/s
█ 357 kB 3.2 MB/s
```

```
[ ] 1 !wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
    2 !tar -xf /content/ngrok-stable-linux-amd64.tgz
    3 !/content/ngrok authtoken 22eEB7a0EsuRdWiU4Zfm9le0vgH_4d8xPKmc9Gvc6LaMM6AWD
```

```
--2022-04-01 15:36:17-- https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
Resolving bin.equinox.io (bin.equinox.io)... 52.202.168.65, 54.161.241.46, 18.205.222.128, ...
Connecting to bin.equinox.io (bin.equinox.io)|52.202.168.65|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13770165 (13M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.tgz'
```

```
ngrok-stable-linux- 100%[=====>] 13.13M 16.0MB/s in 0.8s
```

```
2022-04-01 15:36:18 (16.0 MB/s) - 'ngrok-stable-linux-amd64.tgz' saved [13770165/13770165]
```

```
Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

```
[ ] 1 get_ipython().system_raw('./ngrok http 8050 &')
    2
    3 ! curl -s http://localhost:4040/api/tunnels | python3 -c \
    4     "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

<https://7405-35-230-117-81.ngrok.io>

Альтернативы Google Colab

- **Yandex DataSphere** – в отличие от GC это платный блокнот, в котором тарифицируется фактическое время вычислений.
- **Kaggle Kernels** – кроме Python, сервис Kaggle поддерживает R, интегрируется с Google Cloud Storage, BigQuery и AutoML. При этом время пользования процессорами – девять часов, на три меньше, чем у GC.
- **Azure Notebooks** – тоже поддерживает другие языки (R, F#). Сервисы Microsoft Azure также, как и Яндекса, тарифицируются за фактическое время использования.
- **CoCalc** – предлагает и бесплатный, и платный периоды. В расширенной версии больше памяти и времени простоя, приоритетный доступ к процессорам и техподдержке.

Классификация типов данных в Python

- Простые — числа и строки
 - Коллекции — списки, кортежи и словари
 - Прочее — файлы, итераторы, сокеты, NaN
-
- Неизменяемые типы - числа, строки
 - Изменяемые типы — списки, словари и множества

Типы данных в Python

- Целые числа – `int`. Целые числа могут быть любой длины, они ограничиваются лишь доступной памятью.
- Числа с плавающей точкой – `float`.
- Комплексные числа – `complex`. Комплексные числа записываются в форме $x+uj$, где x — действительная часть числа, а u — мнимая.
- Строки - последовательность символов. Можно использовать одинарные или двойные кавычки для создания строки. Многострочные строки можно обозначить тройными кавычками, `'''` или

Синтаксис Python

- Python не содержит операторных скобок (begin..end в pascal или {...} в Си), вместо этого блоки выделяются отступами: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием.
- Однострочные комментарии начинаются со знака фунта «#», многострочные — начинаются и заканчиваются тремя двойными кавычками «"""».
- Чтобы присвоить значение переменной используется знак «=», а для сравнения — «==».
- Для увеличения значения переменной, или добавления к строке используется оператор «+=», а для уменьшения — «-=».

Синтаксис Python

- While - один из самых универсальных циклов в Python, поэтому довольно медленный. Выполняет тело цикла до тех пор, пока условие цикла истинно.
- Цикл for проходит по любому итерируемому объекту (например строке или списку), и во время каждого прохода выполняет тело цикла.
- Оператор continue начинает следующий проход цикла, минуя оставшееся тело цикла (for или while)
- Оператор break досрочно прерывает цикл.
- Слово else, примененное в цикле for или while, проверяет, был ли произведен выход из цикла инструкцией break, или же "естественным" образом.

Синтаксис Python

- While - один из самых универсальных циклов в Python, поэтому довольно медленный. Выполняет тело цикла до тех пор, пока условие цикла истинно.
- Цикл for проходит по любому итерируемому объекту (например строке или списку), и во время каждого прохода выполняет тело цикла.
- Оператор continue начинает следующий проход цикла, минуя оставшееся тело цикла (for или while)
- Оператор break досрочно прерывает цикл.
- Слово else, примененное в цикле for или while, проверяет, был ли произведен выход из цикла инструкцией break, или же "естественным" образом.

Литература по Python и не только

- «Язык программирования Python», Гвидо ван Россум и др.
- «Изучаем Python», Марк Лутц
- «Высокопроизводительный Python: практическое пособие для людей», Миша Горелик, Ян Освальд
- «Python. К вершинам мастерства», Лучано Рамальо
- «Python. К вершинам мастерства», Лучано Рамальо
- «Мифический человеко-месяц, или Как создаются программные системы», Фредерик Брукс
- ...

Конференции Python

- MOSCOW PYTHON CONF++
- PYCON RUSSIA
- PYTUP
- PYCON WEEKEND
- ...

Спасибо за внимание!