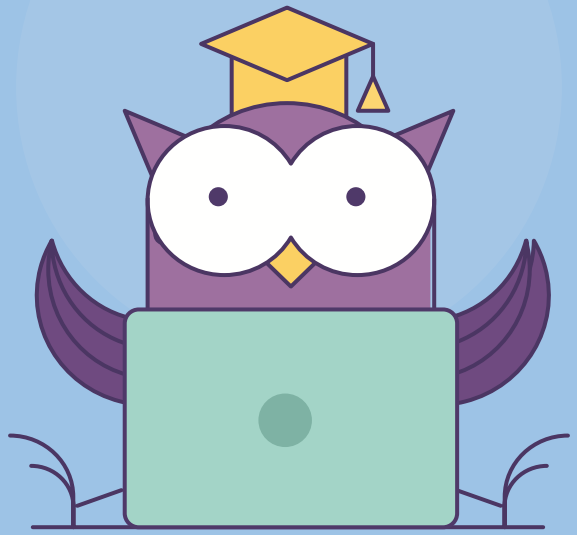
OTUS

Доступ к данным, ноутбуки

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо
Ставьте - если есть проблемы

План занятия

- Что такое Jupyter
- Архитектура
- Обзор интерфейса
- Метаданные
- Cheat sheets
- Практика

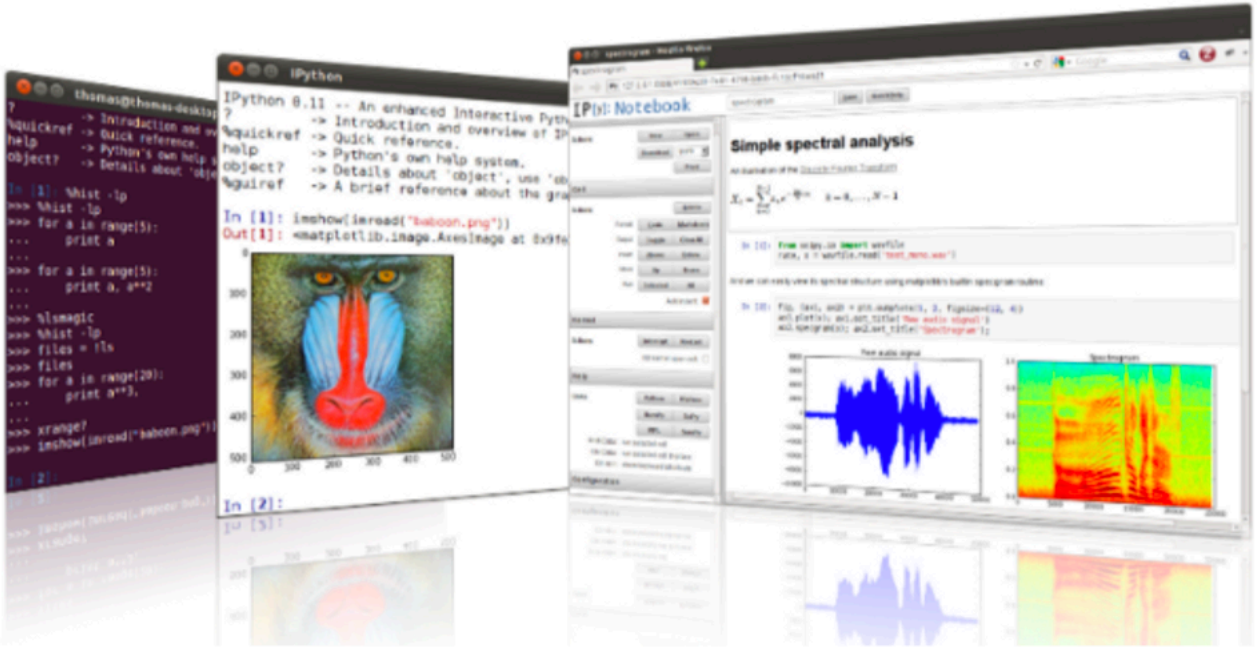
IPython

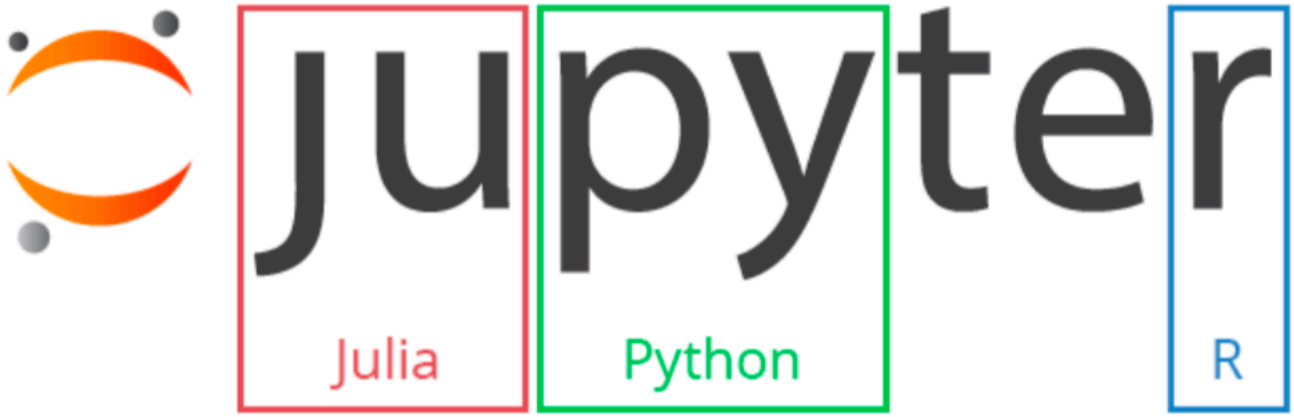
2001

Все в консоли, отображение графиков в новом окне и т.д.

IPython Notebook pre 1.0

18 декабря 2011




jupyter
Julia
Python
R

Что это?

- A web-based shell to an IPython
- A mix of notes, code, html, images, video, ...
- A great tool for debugging, teaching
- Has ability to save, edit and delete notebooks

Как это работает

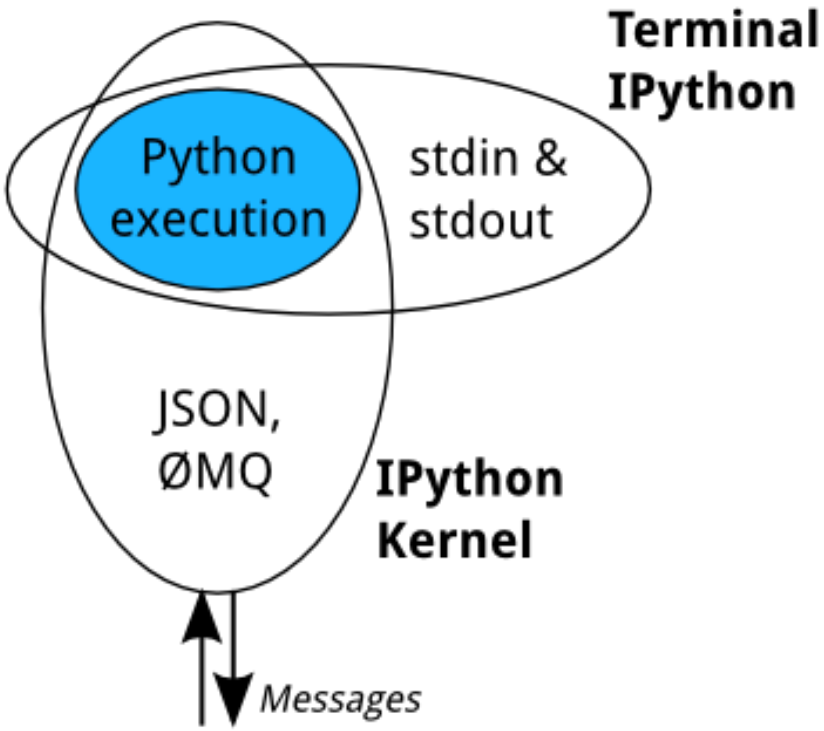
- Terminal IPython - REPL = Read-Eval-Print-Loop
- The IPython kernel – computation and communication with the frontend interfaces, like the notebook

Terminal IPython

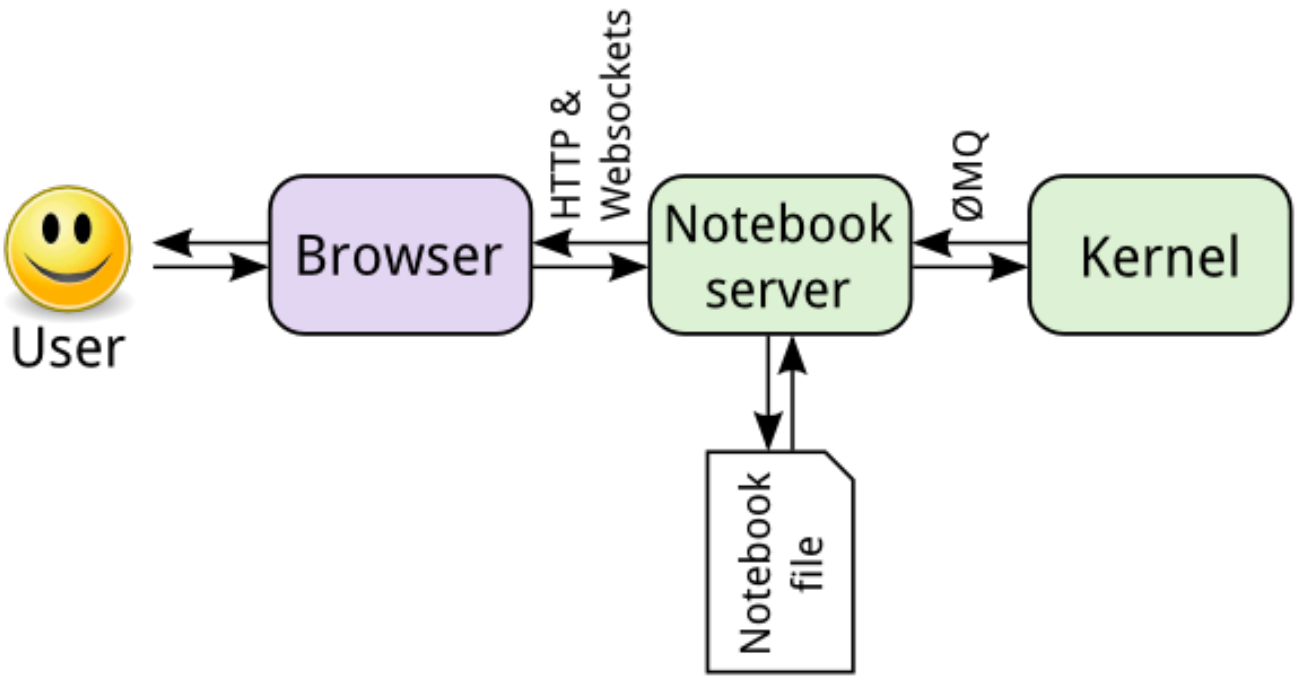
```
while True:
    code = input(">>> ")
    exec(code)
```

```
(base) MacBook-Pro-Artemiy:github artemiykozyr$ ipython
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: 2**128
Out[1]: 340282366920938463463374607431768211456

In [2]:
```

The IPython Kernel

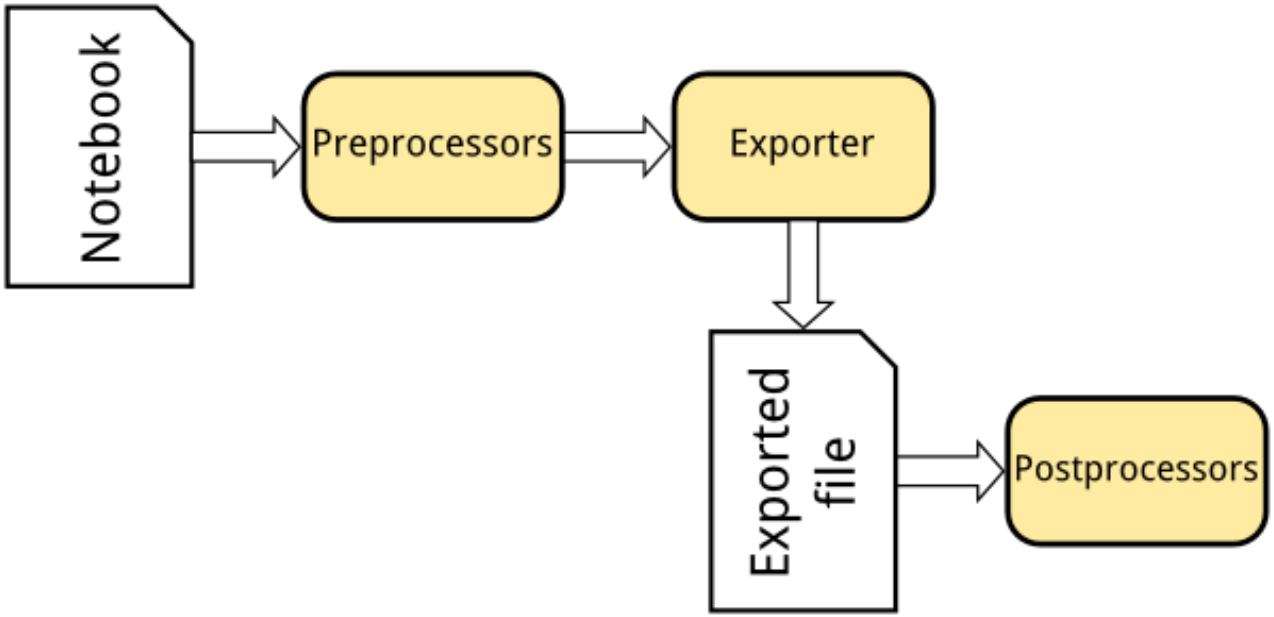


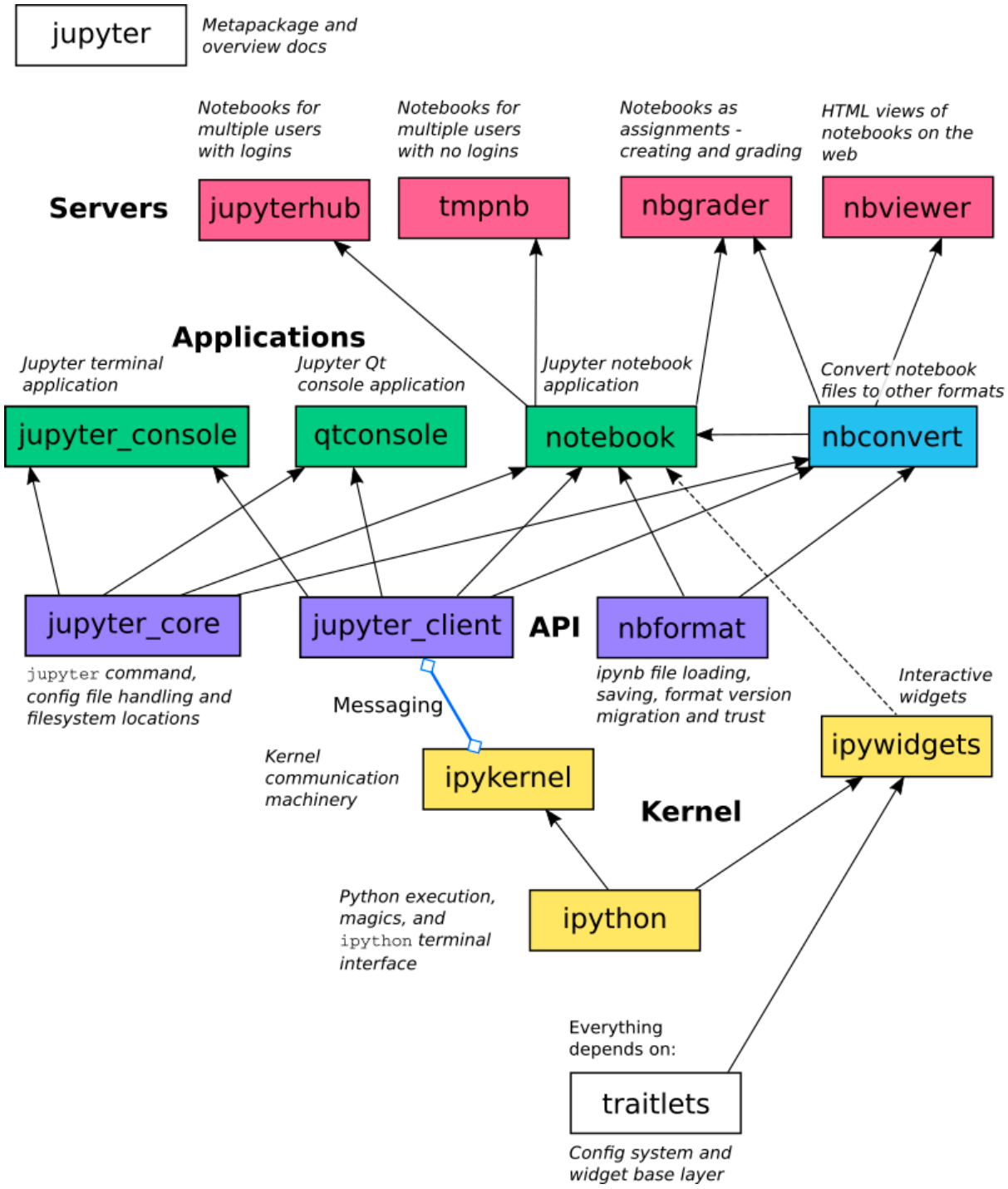
Notebooks


User
Browser
HTTP &
Websockets
Notebook
server
ØMQ
Kernel
Notebook
file

Exporting notebooks

- HTML, LaTeX, or reStructuredText

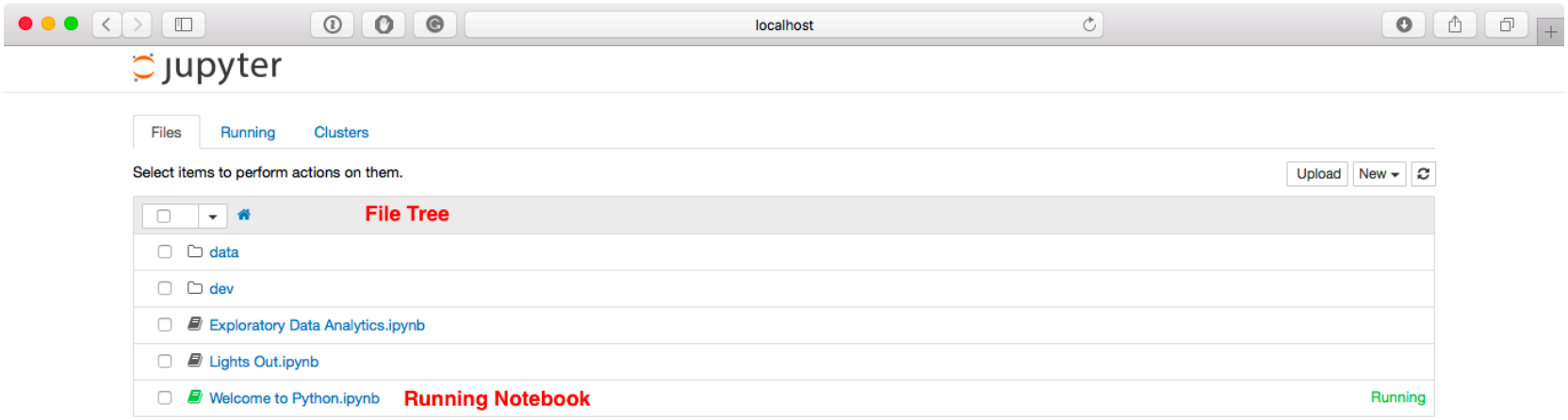



jupyter
Metapackage and overview docs
Notebooks for multiple users with logins
Notebooks for multiple users with no logins
Notebooks as assignments - creating and grading
HTML views of notebooks on the web
Servers
jupyterhub
tmpnb
nbgrader
nbviewer
Applications
Jupyter terminal application
Jupyter Qt console application
Jupyter notebook application
Convert notebook files to other formats
jupyter_console
qtconsole
notebook
nbconvert
jupyter_core
jupyter_client
API
nbformat
jupyter command, config file handling and filesystem locations
Messaging
ipynb file loading, saving, format version migration and trust
Interactive widgets
ipywidgets
Kernel communication machinery
ipykernel
Kernel
Python execution, magics, and ipython terminal interface
ipython
Everything depends on:
traitlets
Config system and widget base layer

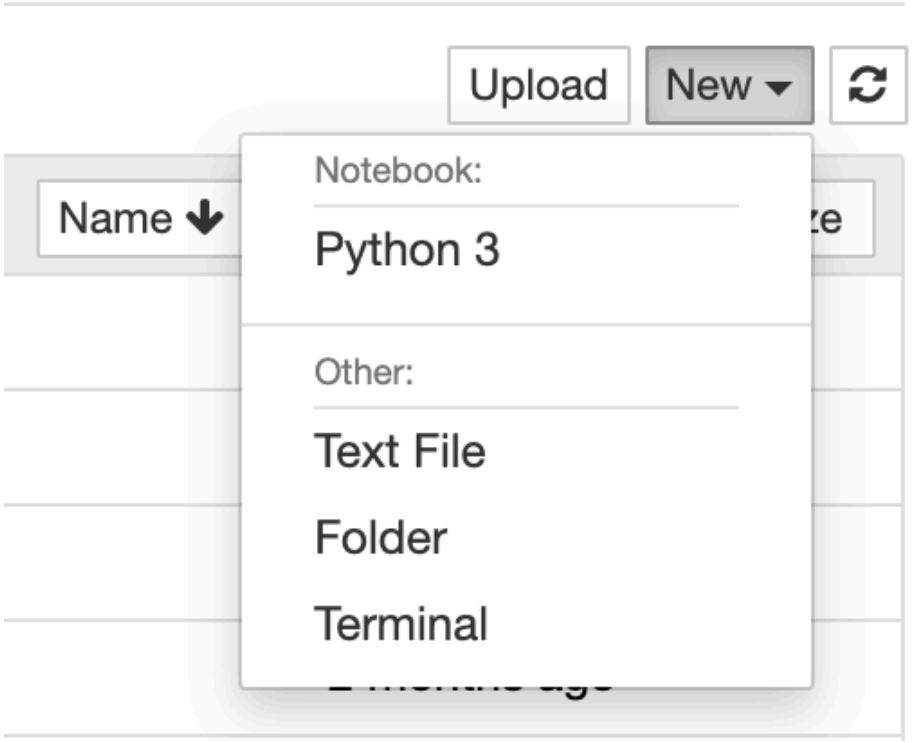
Установка

- Anaconda / Miniconda
- pip3 install jupyter
- Installing Jupyter Notebook

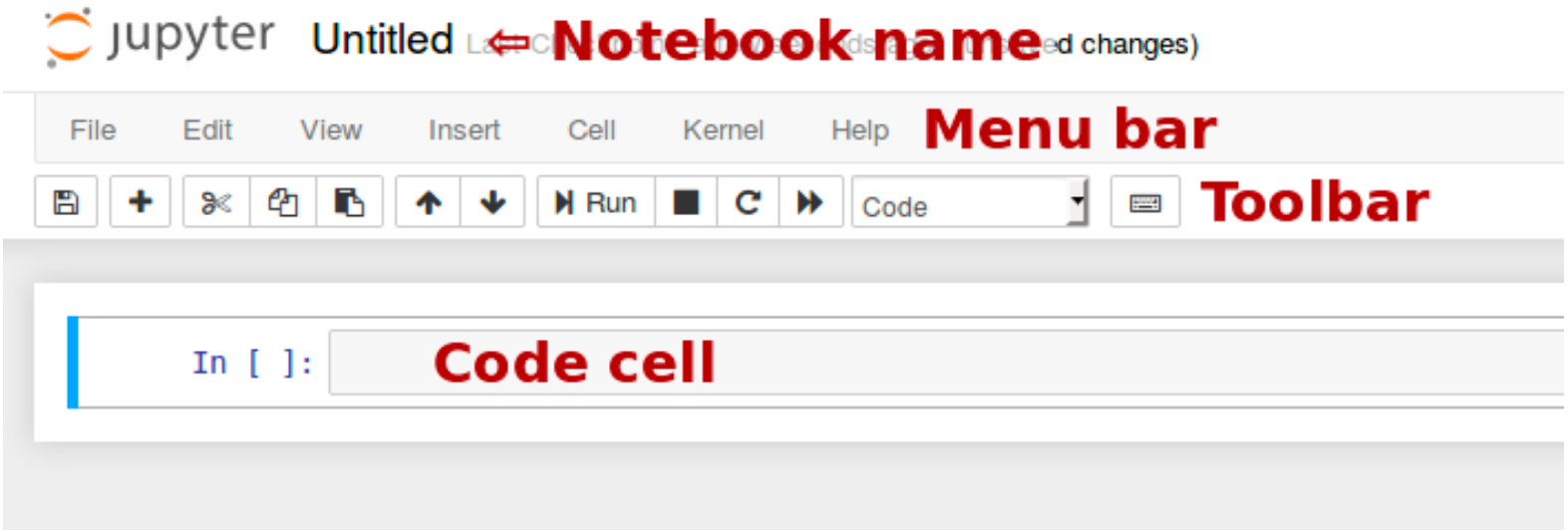
Notebook Dashboard



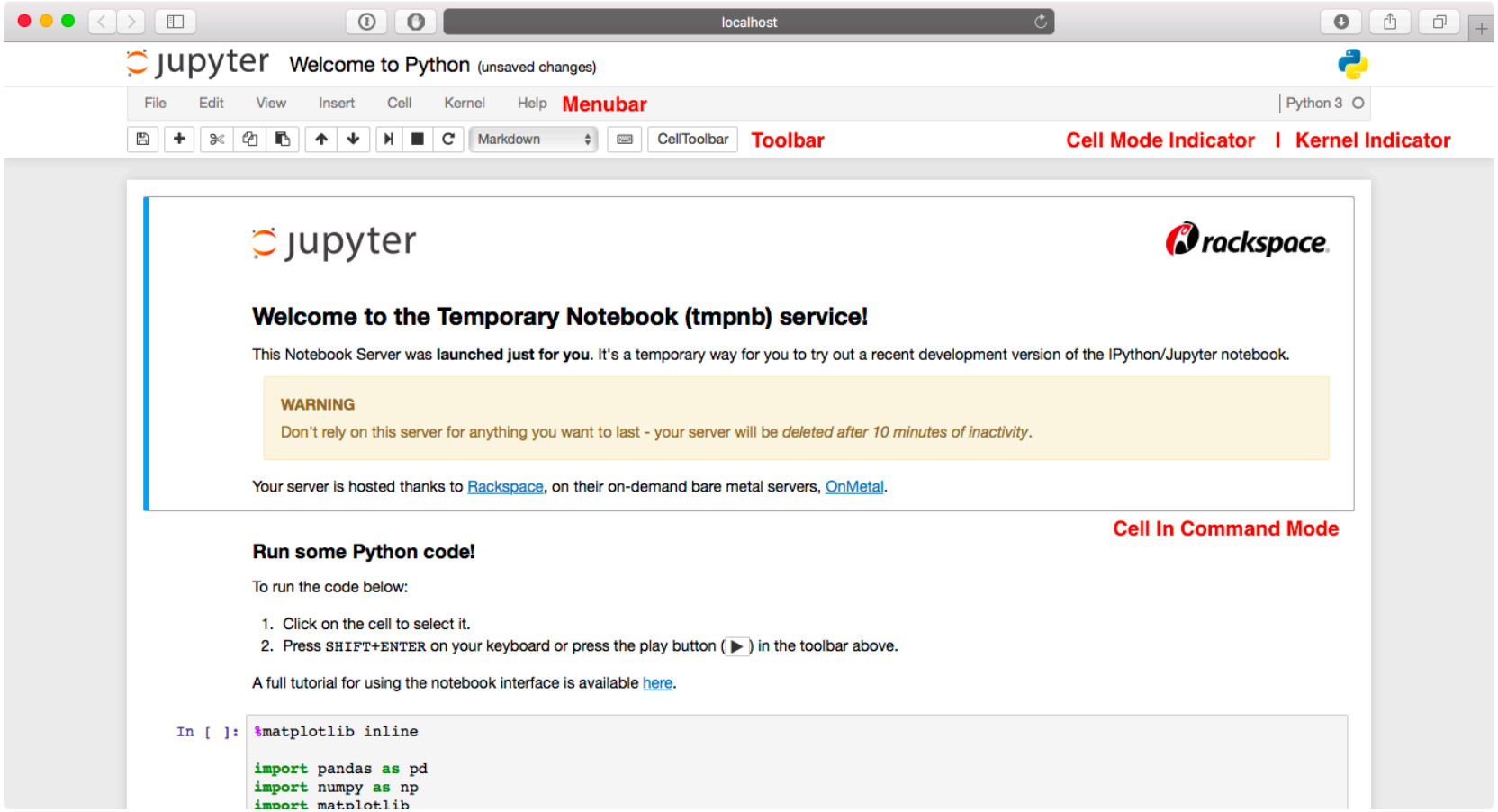
New notebook



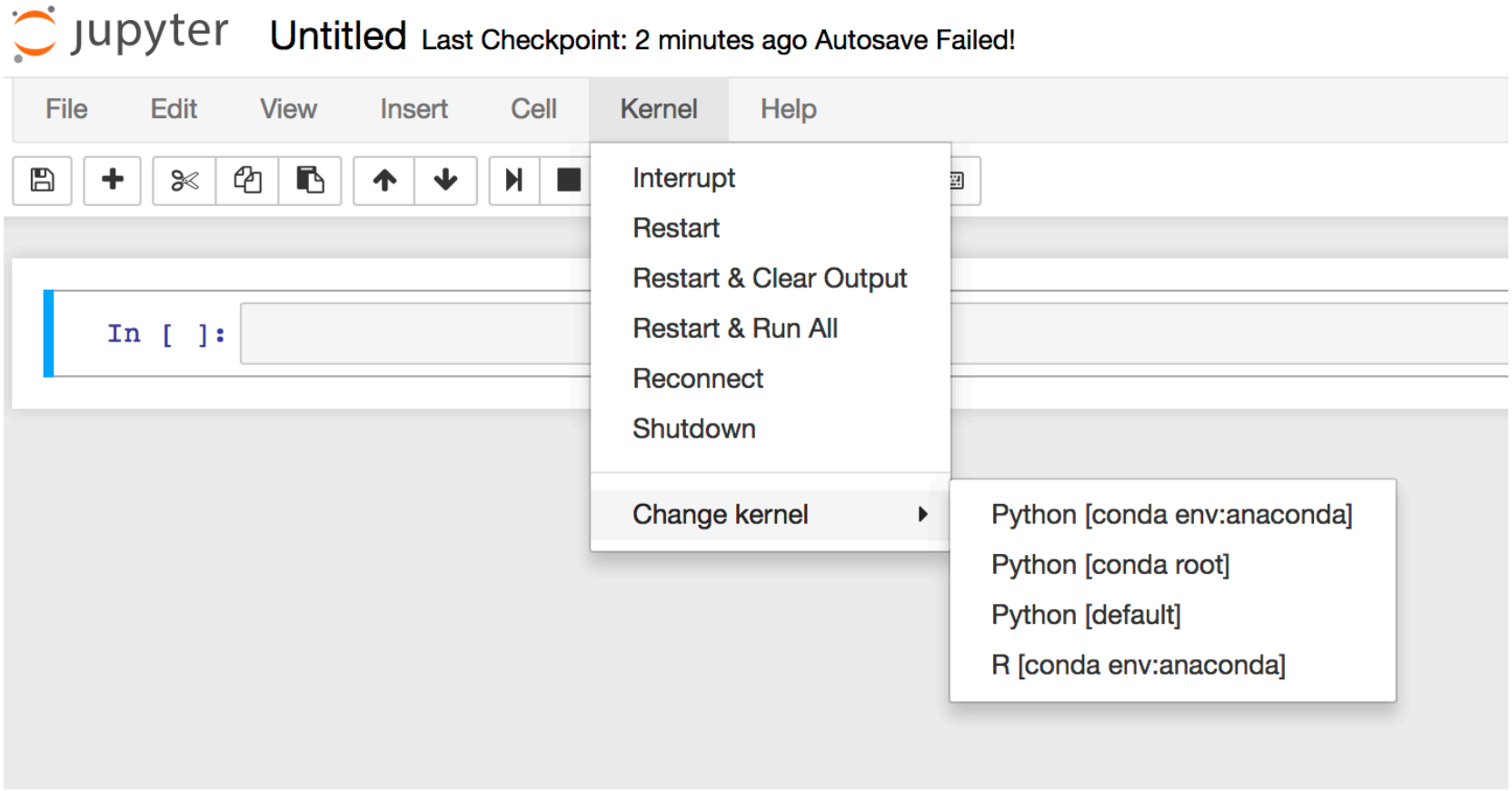
User Interface



Notebook Editor



Jupyter kernels



Структура

- code cells
- markdown cells
- raw cells

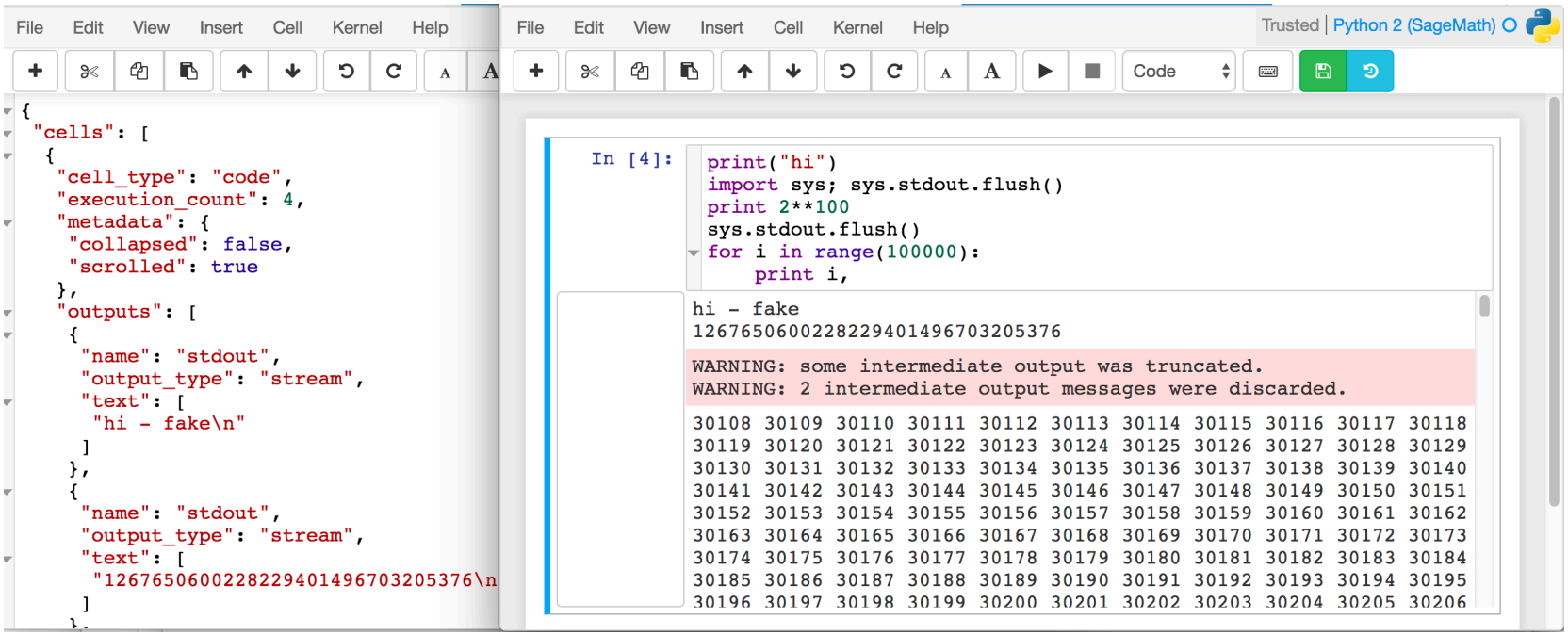
Markdown

```
# This is a level 1 heading
## This is a level 2 heading
This is some plain text that forms a paragraph.
Add emphasis via **bold** and __bold__, or *italic* and _italic_.
Paragraphs must be separated by an empty line.
* Sometimes we want to include lists.
* Which can be indented.
1. Lists can also be numbered.
2. For ordered lists.
[It is possible to include hyperlinks](https://www.example.com)
Inline code uses single backticks: `foo()`, and code blocks use triple backt
```
bar()
```
Or can be indented by 4 spaces:
foo()
And finally, adding images is easy: ![Alt text](https://www.example.com/imag
```

Magic

```
In [14]: %lsmagic

Out[14]: Available line magics:
%alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %c

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript  %%

Automagic is ON, % prefix IS NOT needed for line magics.

In [15]: %quickref
```

.ipynb file


File Edit View Insert Cell Kernel Help
{
"cells": [
{
"cell_type": "code",
"execution_count": 4,
"metadata": {
"collapsed": false,
"scrolled": true
},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"hi - fake\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"126765060022822940149670320537 6\n
File Edit View Insert Cell Kernel Help
Trusted | Python 2 (SageMath)
Code
In [4]:
print("hi")
import sys; sys.stdout.flush()
print 2**100
sys.stdout.flush()
for i in range(100000):
print i,
hi - fake
1267650600228229401496703205376
WARNING: some intermediate output was truncated.
WARNING: 2 intermediate output messages were discarded.
30108 30109 30110 30111 30112 30113 30114 30115 30116 30117 30118
30119 30120 30121 30122 30123 30124 30125 30126 30127 30128 30129
30130 30131 30132 30133 30134 30135 30136 30137 30138 30139 30140
30141 30142 30143 30144 30145 30146 30147 30148 30149 30150 30151
30152 30153 30154 30155 30156 30157 30158 30159 30160 30161 30162
30163 30164 30165 30166 30167 30168 30169 30170 30171 30172 30173
30174 30175 30176 30177 30178 30179 30180 30181 30182 30183 30184
30185 30186 30187 30188 30189 30190 30191 30192 30193 30194 30195
30196 30197 30198 30199 30200 30201 30202 30203 30204 30205 30206

Top-level structure

At the highest level, a Jupyter notebook is a dictionary with a few keys:

- metadata (dict)
- nbformat (int)
- nbformat_minor (int)
- cells (list)

```
{
  "metadata" : {
    "signature": "hex-digest", # used for authenticating unsafe outputs on load
    "kernel_info": {
        # if kernel_info is defined, its name field is required.
        "name" : "the name of the kernel"
    },
    "language_info": {
        # if language_info is defined, its name field is required.
        "name" : "the programming language of the kernel",
        "version": "the version of the language",
        "codemirror_mode": "The name of the codemirror mode to use [optional]"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 0,
  "cells" : [
      # list of cell dictionaries, see below
  ],
}
```

Cell Types

There are a few basic cell types for encapsulating code and text. All cells have the following basic structure:

```
{
  "cell_type" : "name",
  "metadata" : {},
  "source" : "single string or [list, of, strings]",
}
```

Markdown cells

Markdown cells are used for body-text, and contain markdown, as defined in GitHub-flavored markdown, and implemented in marked.

```
{
  "cell_type" : "markdown",
  "metadata" : {},
  "source" : ["some *markdown*"],
}
```

> *Changed in version nbformat:* 4.0
>
> Heading cells have been removed, in favor of simple headings in markdown.

Code cells

Code cells are the primary content of Jupyter notebooks. They contain source code in the language of the document's associated kernel, and a list of outputs associated with executing that code. They also have an execution_count, which must be an integer or `null`.

```
{
  "cell_type" : "code",
  "execution_count": 1, # integer or null
  "metadata" : {
      "collapsed" : True, # whether the output of the cell is collapsed
      "autoscroll": False, # any of true, false or "auto"
  },
  "source" : ["some code"],
  "outputs": [{
      # list of output dicts (described below)
      "output_type": "stream",
      ...
  }],
}
```

Code cell outputs

A code cell can have a variety of outputs (stream data or rich mime-type output). These correspond to messages produced as a result of executing the cell.

All outputs have an `output_type` field, which is a string defining what type of output it is.

stream output

```
{
  "output_type" : "stream",
  "name" : "stdout", # or stderr
  "text" : ["multiline stream text"],
}
```

Changed in version nbformat: 4.0

The keys `stream` key was changed to `name` to match the stream message.

display_data

Rich display outputs, as created by `display_data` messages, contain data keyed by mime-type. This is often called a mime-bundle, and shows up in various locations in the notebook format and message spec. The metadata of these messages may be keyed by mime-type as well.

```
{
  "output_type" : "display_data",
  "data" : {
    "text/plain" : ["multiline text data"],
    "image/png": ["base64-encoded-png-data"],
    "application/json": {
      # JSON data is included as-is
      "json": "data",
    },
  },
  "metadata" : {
    "image/png": {
      "width": 640,
      "height": 480,
    },
  },
}
```

Metadata

Metadata is a place that you can put arbitrary JSONable information about your notebook, cell, or output. Because it is a shared namespace, any custom metadata should use a sufficiently unique namespace, such as `metadata.kaylees_md.foo = "bar"`.

Metadata fields officially defined for Jupyter notebooks are listed here:

Notebook metadata

The following metadata keys are defined at the notebook level:

Key	Value	Interpretation
kernelspec	dict	A kernel specification
signature	str	A hashed signature of the notebook

Cell metadata

The following metadata keys are defined at the cell level:

Key	Value	Interpretation
collapsed	bool	Whether the cell's output container should be collapsed
autoscroll	bool or 'auto'	Whether the cell's output is scrolled, unscrolled, or autoscrolled
deletable	bool	If False, prevent deletion of the cell
format	'mime/type'	The mime-type of a Raw NBConvert Cell
name	str	A name for the cell. Should be unique
tags	list of str	A list of string tags on the cell. Commas are not allowed in a tag

Output metadata

The following metadata keys are defined for code cell outputs:

Key	Value	Interpretation
isolated	bool	Whether the output should be isolated into an IFrame

Исследование данных

```
df.head()
```

	year	rank	company	revenue (in millions)	profit (in millions)
0	1955	1	General Motors	9823.5	806
1	1955	2	Exxon Mobil	5661.4	584.8
2	1955	3	U.S. Steel	3250.4	195.4
3	1955	4	General Electric	2959.1	212.6
4	1955	5	Esmark	2510.8	19.1

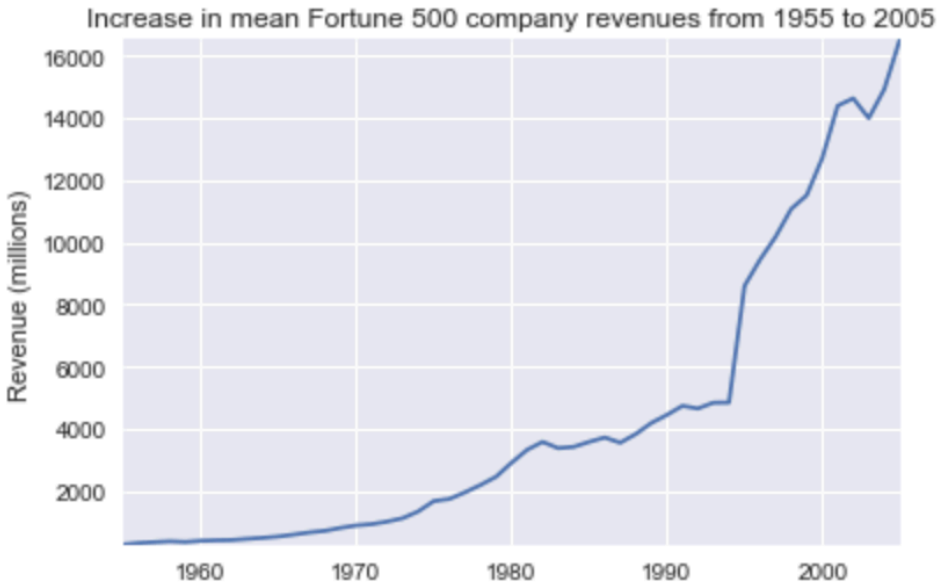
Исследование данных

```
df.dtypes
```

```
year int64
rank int64
company object
revenue float64
profit object
dtype: object
```

Графики на matplotlib

```
y2 = avgs.revenue
fig, ax = plt.subplots()
plot(x, y2, ax, 'Increase in mean Fortune 500 company revenues from 1955 to
```



Cheat sheets

- Jupyter Notebook 1 Page Overview
- Jupyter Notebook Cheatsheet
- Jupyter Notebook Keyboard Shortcuts
- IPhyton Notebook Keyboard Shortcuts
- Markdown for Jupyter notebooks cheatsheet
- Conda Cheatsheet

Learning Notebooks

- Notebook Examples
- IPython Cookbook by Cyrille Rossant
- Google Cloud Datalab samples
- PyCon 2019 Jupyter tutorial

Практика

- Launch Cloud Datalab

План занятия

- Что такое Jupyter
- Архитектура
- Обзор интерфейса
- Метаданные
- Cheat sheets
- Практика

Рефлексия

- Что вам запомнилось больше всего?
- Пройти опрос

Ваши вопросы?