Знакомство с Jupyter Notebook

Jupyter Notebook чем-то похож на текстовый редактор: есть меню, панель инструментов и поле для работы, которое состоит из ячеек (*cells*). Ячейки могут содержать как код, так и текст (неразмеченный и размеченный). Комментарии в ячейках с кодом должны начинаться со знака `#` .

Пример ячейки с кодом:

In [1]:
```
a = 3 # создаем переменную a и присваиваем ей значение 3
b = 0.2

# комментарий 1
# комментарий 2

print(a, b) # выводим на экран значения переменных A и B
```

```
3 0.2
```

По умолчанию тип ячейки *Code*, это можно увидеть на панели под меню. Чтобы изменить тип ячейки, нужно нажать на стрелочку вниз и выбрать нужный вариант. Всего вариантов четыре: *Code*, *Markdown*, *RawNBConvert* и *Heading*.

- *Code*: ячейка с кодом Python;
- *Markdown*: ячейка с размеченным текстом, язык разметки Markdown;
- *RawNBConvert*: неразмеченный «сырой» текст, без курсива/полужирного шрифта;
- *Heading*: устарел, раньше использовался для заголовков, сейчас их нужно создавать в режиме *Markdown*.

Примеры ячеек с размеченным текстом (язык разметки Markdown, почитать про него можно здесь и здесь) — см. ниже.

Что можно делать с помощью Markdown?

Выделять заголовки разных уровней.

На входе:

```
# Заголовок 1
## Заголовок 2
### Заголовок 3
```

На выходе:

Заголовок 1

Заголовок 2

Заголовок 3

Выделять части текста с помощью *курсива* или **полужирного начертания**. Или даже ~~зачеркивать~~. Вот так этот текст выглядит в размеченном виде:

```
Выделять части текста с помощью *курсива* или **полужирного начертания**. Или даже <s>зачеркивать</s>.
```

Добавлять списки разного вида.

Ненумерованный список.

На входе:

```
* пункт 1
* пункт 2
* пункт 3
```

На выходе:

- пункт 1
- пункт 2
- пункт 3

Нумерованный список:

На входе:

```
1. Во-первых,...
2. Во-вторых,...
3. В-третьих,...
```

На выходе:

1. Во-первых,...
2. Во-вторых,...
3. В-третьих,...

Для тех, кто знаком с LaTeX: можно красиво оформлять формулы, используя синтаксис, принятый в LaTeX (наличие установленного на компьютере LaTeX не требуется, текст компилируется в самом Jupyter Notebook):

$$(x + y)^2 = x^2 + 2xy + y^2$$

$$\Delta = I_{2018} - I_{2017}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$\sin(\alpha)^2 + cos(\alpha)^2 = 1$$

Формулы были получены таким образом:

```
$$(x+y)^2 = x^2 + 2xy + y^2$$
```

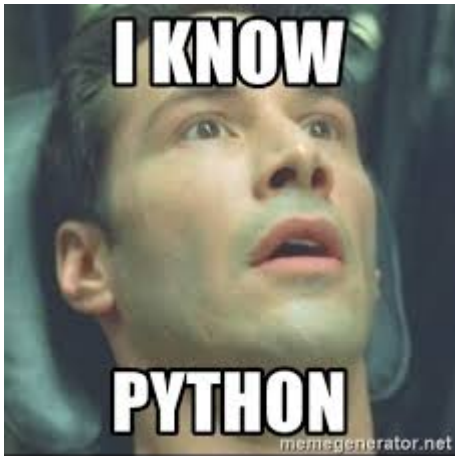
```
$$\Delta = I_{2018} - I_{2017}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$\sin(\alpha)^2 + cos(\alpha)^2 = 1$$
```

Еще можно добавлять **ссылки**. Либо текст ссылки и саму ссылку: текст `[Jupyter](http://jupyter.org)` даст ссылку Jupyter. Либо саму ссылку как есть: текст `<http://jupyter.org/>` вернет ссылку http://jupyter.org/.

И, конечно, можно добавлять картинки!

Например, так:

```
![title](https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcReyae6v_NjByqshfjUhNtMgE-7c_zeYPt81IWrZ4dOqH5pYDvw)
```



Или так (синтаксис, как в html, кто знаком):

```
<img src ="https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcReyae6v_NjByqshfjUhNtMgE-7c_zeYPt81IWrZ4dOqH5pYDvw" alt ="Test picture">
```

Примечание: часть с *alt* нужна для того, чтобы в случае, если картинки с таким названием нет, выводился какой-то альтернативный текст вместо нее.

Горячие клавиши

В Jupyter Notebook есть свои горячие клавиши, которые позволяют добавлять или удалять ячейки, менять их тип и так далее. Чтобы работать в режиме горячих клавиш, нужно выйти из редактирования ячейки (нажать на *Esc*), а потом набрать нужную комбинацию клавиш.

Чтобы вернуться в режим редактирования ячейки, можно кликнуть на нее два раза или, выбрав ее, нажать *Enter*. Все полезные комбинации клавиш можно найти в меню (*Help - Keyboard Shortcuts*) или нажав *Esc*, а потом *H* или *P*.

Если потерялись

И наконец: как найти, куда Jupyter сохраняет ноутбуки — файлы с расширением `.ipynb`. Можно импортировать модуль *os* и узнать текущую рабочую папку (*current working directory*) — в ней будут лежать ноутбуки.

In [ ]:
```
import os
os.getcwd()
```

С помощью `import` мы будем импортировать любые модули или библиотеки Python, а также отдельные функции при необходимости. Кроме того, можем импортировать текст философии Python, которая имеет название "The Zen of Python":

In [5]:
```
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!
```