



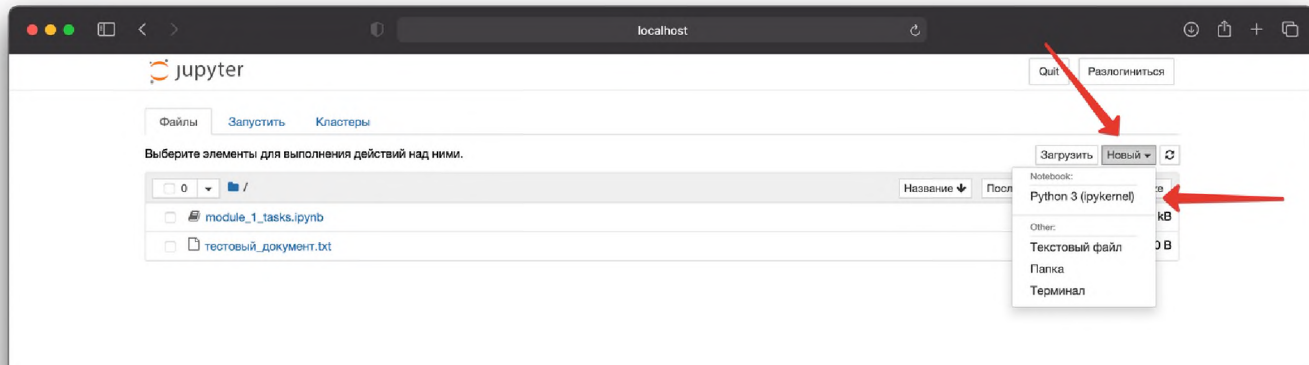
Среда разработки Jupyter Notebook

Среда разработки Jupyter Notebook

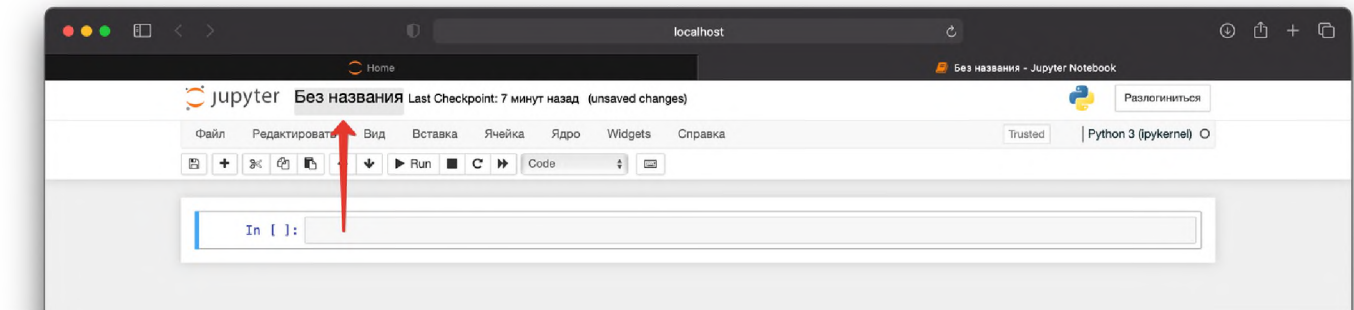
Блокноты Jupyter

Объекты, с которыми мы работаем в среде Jupyter Notebook, называются **блокнотами**.

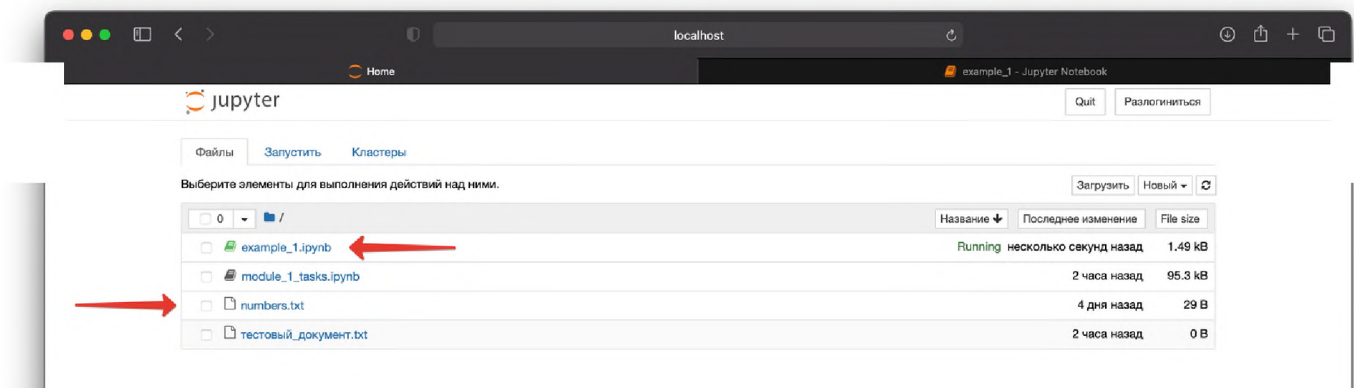
Чтобы создать новый блокнот в рабочей папке, нужно нажать на кнопку «Новый» и выбрать вариант «Python 3». Это соответствует тому, что вы создаёте блокнот для языка программирования Python (бывают блокноты и для других языков программирования):



актуальное название (например, `example_1`):



Каждый новый просматриваемый блокнот открывается в новой вкладке браузера. Давайте вернёмся на первоначальную вкладку браузера и убедимся, что новый блокнот создался в нашей рабочей папке:



Видим, что в рабочей папке действительно появился новый файл `example_1.ipynb` — это как раз наш блокнот.

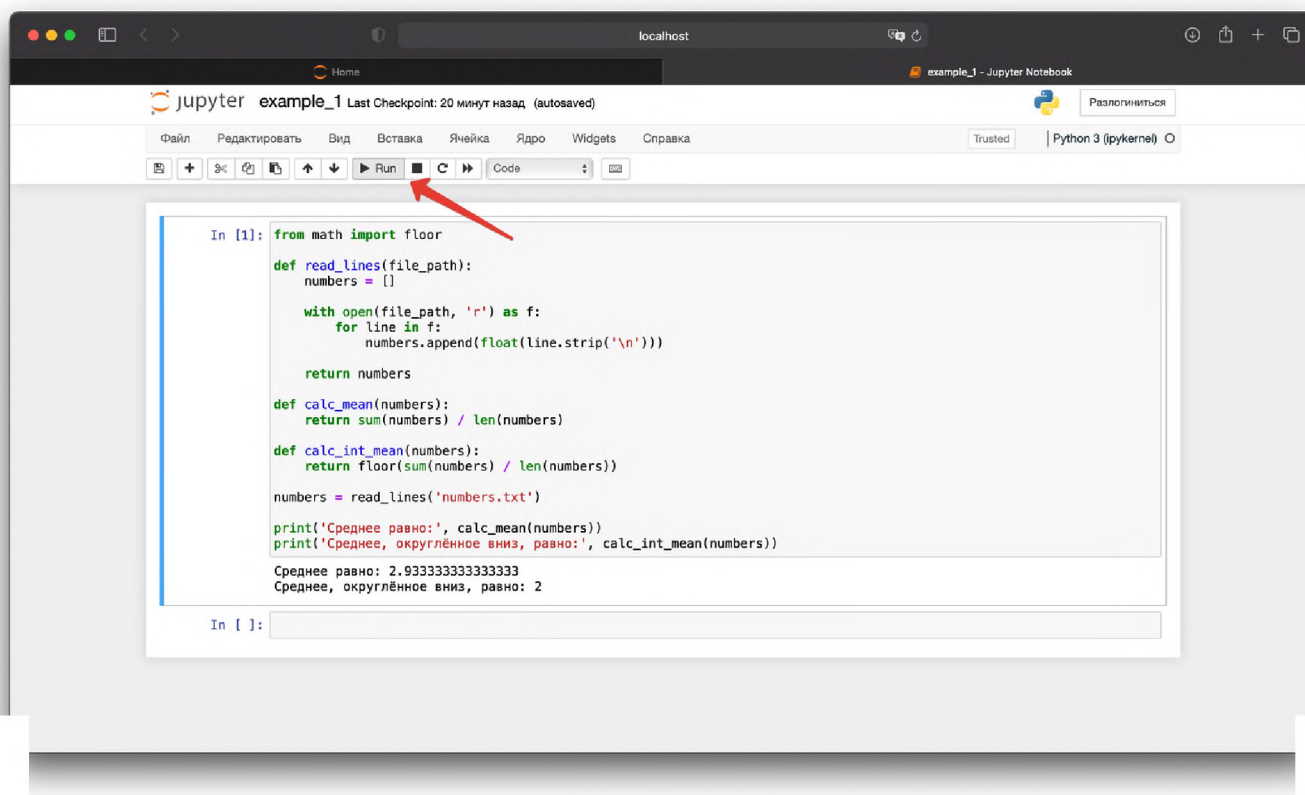
Важно, что любому блокноту, который находится в нашей рабочей папке, будут доступны (например, для чтения) все файлы, которые находятся в этой папке. В частности, внутри нашего блокнота `example_1` мы будем активно использовать файл `numbers.txt`, который тоже лежит в нашей рабочей папке.

Ячейки

Каждый блокнот состоит из **ячеек**. Ячейки в первую очередь используются для того, чтобы писать в них код.

Запуск

чтобы исполнить код, который написан в ячейке, нужно нажать на кнопку «Run» (или «Запуск»).

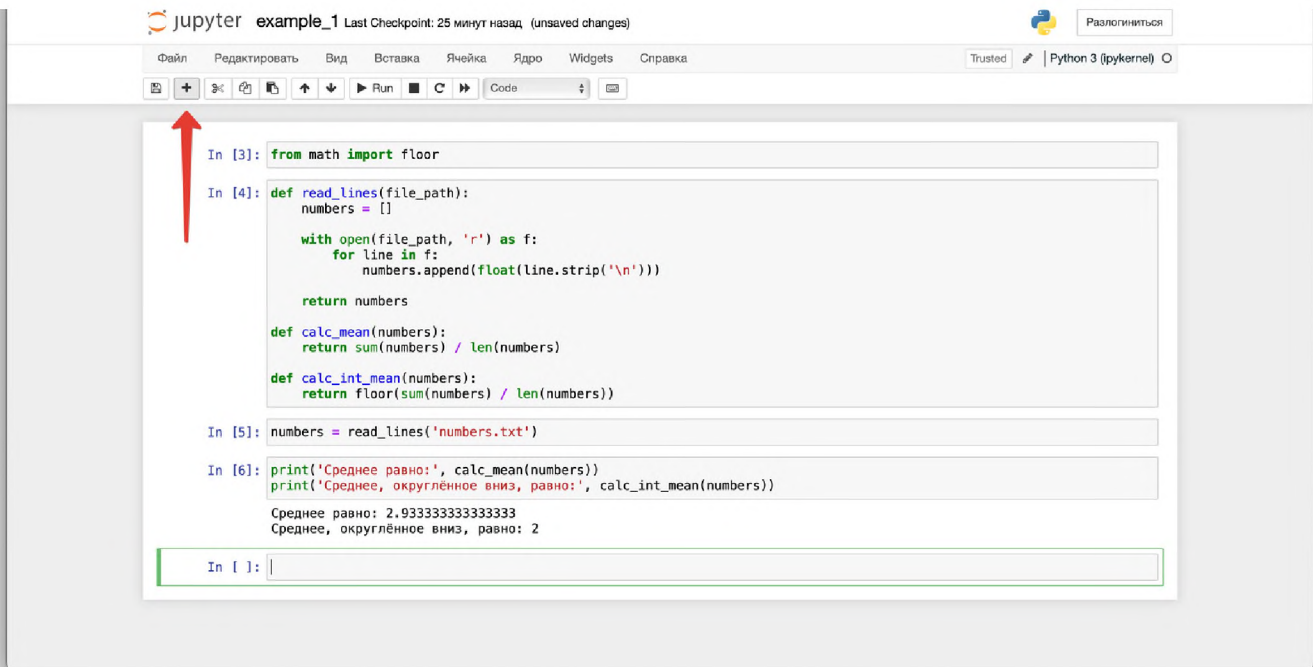


Также для запуска ячейки с кодом можно нажать на ячейку и воспользоваться горячим сочетанием клавиш **Shift+Enter**. Данное действие эквивалентно запуску ячейки по отдельной кнопке.

Создание

Большое преимущество блокнотов Jupyter заключается в том, что в них можно создавать много ячеек. В отдельные ячейки принято помещать отдельные кусочки логики программы. Новую ячейку можно создать, воспользовавшись кнопкой «+» в верхнем левом углу экрана.

В случае нашего примера логично вынести в отдельную ячейку импорты, в отдельную — функции, а также перенести в отдельные ячейки чтение информации из файла `numbers.txt` и вывод результата работы функций на экран. Получится следующий блокнот:



Переиспользование вычислений

Одной из особенностей того, что программу можно представить в виде системы ячеек, является возможность переиспользовать вычисления из отдельных ячеек.

Если в какой-то ячейке происходит запись значения в переменную, то значение переменной сохраняется в общую память блокнота и становится доступно всем другим ячейкам.

Например, в нашем случае любая ячейка может использовать переменную `numbers` даже при том, что в самих этих ячейках переменная `numbers` не определена:

```
jupyter example_1 Last Checkpoint: 35 минут назад (unsaved changes)
Файл Редактировать Вид Вставка Ячейка Ядро Widgets Справка Trusted Python 3 (ipykernel)

with open(file_path, 'r') as f:
    for line in f:
        numbers.append(float(line.strip('\n')))

return numbers

def calc_mean(numbers):
    return sum(numbers) / len(numbers)

def calc_int_mean(numbers):
    return floor(sum(numbers) / len(numbers))

In [5]: numbers = read_lines('numbers.txt')

In [6]: print('Среднее равно:', calc_mean(numbers))
        print('Среднее, округлённое вниз, равно:', calc_int_mean(numbers))
Среднее равно: 2.933333333333333
Среднее, округлённое вниз, равно: 2

In [7]: calc_mean(numbers)
Out[7]: 2.933333333333333

In [8]: calc_int_mean(numbers)
Out[8]: 2

In [ ]:
```

Это очень удобно в случае, если вычисление в какой-то ячейке занимает очень много времени и при этом в дальнейшем будет часто переиспользоваться.

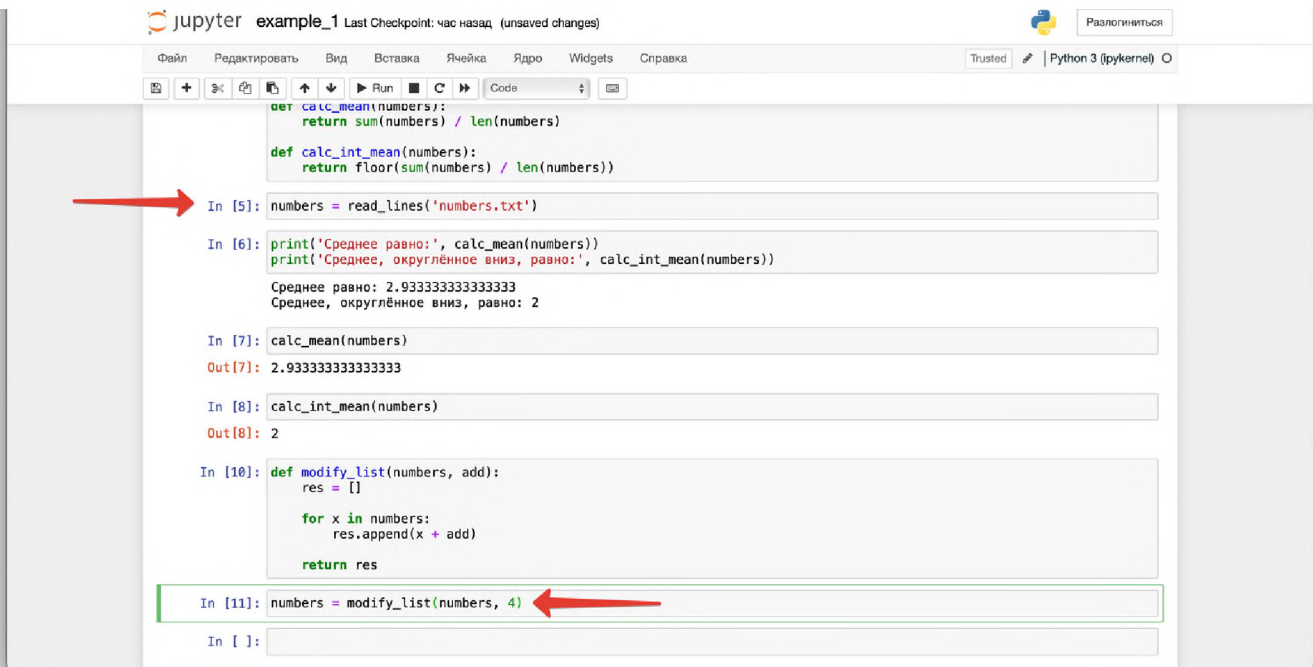
Например, в нашем случае чтение файла `numbers.txt` могло бы занимать какое-то значительное время (например, если бы файл был очень большим). Поэтому разумно вынести его в отдельную ячейку.

В этом случае в любой другой ячейке при обработке значений из файла `numbers` мы не тратим времени на его чтение, используя уже хранящееся в памяти значение.

Порядок вычислений

Ещё одной важной особенностью работы с ячейками является то, что порядок их исполнения задаётся порядком, в котором они вычисляются, а не порядком их следования.

Например, ячейка, которая расположена в блокноте ниже другой ячейки, может повлиять на результат её вычисления в будущем. В нашем примере с помощью функции `modify_list` мы меняем список `numbers`:



The screenshot shows a Jupyter Notebook window titled "example_1 Last Checkpoint: час назад (unsaved changes)". The interface includes a menu bar with options like "Файл", "Редактировать", "Вид", "Вставка", "Ячейка", "Ядро", "Widgets", and "Справка". Below the menu is a toolbar with icons for file operations and execution. The main area contains a code editor with the following Python code:

```
def calc_mean(numbers):  
    return sum(numbers) / len(numbers)  
  
def calc_int_mean(numbers):  
    return floor(sum(numbers) / len(numbers))
```

Below the code editor, there are several input and output cells. A red arrow points to the input cell for `In [5]: numbers = read_lines('numbers.txt')`. The output of this cell is not shown. The next input cell is `In [6]: print('Среднее равно:', calc_mean(numbers))`, which has an output of `Среднее равно: 2.933333333333333`. The next input cell is `In [7]: calc_mean(numbers)`, which has an output of `Out[7]: 2.933333333333333`. The next input cell is `In [8]: calc_int_mean(numbers)`, which has an output of `Out[8]: 2`. The next input cell is `In [10]: def modify_list(numbers, add):`, which has an output of `Out[10]: 2`. The final input cell is `In [11]: numbers = modify_list(numbers, 4)`, which is highlighted with a green border and a red arrow pointing to it. The output of this cell is not shown.

Это приводит к тому, что значение переменной `numbers` меняется в общей памяти блокнота. Из-за этого любая ячейка, которая использует значение переменной и вычисляется после как мы изменили это значение, будет работать с обновлённой версией `numbers`.

тности, если теперь мы перезапустим вычисление ячейки, которая рассчитывает среднее значение для списка `numbers` и которую мы уже когда-то запускали, среднее изменится. Ровно потому, что теперь в `numbers` представлены другие значения.

The screenshot shows a Jupyter Notebook titled 'example_1' with a 'Last Checkpoint: час назад (unsaved changes)' status. The interface includes a menu bar (Файл, Редактировать, Вид, Вставка, Ячейка, Ядро, Widgets, Справка) and a toolbar with icons for file operations, running, and saving. The code editor contains the following Python code:

```
def read_lines(file_path):  
    numbers = []  
  
    with open(file_path, 'r') as f:  
        for line in f:  
            numbers.append(float(line.strip('\n')))  
  
    return numbers  
  
def calc_mean(numbers):  
    return sum(numbers) / len(numbers)  
  
def calc_int_mean(numbers):  
    return floor(sum(numbers) / len(numbers))
```

The notebook shows the execution of several cells:

- In [5]: `numbers = read_lines('numbers.txt')`
- In [12]: `print('Среднее равно:', calc_mean(numbers))`
`print('Среднее, округлённое вниз, равно:', calc_int_mean(numbers))`
Output: `Среднее равно: 6.933333333333334`
`Среднее, округлённое вниз, равно: 6`
- In [7]: `calc_mean(numbers)`
Output [7]: `2.933333333333333`
- In [8]: `calc_int_mean(numbers)`
Output [8]: `2`
- In [10]: `def modify_list(numbers, add):`
`res = []`
`for x in numbers:`
`res.append(x + add)`

A red arrow points to the output of the `calc_mean` function in the second cell, which is `2.933333333333333`.

То есть при работе с ячейками важно помнить, что порядок их выполнения совпадает с порядком, в котором вы их запускаете, а не с их порядком следования в блокноте. И запускать можно в абсолютно любом порядке, не обязательно сверху—вниз.

данные из лекции

- Блокнот ([example 1.ipynb](#)), который использовался в лекции.
- Файл `numbers.txt` ([numbers.txt](#)).