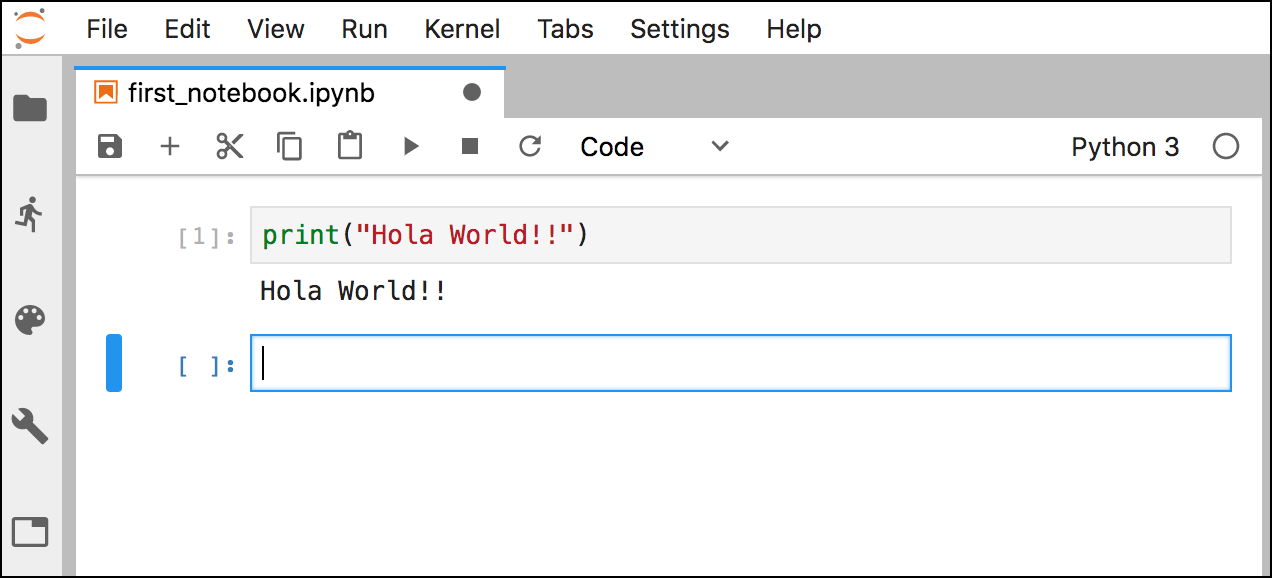
**Поиск угроз с помощью Jupyter Notebook.**

**Часть 1. Ваш первый Notebook.**



Когда речь заходит об обнаружении угрозы, сколько раз вы слышали, как кто-то говорит: «Всё в моей голове, просто спросите меня, если у вас есть какие-либо вопросы!» или «Только он / она / они знает(ют), как это сделать!» Много раз, верно? Отсутствие документирования, стандартизации или обмена информацией о том, как анализировать данные для обнаружения потенциальных вторжений в сеть, встречается чаще, чем вы думаете, особенно если команда разнообразна с технической и экспертной точки зрения. Это влияет не только на стратегии обнаружения, но и на динамику вашей команды.

Теперь, сколько раз вы также думали о более эффективном, интуитивном или творческом способе анализа событий безопасности, которые собирает ваша организация? И считали ли вы возможности панели поиска, зависящей от одного языка, ограниченными?

Эта статья является частью из пяти статей, в которой будет представлена ​​концепция использования **Jupyter Notebook** для более динамичного, гибкого и независимого от языка способа анализа событий безопасности. В это же время она поможет вашей команде документировать, стандартизировать и совместно использовать сценарии по обнаружению. Что-то, что вы можете интегрировать с такими проектами, как [ThreatHunter-Playbook](https://github.com/Cyb3rWard0g/ThreatHunter-Playbook), и развёртывать его дома бесплатно и в течение неограниченного времени с такими проектами с открытым исходным кодом, как [HELK](https://github.com/Cyb3rWard0g/HELK).

В первой статье я расскажу об основах работы **Jupyter Notebook**, о том, как создать свой первый «notebook» (блокнот) и о том, как выполнить несколько начальных базовых команды в Python.

Остальные части можно найти по следующим ссылкам:

* Поиск угроз с помощью Jupyter Notebook. Часть 2. Базовый анализ данных с помощью Pandas.
* Поиск угроз с помощью Jupyter Notebook. Часть 3. Запросы Elasticsearch через Apache Spark.
* Поиск угроз с помощью Jupyter Notebook. Часть 4. SQL JOIN через Apache SparkSQL.
* Поиск угроз с помощью Jupyter Notebook. Часть 5. Документирование, совместное использование и запуск книг-охотников за угрозами!

**Что такое Notebook?**

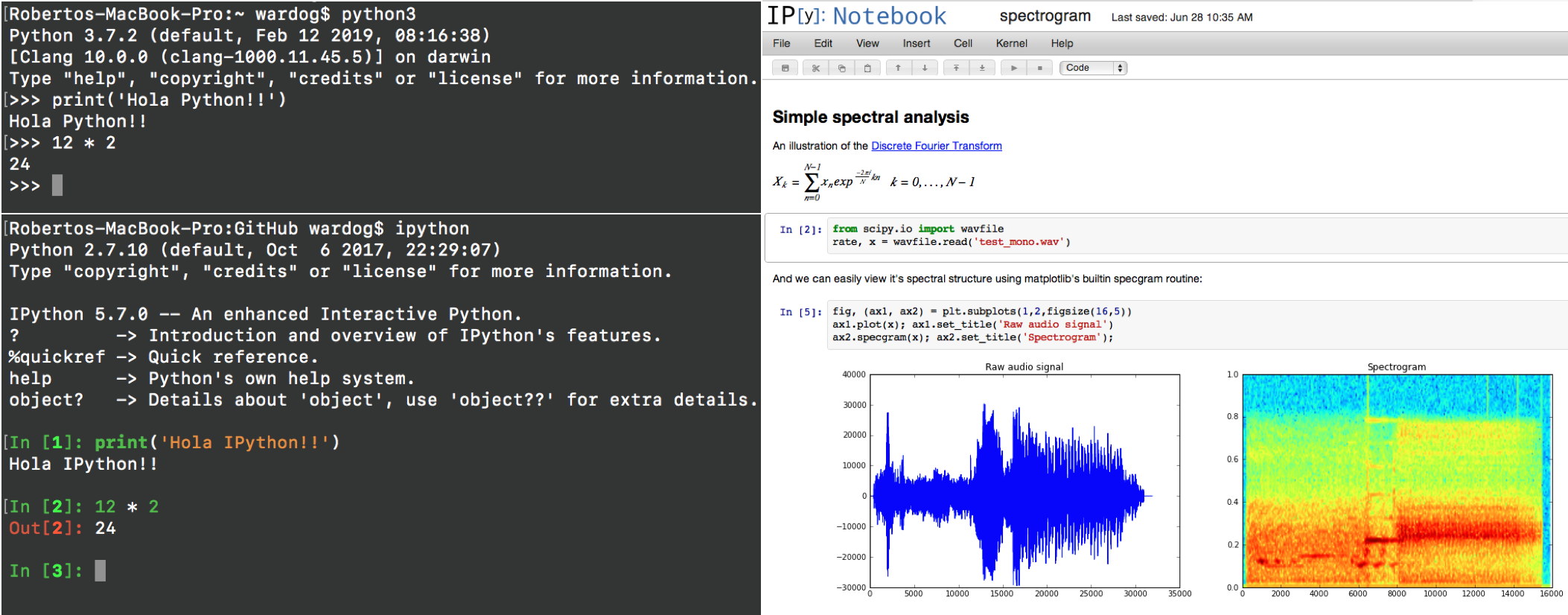
Представляйте блокнот как документ, к которому вы можете получить доступ через веб-интерфейс, позволяющий сохранять **вход**(т. е. живой код) и **выход**(т. е. результаты выполнения кода / вывод оцененного кода) интерактивных сеансов, а также важные примечания. Такие примечания необходимы для объяснения методологии и шагов, предпринятых для выполнения конкретных задач (например, анализ данных).

**Что такое Jupyter Notebook?**

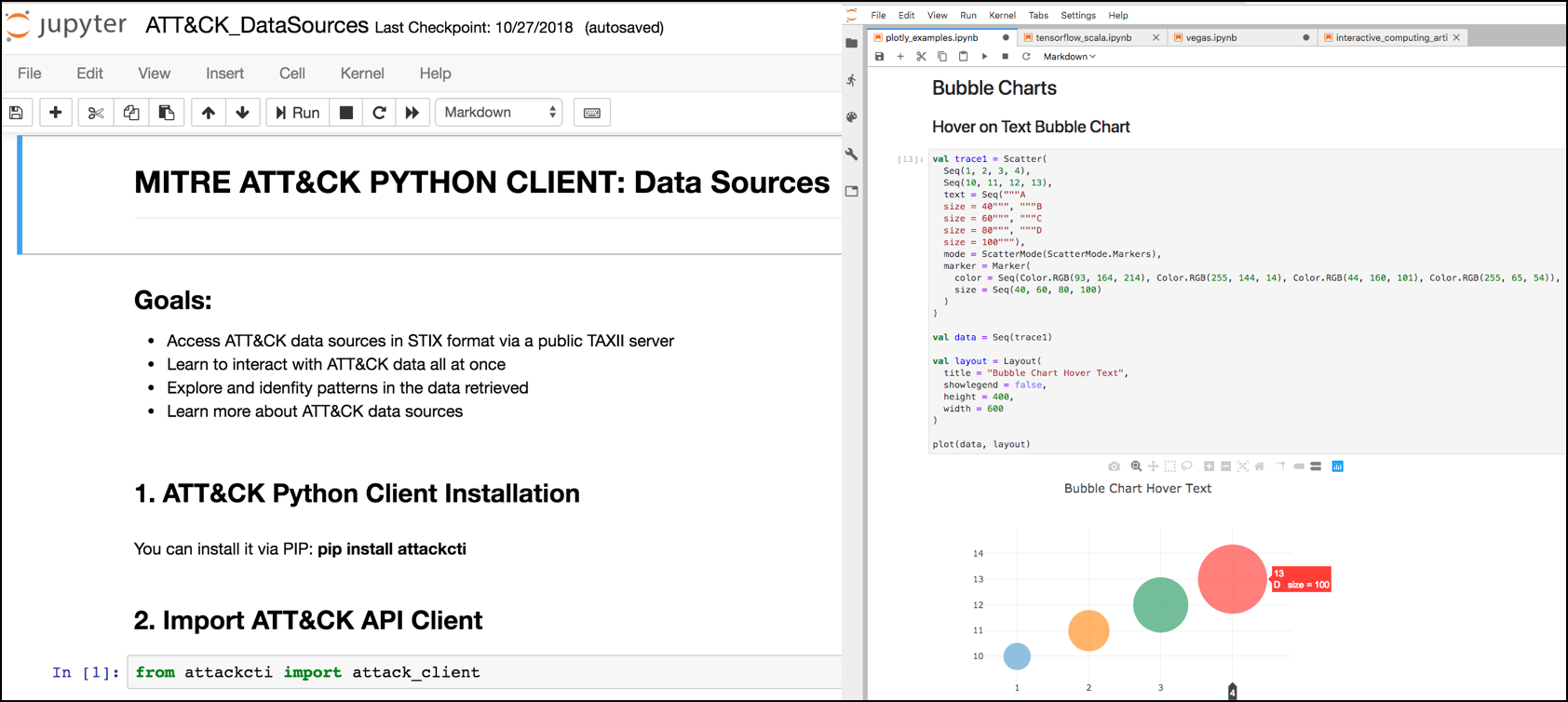
Jupyter Notebook — это веб-приложение с открытым исходным кодом, с помощью которого вы сможете создавать и обмениваться документами, содержащими живой код, уравнения, визуализации и текст повествования. Сюда входит:

* очистка и преобразование данных,
* численное моделирование,
* статистическое моделирование,
* визуализация данных,
* машинное обучение и многое другое.

Проект Jupyter Notebook представляет собой эволюцию библиотеки [**IPython Notebook**](https://ipython.readthedocs.io/en/stable/overview.html), которая, в первую очередь, была разработана для улучшения **стандартной интерактивной консоли python**. Эта консоль позволяет выполнять научные операции и предоставляет расширенные возможности анализа данных с помощью совместно используемых веб-документов.

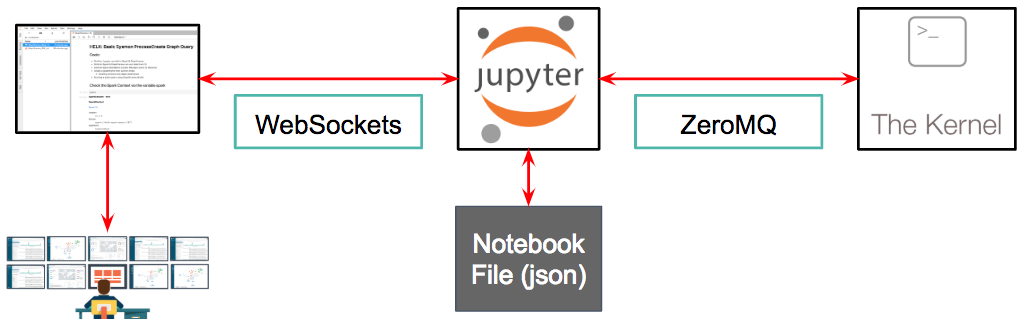


В настоящее время проект Jupyter Notebook поддерживает не только Python, но и более 40 языков программирования, таких как R, Julia, Scala и PySpark. На самом деле, его название первоначально произошло от трёх языков программирования: Julia, Python и R, что сделало его одним из первых приложений для ноутбуков, не зависящих от языка, и теперь считается одной из наиболее предпочтительных сред для исследователей и инженеров в сообществе для изучения и анализа данных..



**Как работает Jupyter** **Notebook?**

Jupyter Notebook работает с так называемой **двухпроцессной моделью**, основанной на инфраструктуре **ядро-клиент**. Эта модель применяет аналогичную концепцию к среде программирования **Read-Evaluate-Print Loop (REPL)**, которая принимает входные данные одного пользователя, оценивает их и возвращает результат пользователю.



Основываясь на концепции **двухпроцессной модели**, объяснить основные компоненты Jupyter можно следующим образом:

**Jupyter Client**

* Позволяет пользователю отправлять код в ядро ​​в виде **Qt Console** или браузера через **документы блокнота**.
* С точки зрения REPL, клиент выполняет операции read и print.
* Блокноты размещаются на веб-сервере Jupyter, который использует **Tornado** для обслуживания HTTP-запросов.

**Jupyter Kernel**

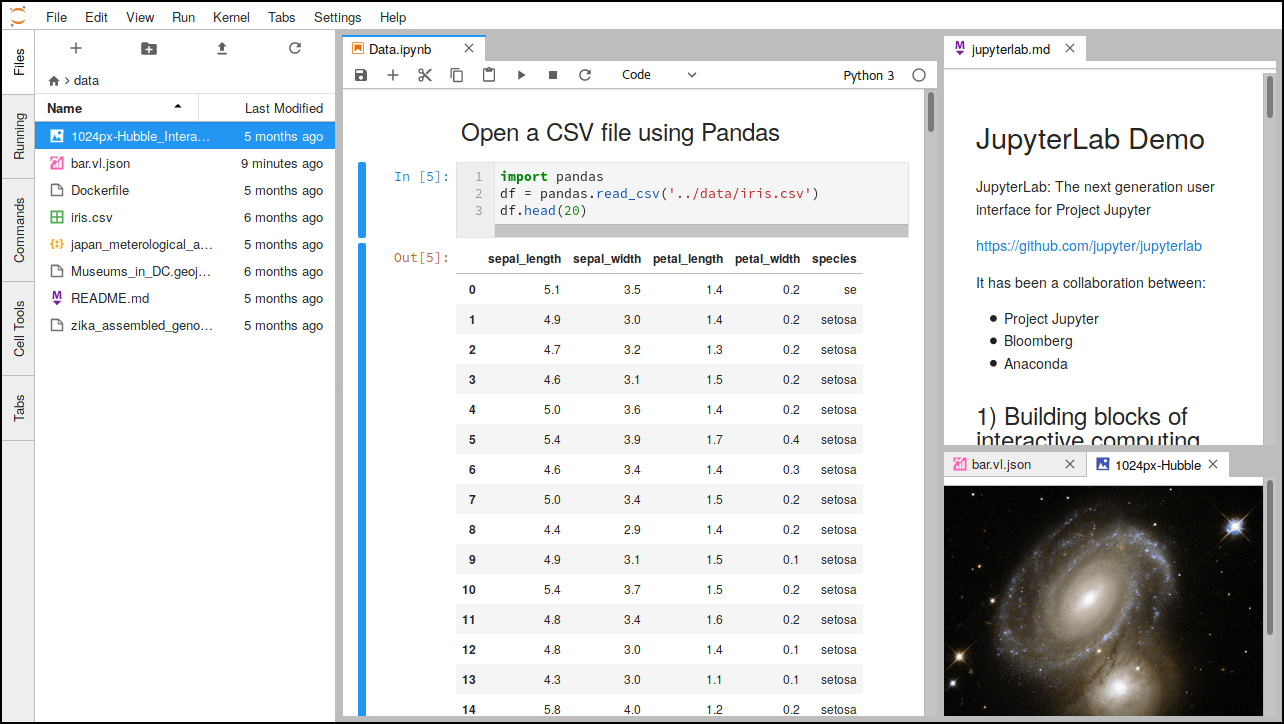
* Получает код, отправленный клиентом, выполняет его и возвращает результаты обратно клиенту для отображения. Процесс ядра может иметь несколько клиентов, взаимодействующих с ним, поэтому эту модель также называют **разделенной двухпроцессной моделью**.
* С точки зрения REPL ядро ​​выполняет операцию evaluate.
* Ядро и клиенты обмениваются данными по протоколу интерактивных вычислений на основе библиотеки асинхронных сообщений **ZeroMQ** (низкоуровневый транспортный уровень) и WebSockets (на основе TCP).

**Документ Jupyter Notebook**

* Блокноты автоматически сохраняются и хранятся на диске в формате JavaScript Object Notation (JSON) с открытым исходным кодом и расширением .ipynb.

**Jupyter Lab**

* Это пользовательский веб-интерфейс следующего поколения для Project Jupyter.
* JupyterLab в конечном итоге заменит классический Jupyter Notebook.
* Мы будем использовать расширение Jupyter Lab на протяжении всех статей.



**Установка Jupyter**

Я уверен, что вам не терпится установить Jupyter и начать изучать его возможности, но сначала вы должны решить, хотите ли вы установить сервер Jupyter Notebook непосредственно в вашей системе, разместить его на виртуальной машине или в док-контейнере.

Я считаю, что важно предоставить вам варианты, чтобы вы почувствовали себя комфортно при работе с инструментами, какими бы вам ни хотелось. Если вы хотите выполнить классическую установку непосредственно в вашей системе, следуйте официальным **документам Jupyter Install**.

**Запуск Jupyter через Docker**

В этой статье мы будем использовать проект HELK. Мне нравится делиться стандартизированной и рабочей средой через образы докеров, которые помогают сосредоточиться на возможностях приложения, а не тратить время на устранение неполадок при установке сервера.

**Требования**

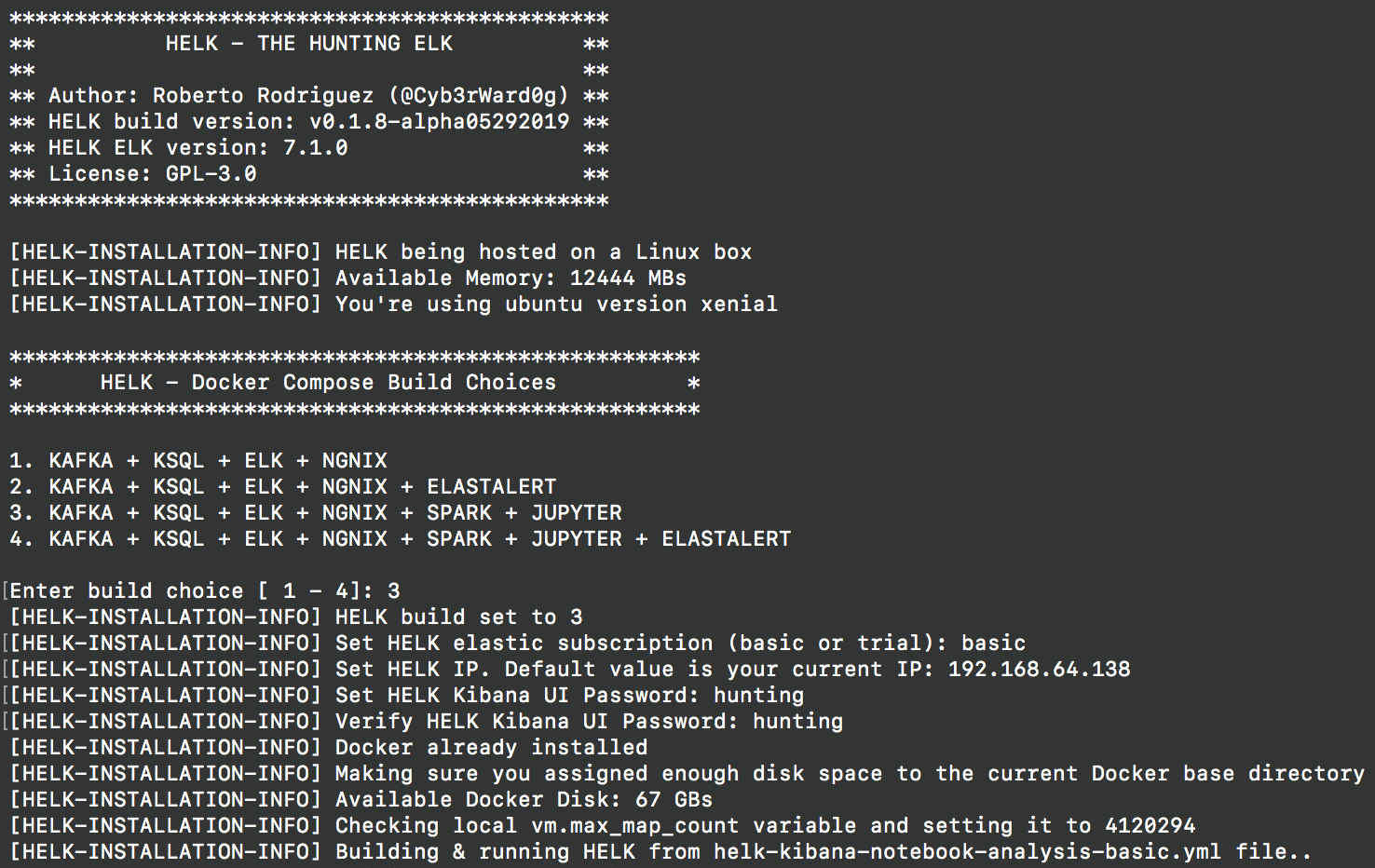
* Проект HELK

**Начальные шаги**

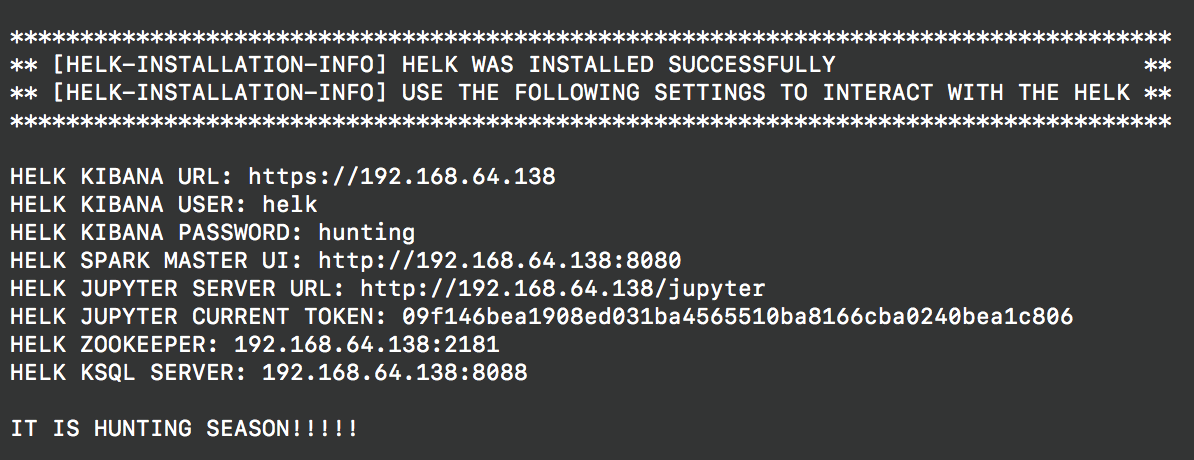
* Скопируйте последний репозиторий HELK, измените текущий каталог на HELK / docker и запустите сценарий helk\_install.sh с именем sudo.

git clone [https://github.com/Cyb3rWard0g/HELK.git](https://github.com/Cyb3rWard0g/demos.git)  
cd HELK/docker  
sudo ./helk\_install.sh

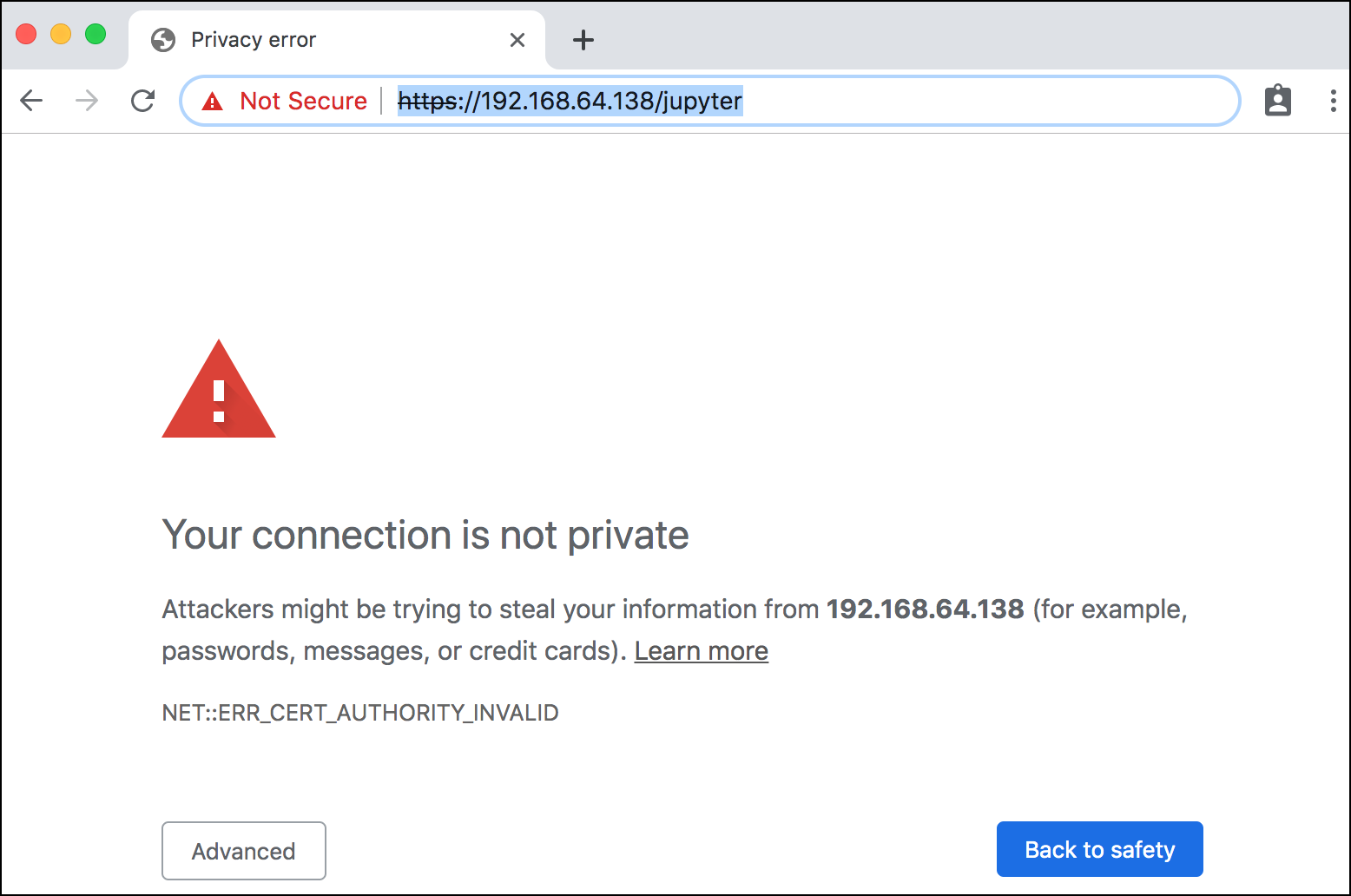
* Для этой статьи мы будем использовать вариант 3 с лицензией basic.

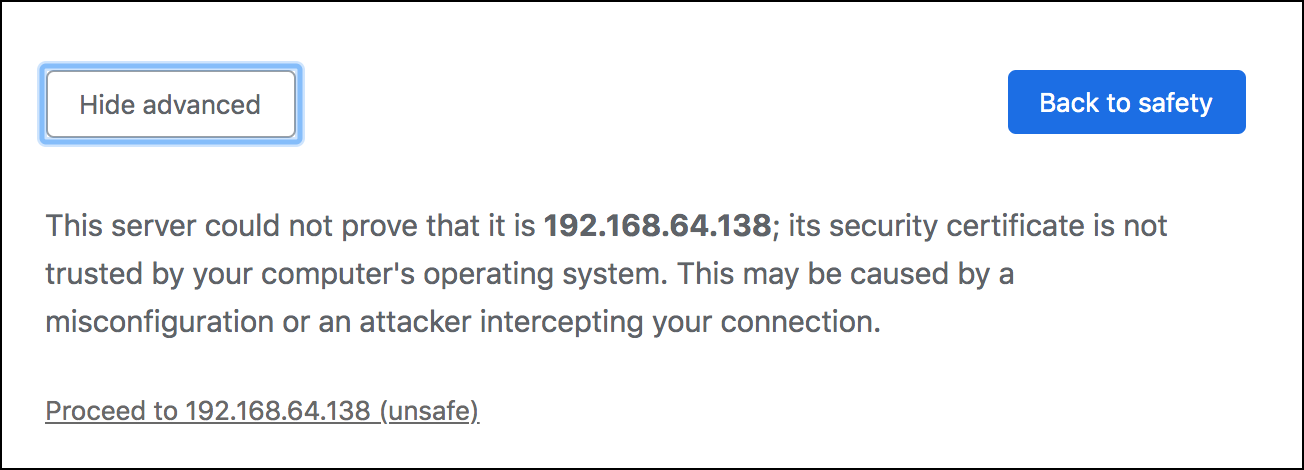


* Как только сценарий завершится (около 5–10 минут), вы увидите похожий вывод:

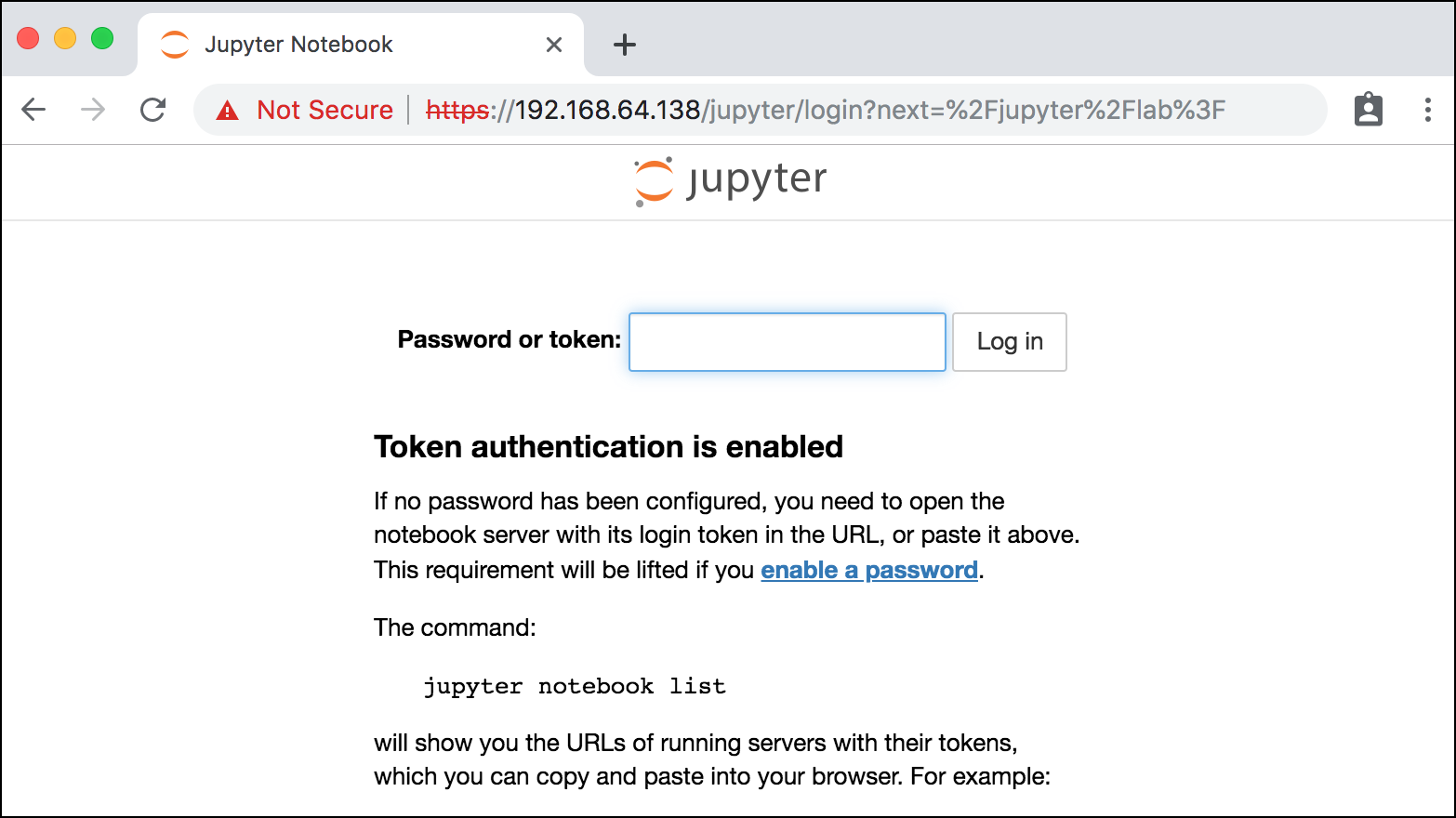


* Перейдите по ссылке HELK JUPYTER SERVER URL.
* Возможно получить Privacy Error. Просто проигнорируйте её, нажмите Advanced, а затем Proceed.

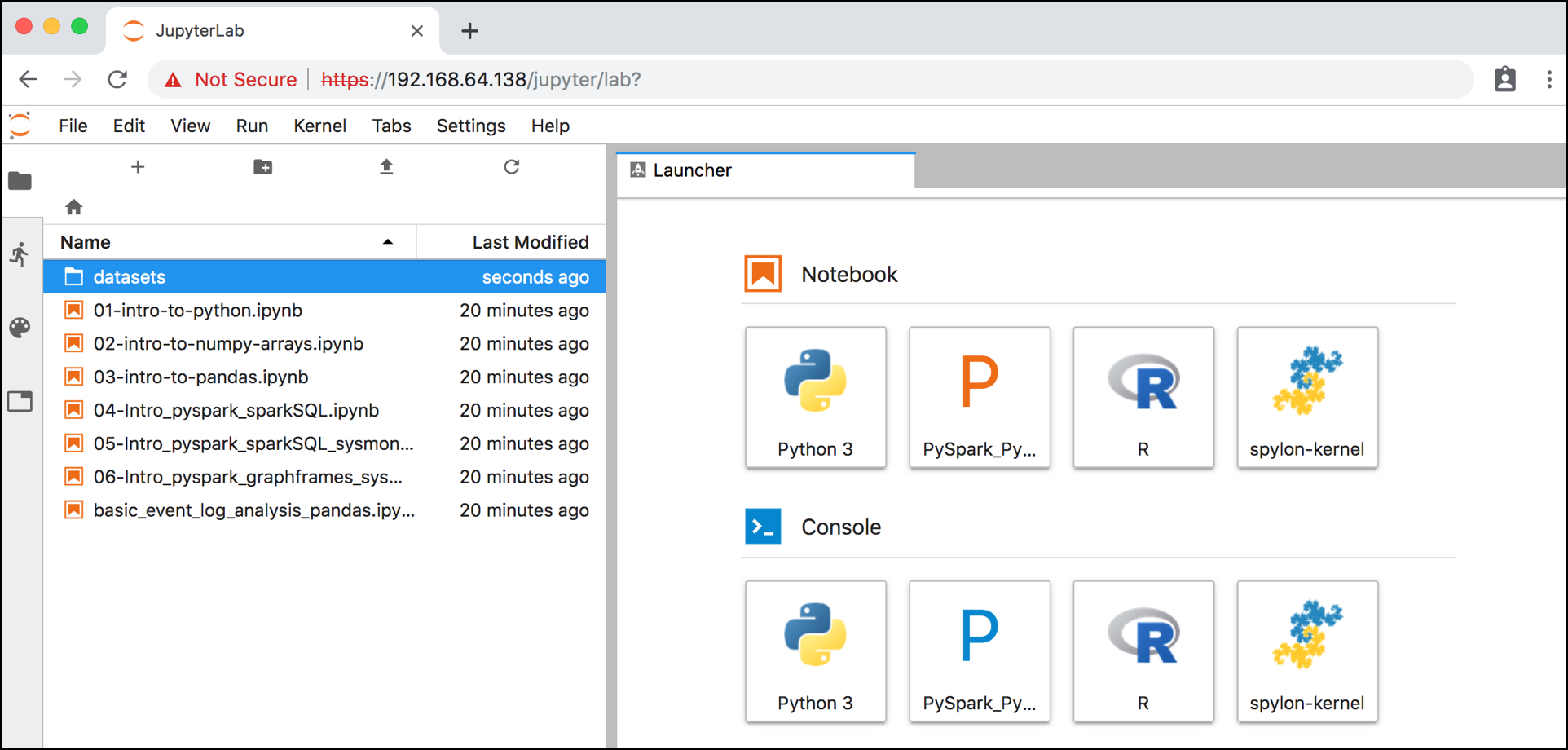




COPY JUPYTER CURRENT TOKEN, который отображается на вашей консоли, и PASTE TOKEN в поле Password or token и нажмите Log in.



* Вы получите доступ к главному меню Jupyter Lab.

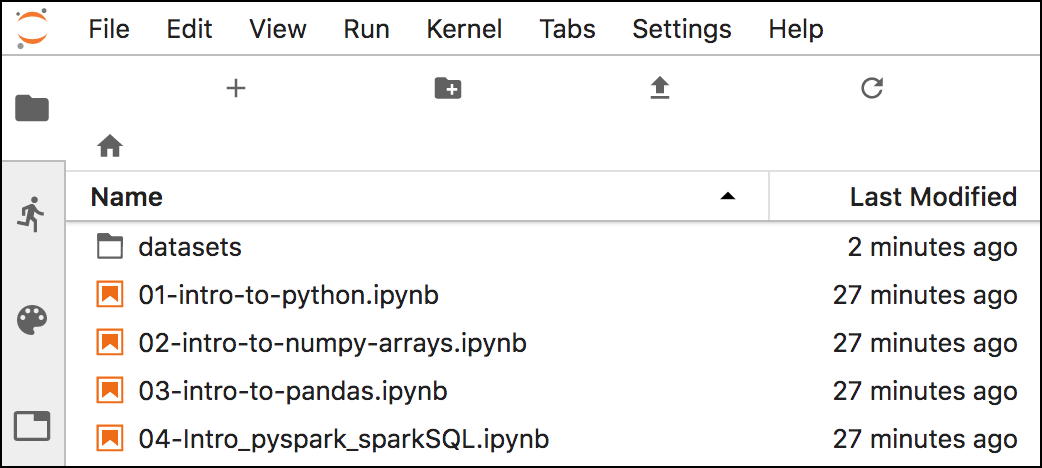


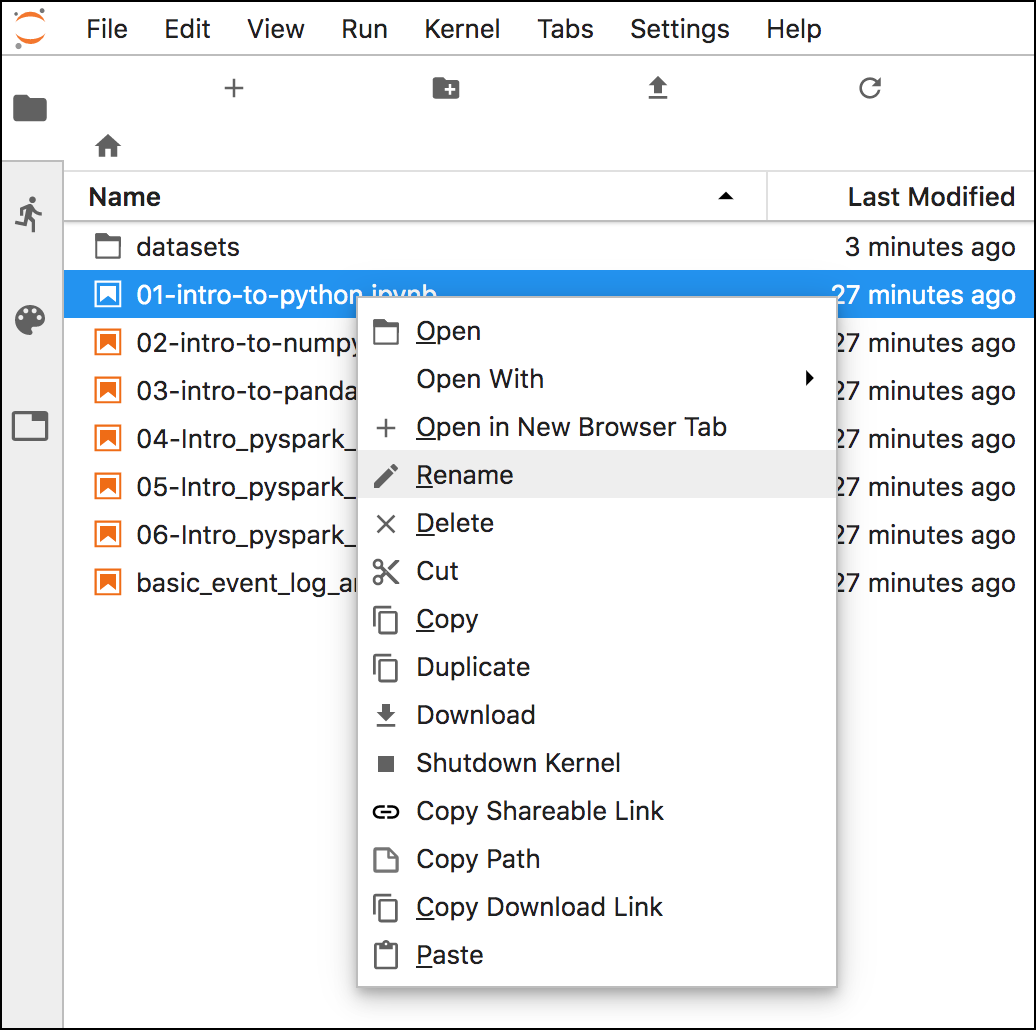
* Как вы видели (изображение выше), есть папка с именем datasets. Она содержит набор данных Mordor по умолчанию, который мы будем использовать в следующих статьях. Также есть несколько блокнотов для вас. Я сделал все это для того, чтобы вы ознакомились с концепциями, которыми я буду делиться.

**Изучение основного интерфейса Jupyter**

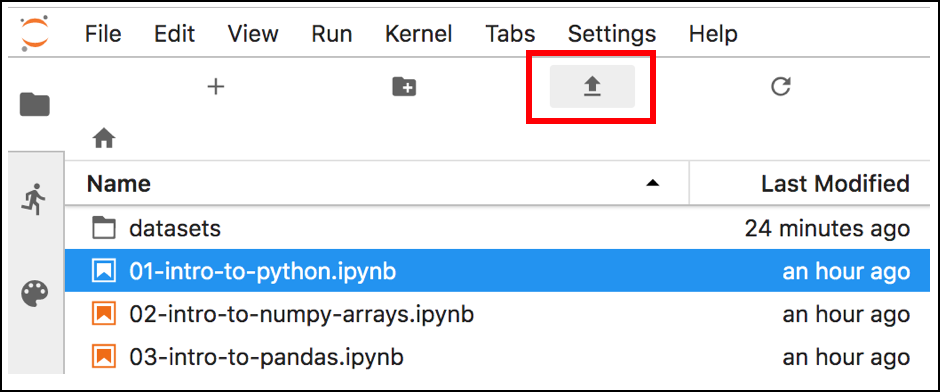
**Раздел браузера файлов**

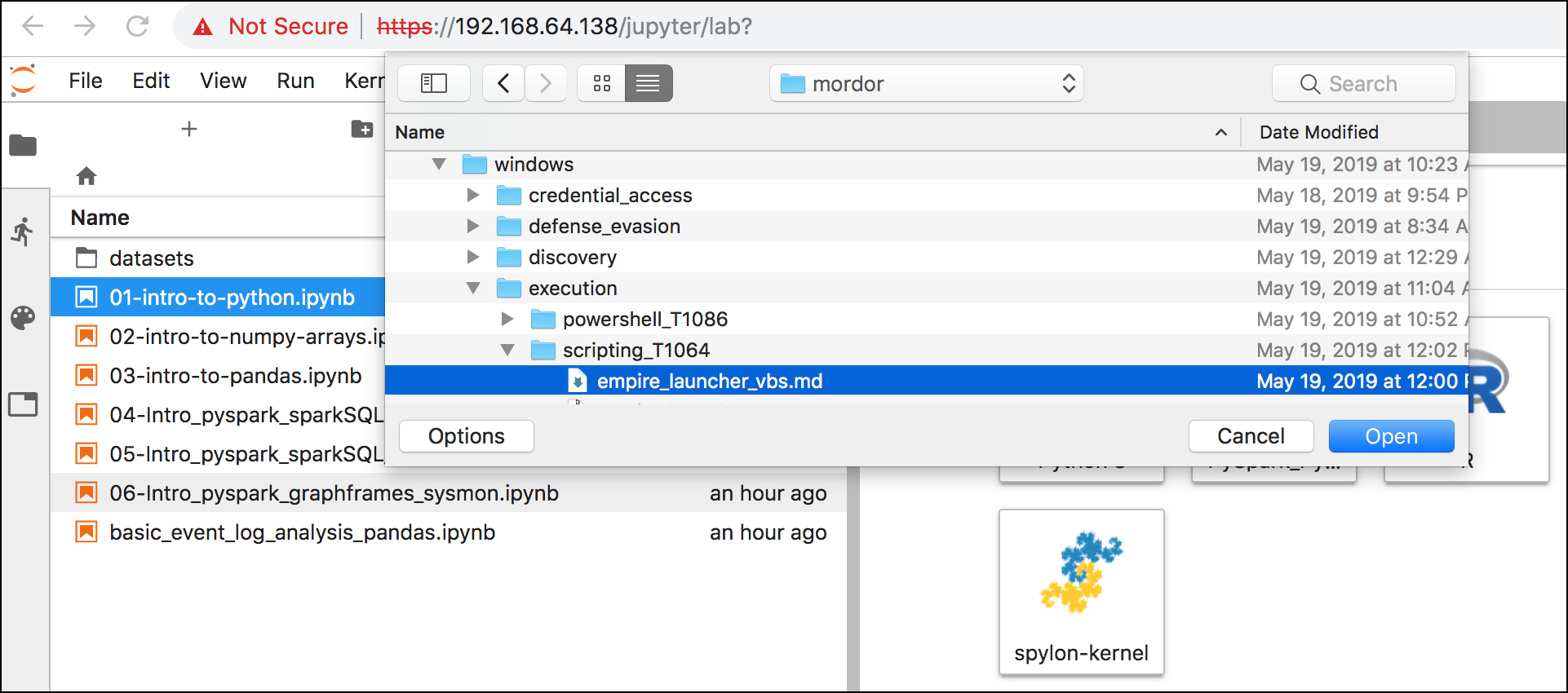
* В этом разделе показаны доступные объекты, такие как папки или файлы, которые вы можете использовать. Вы можете rename, move, download, или delete любые папки или файлы, щелкая правой кнопкой мыши по объектам.



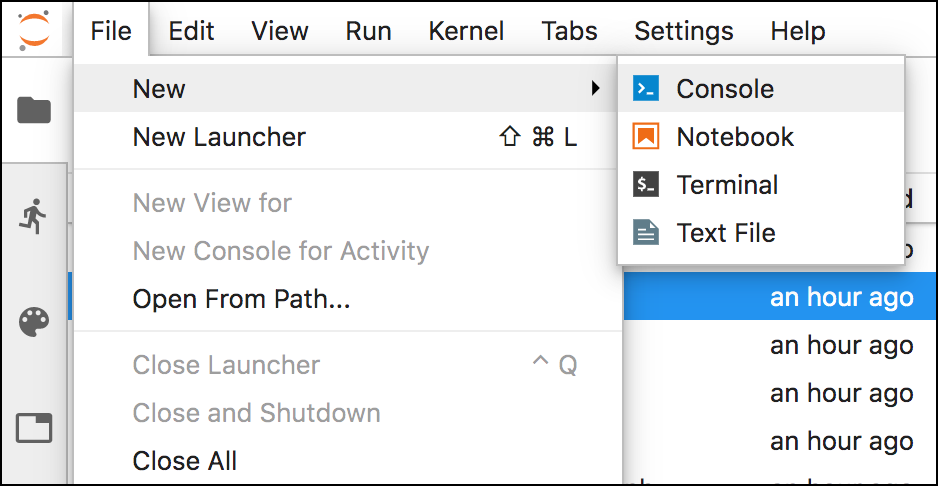


* Также вы можете загрузить файл, который существует локально в вашей системе, нажав на значок Upload, как показано ниже.

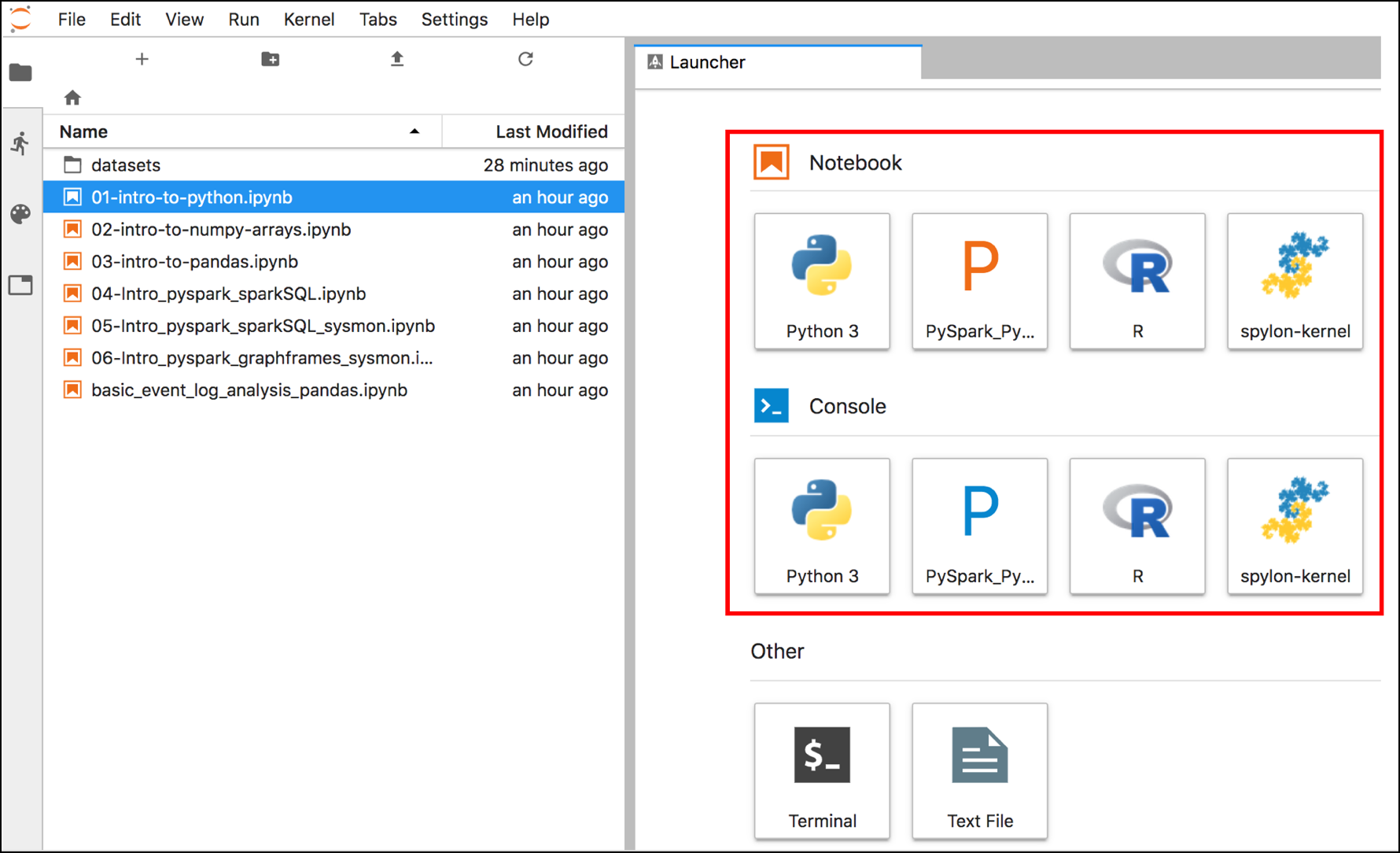




* Кроме того, вы сможете создавать новые объекты, такие как Notebook, текстовые файлы и даже запускать терминал bash.



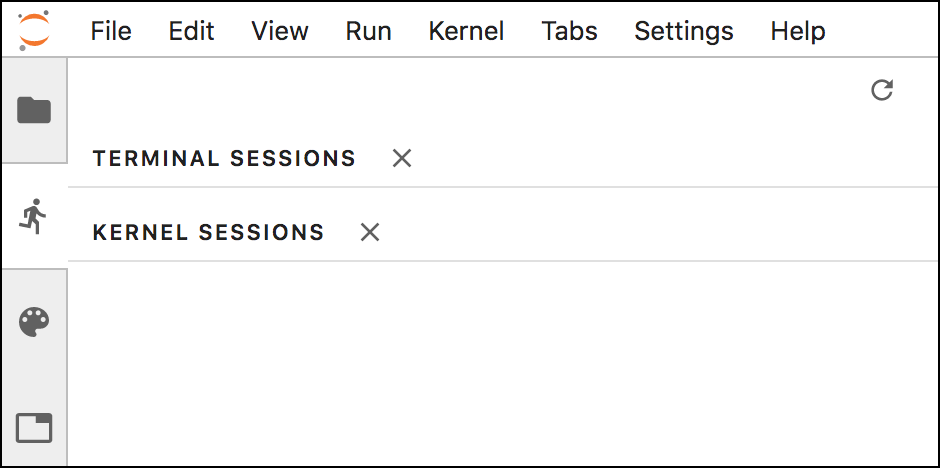
* Сделать это можно также из панели управления, как показано ниже.



* Как вы можете видеть на изображении выше, наш сервер Jupyter имеет четыре доступных ядра: **Python** 3, **PySpark**, **R** и **Syplon**.
* По умолчанию Jupyter поставляется с ядром Python 3 (IPython). Команда Jupyter поддерживает ядро ​​IPython, поскольку сервер Jupyter Notebook зависит от функциональности ядра IPython. Многие другие языки, помимо Python, можно использовать в блокноте. Это то, что подразумевается под языковым подходом. Если вы хотите узнать больше о других ядрах, посмотрите здесь.

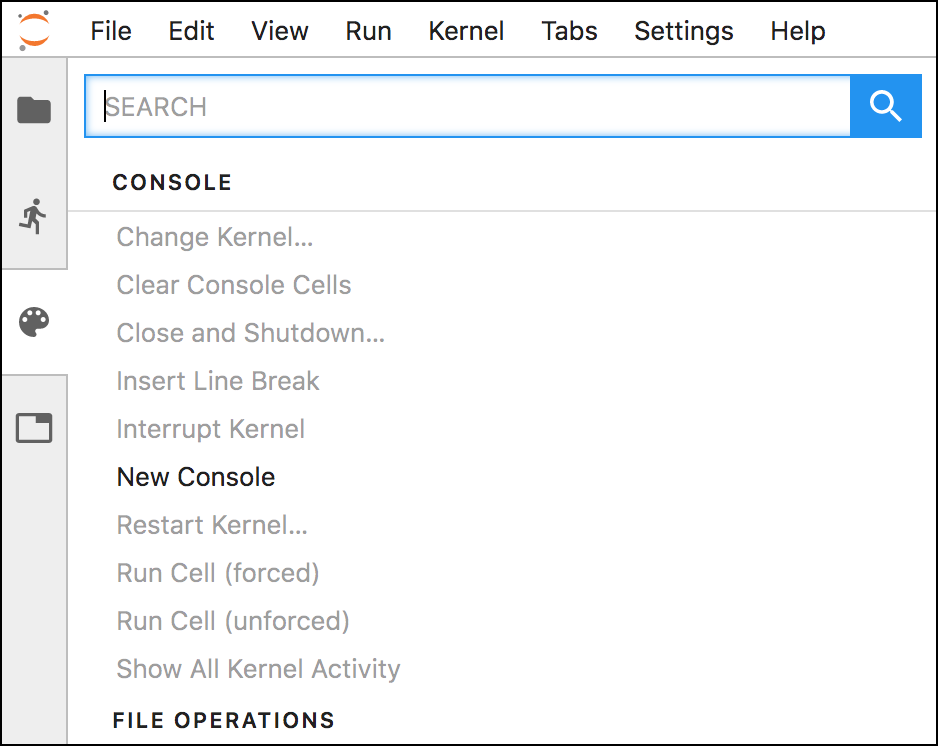
**Секция работы терминалов и ядер**

* В этом разделе представлена ​​информация о текущих запущенных процессах Jupyter. Когда вы создадите блокнот или начнёте сеанс терминала, просмотреть эти процессы можно здесь.



**Раздел «Команды»**

* В этом разделе представлена ​​информация о нескольких командах, доступных для взаимодействия с файлами новых консолей, ядра и др.

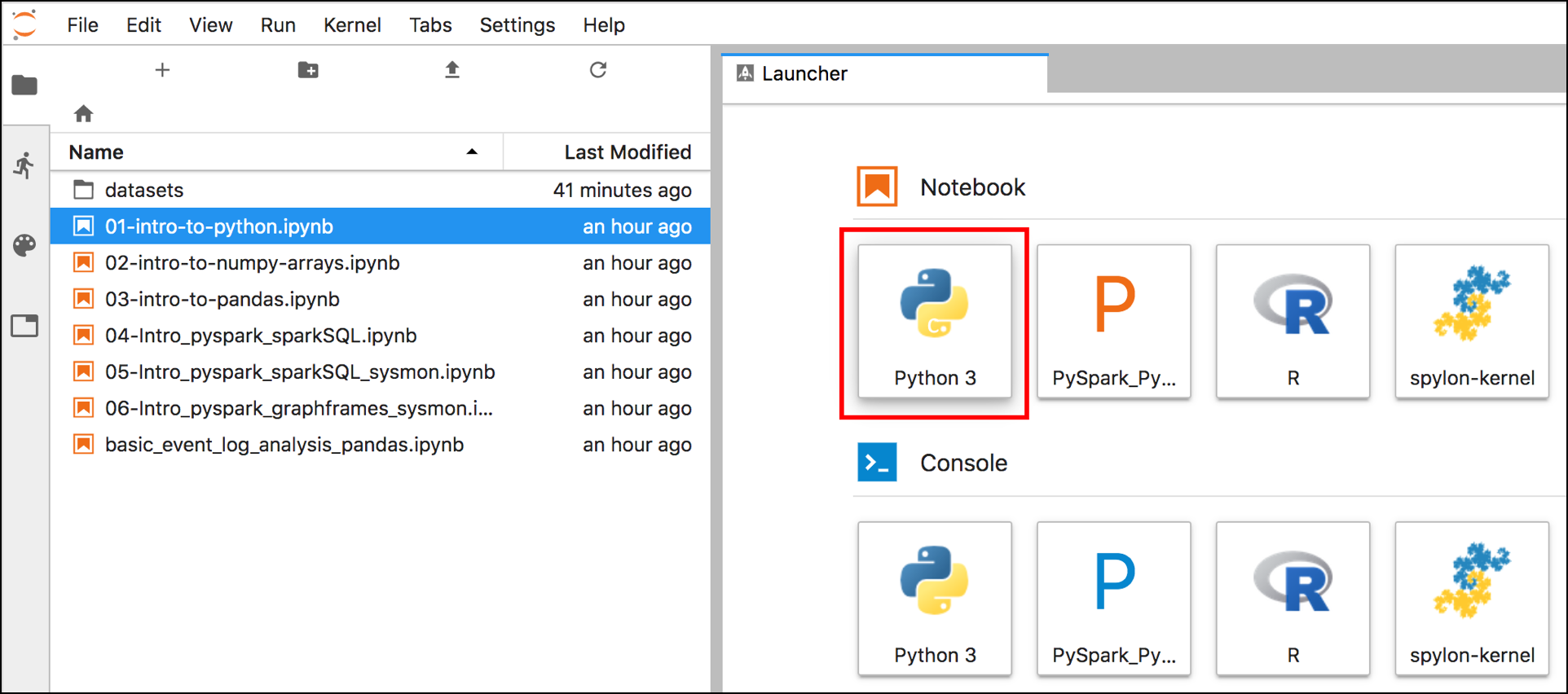




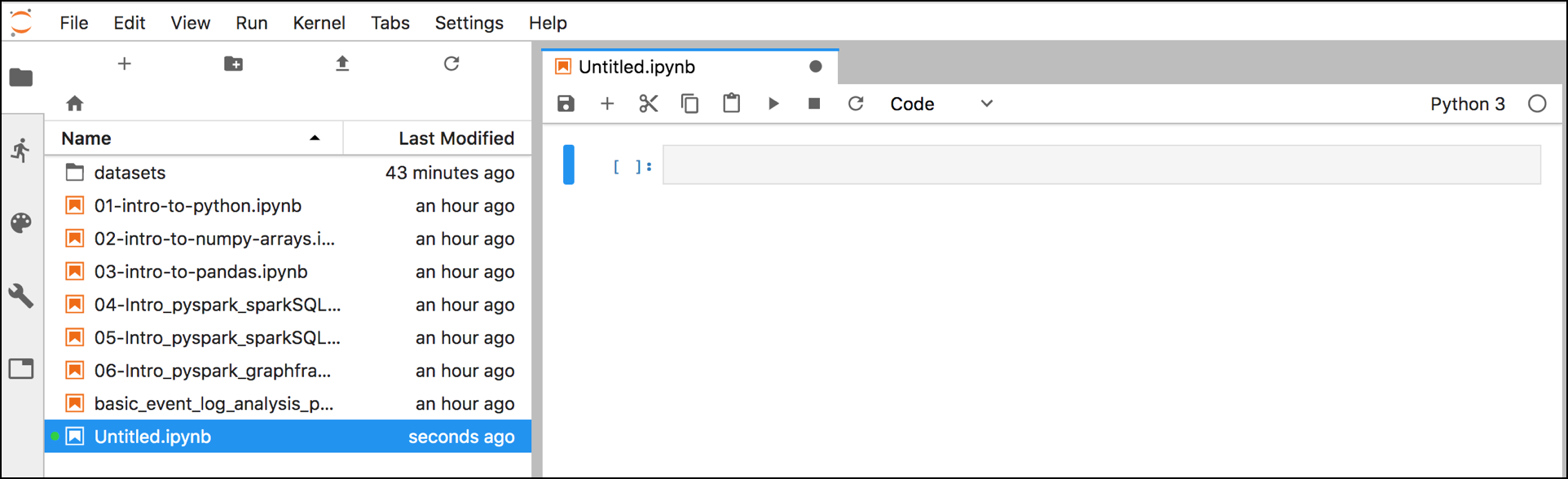
**Ваш первый Notebook!**

Давайте создадим наш первый блокнот и познакомимся с дополнительными опциями.

* Вернитесь в раздел **Launcher** и нажмите на конкретное ядро, которое вы хотите использовать для инициализации нового блокнота. Давайте выберем Python 3.

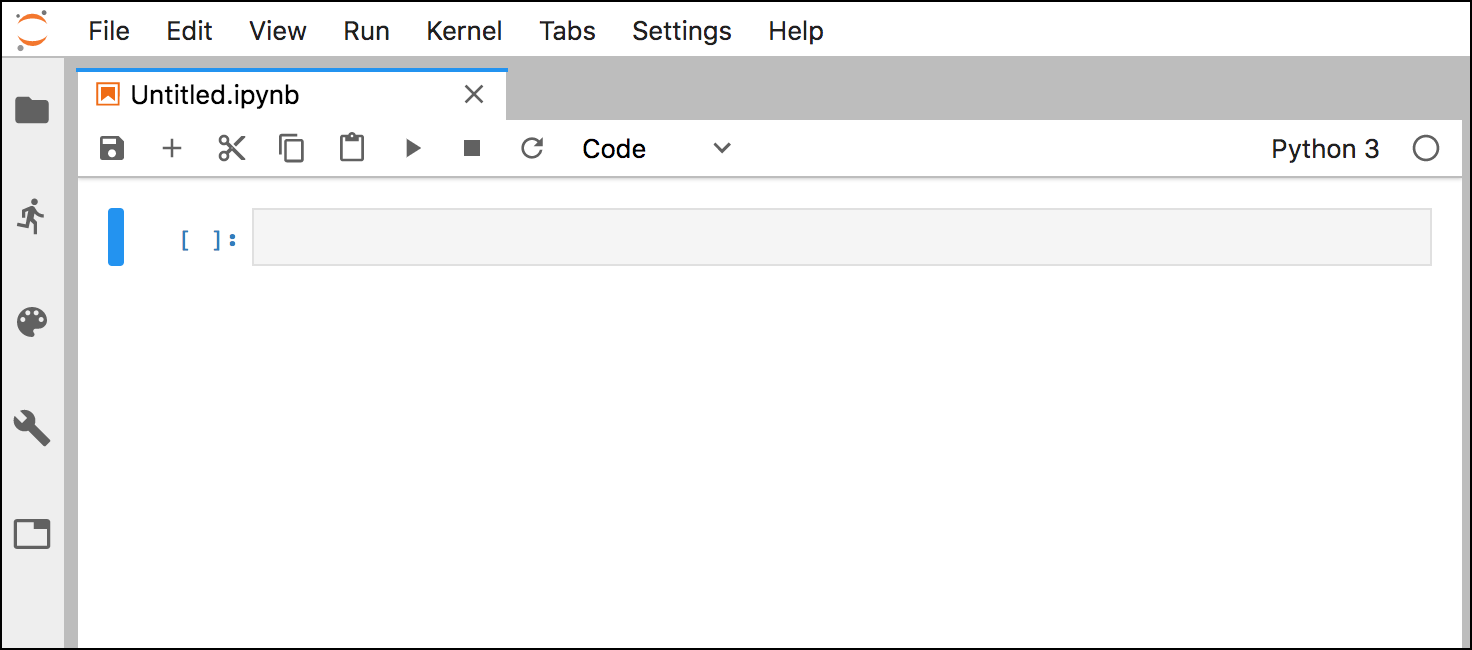


* Вы попадёте в основной веб-интерфейс блокнота с исходным именем блокнота Unititled и одной доступной ячейкой input(In), в которую можно добавить код для запуска ядра Python 3.
* Вы также увидите новый файл c названием Unititled ирасширением ipynb. Блокноты сохраняются автоматически.

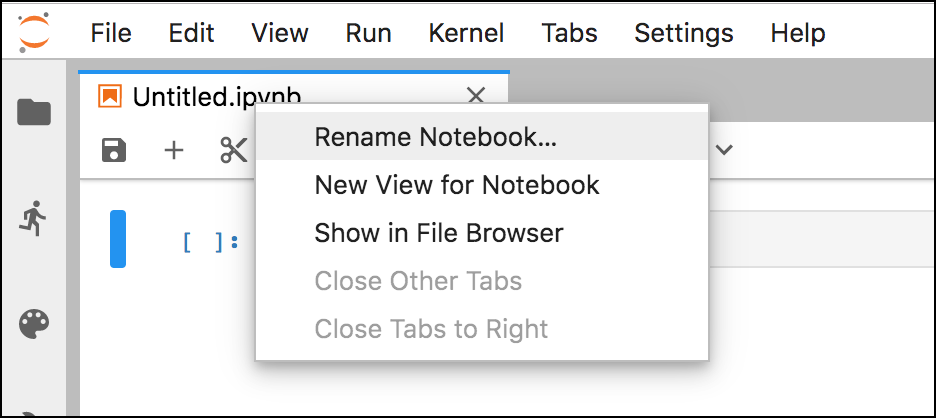


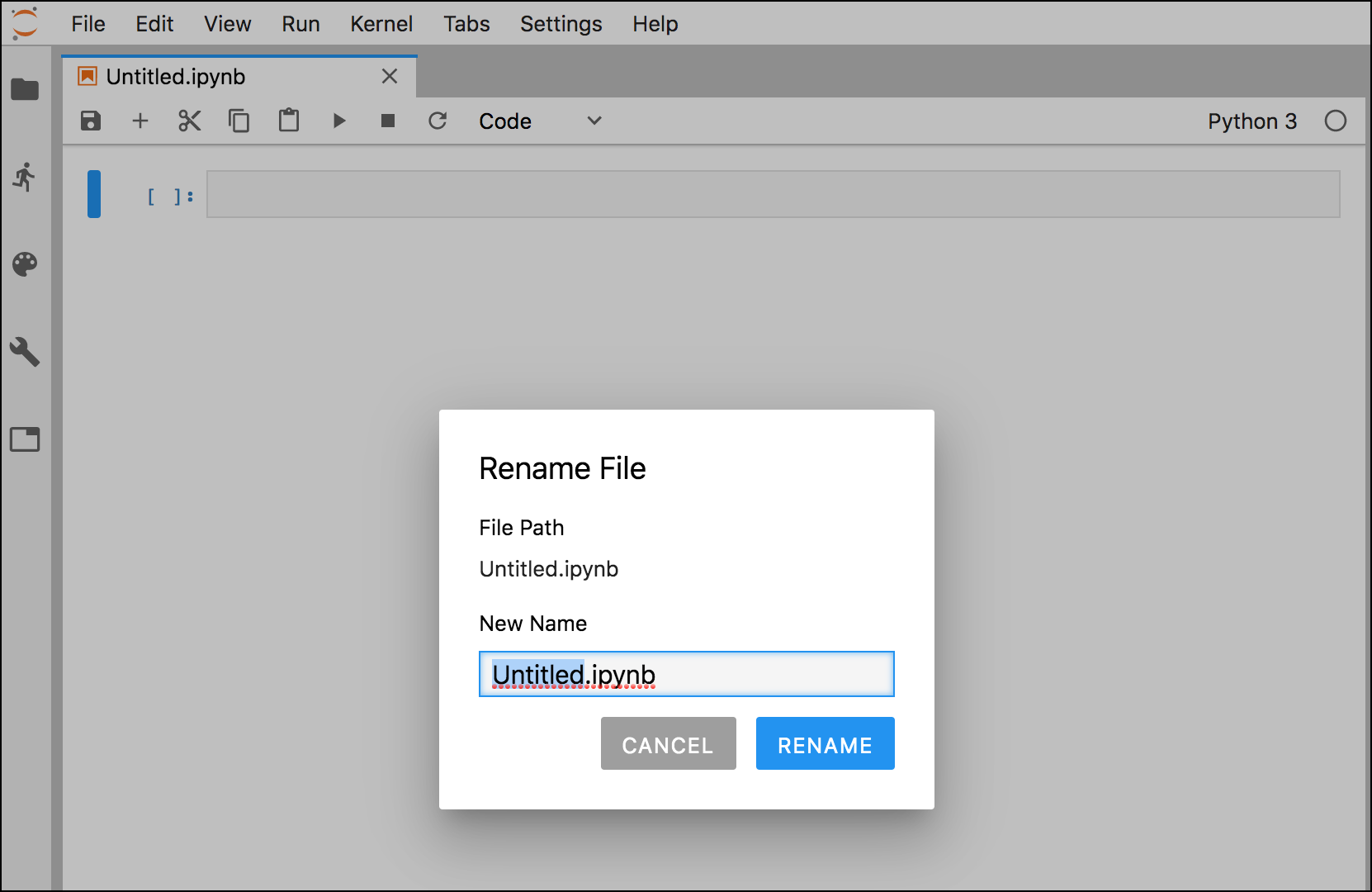
* Раздел **File Browser можно свернуть**, нажав на его значок.

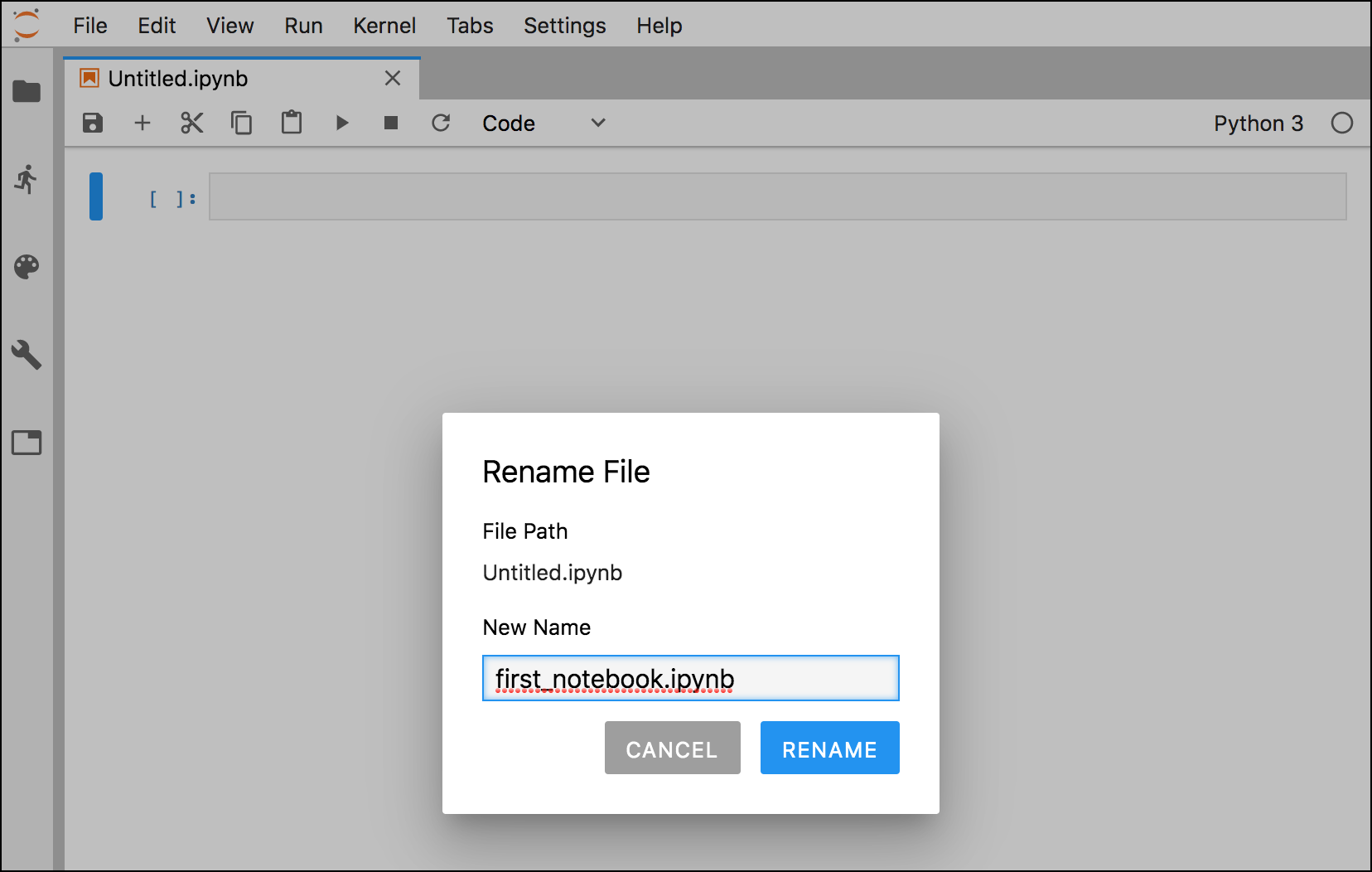




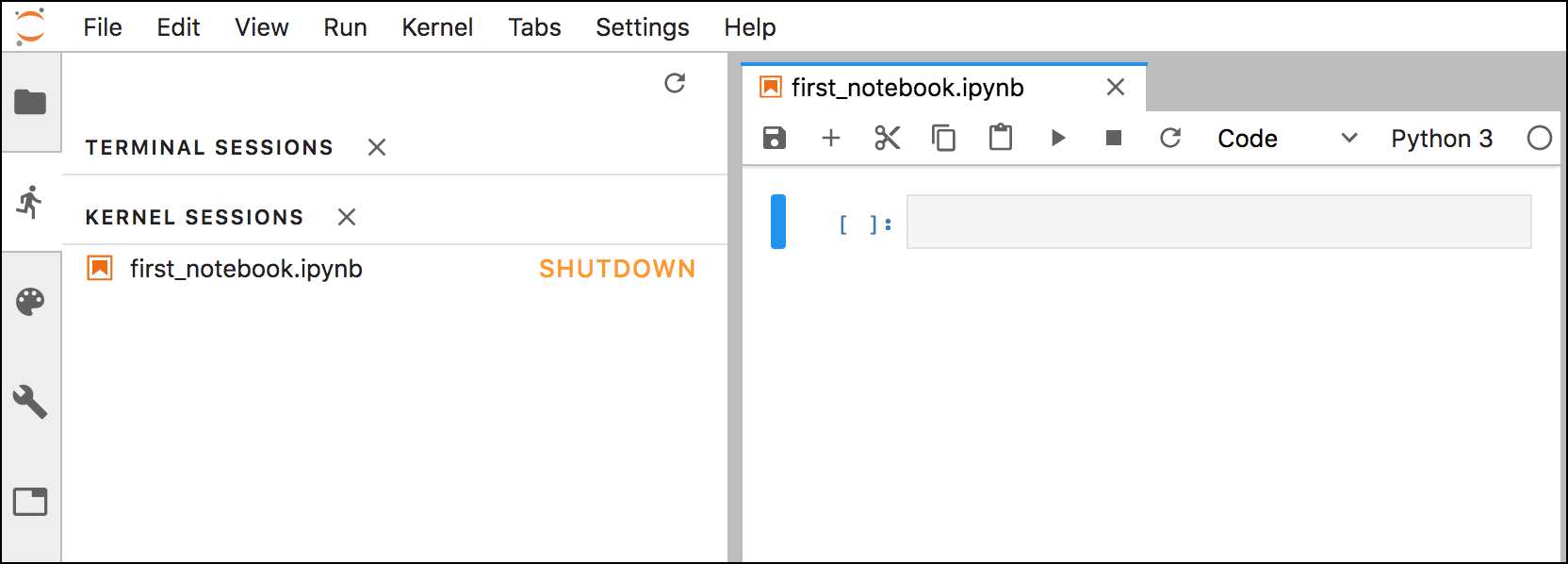
* Свой блокнот можно переименовать, щелкнув правой кнопкой мыши на заголовке блокнота и выбрав Rename Notebook.







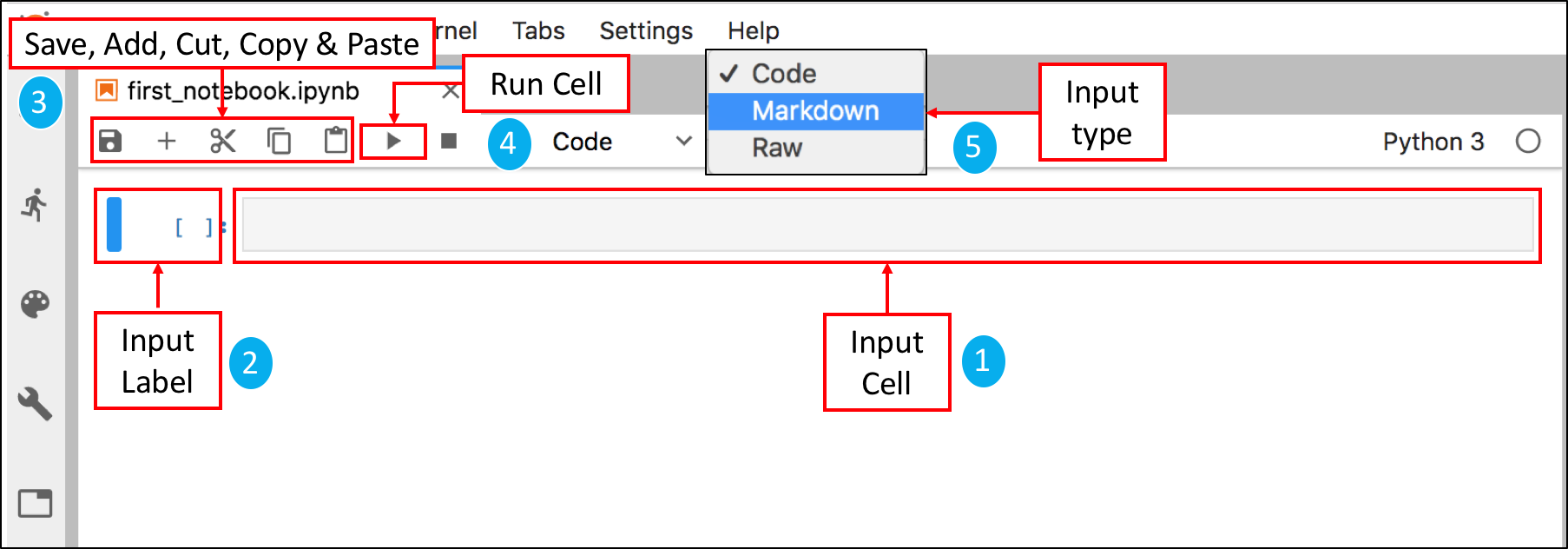
* Если вы проверите сеансы ядра Jupyter, вы увидите, что ваш блокнот работает с возможностью отключения ядра.



**Изучение интерфейса Notebook**

В интерфейсе блокнота доступно несколько опций, и большинство из них очень просты и работают так же, как и на обычной панели инструментов документа, где вы можете сохранять, открывать или закрывать файлы. Однако есть несколько вариантов и концепций, которые очень важно понять:

**Ячейки интерфейса**



1️⃣ Одна из основных частей интерфейса — это input cell container, в котором можно ввести код (например, Python) или текст (например, Markdown), а ядро ​​ оценивает и выполняет запрос.

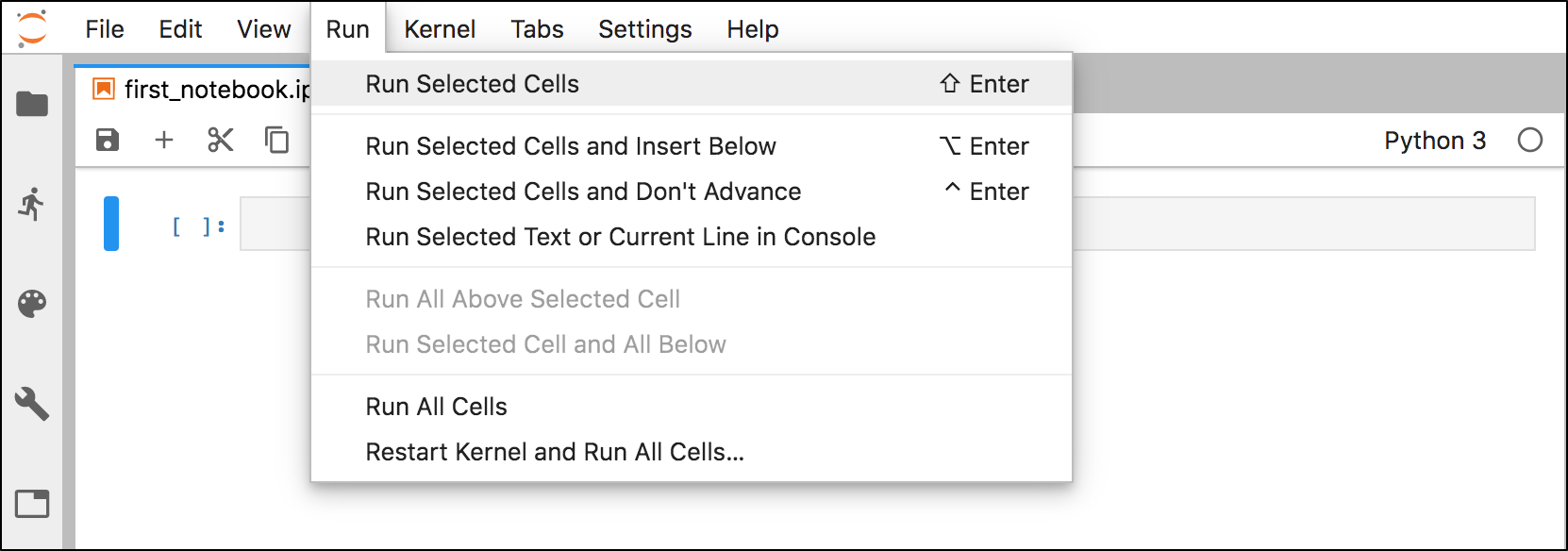
2️⃣ Слева от ячейки input cell находится input label, которая отслеживает выполнение кода через последовательность чисел, начиная с [1]. Если запрос ещё не запущен, ячейка отображает пустые скобки [ ]. Если в ядре выполняется код, внутри скобок [] появляется звёздочка (\*).

3️⃣ Вы можете сохранить содержимое блокнота, добавить новые ячейки (➕), вырезать / удалить ячейку (✂) ️, скопировать и вставить ячейки.

4️⃣ Вы можете запустить код, выбрав input cell container и нажав на run cell button. А также открыть ячейку с помощью SHIFT + ENTER.

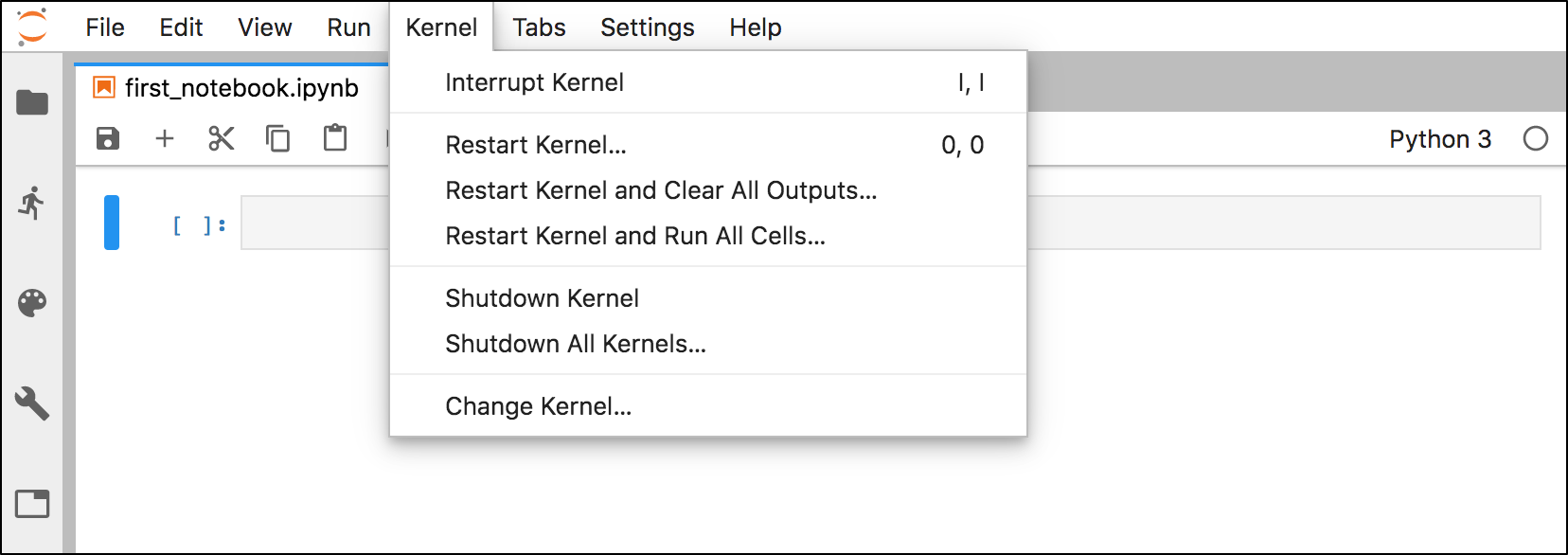
5️⃣ С помощью выпадающей кнопки code вы можете выбрать тип ввода, с которым будете работать. В настоящее время существует три типа ввода ячеек. Тип ввода code разрешает пользователю запускать код на определённом языке программирования, заданном в ядре. В данным случае — это Python 3.

Также получить доступ к дополнительным параметрам можно на вкладке Run, как показано ниже:



**Ядро интерфейса**

* Получить доступ к параметрам ядра можно через вкладку Kernel.

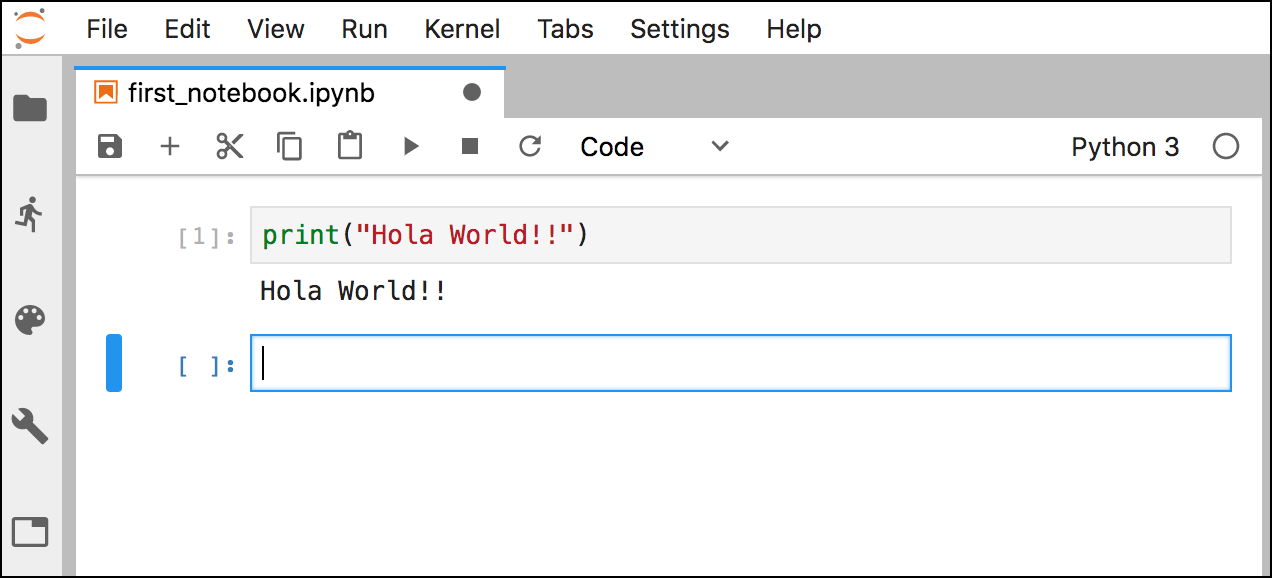


**Запустим код**

Теперь, когда мы понимаем, как взаимодействовать с интерфейсом Jupyter Notebook, давайте запустим какой-нибудь базовый код Python в ячейке input cell container.

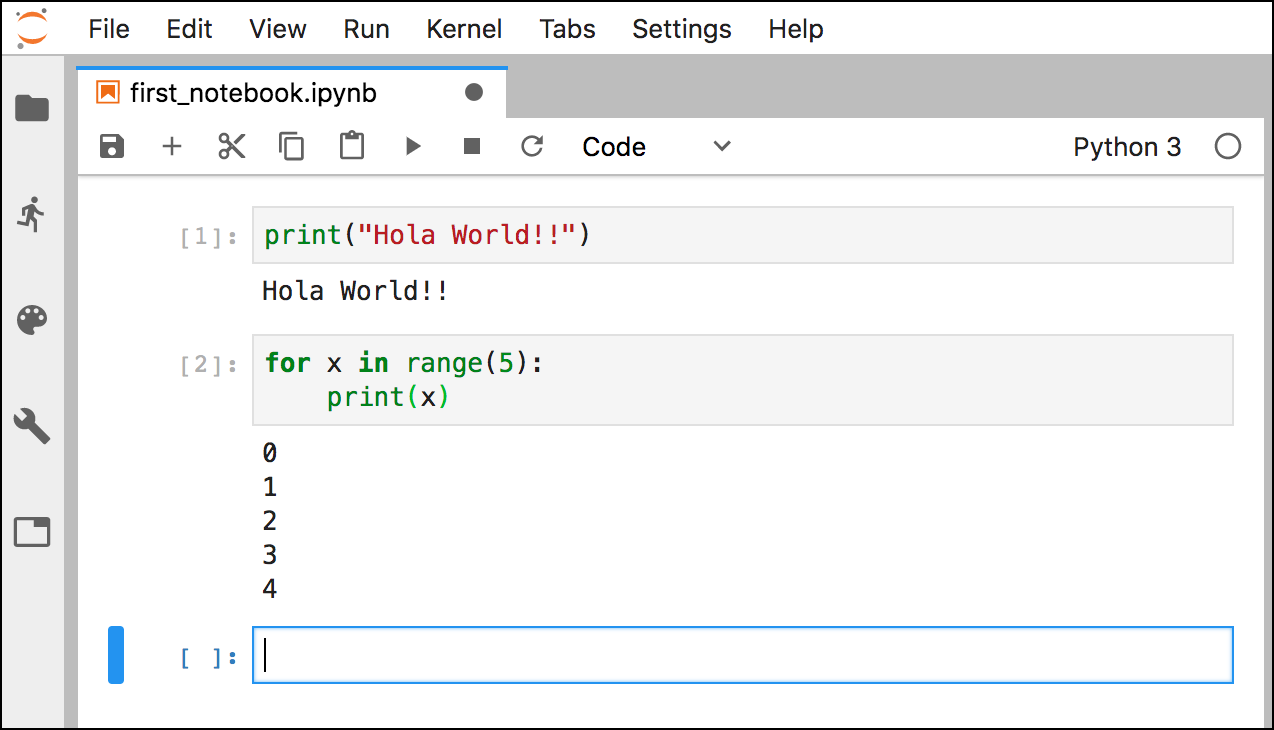
* Введите PRINT - простое утверждение для python - и запустите ячейку (SHIFT + ENTER)

print("Hola World!!")

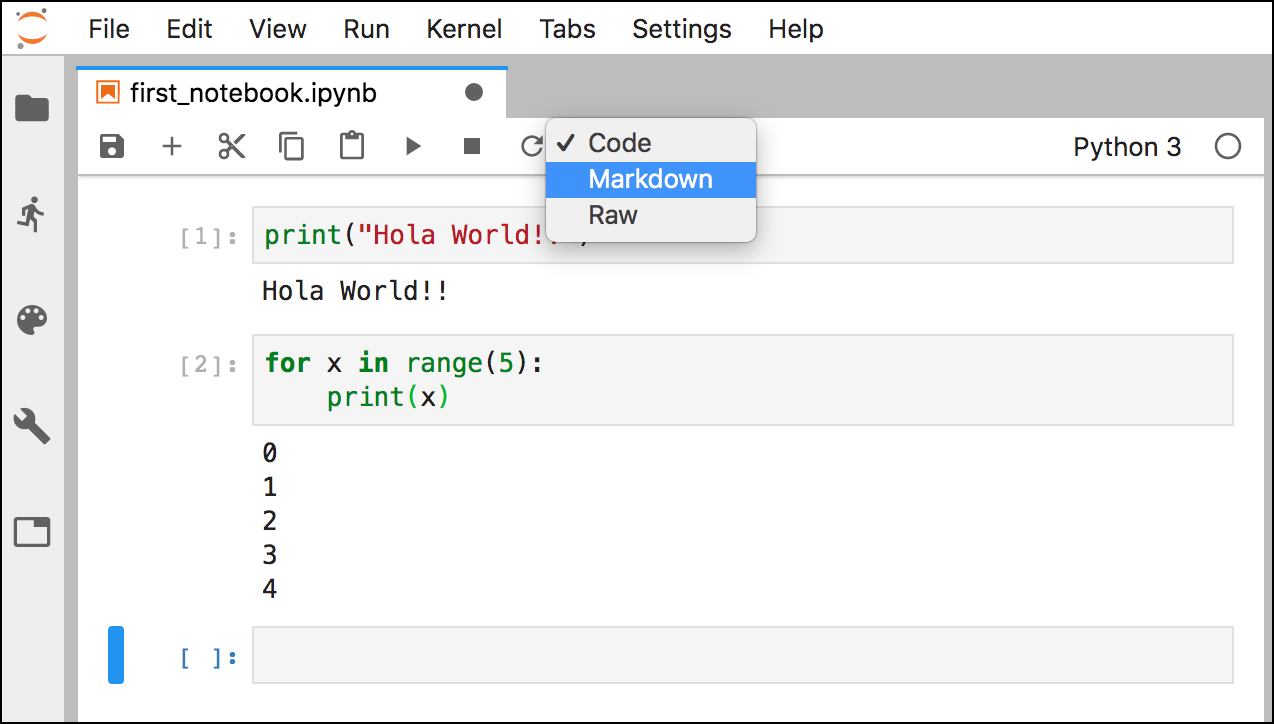


* Попробуйте запустить цикл FOR loop в последовательности из 5 чисел, сгенерированных RANGE.

for x in range(5):  
 print(x)



* Переключите новый cell type на Markdown, чтобы изменить вид разметки.



* COPY, PASTE, и RUN выполняют в ячейке Markdown:

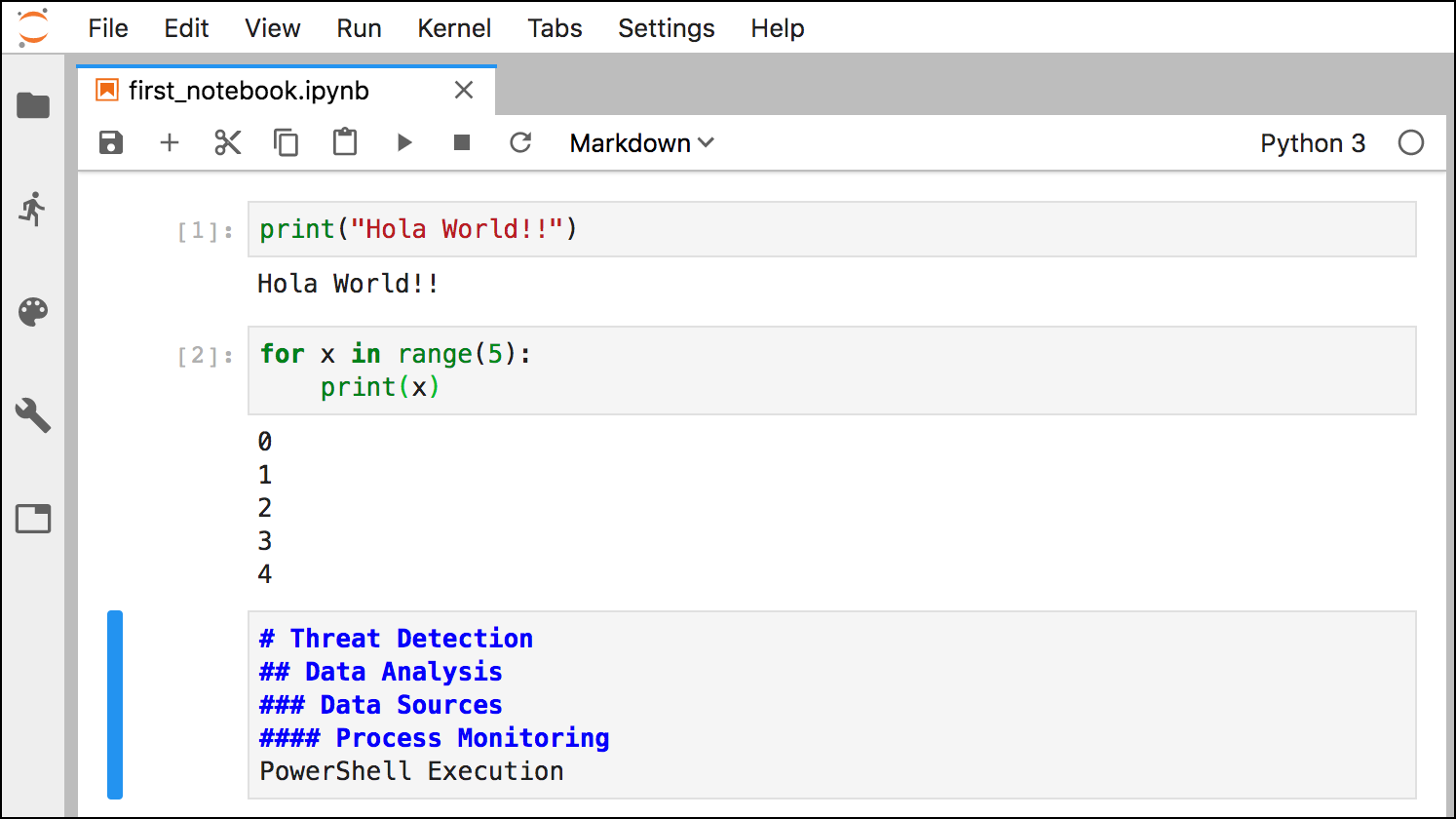
# Threat Detection

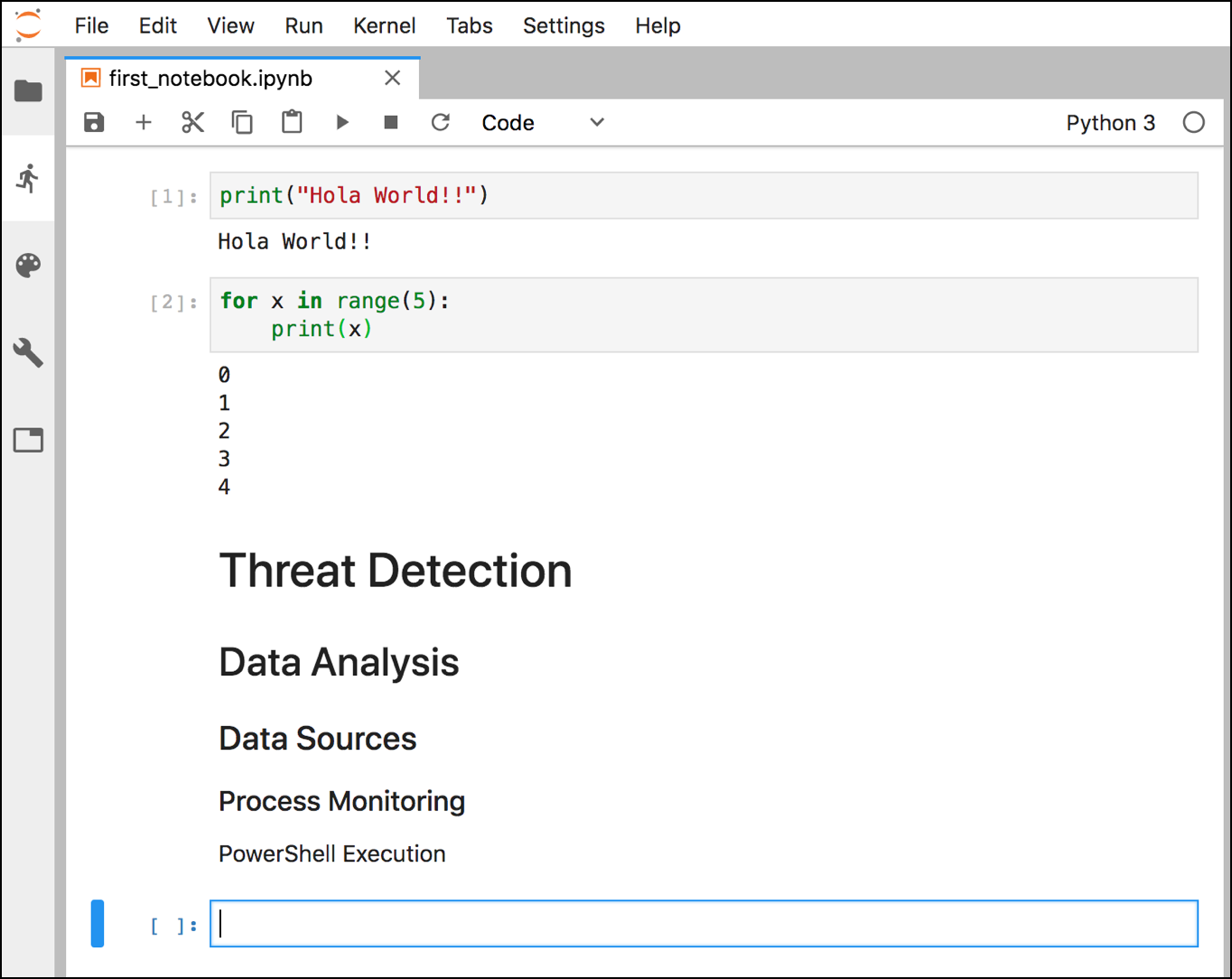
## Data Analysis

### Data Sources

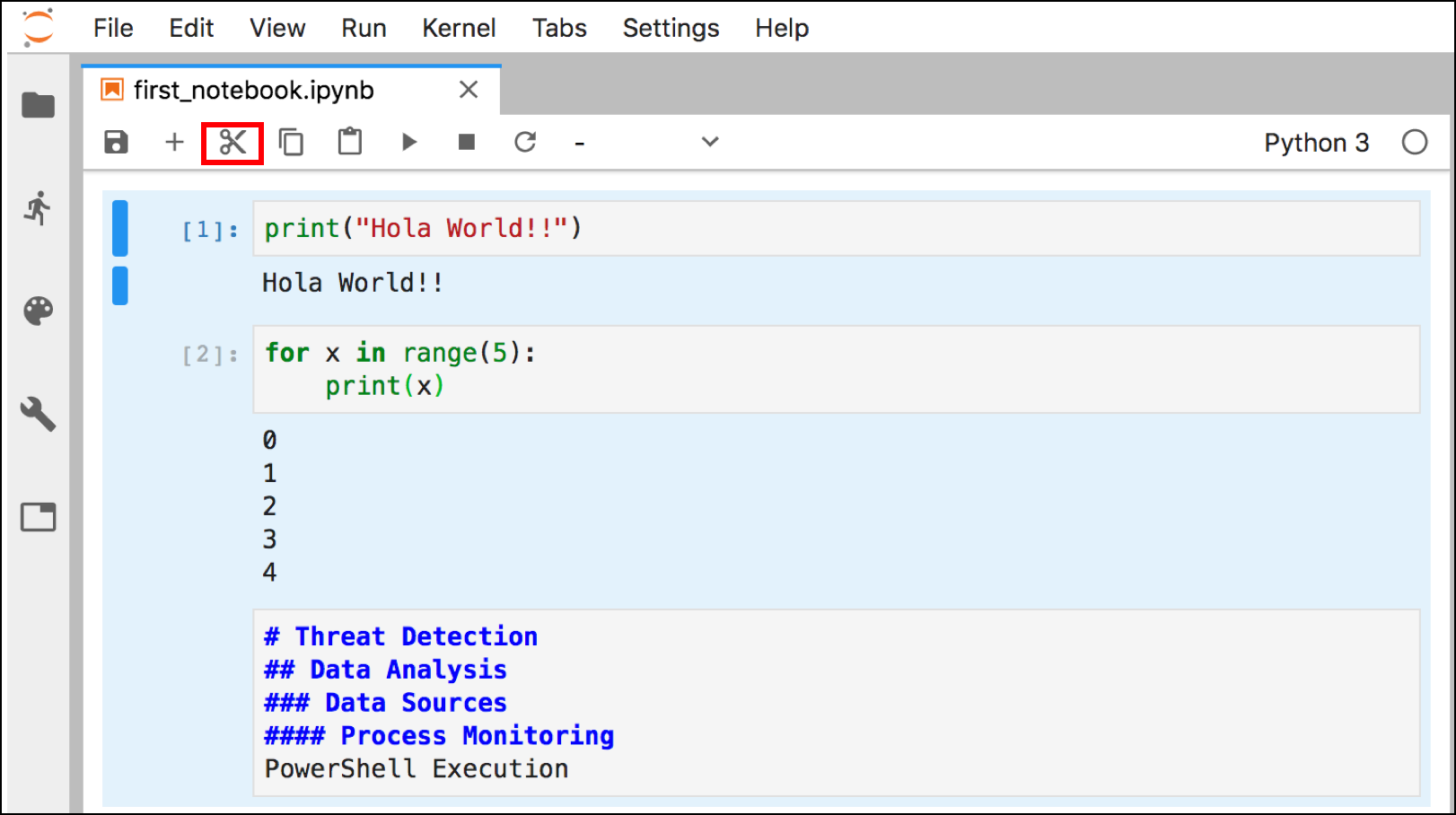
#### Process Monitoring

PowerShell Execution





* Выделите все текущие ячейки с помощью SHIFT + UP и удалите их.



* Сделайте переменную доступной для нескольких входных ячеек, создав переменную в одной ячейке. Запустите её прежде, чем вызвать переменную из другой ячейки.

dog\_name = 'Pedro'

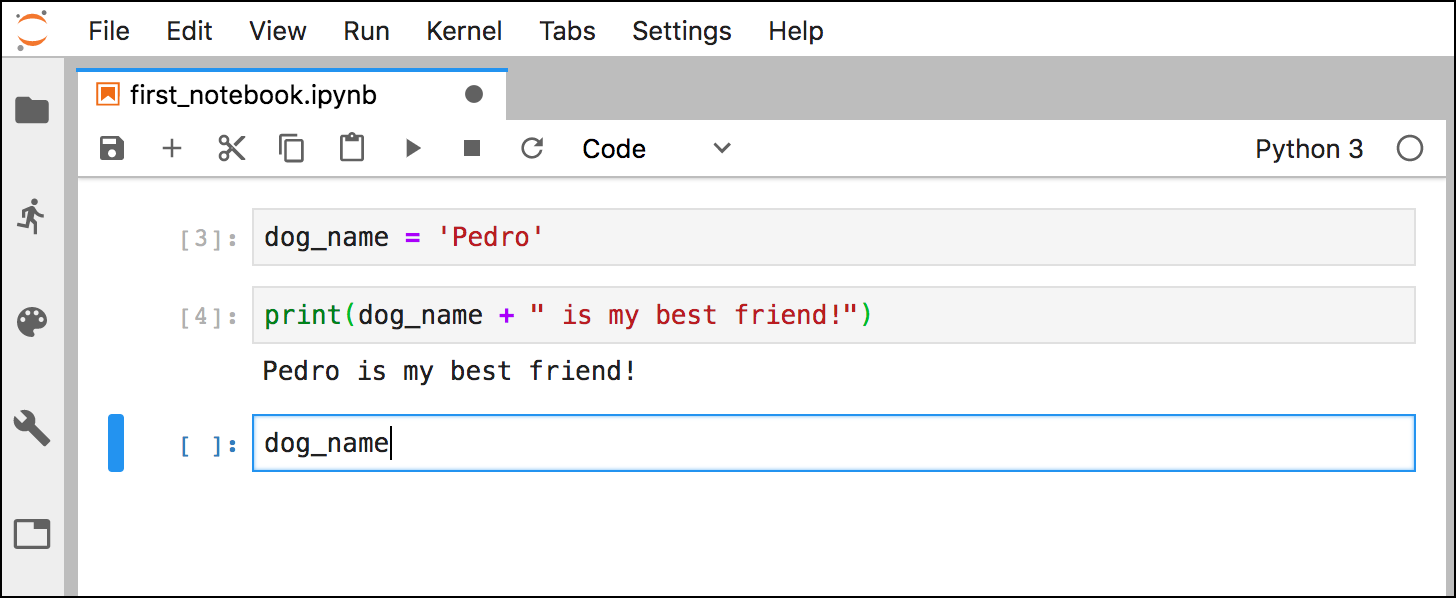
print(dog\_name + " is my best friend!")



Одной из полезных функций стандартной оболочки Python является **завершение табуляции**.

* Введите первые буквы переменной dog\_name и нажмите клавишу **tab** на клавиатуре. Так вы отыщите любые переменные, соответствующие буквам, которые вы уже ввели, в пространстве имён.

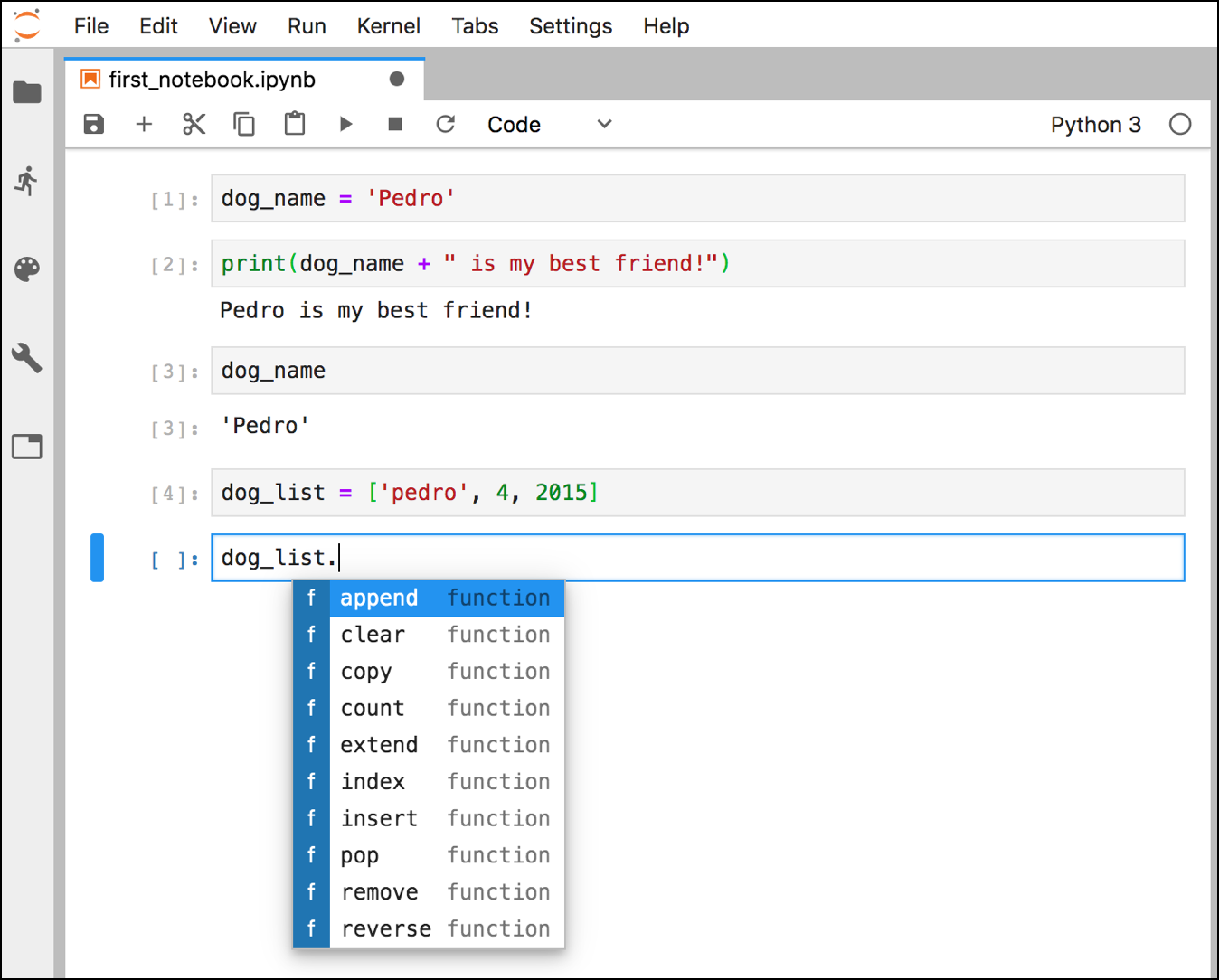
dog\_<tab>



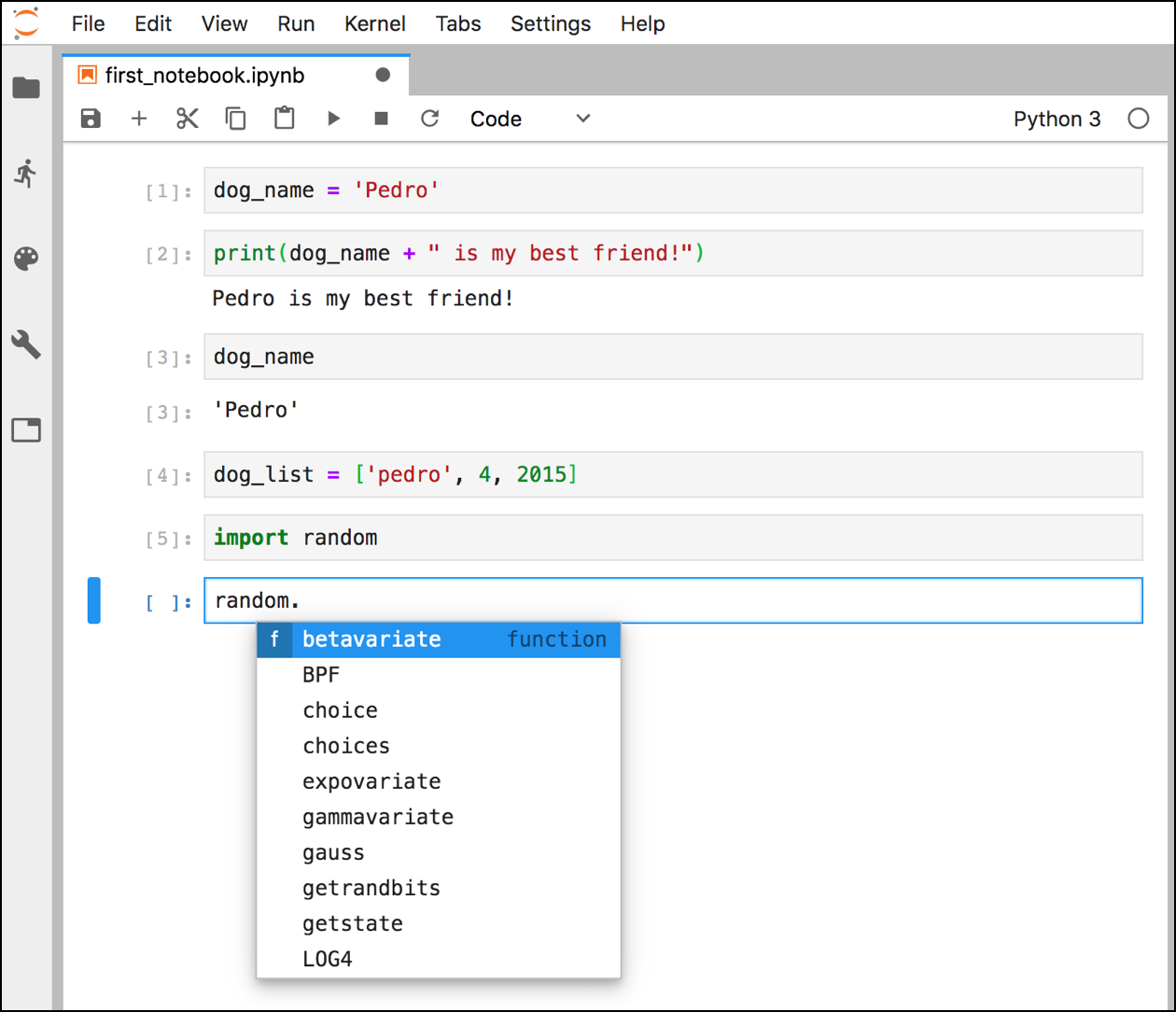
* Также можно обозначить полные методы или атрибуты, доступные в объекте.
* Давайте определим список с элементами о моей собаке и сохраним его в переменной.

dog\_list = ['pedro, 4, 2015]

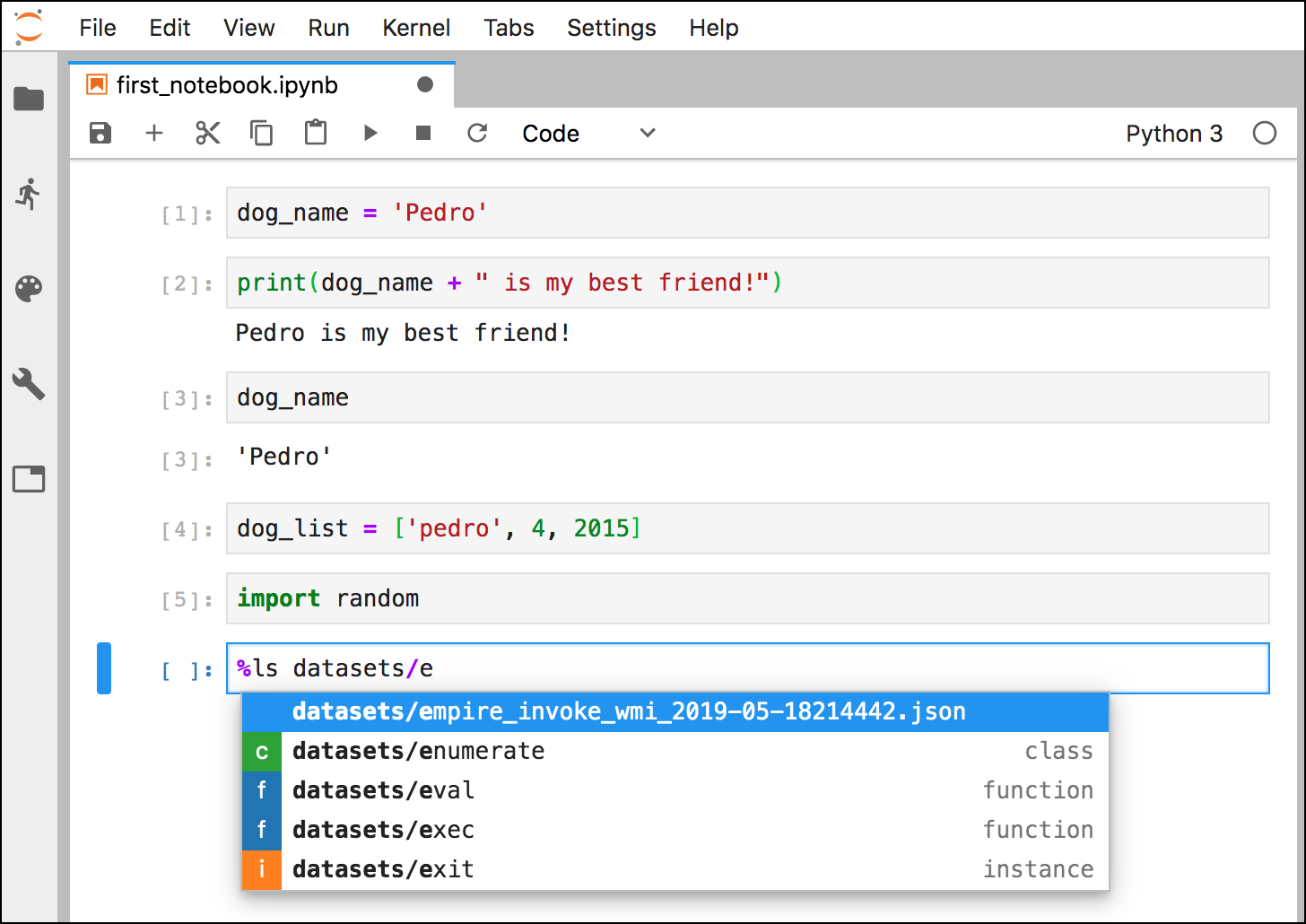
* Затем можно ввести dog\_list с **точкой** после и нажать клавишу **tab**, чтобы просмотреть методы, которые применимы к этому **списку**. Так, например, вы сможете добавить новый элемент в список, используя функцию **append**.



* То же самое можно повторить и для модулей. Давайте импортируем модуль в случайном порядке и заполним доступные методы или функции.

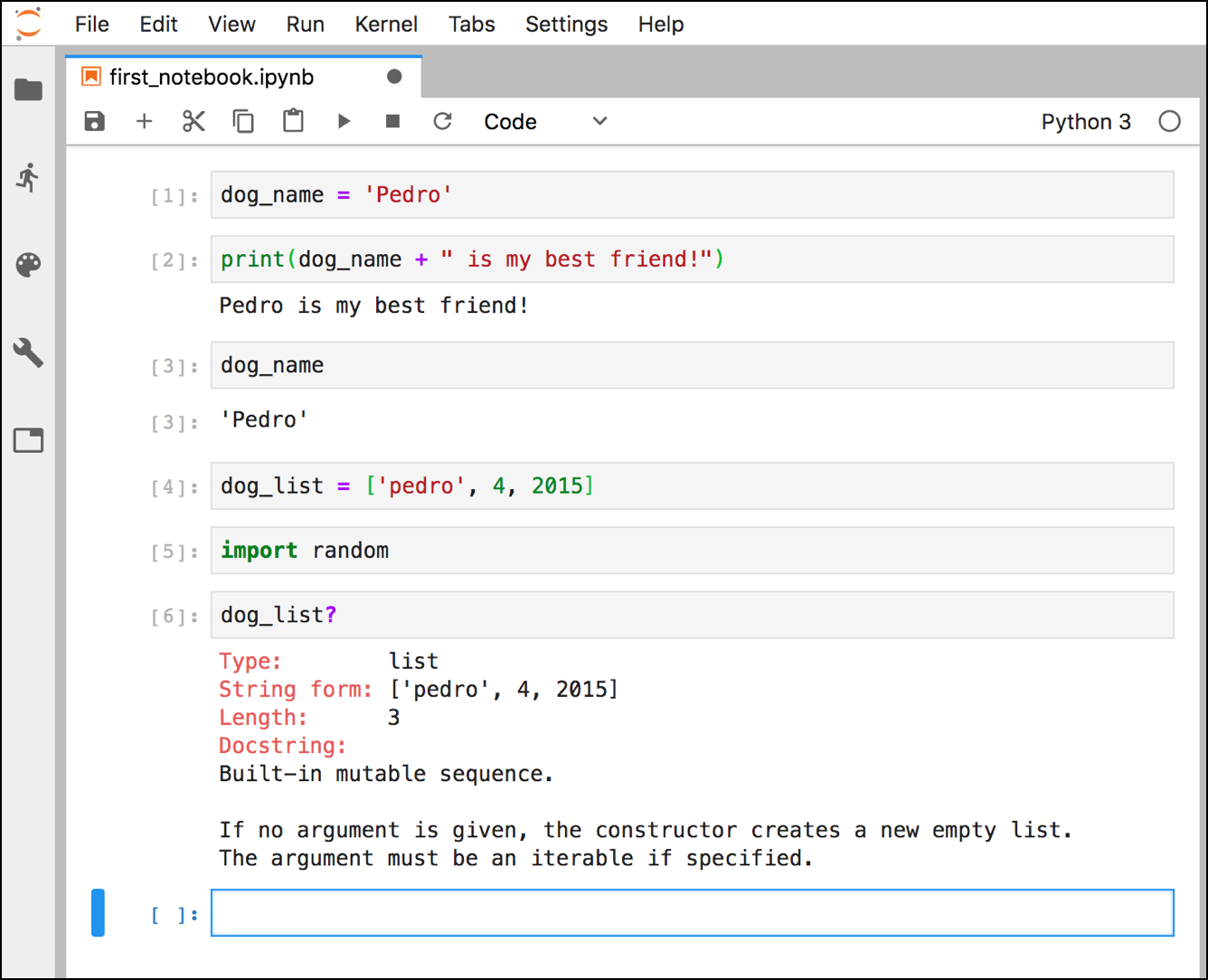


* Функция завершения табуляции также работает для путей к файлам. Мы можем проверить это с помощью папки datasets.

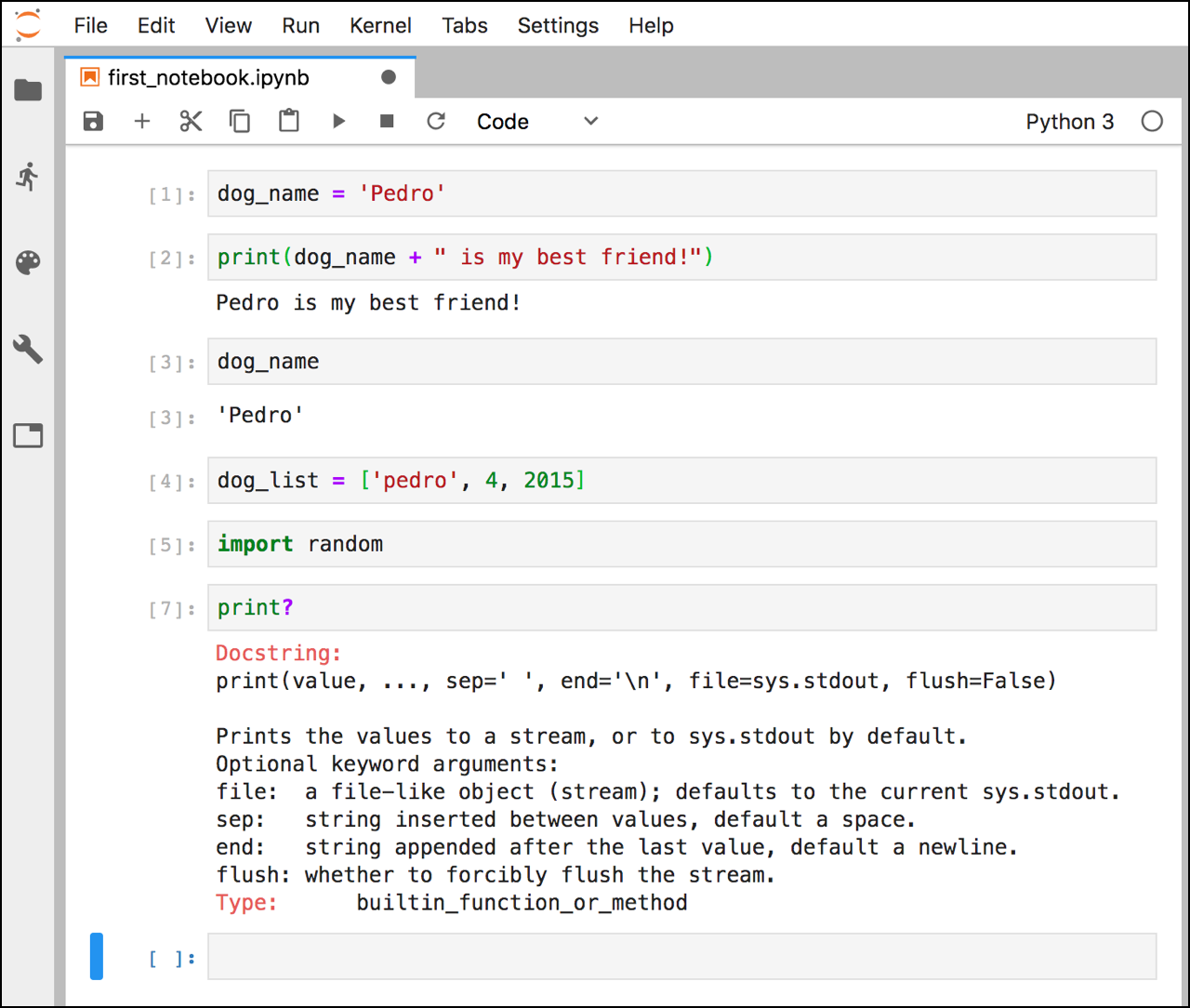


Еще одна интересная функция — это **интроспекция**, и она используется для получения информации об объекте (то есть списке, функциях и т. д.). Просто введите знак вопроса (?) до или после объекта.

* Давайте использовать его для нашей переменной dog\_list (список).



* Что насчёт функции? Давайте сравним её с функцией print. Подставив один вопросительный знак (?), появится функция docstring .



Было очень легко, правда? Если вы впервые использовали Jupyter Notebook, я надеюсь, что эта статья помогла вам ознакомиться с основными концепциями и ускорить применение вашей **первой среды Jupyter**.

Если вы снова захотите получить токен Jupyter, вам необходимо выполнить следующее:

sudo docker exec -ti helk-jupyter jupyter notebook list | grep "token" | sed 's/.\*token=\([^ ]\*\).\*/\1/'

В следующей статье мы воспользуемся доступными блокнотами в контейнере jupyter HELK, чтобы немного больше узнать об анализе данных журналов событий безопасности с помощью библиотеки python с именем Pandas.

**Ссылки**

https://jupyter.org/

https://jupyter4edu.github.io/jupyter-edu-book/

https://jupyter.readthedocs.io/en/latest/architecture/how\_jupyter\_ipython\_work.html

https://ipython-books.github.io/chapter-3-mastering-the-jupyter-notebook/

https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print\_loop

https://jupyterlab.readthedocs.io/en/stable/getting\_started/overview.html#overview