



NUMPY – PANDAS – TIME SERIES

Ссылка для скачивания материалов курса:
https://github.com/IgorP-IP/Numpy-Pandas-Times_Series

**Автор: Приходько Игорь Анатольевич
г.Санкт-Петербург**



Оглавление

№ п/п	Разделы и темы	Кол-во часов по программе
1.	Введение. Jupyter Notebook	
2.	Основы NumPy	
3.	Основы Pandas	
4.	Основы Time Series	
	ИТОГО	

Условные обозначения:

- 💡 - **определение: важно понимать, но можно не запоминать**
- ⚠ - **правило в Python: важно знать, иначе Ваша программа не будет работать!**
- 💡 - **Technical English: например, list – список**

**Lesson topic: 1.1. Что такое данные и как их анализировать****1. Данных в мире всё больше****«Данные — это новая нефть»,**

- сказал однажды Джек Ма, создатель ИТ-гиганта Alibaba.

Фразу мы теперь слышим из каждого утюга, но так ли это на самом деле?

Попробуем разобраться по-простому, чтобы было понятно тем, кто присматривается к новой профессии и планирует стать аналитиком данных или data-сайентистом — работать с нейросетями, алгоритмами машинного обучения и анализом данных.

О каких данных идёт речь?

Не о всяких. Большие данные, или Big Data, — гигантские массивы разнородной информации, которые нельзя обработать вручную или обычными программами типа MS Excel.

Почему Big Data сравнивают с нефтью?

Потому что нефть — основа экономики. Big Data, как и нефть, проникли во все сферы нашей жизни, становятся её неотъемлемой частью. Современный цифровой мир построен на данных. Нет данных — нет движения.

Гиганты вроде Amazon, Google, Apple, Microsoft, Tesla, Яндекс, Сбербанк и другие компании конкурируют за Big Data. Зачем им это нужно?

У кого больше данных — у того больше преимуществ. Эксперты в этой области считают, что в будущих технологиях нейросети и алгоритмы машинного обучения будут играть всё большую роль.

«Кто владеет информацией —**тот владеет миром»,**

- эту легендарную фразу произнёс Натан Ротшильд.

Хрестоматийной стала история о том, как сыновья Ротшильда сделали целое состояние на поражении Наполеона при Ватерлоо 18 июня 1815 года.



Ротшильды заработали на этой новости 40 миллионов фунтов стерлингов.

Реальная информация, полученная раньше других, позволила Ротшильдам вести беспрогрышную игру на бирже.

Итак, уже сегодня мы живем в мире, в котором огромное количество данных. И с каждым годом этих данных становится всё больше!

- Веб-сайты отслеживают любое нажатие любого пользователя.
- Смартфоны накапливают сведения о вашем местоположении и скорости в ежедневном и ежесекундном режиме.
- "Оцифрованные" селфера носят шагомеры, которые, не переставая, записывают их сердечные ритмы, особенности движения, схемы питания и сна.
- Умные авто собирают сведения о манерах вождения своих владельцев,
- умные дома - об образе жизни своих обитателей,
- а умные маркетологи - о наших покупательских привычках.

Сам Интернет представляет собой огромный граф знаний, который, среди всего прочего, содержит обширную гипертекстовую энциклопедию, специализированные базы данных о фильмах, музыке, спортивных результатах, игровых автоматах, мемах и коктейлях ... и слишком много статистических отчетов (причем некоторые почти соответствуют действительности!) от слишком большого числа государственных исполнительных органов, и все это для того, чтобы вы объяли необъятное.

В этих данных кроются ответы на бесчисленные вопросы, которые никто никогда не думал задавать. Этот курс научит Вас, как их находить.

Существует шутка,

что исследователь данных - это тот, кто знает статистику

лучше, чем инженер-информатик,

а информатику - лучше, чем инженер-статистик.

Не утверждаю, что это хорошая шутка, но на самом деле (в практическом плане) некоторые исследователи данных действительно являются инженерами-статистиками, в то время как другие почти неотличимы от инженеров программного обеспечения. Некоторые являются экспертами в области машинного обучения, в то время как другие не смогли бы машинно-обучиться, чтобы найти выход из детского сада. Некоторые имеют ученые степени доктора наук с впечатляющей историей публикаций, в то время



как другие никогда не читали академических статей (хотя им должно быть стыдно). Короче говоря, в значительной мере не важно, как определять понятие науки о данных, потому что всегда можно найти практикующих исследователей данных, для которых это определение будет всецело и абсолютно неверным.

Тем не менее этот факт не остановит нас от попыток. Мы скажем, что

- ❖ **исследователь данных - это тот, кто извлекает ценные наблюдения из запутанных данных.**

В наши дни мир переполнен людьми, которые пытаются превратить данные в сущностные наблюдения.

Приведу лишь несколько примеров:

- Во многих компаниях во многих странах аналитики данных занимаются поисками наилучшего способа, как заставить людей щелкать на рекламных баннерах, что в конечном счете обычно приводит к росту продаж компаний.
- Социальные сети просят вас указывать свой родной город и нынешнее местоположение, якобы чтобы облегчить вашим друзьям находить вас и связываться с вами.

Но эти социальные сети также анализируют эти местоположения, чтобы определить схемы глобальной миграции и места проживания фанатов различных футбольных команд.

- Крупные операторы розничной торговли отслеживают покупки и взаимодействия онлайн и в магазине. Он использует данные, чтобы строить предсказательные модели в отношении того, например, какие клиентки беременны, чтобы лучше продавать им товары, предназначенные для младенцев.
- В 2012 г. избирательный штаб Барака Обамы нанял десятки исследователей данных, которые вовсю копали и экспериментировали, чтобы определить избирателей, которым требовалось дополнительное внимание, при этом подбирая оптимальные обращения и программы по привлечению финансовых ресурсов, которые направлялись в адрес конкретных получателей, и сосредотачивая усилия по повышению явки избирателей там, где эти усилия могли быть наиболее успешными. А в предвыборной кампании Трампа 2016 года его команда протестировала ошеломляющее разнообразие онлайновых объявлений и проанализировала данные с целью выявления того, что сработало, а что нет.



- В настоящий момент все больше аналитиков данных используют свои навыки, делая правительство в своих странах эффективнее и помогая бездомным.
- В последние годы анализ больших данных и применение искусственного интеллекта всё больше находит своё место в совершенствовании здравоохранения.

2. Терминология

Этот курс посвящен исследованию данных (**data**) – больших данных (**big data**) - с помощью языка программирования **Python**. Сразу же возникает вопрос: что же такое наука о данных (**data science**)?

Ответ на него дать непросто — настолько данный термин многозначен. Долгое время активные критики отказывали термину «наука о данных» в праве на существование либо по причине его избыточности (в конце концов, какая наука не имеет дела с данными?), либо расценивая этот термин как «модное словечко» для придания красоты резюме и привлечения внимания агентов по найму кадров.

В подобных высказываниях критики упускается нечто очень важное.

Междисциплинарность — ключ к ее пониманию.

Говоря об анализе больших данных, мы будем встречать следующие термины:

- ▢ **Big Data (большие данные)** — это структурированные, частично структурированные или неструктурированные большие массивы данных. Также под этим термином понимают обработку, хранение и анализ огромных объемов данных.
- ▢ **Data Science (наука о данных)** — раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме.
- ▢ **Data Scientist** — это специалист, который с помощью математических алгоритмов и программных инструментов анализирует данные, которые собрала компания.
- ▢ **Data Mining (анализ данных)** — это процесс проверки, очистки, преобразования и моделирования данных с целью обнаружения полезной информации, обоснования выводов и поддержки принятия решений. Анализ данных имеет множество аспектов и подходов, охватывая различные методы под разными названиями, и используется в различных областях бизнеса, науки и социальных наук. В современном деловом мире анализ данных играет важную роль в принятии более научных решений и помогает бизнесу работать более эффективно.

Data Mining — это особый метод анализа данных, который фокусируется на статистическом моделировании и обнаружении знаний для предиктивных, а не чисто описательных целей, в то время как бизнес-аналитика охватывает анализ



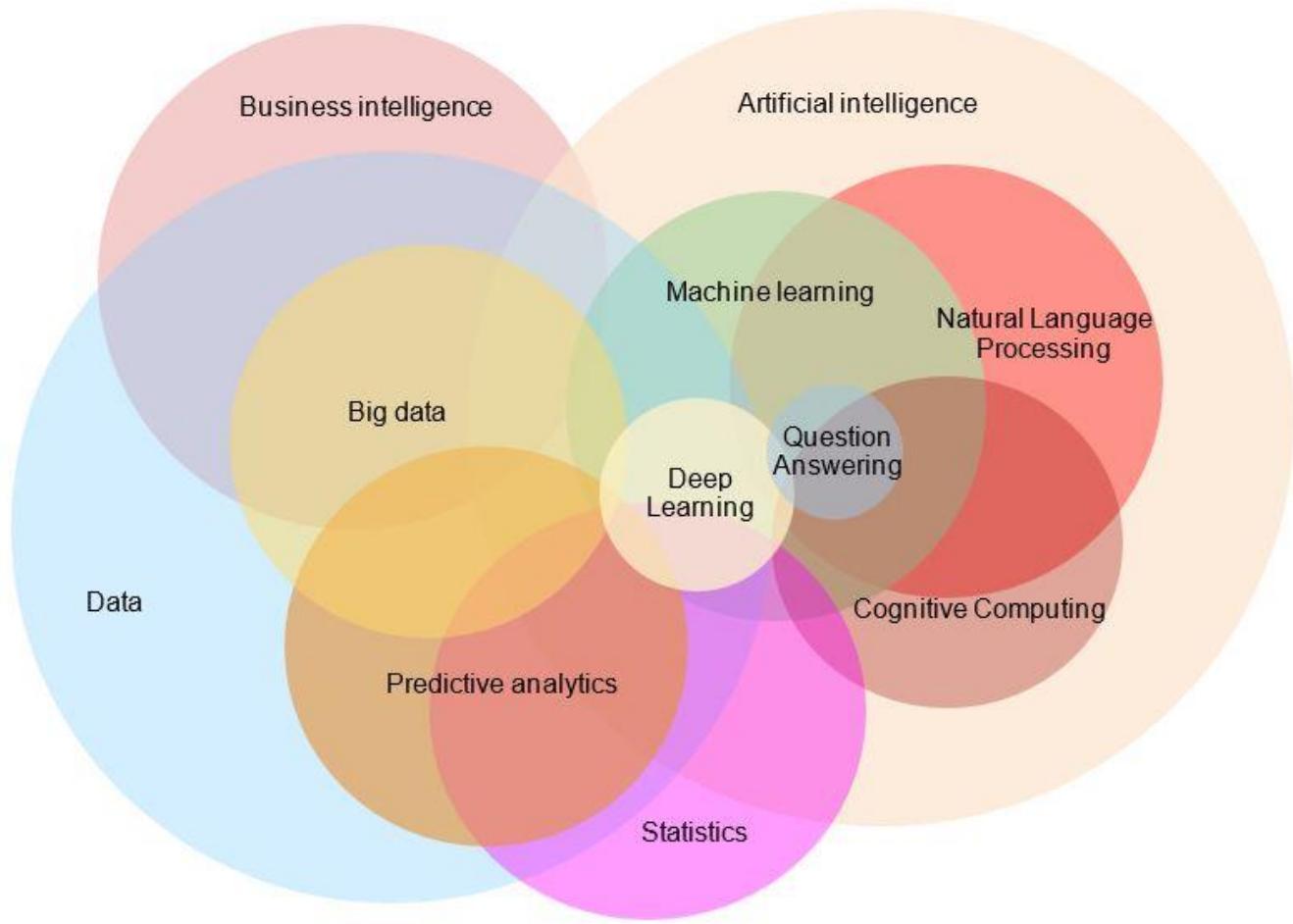
данных, который в значительной степени опирается на агрегацию, фокусируясь в основном на деловой информации.

- ▢ **Machine Learning, ML (машинное обучение)** — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.
- ▢ **Deep Learning, DL (глубокое обучение)** — совокупность методов машинного обучения (с учителем, с частичным привлечением учителя, без учителя, с подкреплением), основанных на обучении представлениям (англ. feature/representation learning), а не специализированных алгоритмах под конкретные задачи.
- ▢ **Artificial intelligence, AI (Искусственный интеллект, ИИ)** — свойство искусственных интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека (не следует путать с искусственным сознанием); наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.

Искусственный интеллект связан со сходной задачей использования компьютеров для понимания человеческого интеллекта, но не обязательно ограничивается биологически правдоподобными методами.

Существующие на сегодня интеллектуальные системы имеют довольно узкие области применения. Например, программы, способные обыграть человека в шахматы, как правило, не могут отвечать на вопросы

Для наглядности все эти термины можно представить в виде подобной графической схемы:



3. Использование Python для анализа данных

Простой и понятный язык программирования Python идеально подходит для получения и понимания данных любого типа, а также для выполнения с ними различных действий. Он сочетает в себе богатый набор встроенных структур данных для базовых операций и надежную экосистему библиотек с открытым исходным кодом для анализа и работы с данными любого уровня сложности.

В этом курсе мы рассмотрим такие библиотеки:

- **NumPy**,
- **Pandas**,
- **Matplotlib**,
- **Scikit-learn** и некоторых других.

На языке Python вы сможете писать лаконичный и интуитивно понятный код с минимальными затратами времени и усилий, реализуя большинство идей всего в нескольких строках. На самом деле гибкий синтаксис позволяет реализовать несколько



операций с данными даже в одной строке. Например, можно написать строку кода, которая одновременно фильтрует, преобразует и агрегирует данные.

Будучи языком общего назначения, Python подходит для решения широкого круга задач. Работая с этим языком, можно легко интегрировать анализ данных с другими задачами для создания полнофункциональных, хорошо продуманных приложений. Например, можно создать бота, который выдает прогнозы фондового рынка в ответ на запрос пользователя на естественном языке. Что-бы создать такое приложение, понадобится API для бота, прогнозная модель машинного обучения и инструмент обработки естественного языка (NLP) для взаимодействия с пользователями. Для всего этого существуют мощные библиотеки Python.

4. О чём этот курс?

Этот курс посвящен вопросам преобразования, обработки, очистки данных и вычислениям на языке Python. Моя цель – предложить руководство по тем частям языка программирования Python и экосистемы его библиотек и инструментов, относящихся к обработке данных, которые помогут Вам анализировать большие данные.

5. Какого рода данные?

Говоря «данные», я имею в виду прежде всего структурированные данные; это намеренно расплывчатый термин, охватывающий различные часто встречающиеся виды данных, как то:

- табличные данные, когда данные в разных столбцах могут иметь разный тип (строки, числа, даты или еще что-то). Сюда относятся данные, которые обычно хранятся в реляционных базах или в файлах с запятой в качестве разделителя;
- многомерные списки (матрицы);
- данные, представленные в виде нескольких таблиц, связанных между собой по ключевым столбцам (то, что в SQL, Structured Query Language — «язык структурированных запросов», называется первичными и внешними ключами);
- равноотстоящие и неравноотстоящие временные ряды.

Этот список далеко не полный. Значительную часть наборов данных можно преобразовать к структурированному виду, более подходящему для анализа и моделирования, хотя сразу не всегда очевидно, как это сделать.

6. Почему именно Python?



Для многих людей (и меня в том числе) Python – язык, в который нельзя не влюбиться. С момента своего появления в 1991 году Python стал одним из самых популярных динамических языков программирования наряду с Perl, Ruby и другими.

С 1991 года язык Python претерпел много улучшений, особенно версия 3, выпущенная в декабре 2008 года. Отличия оказались такими существенными, что программы, написанные на версиях 1 и 2, далеко не всегда можно было запустить на интерпретаторе языка Python 3. Мы с Вами будем изучать именно эту последнюю современную версию - Python 3. И Вам не стоит переживать о несовместимости ранних версий Python с его текущей 3-ей версией, так как с 2008 года прошло уже достаточно много времени, и самые важные программы были адаптированы под 3-ю версию Python.

Недавнее появление улучшенных библиотек для Python (прежде всего numpy и pandas) сделало его серьезным конкурентом в решении задач манипулирования данными. В сочетании с достоинствами Python как универсального языка программирования это делает его отличным выбором для создания приложений обработки данных. Своим успехом в области научных расчетов Python также обязан простоте интеграции с кодом на C, C++ и FORTRAN. Во многих современных вычислительных средах применяется общий набор унаследованных библиотек, написанных на FORTRAN и C, содержащих реализации алгоритмов линейной алгебры, оптимизации, интегрирования, быстрого преобразования Фурье и других.

Поэтому многочисленные компании и национальные лаборатории используют Python как «клей» для объединения написанных за много лет программ.

Основной недостаток Python: поскольку Python – интерпретируемый язык программирования, в общем случае написанный на нем код работает значительно медленнее, чем эквивалентный код на компилируемом языке типа Java или C++

7.Необходимые библиотеки для Python

Краткий обзор необходимых библиотек.

IPython и Jupyter



Проект IPython (<http://ipython.org/>) начал свою работу в 2001 году как побочный проект, имеющий целью создать более удобный интерактивный интерпретатор Python.

В 2014 году команда разработки IPython анонсировали проект Jupyter (<https://jupyter.org/>) – широкую инициативу проектирования языково-независимых средств интерактивных вычислений.

Сам IPython стал компонентом более широкого проекта Jupyter с открытым исходным кодом, предоставляющего продуктивную среду для интерактивных исследовательских вычислений.

Jupyter-блокноты позволяют создавать контент на языках разметки Markdown и HTML, т. е. готовить комбинированные документы, содержащие код и текст.

NumPy

NumPy (<https://numpy.org/>), сокращение от «Numerical Python», – основной пакет для выполнения научных расчетов на Python (в первую очередь используется для структурированных данных) в отношении **однородных данных**. Большая часть этой курса базируется на NumPy и построенных поверх него библиотеках.

pandas

Библиотека pandas (<https://pandas.pydata.org/>) предоставляет структуры данных и функции, призванные сделать работу со структурированными данными простым, быстрым и выразительным делом. Pandas даёт возможность обрабатывать **неоднородные данные**.

matplotlib

Библиотека matplotlib (<https://matplotlib.org/>) – самый популярный в Python инструмент для создания графиков и других способов визуализации двумерных данных.

scikit-learn

Проект scikit-learn (<https://scikit-learn.org/stable/>), запущенный в 2007 году, с самого начала стал основным инструментарием машинного обучения для программистов на Python.

8. Установка и настройка для Windows

8.1. Установка Python на компьютер

Для установки интерпретатора Python на Ваш компьютер Вы должны знать конфигурацию своего компьютера.



Нажмите две клавиши Windows + Pause.

Вы увидите на мониторе своего компьютера следующее:

The screenshot shows the Windows System Properties window. On the left, there's a sidebar with various system categories like Display, Sound, Notifications, Focus, Power, Battery, Memory, Tablet, Multitasking, Projection, General, and Clipboard. The 'System' category is selected. The main pane is titled 'About this computer' and contains sections for 'Device characteristics' and 'Windows characteristics'. In 'Device characteristics', it lists the device as 'Aspire ES1-531' with an Intel Celeron N3050 processor and 4GB RAM. In 'Windows characteristics', it shows Windows 10 Home edition installed on April 18, 2021. There are also links for remote desktop, system protection, additional system parameters, renaming the PC, getting help, and sending feedback.

Параметры

>Main menu

Найти параметр

Система

- Дисплей
- Звук
- Уведомления и действия
- Фокусировка внимания
- Питание и спящий режим
- Батарея
- Память
- Планшет
- Многозадачность
- Проектирование на этот компьютер
- Общие возможности
- Буфер обмена

Главная

Найти параметр

О программе

Характеристики устройства

Aspire ES1-531	LAPTOP-8SPDRKAM
Имя устройства	Intel(R) Celeron(R) CPU N3050 @ 1.60GHz 1.60 GHz
Процессор	
Оперативная память	4,00 ГБ (доступно: 3,83 ГБ)
Код устройства	757051D8-DA9B-4645-9EA0-D52002AC5295
Код продукта	00327-30270-43164-AAOEM
Тип системы	64-разрядная операционная система, процессор x64
Перо и сенсорный ввод	Для этого монитора недоступен ввод с помощью пера и сенсорный ввод

Копировать

Переименовать этот ПК

Характеристики Windows

Выпуск	Windows 10 Домашняя для одного языка
Версия	20H2
Дата установки	18.04.2021

Удаленный рабочий стол

Защита системы

Дополнительные параметры системы

Переименовать этот ПК (для опытных пользователей)

Получить помощь

Отправить отзыв

В характеристиках устройства Вы найдёте тип системы (у меня это 64-разрядная операционная система).

В характеристиках Windows Вы найдёте свою операционную систему (у меня это Windows 10).

Эта информация Вам понадобится при установке интерпретатора Python на Ваше устройство.

Конечно у Вас может быть свое устройство (не обязательно стационарный компьютер) и другая операционная система. Это не существенно.

В любом случае для установки интерпретатора Python нам нужно перейти на официальный сайт

www.python.org:

1.идём на официальный сайт www.python.org

Welcome to Python.org — Mozilla Firefox

https://www.python.org

2. выбираем во вкладке Downloads свою операционную систему (Windows):

Python Releases for Windows | Python.org — Mozilla Firefox

https://www.python.org/downloads/windows/

Python Releases for Windows

■ Latest Python 3 Release - Python 3.9.6
■ Latest Python 2 Release - Python 2.7.18

Stable Releases

■ Python 3.9.6 - June 28, 2021
Note that Python 3.9.6 cannot be used on Windows 7 or earlier.
■ Download Windows embeddable package (32-bit)
■ Download Windows embeddable package (64-bit)
■ Download Windows help file
■ Download Windows installer (32-bit)
■ Download Windows installer (64-bit)

■ Python 3.8.11 - June 28, 2021
Note that Python 3.8.11 cannot be used on Windows XP or earlier.
■ No files for this release.

■ Python 3.6.14 - June 28, 2021
Note that Python 3.6.14 cannot be used on Windows XP or earlier.

Pre-releases

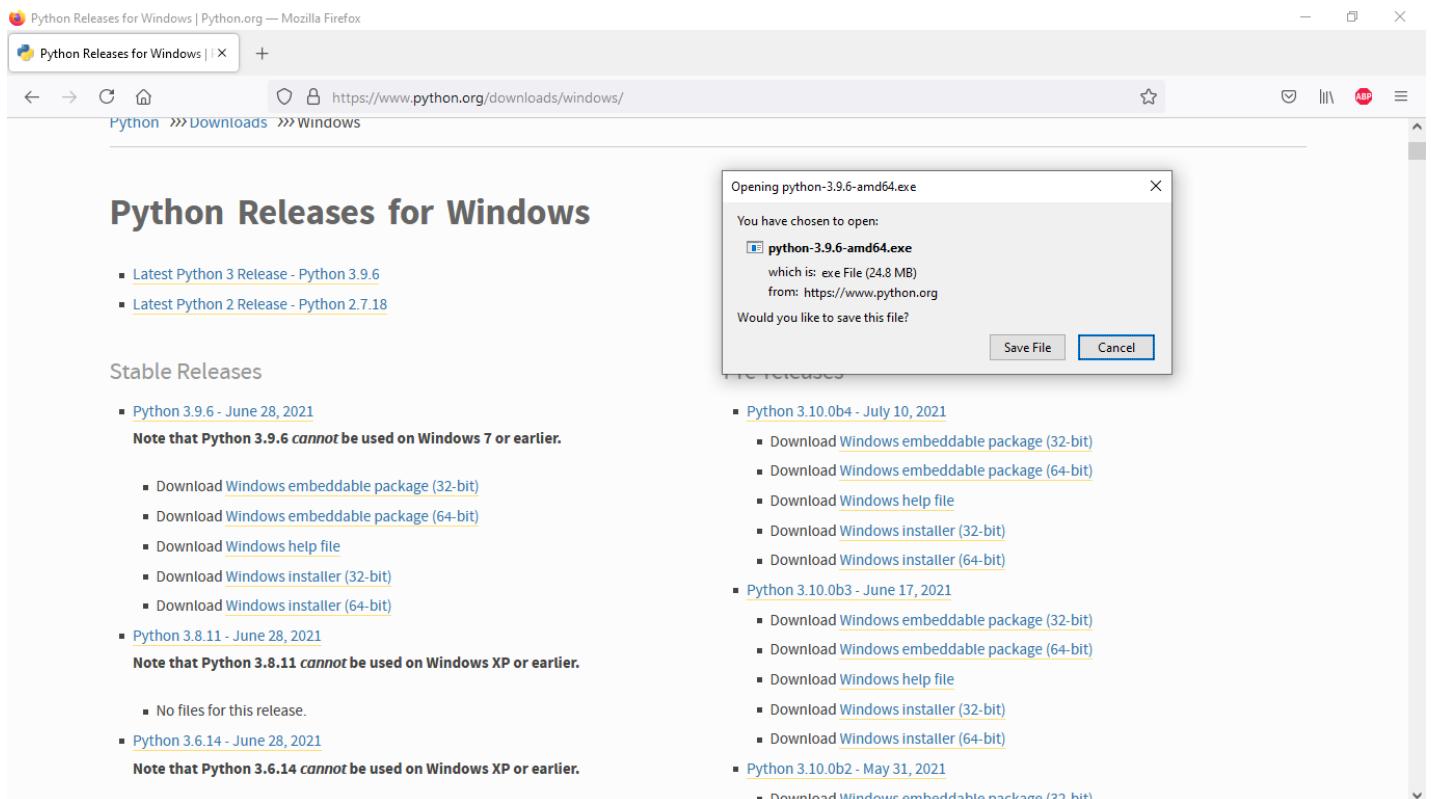
■ Python 3.10.0b4 - July 10, 2021
■ Download Windows embeddable package (32-bit)
■ Download Windows embeddable package (64-bit)
■ Download Windows help file
■ Download Windows installer (32-bit)
■ Download Windows installer (64-bit)

■ Python 3.10.0b3 - June 17, 2021
■ Download Windows embeddable package (32-bit)
■ Download Windows embeddable package (64-bit)
■ Download Windows help file
■ Download Windows installer (32-bit)
■ Download Windows installer (64-bit)

■ Python 3.10.0b2 - May 31, 2021
■ Download Windows embeddable package (32-bit)

3. выбираем свою разрядность системы (Windows 10, 64-разрядная система)

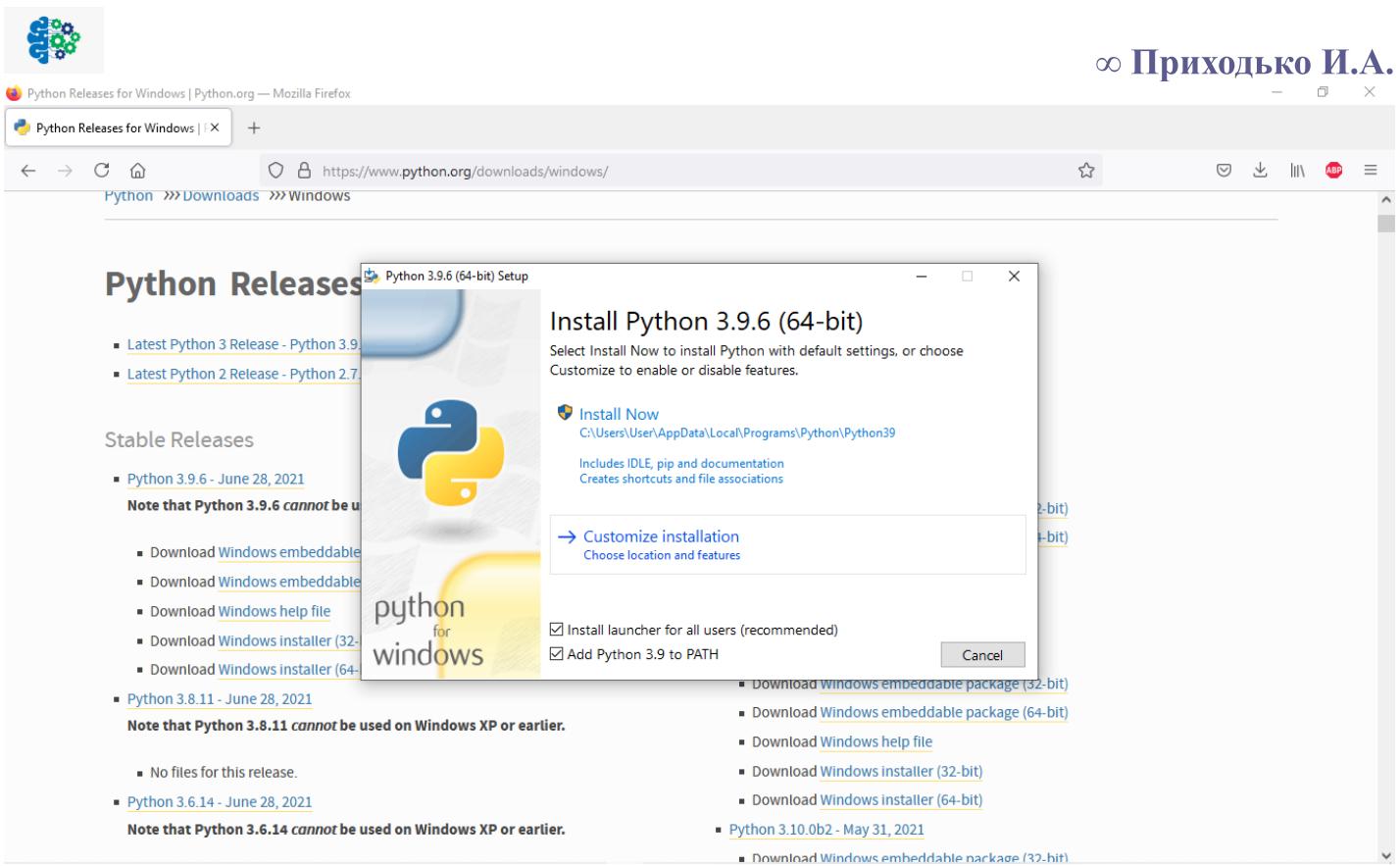
4. Скачиваем последнюю стабильную версию интерпретатора Python (версия 3.9.6 на момент создания курса Python). После скачивания установщика запускаем этот установщик (файл с расширением .exe)



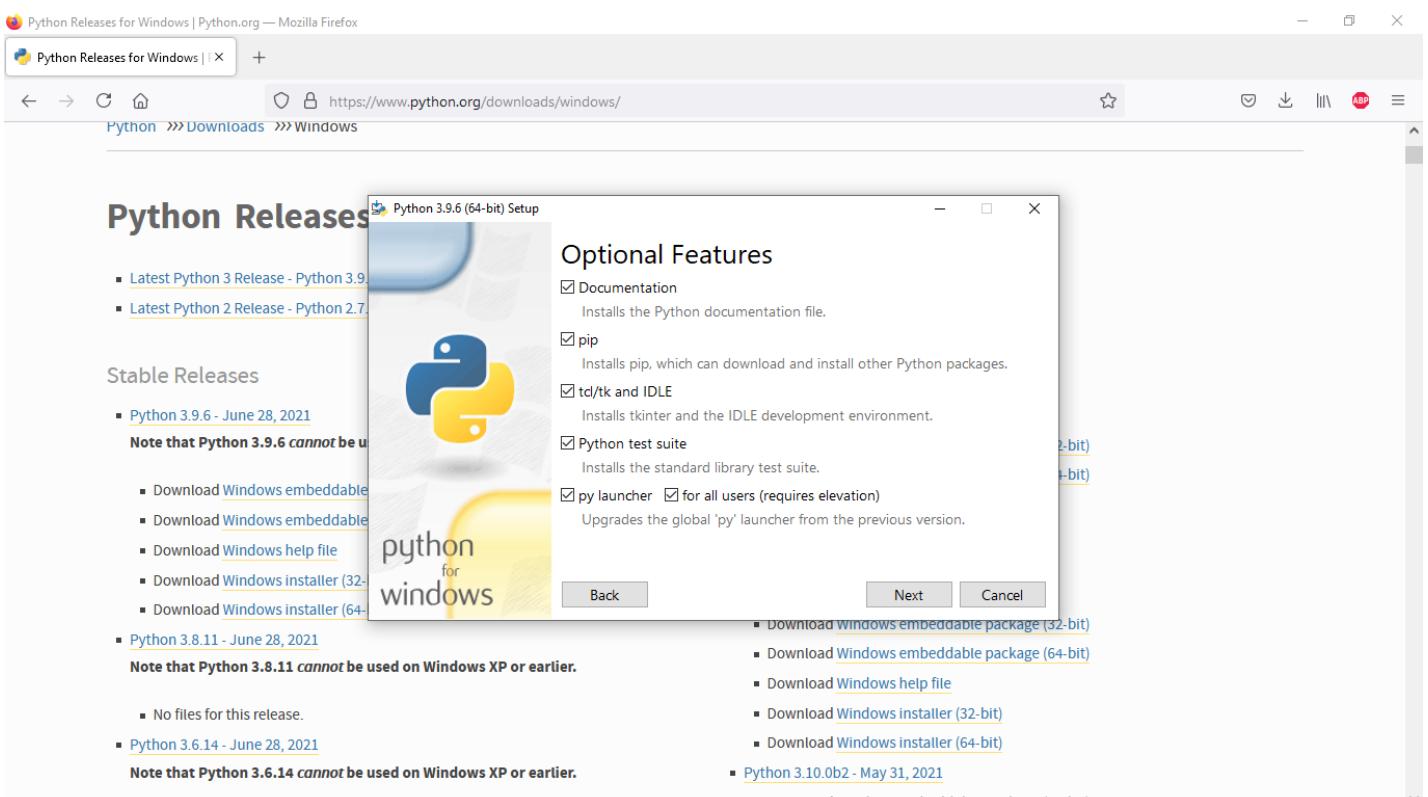
5. Нажимаем Save file

6. Устанавливаем загруженный файл на свой компьютер

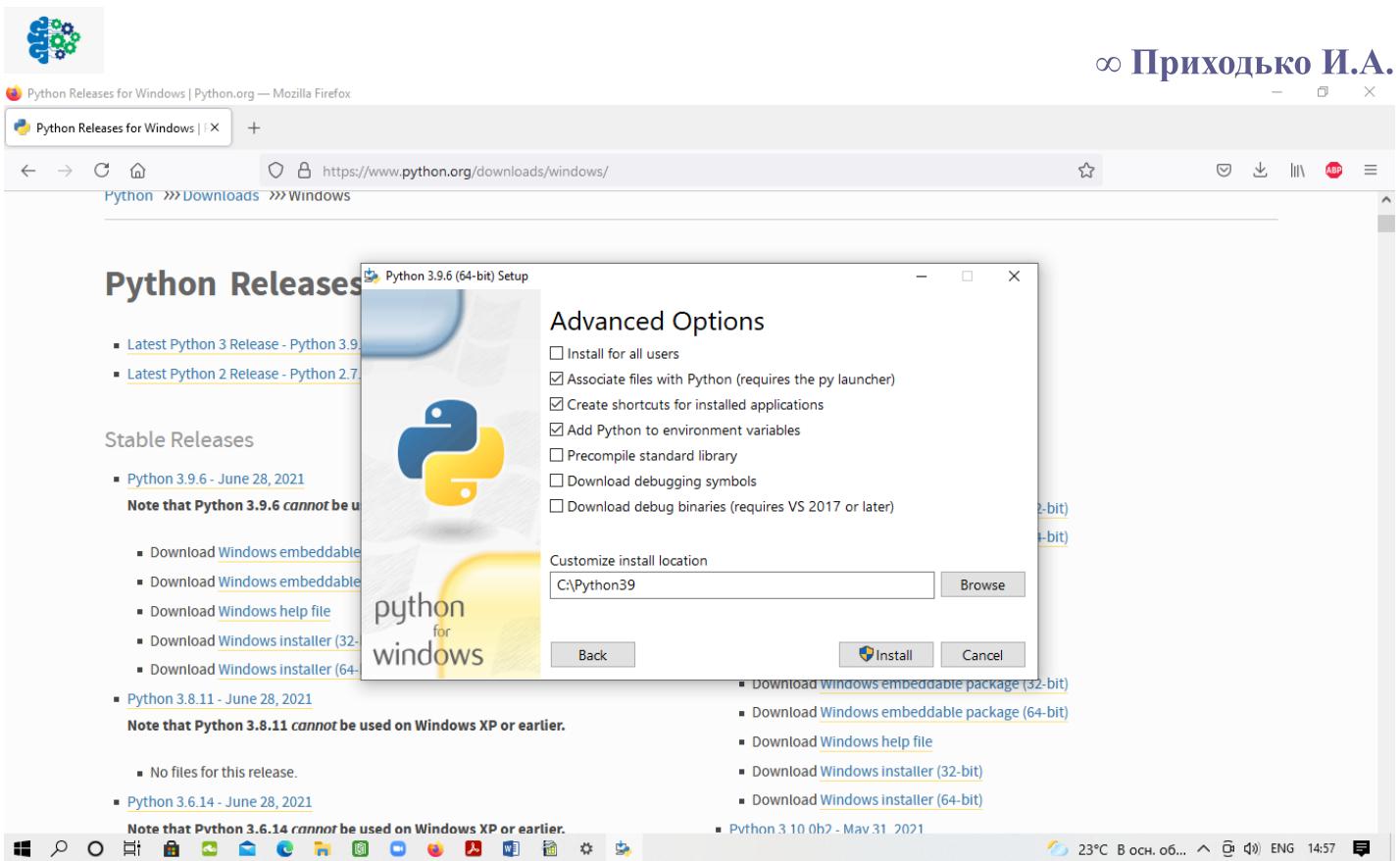
7. Отмечаем галочкой опцию Add Python 3.9 to PATH (на данный момент Вы будете устанавливать на свой компьютер самую последнюю версию, сейчас это версия Python – 3.12.2) и выбираем режим Customize Installation



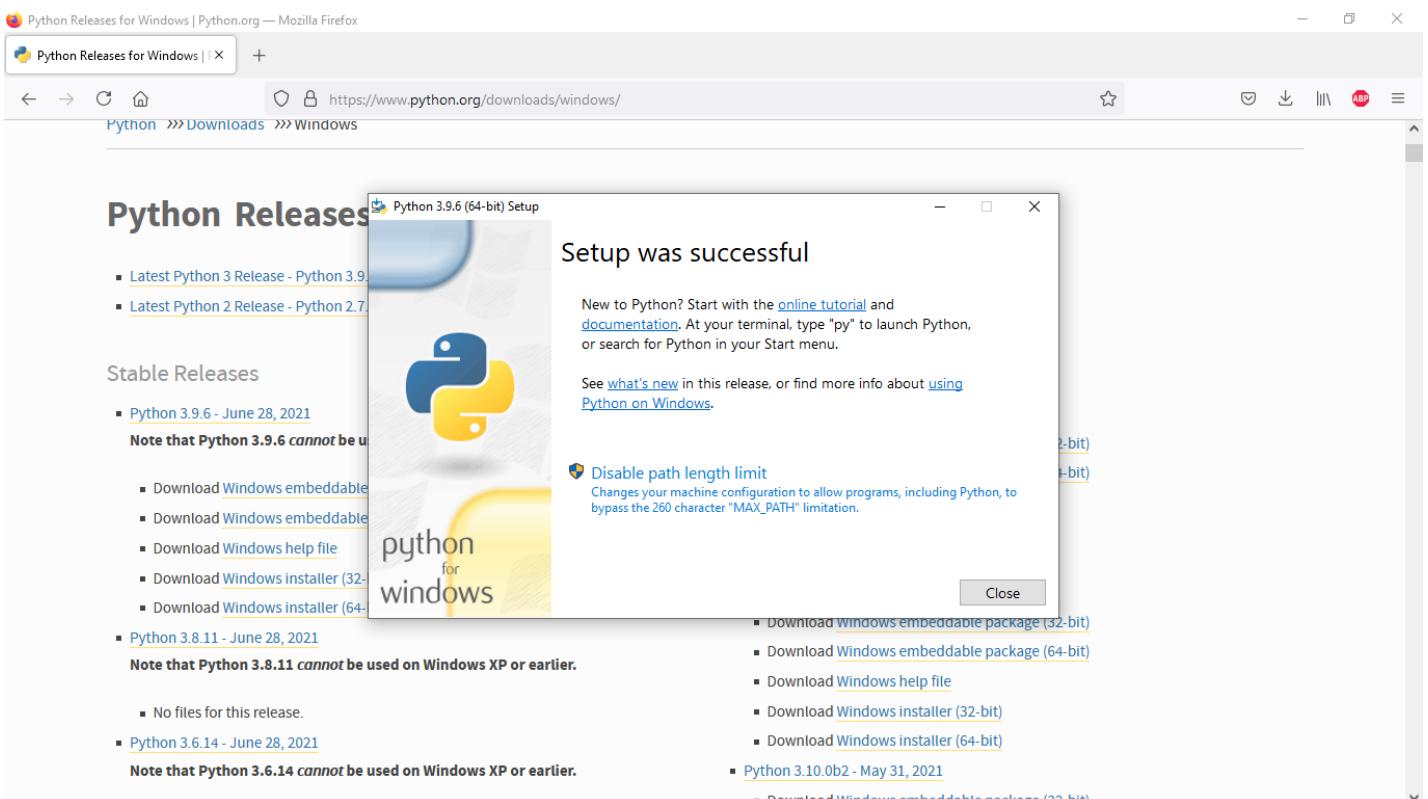
8. В появившемся окне проверьте, что галочки стоят во всех опциях. После этого нужно нажать Next



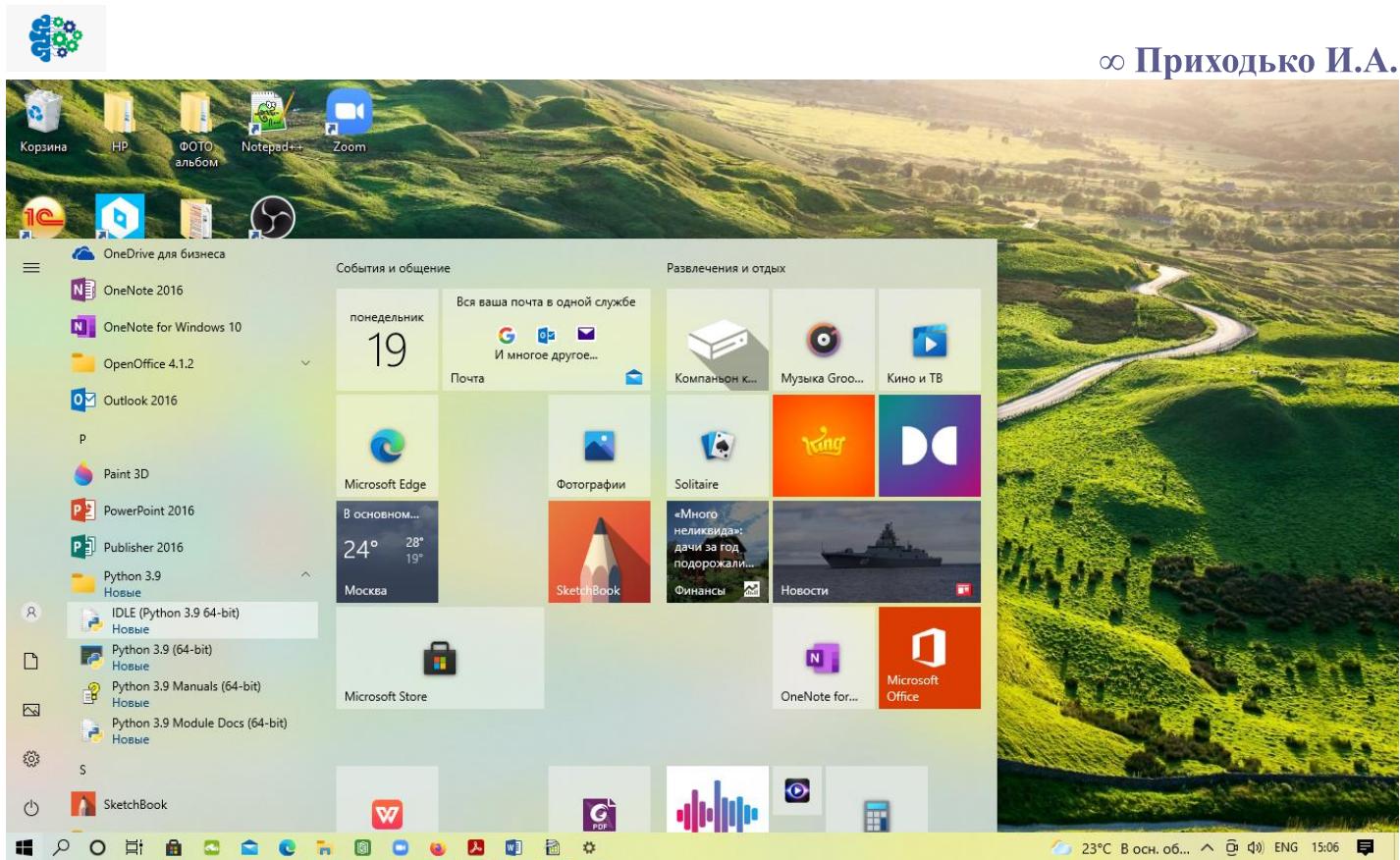
9. Выбираем путь (вместо предлагаемого пути, лучше сделать указать следующий путь: C:\Python39) и нажимаем Install:



10. После установки интерпретатора Python на Ваше устройство нажать клавишу Close:



11. Проверить установку интерпретатора нужно через кнопку Пуск:



12. Можно создать ярлык IDLE на рабочем столе Вашего (а можно этого не делать).

Проверяем работоспособность установленного интерпретатора Python.

На своем компьютере нажмите одновременно клавиши Windows+R : появится окно.

Внутри этого окна наберите команду cmd : появится Консольное окно (Командная строка).

В этой консоли наберите команду python : запустится интерпретатор языка Python.

После знака >>> (приглашение) давайте выполним несколько простых команд:

3+5 Enter (8)

3-5 Enter (-2)

3*5 Enter (15)

3/5 Enter (0.6)

О всех этих командах мы будем говорить на следующих наших занятиях.

Если всё это работает, то значит установка прошла успешно.

Такой режим работы называется интерактивным. При работе в интерактивном режиме результаты выполнения ваших инструкций будут



выводиться сразу же после нажатия клавиши Enter вслед за строкой с **приглашением >>>**.

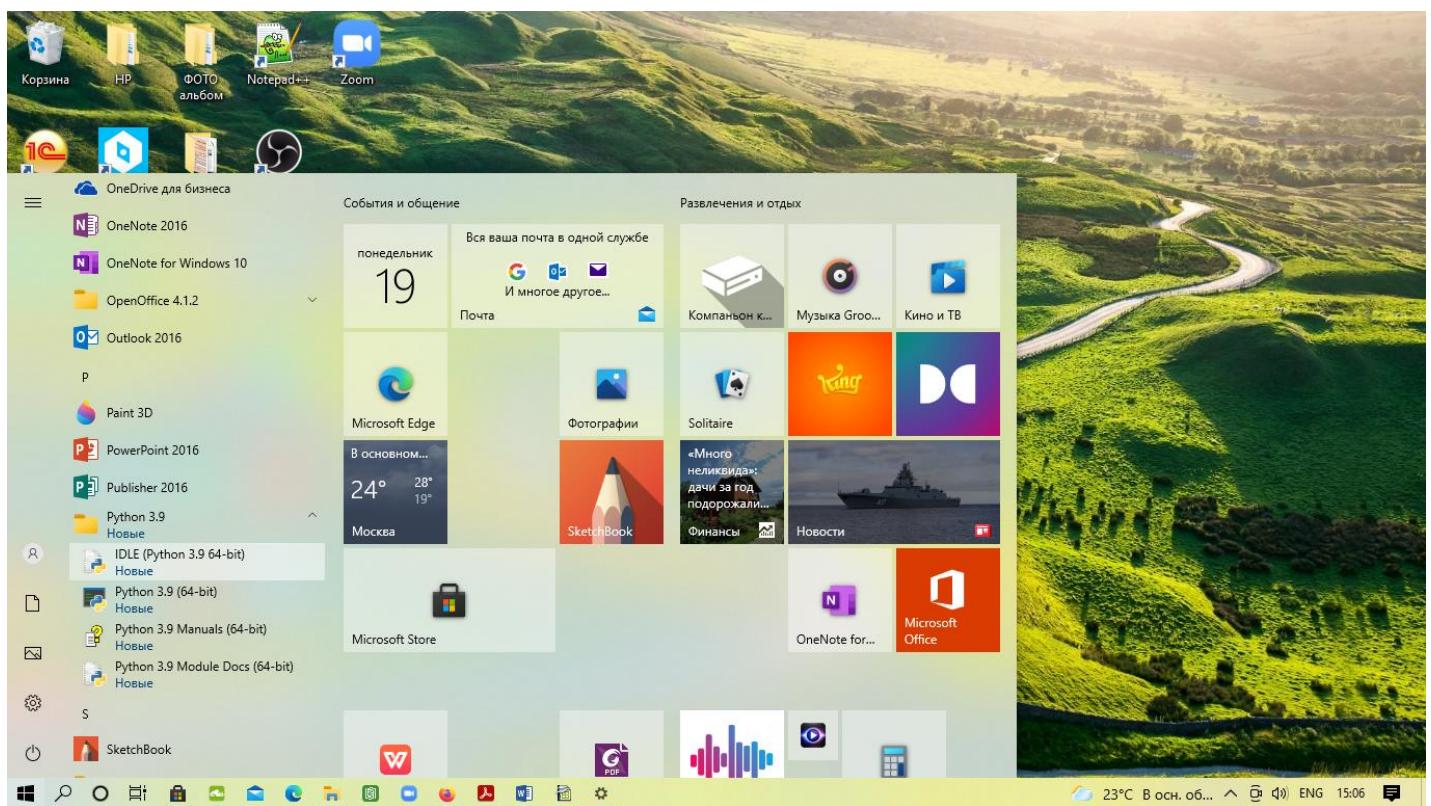
Чтобы выйти из Командной строки достаточно нажать комбинацию клавиш Ctrl+Z, после этого Командную строку можно закрыть, нажав на кнопку X в верхнем правом ее углу.

Вместо Командной строки мы будем использовать интегрированную среду разработки IDLE.

💡 IDE - integrated development environment.

Официально название IDLE считается искаженной аббревиатурой IDE, но в действительности она была названа так в честь члена труппы цирка Монти Пайтона (Monty Python) – Эрика Айдла (Eric Idle).

IDLE – это один из 4-х пунктов, которые можно найти через Пуск в установленном интерпретаторе Python 3.9:



где пункт IDLE – запускает интегрированную среду разработки с графическим интерфейсом,

пункт Python 3.9 – запускает простой интерактивный сеанс работы с интерпретатором, пункт Python 3.9 Manuals - вызывает стандартное справочное руководство,



пункт Python 3.9 Module Docs – запускает механизм документирования PyDoc.

Чтобы запустить IDLE, нужно один раз кликнуть курсором на его иконке.

При запуске IDLE интерпретатор Python запускается автоматически.

После знака >>> (приглашение) давайте выполним те же несколько простых команд:

3+5 Enter (8)

3-5 Enter (-2)

3*5 Enter (15)

3/5 Enter (0.6)

Мы получим те же результаты, которые мы раньше для этих примеров получили уже в командной строке.

Такой режим работы в IDLE также называется интерактивный.

Но при выходе из IDLE все, что мы напрограммировали, просто пропадёт.

А если мы хотим пользоваться созданным нами кодом много раз, но при этом мы не хотим всякий раз этот код набирать заново? Как быть в этом случае?

- 💡 На практике чаще пользуются **файловым режимом** – когда программа написана в каком-либо текстовом редакторе а затем выполняется в этой среде.

Возвращаясь к файловому методу, продемонстрируем его работу в IDLE.

В самом простом варианте в IDLE мы можем выбрать меню File, в нем выбрать пункт New file: появится окно с текстовым редактором, где мы сможем написать текст нашей программы.

Например, напишем команду `print("Hello world!")`, после этого сохраним ее с расширением .py - program1.py (**расширением .py обозначаются все файлы, написанные на языке Python**). Для сохранения своих файлов предлагаю на Рабочем столе создать папку под своим именем и туда сохранять все свои программы!).

Теперь мы можем запустить этот файл прямо из IDLE: нажимаем Run - Run Module - выбираем модуль, который необходимо запустить. И теперь наша программа будет выполнена:

The screenshot shows the Python IDLE interface. On the left is a script editor window titled "program1.py" containing the code: `print("Hello world!")`. On the right is an interactive shell window titled "IDLE Shell 3.9.6" showing the output of running the script: `Hello world!`.

Теперь если мы закроем нашу программу, то впоследствии мы всегда можем многократно запустить её из IDLE: выбираем меню File – Open – указываем путь к требуемому модулю – в открывшемся модуле выбираем Run – Run Module – модуль выполняется.

8.2. Установка Jupyter Notebook на компьютер

Вся информация по установке Jupyter Notebook есть на его официальном сайте:

The screenshot shows a web browser window with the URL jupyter.org/install. The page contains instructions for installing JupyterLab, Jupyter Notebook, and Voilà. It includes code snippets for terminal commands like `jupyter lab`, `pip install notebook`, and `voila`.

Note: If you install JupyterLab with conda or mamba, we recommend using [the conda-forge channel](#).

Once installed, launch JupyterLab with:

```
jupyter lab
```

Jupyter Notebook

Install the classic Jupyter Notebook with:

```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

Voilà

Install Voilà with:

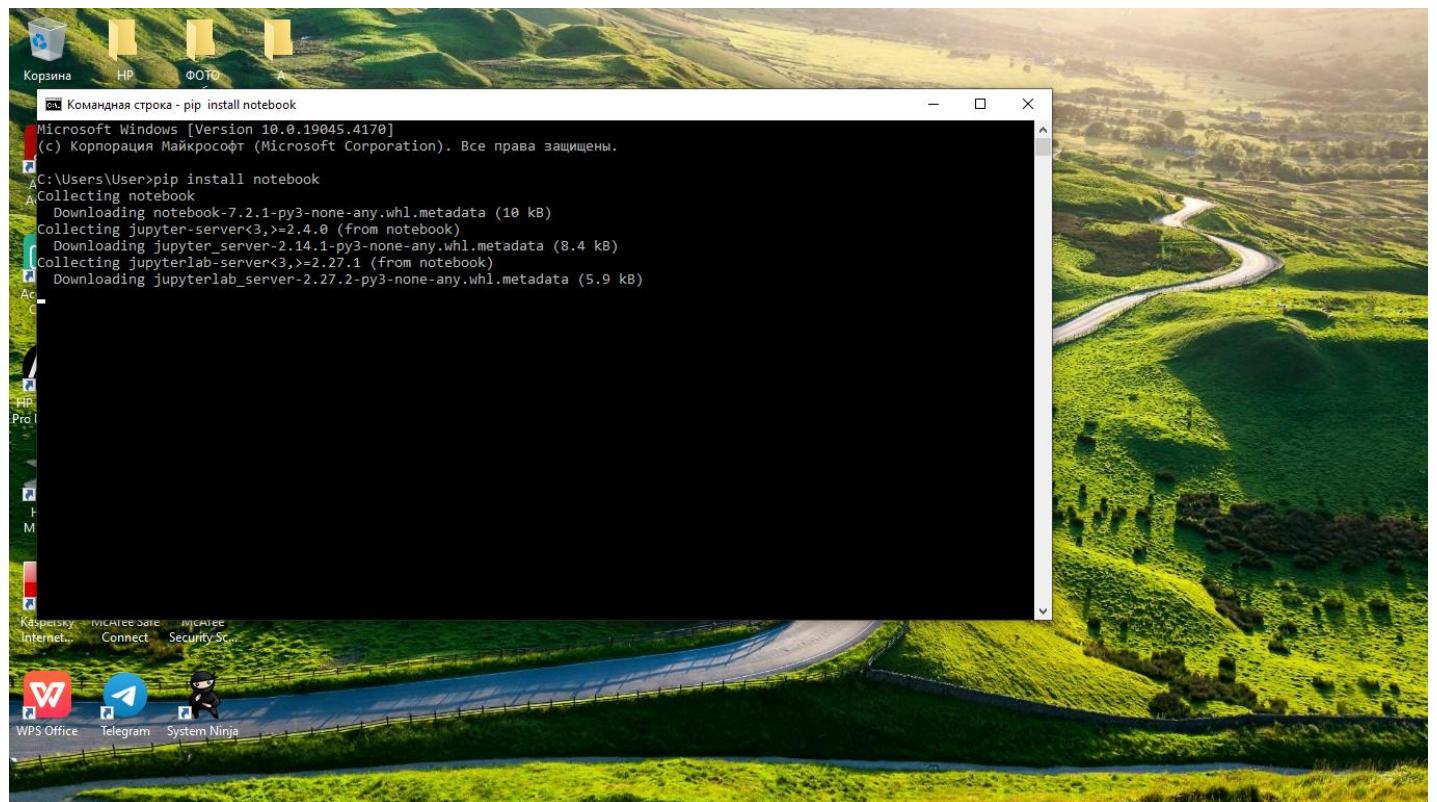
```
pip install voila
```

Once installed, launch Voilà with:

```
voila
```

Для установки Jupyter Notebook согласно официального сайта нужно в командной строке ввести команду:

```
pip install notebook
```





В этом нам поможет специальная утилита — pip.

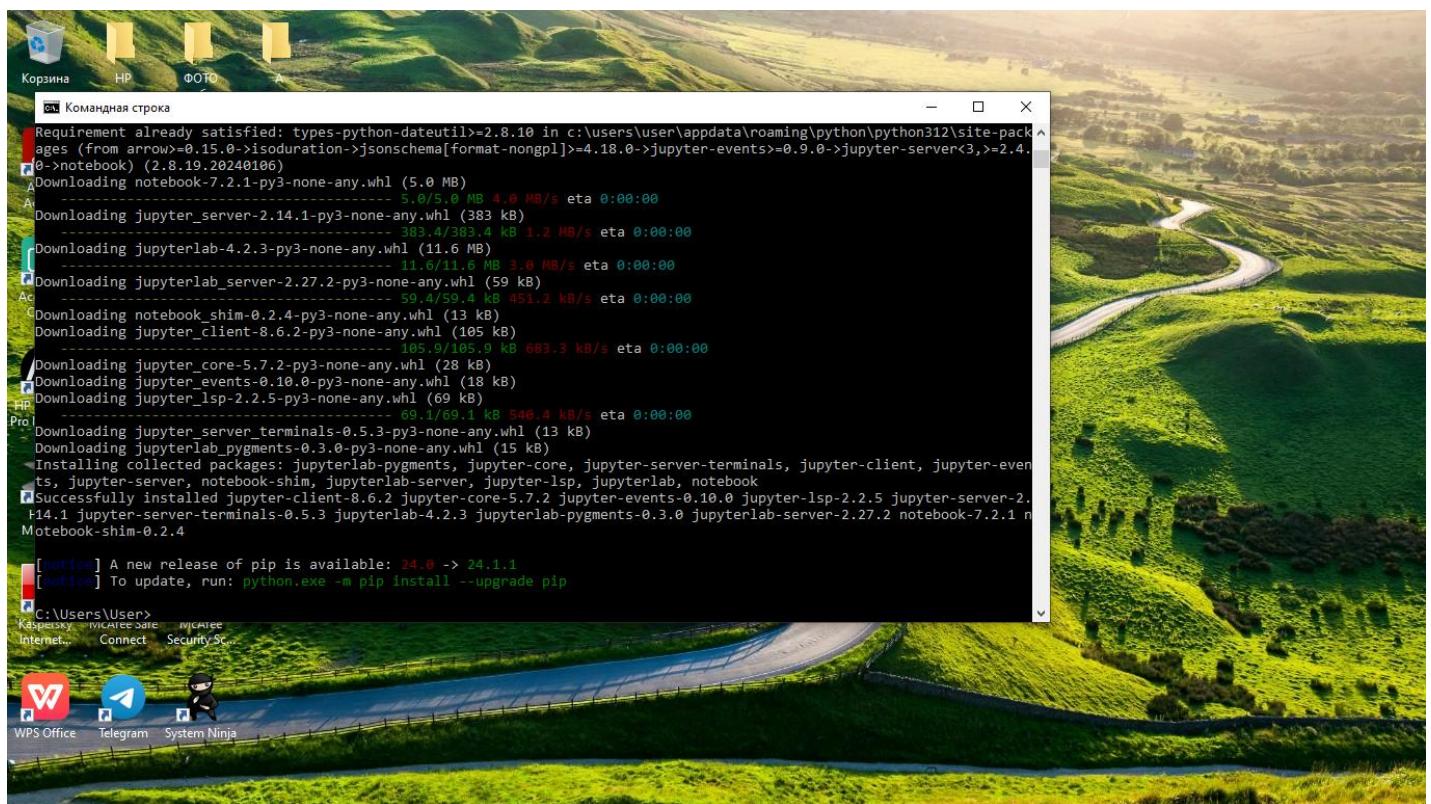
PIP или Python Installs Packages – это пакетный менеджер Python, представляет собой отдельное решение в виде модуля для управления пакетами (установка, удаление и др.) , библиотеками для проектов. Вызывается простым синтаксисом:

pip install package-label (для установки пакета)

pip uninstall package-label (для удаления пакета)

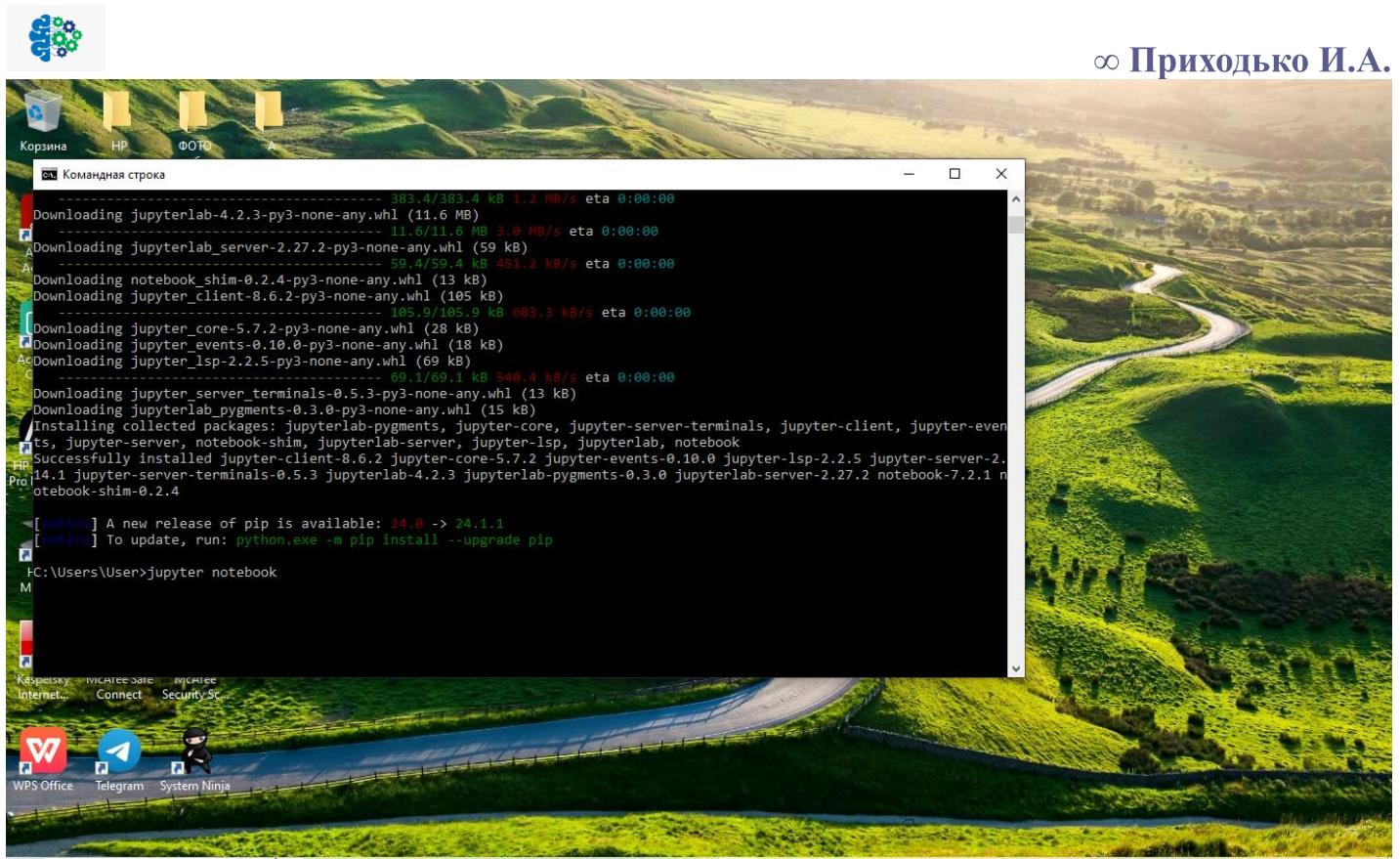
PIP автоматизирует все вышеперечисленные задачи.

После успешной установки Вы получите сообщение:

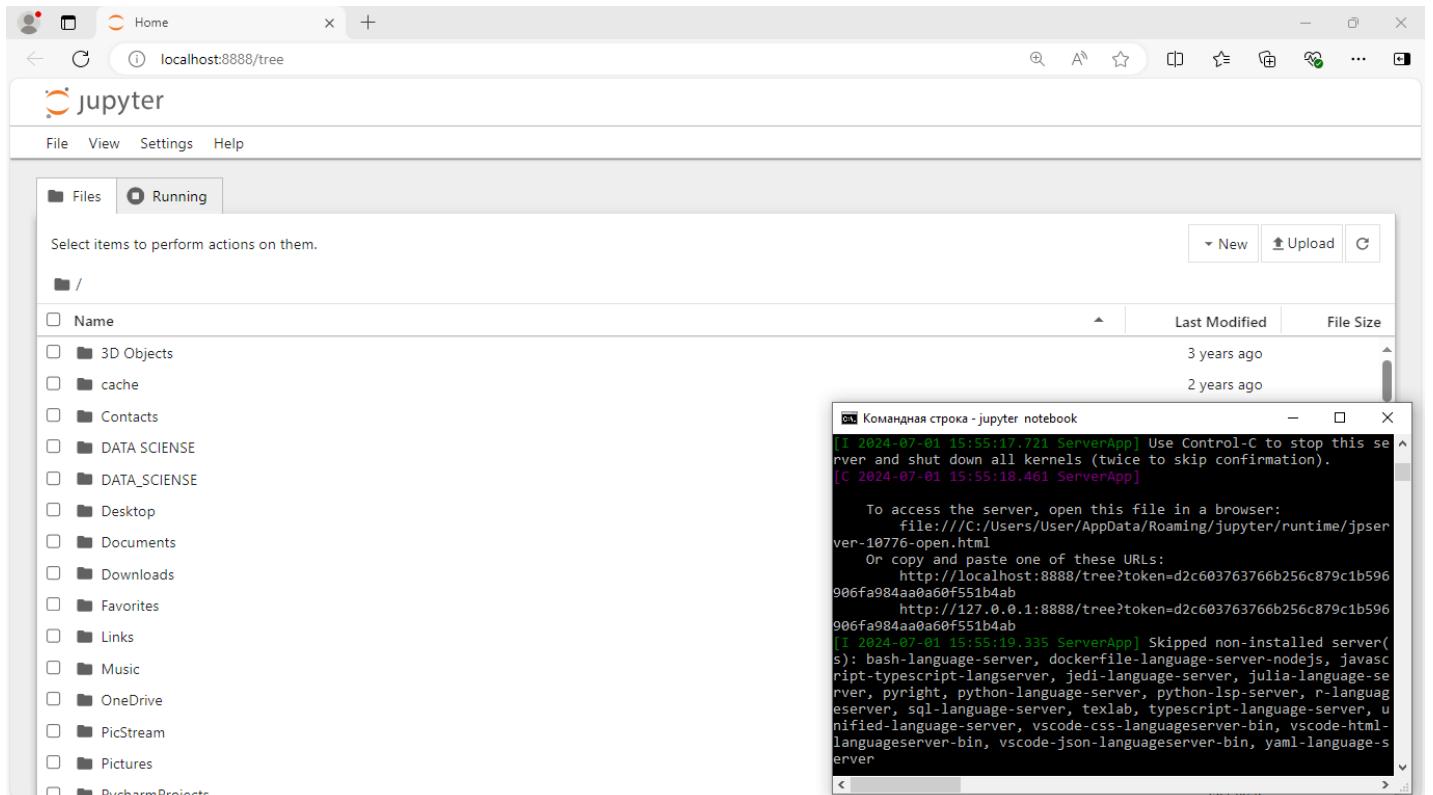


Для запуска Jupyter Notebook согласно официального сайта нужно в командной строке ввести команду:

```
jupyter notebook
```



В результате должен запуститься локальный сервер с открытием браузера:



Если у Вас получился похожий результат, то у Вас все получилось.

Закрыть Jupyter Notebook, а после этого командную строку (консоль) можно, нажав символ X в правом верхнем углу этих окон.



На этом мы заканчиваем сегодняшнее задание.

Что мы сегодня узнали:

- 1.Данных в мире всё больше;**
- 2.Терминология;**
- 3.Использование Python для анализа данных;**
- 4.О чём этот курс?**
- 5.Какого рода данные?**
- 6.Почему именно Python?**
- 7.Необходимые библиотеки для Python;**
- 8.Установка и настройка для Windows.**

Домашнее задание:

- 1.Установите интерпретатор языка **Python** и **Jupyter Notebook** на свое домашнее устройство. Желательно, чтобы оно имело полноценную клавиатуру и манипулятор – мышь (не обязательно, но желательно), а также периодически требуется выход в сеть Интернет;
- 2.Попробуйте выполнить в среде IDLE те же самые команды, что мы с Вами проделали на занятии.

Всем спасибо за внимание.

Желаю Вам успехов в освоении курса!

До встречи на следующем занятии.



2 Lesson.

Lesson topic: 1.2.Jupyter Notebook

1.Что такое Jupyter Notebook и где применяется

В мире Data Science Jupyter Notebook уже несколько лет считается одним из популярных инструментов для анализа данных и быстрого прототипирования.

С Jupyter Notebook аналитические отчёты получаются нагляднее: можно выводить вместе код, изображения, формулы, диаграммы и графики

В Jupyter Notebook две основные части: веб-приложение и ноутбуки — файлы, в которых работают с исходным кодом программы, запускают его и выводят данные в разных форматах. Для экспорта ноутбуков используют два формата — PDF и HTML.

Что можно делать в веб-приложении:

- запускать и редактировать код в браузере;
- показывать результаты расчётов, используя схемы и графики;
- использовать язык разметки Markdown и LaTeX.

Один из плюсов этого инструмента в том, что код можно разделить на кусочки и работать над ними в любом порядке. Например, написать скрипт и сразу посмотреть, как он работает. Остальные фрагменты кода при этом запускать не нужно, результат появляется тут же, под частью кода. Так специалисты по Data Science выводят предварительные результаты исследований, строят графики и диаграммы.

Технически во время работы пользователя с Jupyter Notebook происходят следующие процессы:

- после включения программы Jupyter Notebook на Вашем компьютере запускается локальный сервер с открытием браузера;
- Пользователь подключается к серверу через браузер и создаёт проект, а код отправляется через сервер в ядро (IPython);
- ядро запускает код и отправляет результат через сервер обратно в браузер;
- сервер сохраняет проект в виде файла с расширением .ipynb ;



2.Поддерживаемые языки

Чаще всего с Jupyter-ноутбуками работают на [Python](#), но другие языки программирования тоже поддерживаются, например R, Ruby, Pearl и даже JavaScript — всего около 40 языков. Устанавливать их не нужно — работать с кодом из другого ядра помогают магические команды Python. Они так и называются — magic-команды и бывают двух видов:

- строчные — начинаются с %;
- ячеичные — начинаются с %%.

Чтобы, например, перейти на Ruby, нужно ввести %%ruby.

Опытные пользователи объединяют [SQL](#) и Python внутри блокнотов Jupyter. SQL хорош для извлечения данных и работы со статистикой, а Python подключается, когда нужно эти данные проанализировать. Jupyter Notebook можно подключить к любым базам данных SQL, например MySQL, Postgres, Snowflake, MariaDB, Azure, а затем писать SQL-запросы с помощью нескольких строк кода.

3.Как запустить Jupyter Notebook

Jupyter-ноутбук можно запустить **двумя способами: локально и в облаке**.

1) Установка на локальный компьютер займёт больше времени, но код будет работать быстрее.

На прошлом (первом занятии нашего курса) мы уже установили Jupyter Notebook на свой компьютер.

Этот первый вариант установки подойдёт начинающим. Вы на данном этапе являетесь начинающими специалистами, поэтому мы в ходе изучения настоящего курса будем использовать именно этот вариант. Для опытных специалистов есть альтернативный способ установить Jupyter Notebook на компьютер — загрузить **Anaconda**. Это дистрибутив Python и репозиторий файлов, который используют в основном для анализа данных и машинного обучения. **Jupyter Notebook** и его расширение **JupyterLab** как раз входят в этот дистрибутив.

2) В облаке ноутбук работает медленнее, зато можно сразу садиться и кодить.

Одна из **облачных версий Jupyter Notebook** — программа **Google Colab**.



Её часто используют в учебных программах по машинному обучению. Она работает в любом браузере и не требует специальных настроек.

4.Как работать с Jupyter Notebook

Разберём несколько задач, которые помогут разобраться в работе с Jupyter Notebook.

Чтобы начать новый проект, нужно запустить Jupyter Notebook.

На своем компьютере нажмите одновременно клавиши

Windows+R (появится окно)

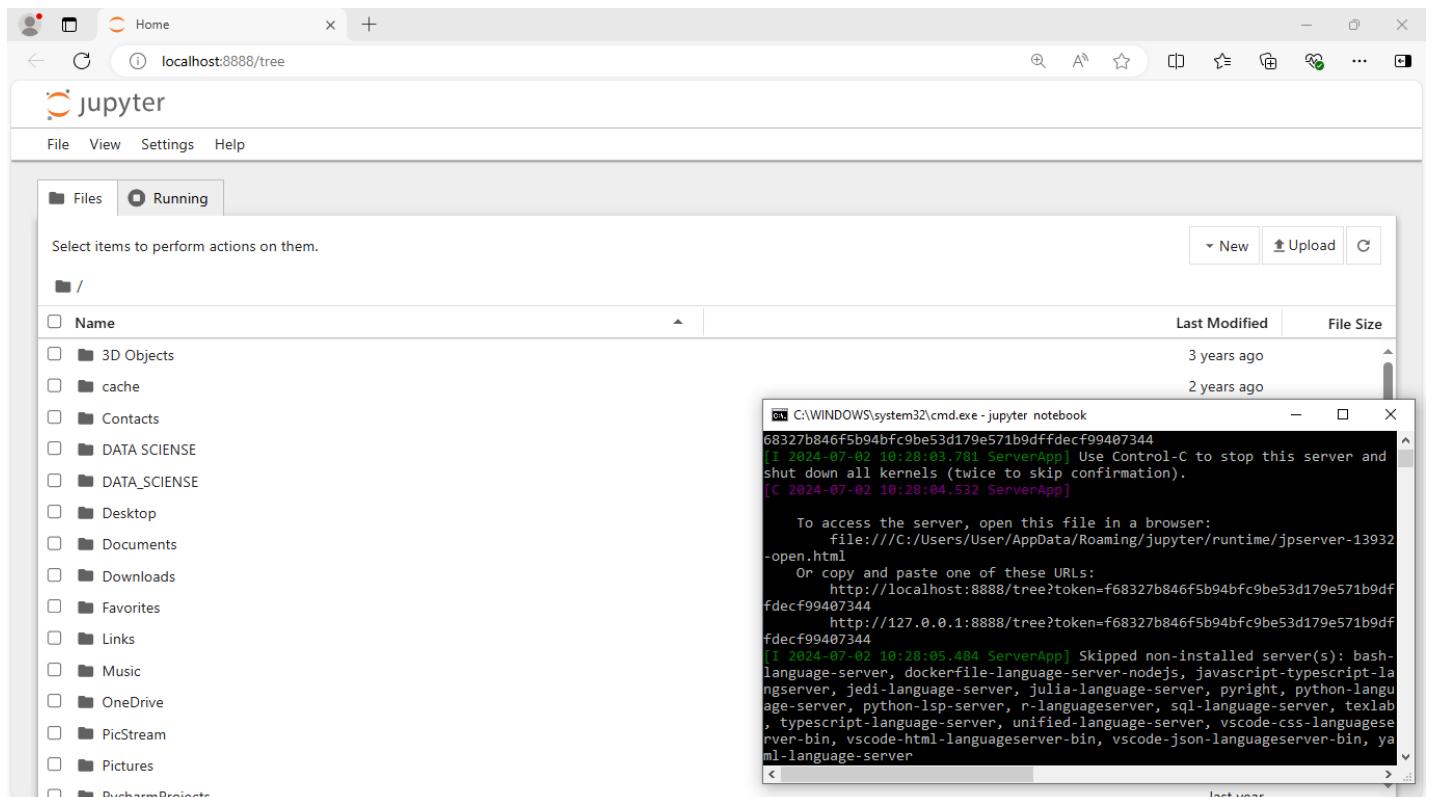
Внутри этого окна наберите команду

cmd (появится Консольное окно - Командная строка).

Для запуска Jupyter Notebook согласно официального сайта нужно в командной строке (консоли) ввести команду:

```
jupyter notebook
```

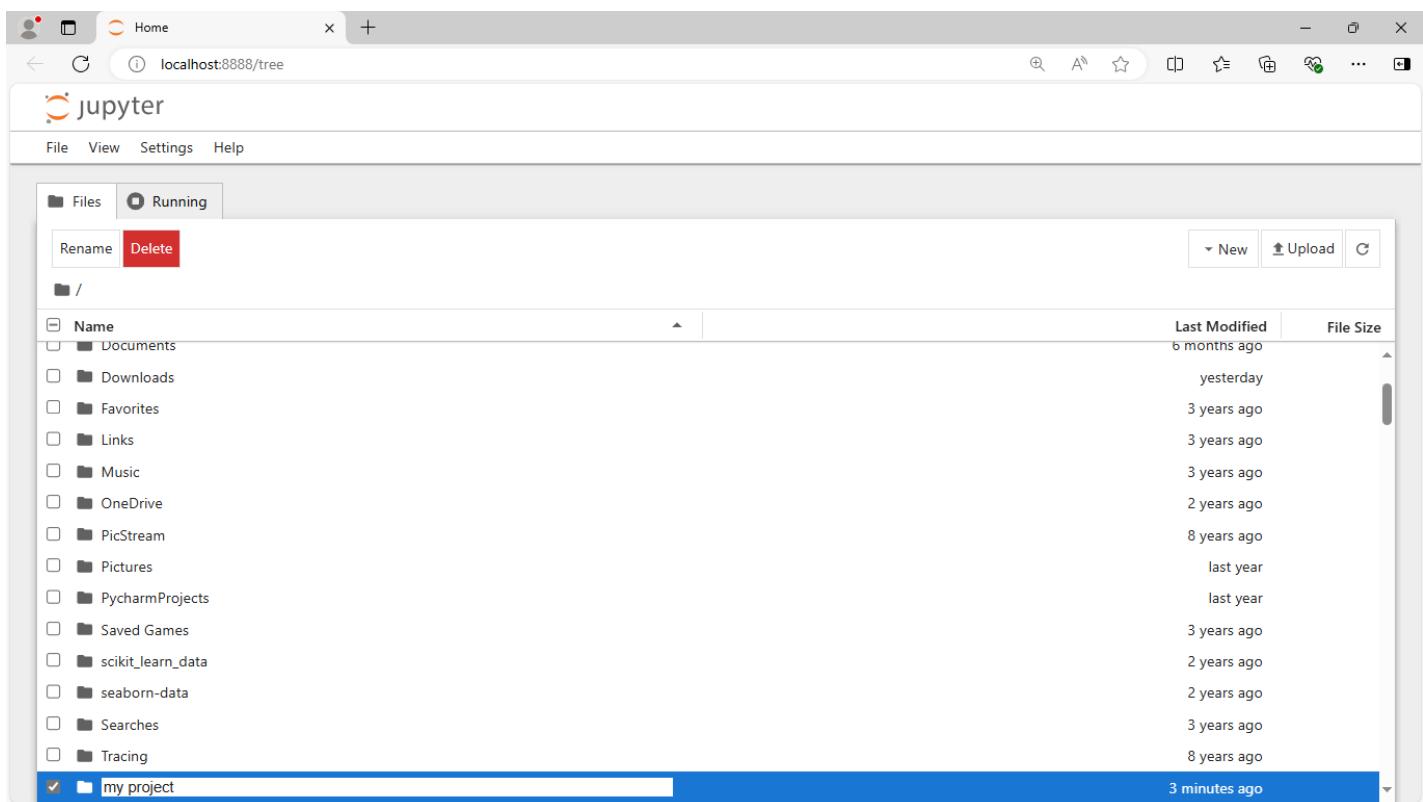
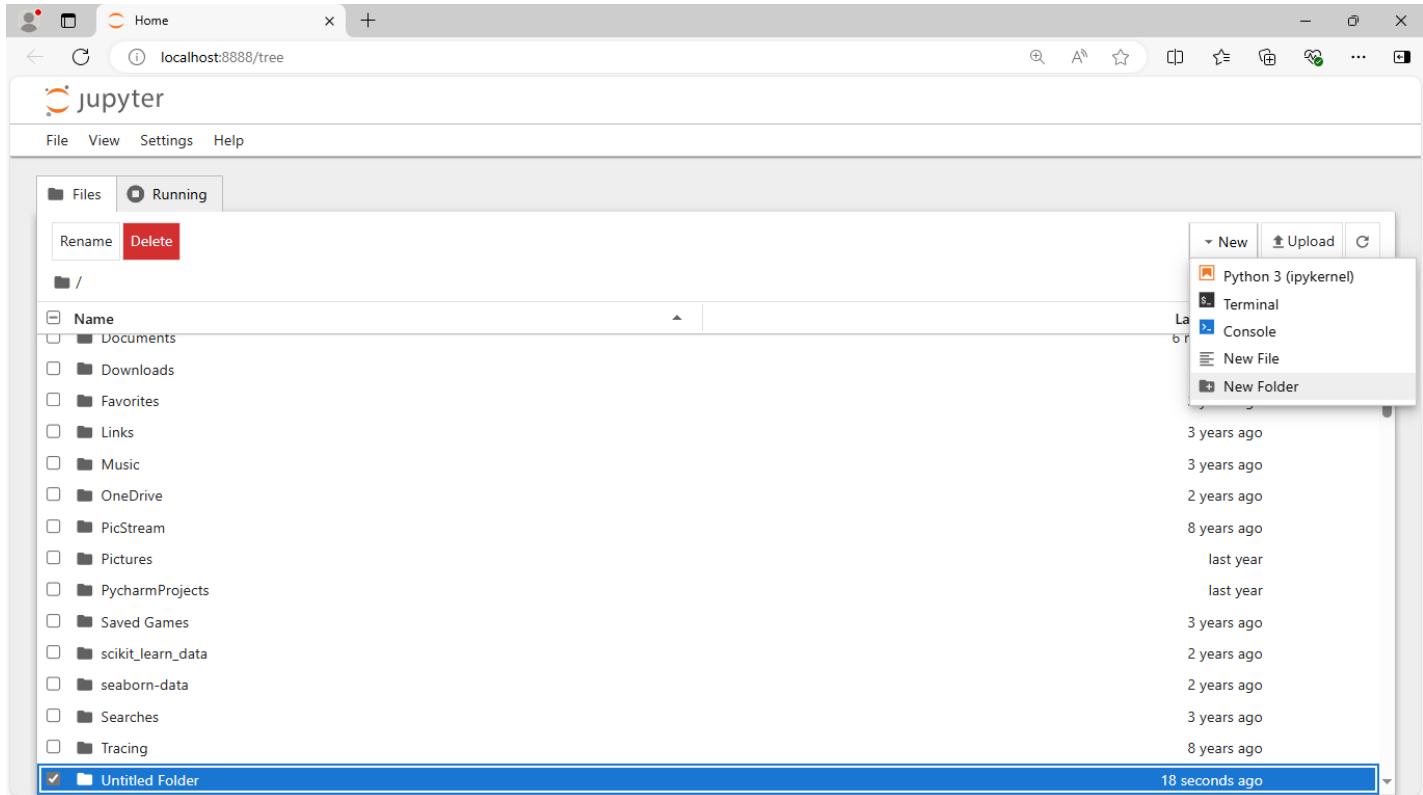
В результате должен запуститься локальный сервер с открытием браузера:



Далее работая с Jupyter Notebook, мы не будем обращаться к консоли. А все основные наши действия мы будем осуществлять в браузере.



Чтобы начать новый проект, нужно запустить Jupyter Notebook (мы уже это сделали) и создать папку для проектов. Затем нажать **New** в правой части экрана и выбрать в списке меню **Folder**. Новая папка автоматически будет названа **Untitled folder**. Чтобы назвать её по-другому, нужно поставить галочку напротив имени и нажать **Rename**

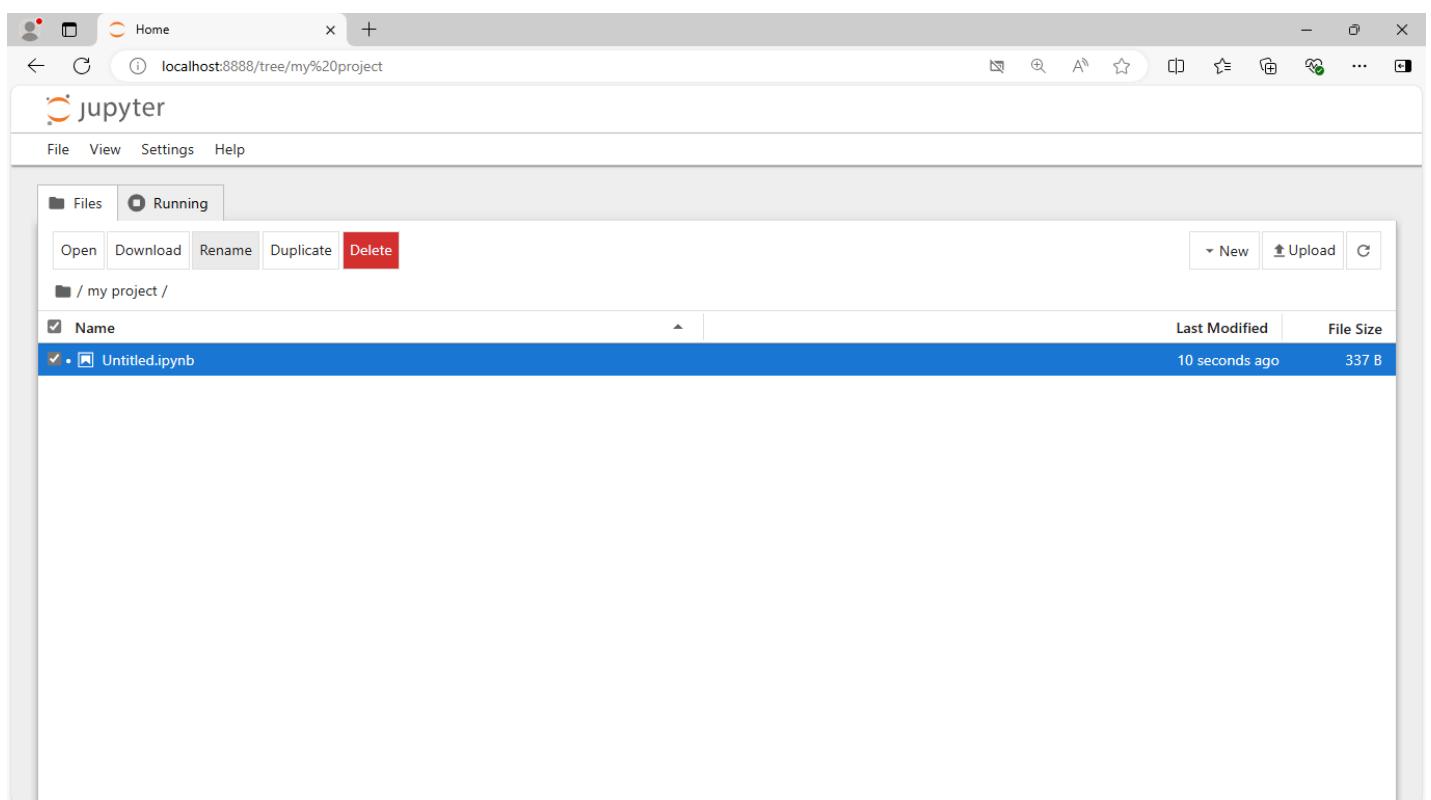
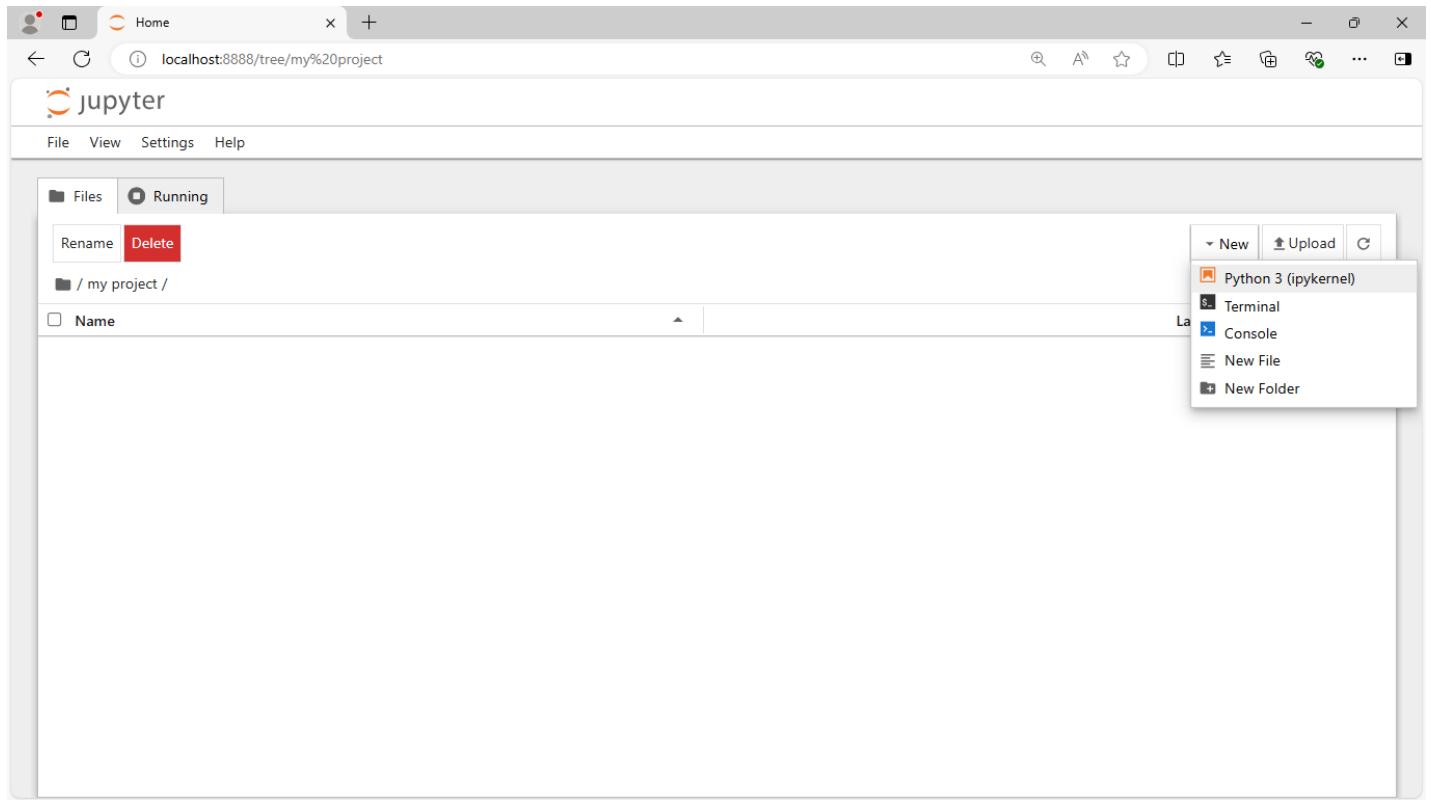




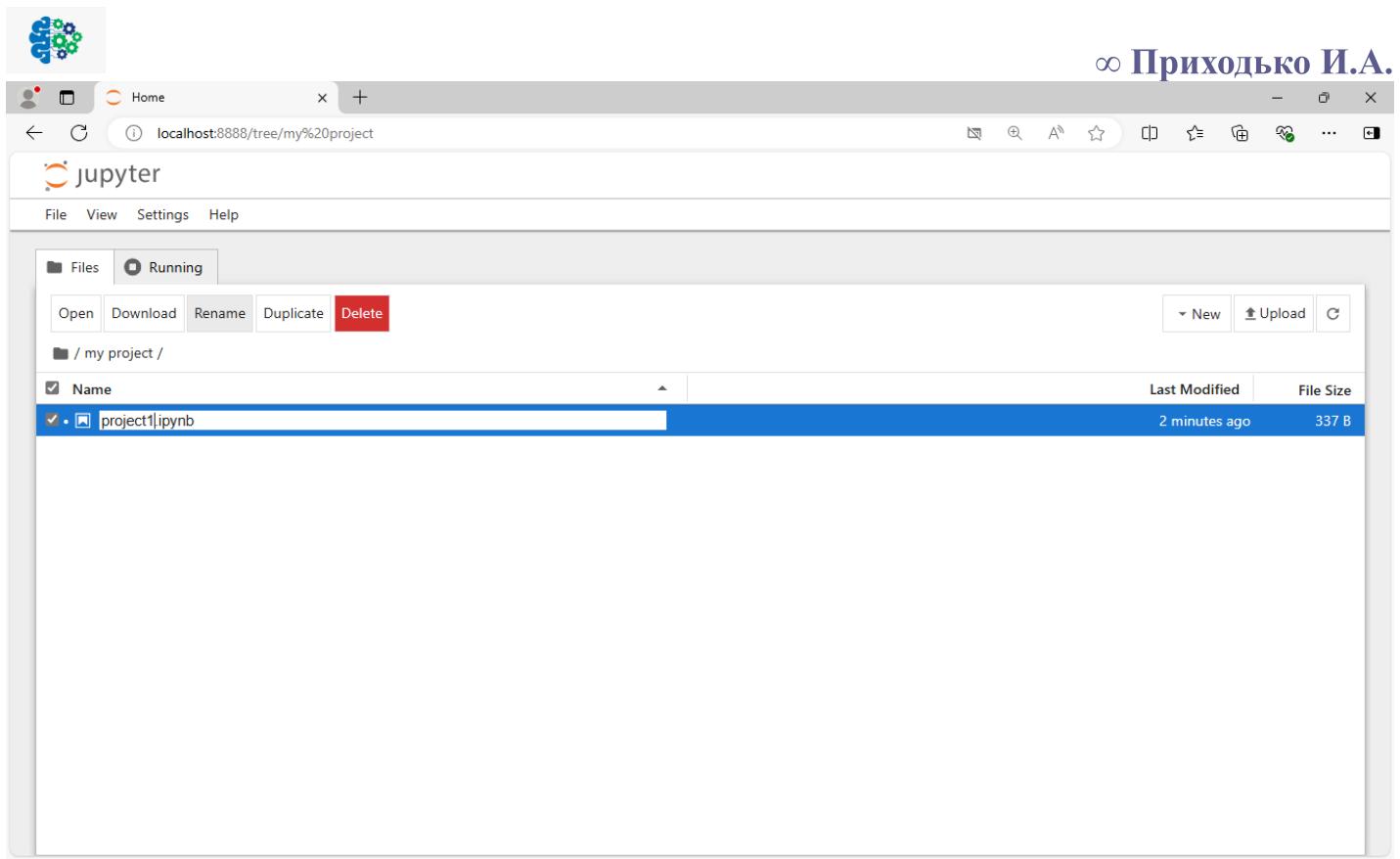
∞ Приходько И.А.

Кликаем на Вашу новую папку (оказываемся внутри этой папки).

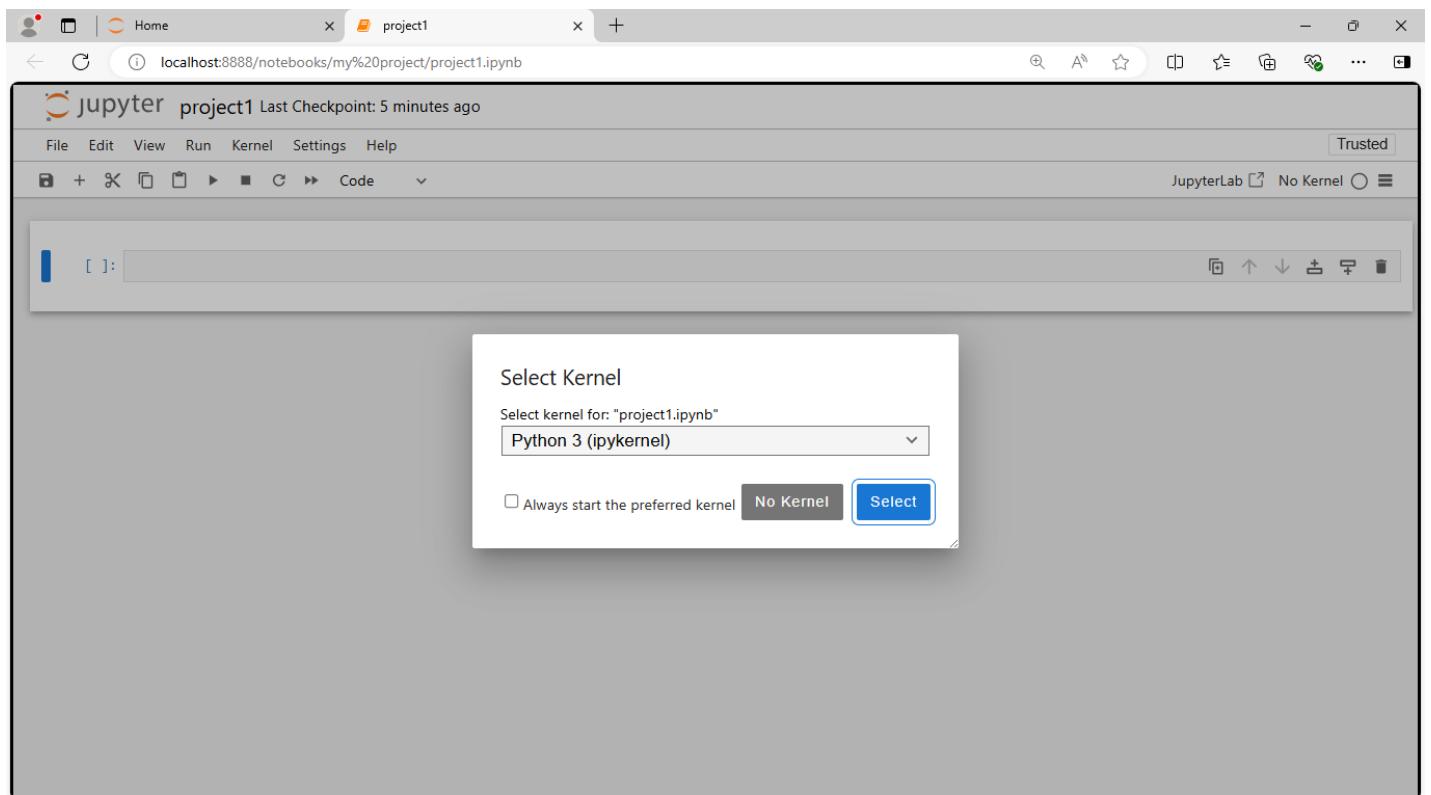
Чтобы создать свой новый ноутбук, нужно снова использовать **New** и выбрать **Python 3**.



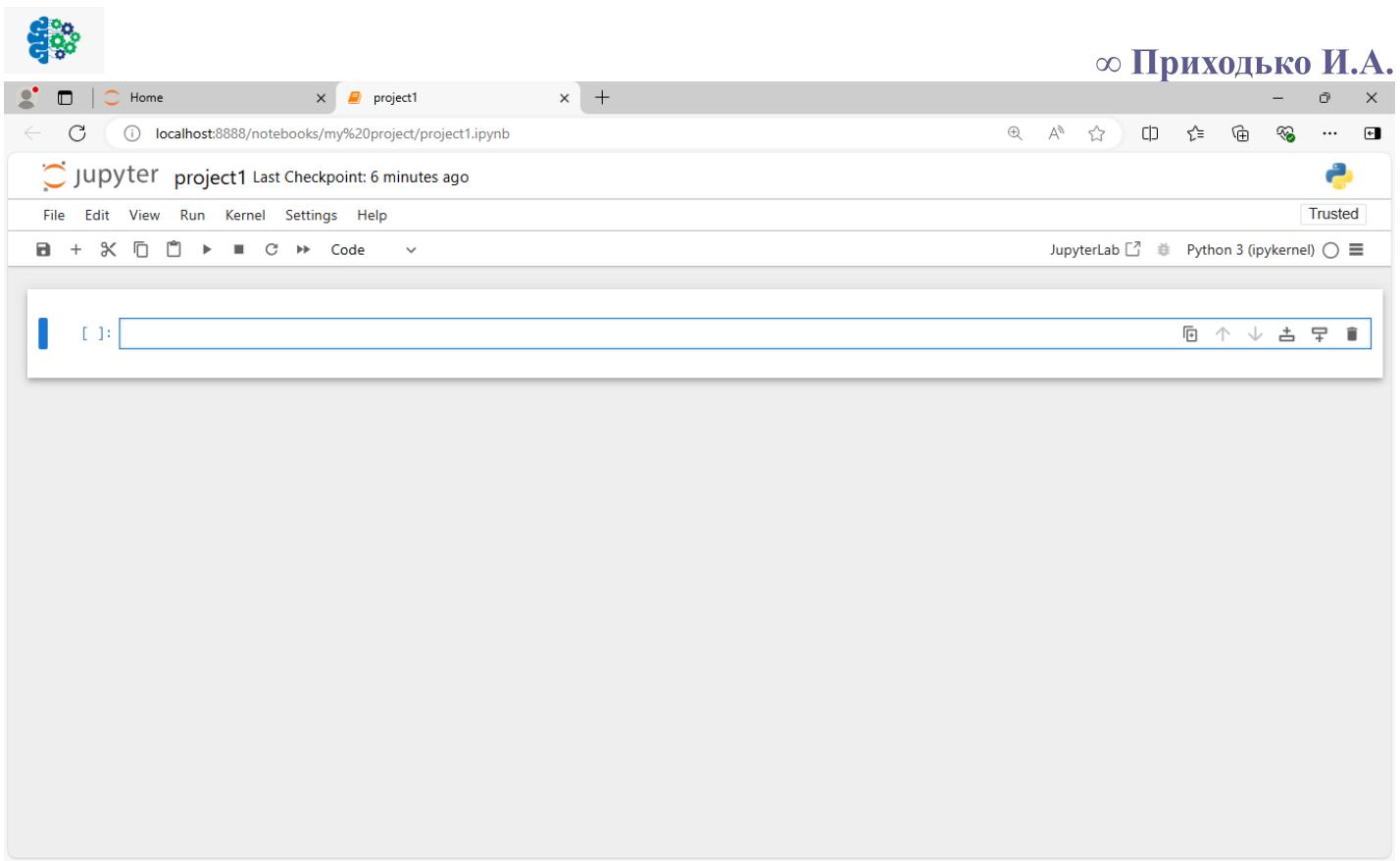
Даём название своему новому ноутбуку: нужно поставить галочку напротив имени и нажать **Rename**



Дважды кликаем по своему новому ноутбуку и Ваш новый ноутбук должен открыться:

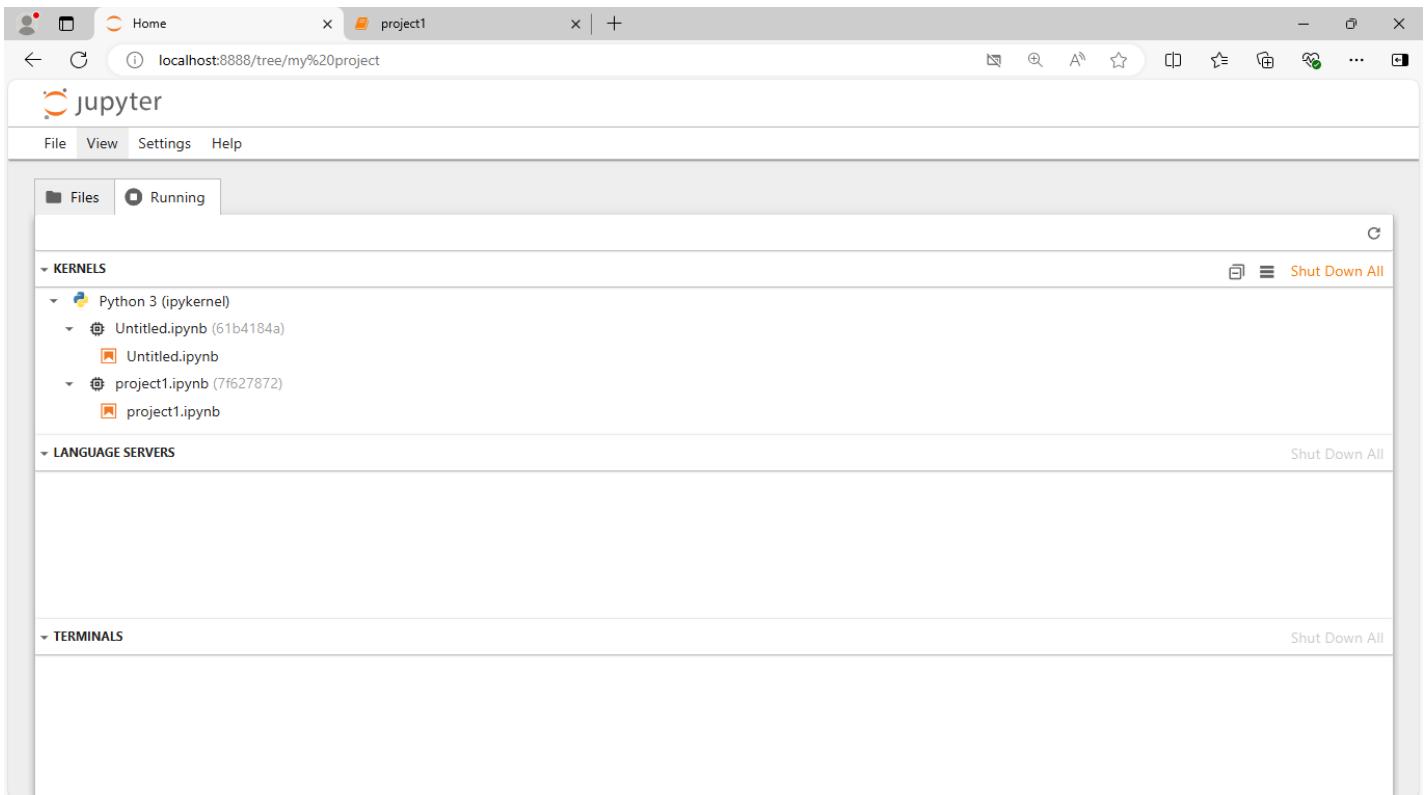


Выбираем **Python 3 (ipykernel)** и жмём **Select**:

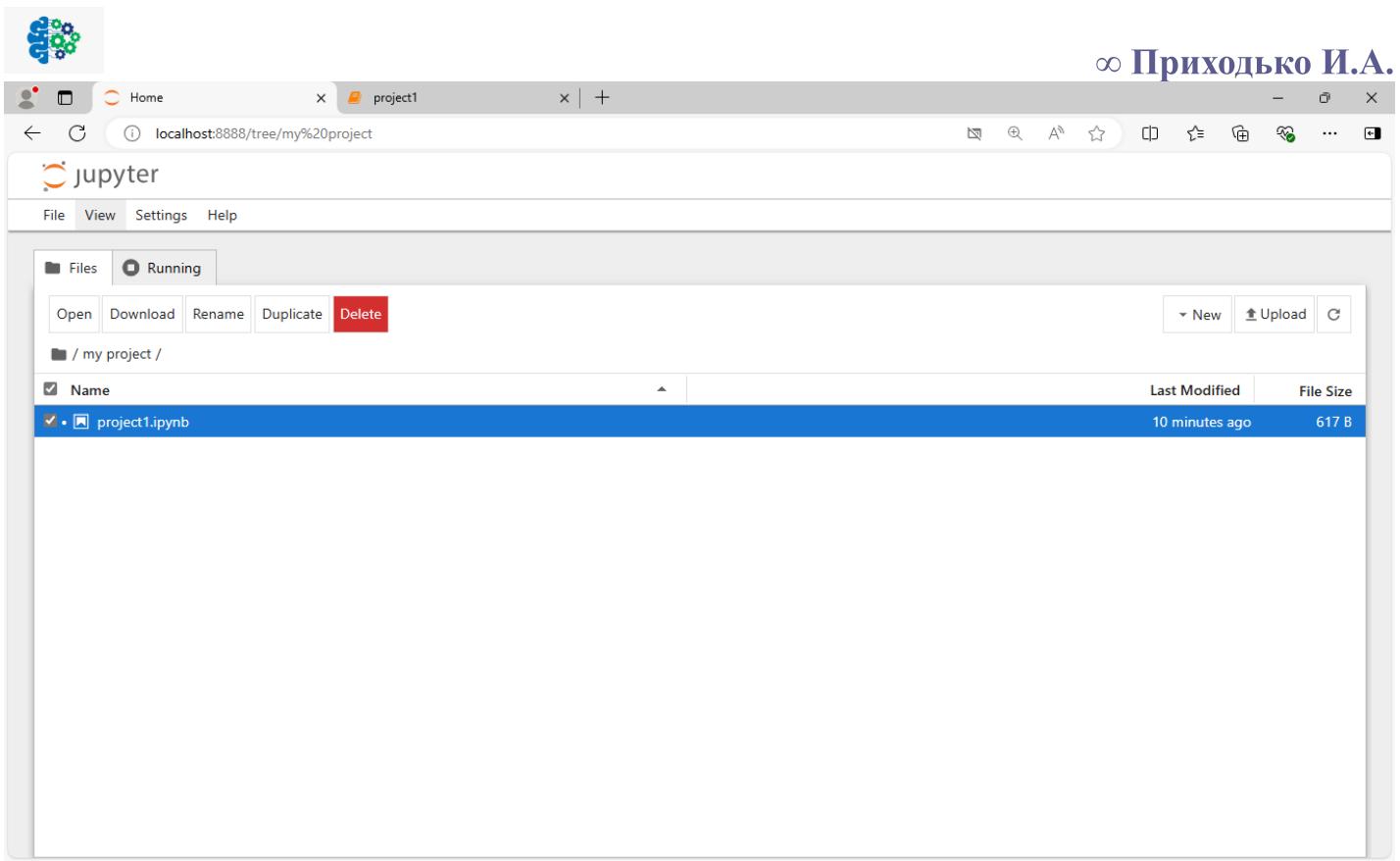


Вы создали свой первый новый ноутбук. Поздравляю Вас!

Каждый ноутбук использует новую вкладку — открывайте хоть сотню проектов одновременно. Чтобы найти все рабочие блокноты, нажмите вкладку «Running»:



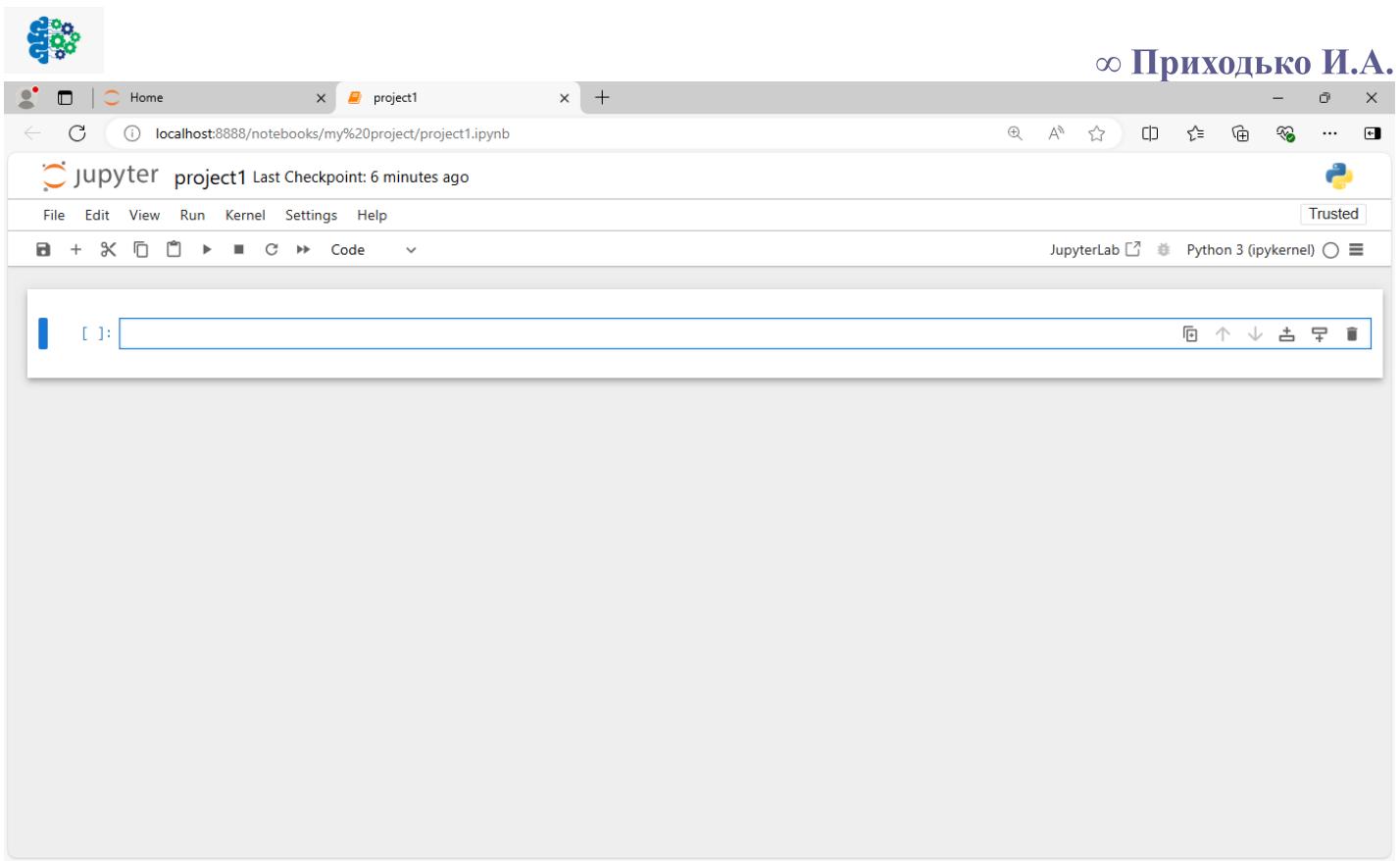
Чтобы найти все рабочие файлы в текущей папке, нажмите вкладку «Files»:



Вернёмся в Ваш новый созданный ноутбук.

Интерфейс любого ноутбука состоит из:

- заголовка,
- меню,
- инструментов,
- ячеек.



Сейчас в Вашем ноутбуке единственная **cell** (ячейка).

- Основным блоком в Jupyter Notebook является **ячейка**. Их существует несколько типов:

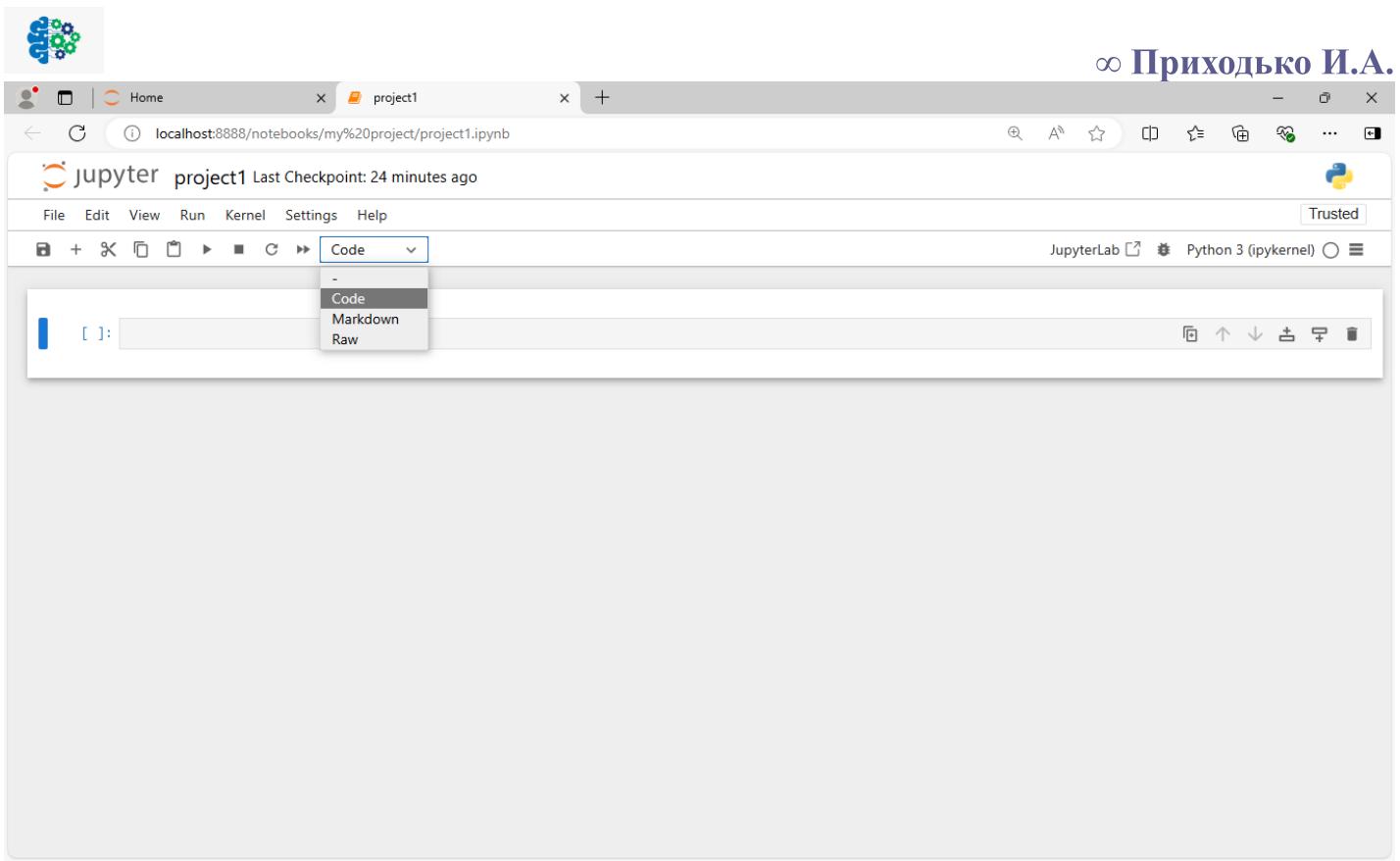
Code: ячейка с кодом на Python или другом языке, который поддерживается вашим Jupyter Notebook.

Markdown: ячейка с разметкой Markdown для создания структурированного текста.

Raw: ячейка, содержащая текст без форматирования.

Теперь можно перейти на следующий уровень и попробовать решить простую задачу — сложить 2 и 5.

Для этого нужно выставить свойство **Code** (режим работы):

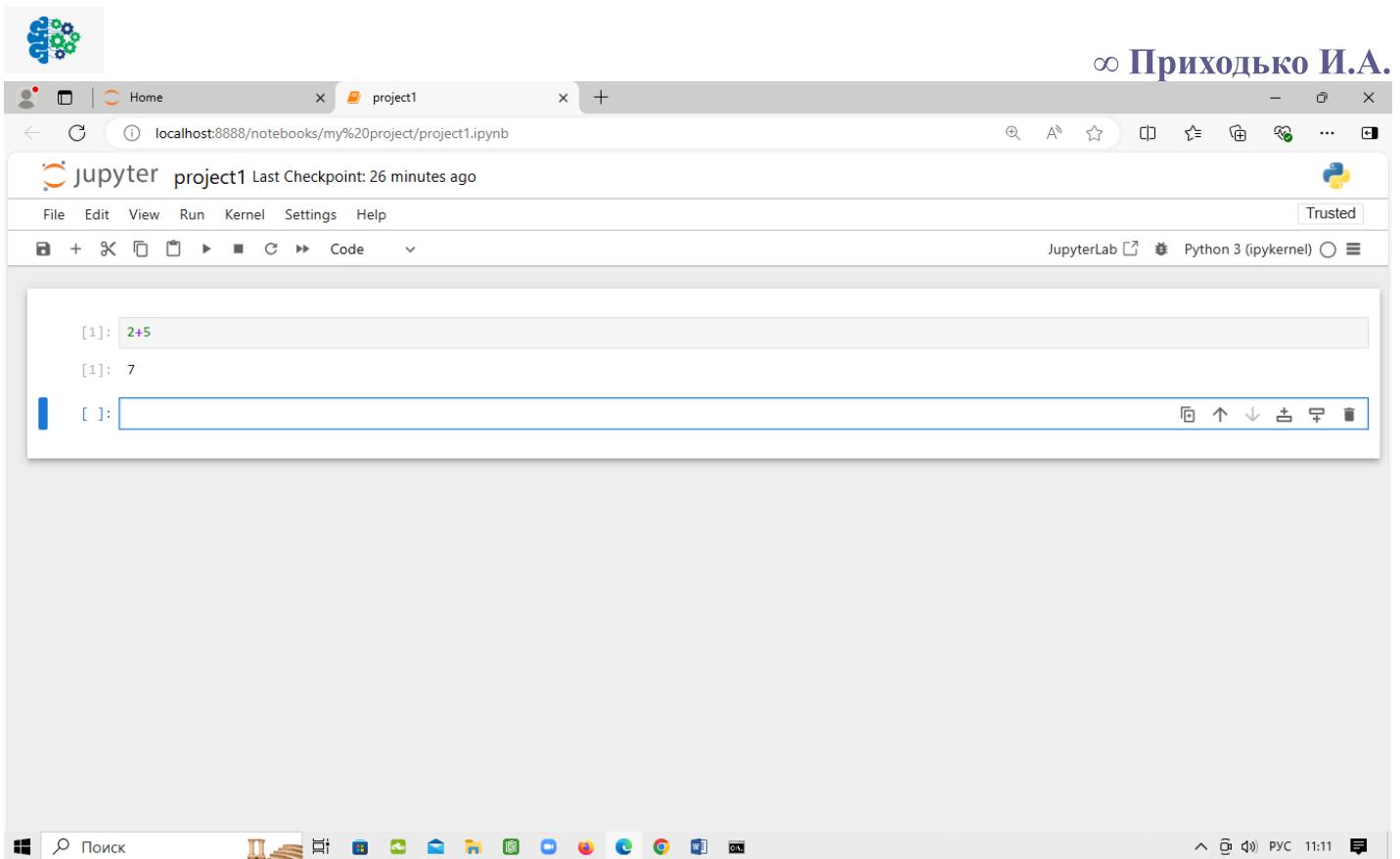


Далее ввести в ячейке **2 + 5** и нажать

Ctrl + Enter (код выполнит интерпретатор Python, но курсор останется в ячейке)

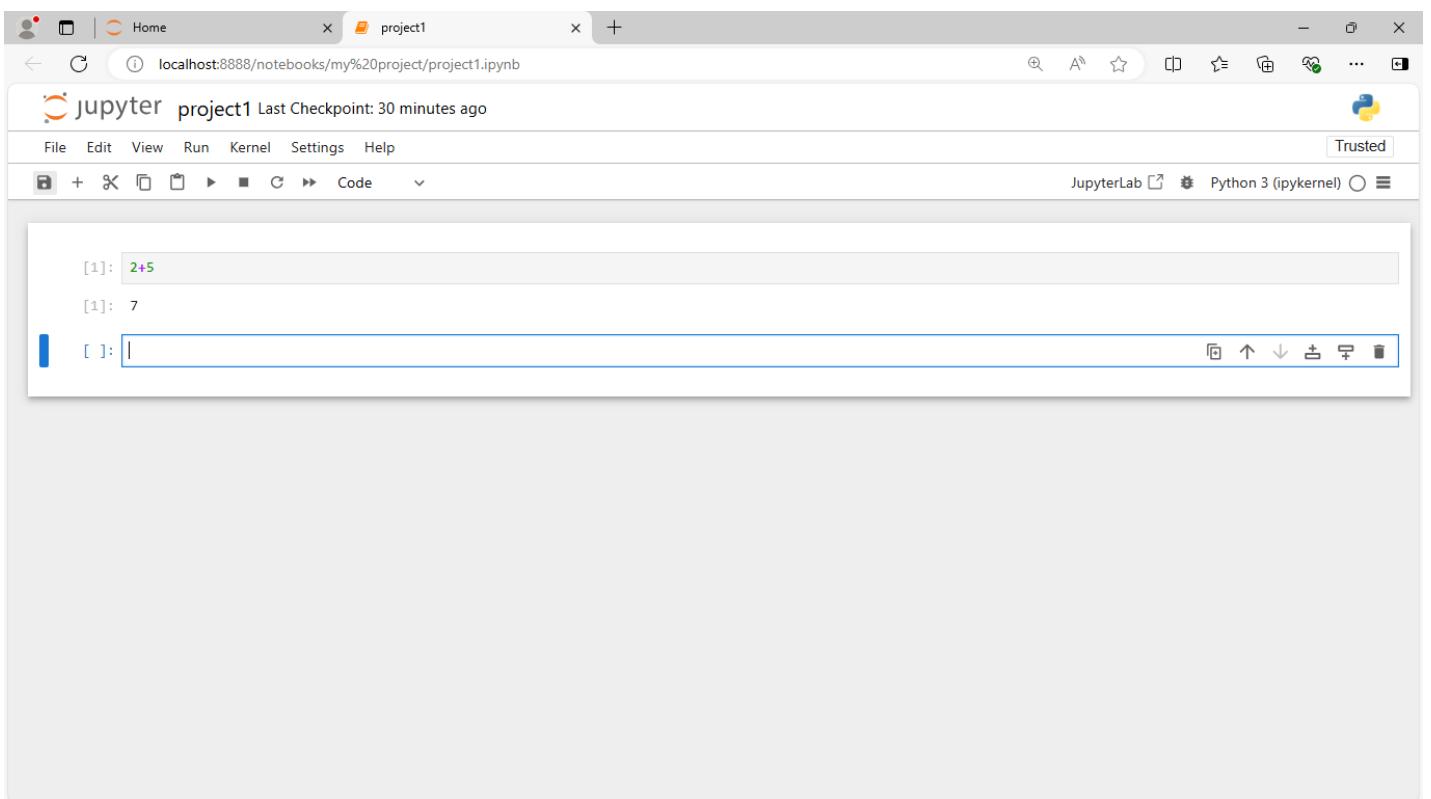
или

Shift + Enter (равносильно нажатию кнопки **Run** или инструмента ►, будет выполнен код и появится ещё одна ячейка).



Чтобы запустить код в облачной версии Jupyter, достаточно нажать Run → Run Selected Cells.

Сохраните изменения в Вашем ноутбуке, нажав на **дискету**:



Вообще каждая ячейка может выполнять инструкции (команды) языка Python.



Давайте выполним еще несколько других команд языка Python.

```
print('Hello, World!')
```

The screenshot shows a Jupyter Notebook interface with the title "project1". The code cell [1] contains the expression `2+5`, which evaluates to `7`. The code cell [2] contains the command `print('Hello, World!')`, which outputs `Hello, World!`. The interface includes a toolbar with various icons and a status bar at the bottom.

```
# Вывод: 'Hello, Mark. You are 14.'
```

```
name = "Mark"
```

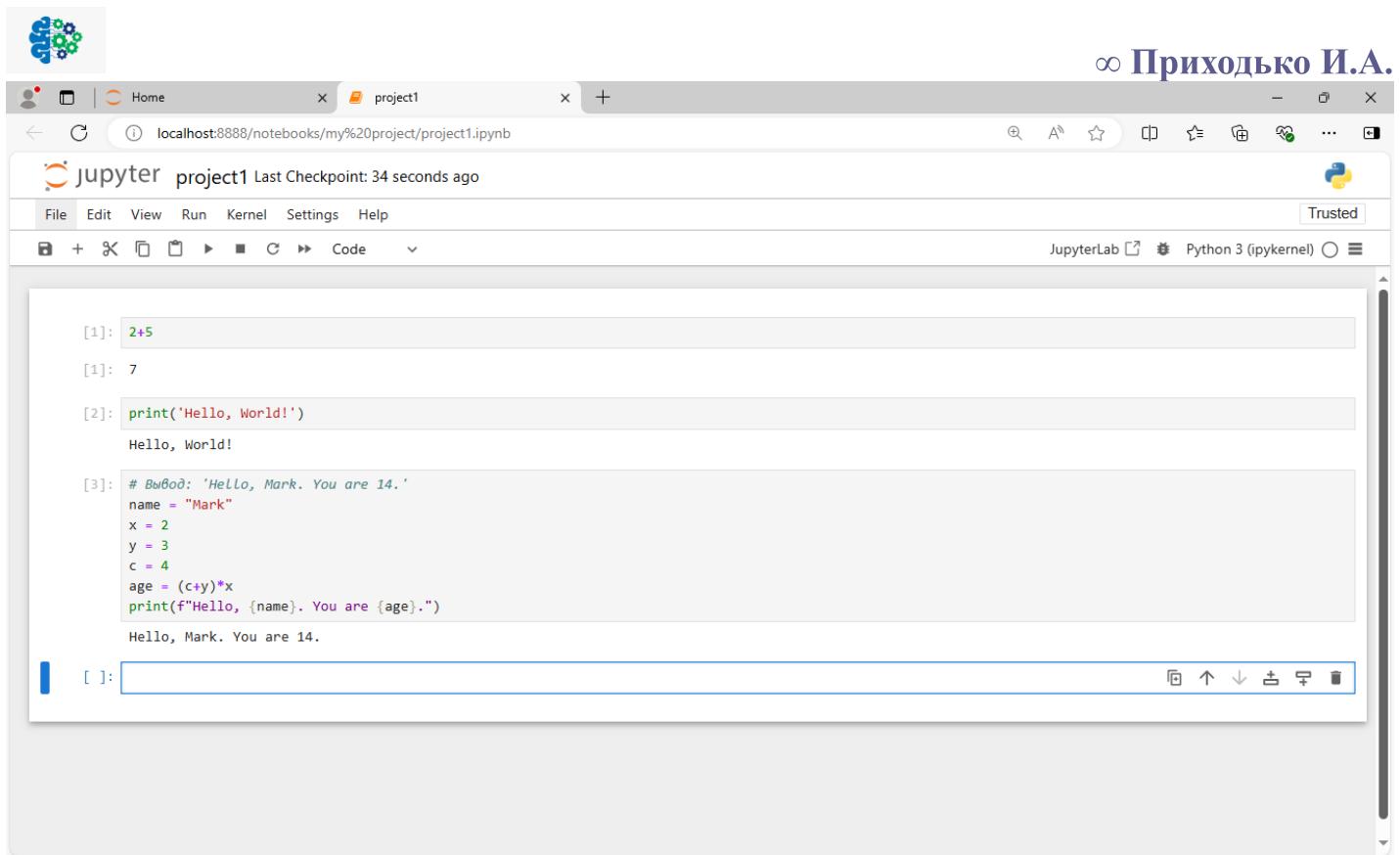
```
x = 2
```

```
y = 3
```

```
c = 4
```

```
age = (c+y)*x
```

```
print(f"Hello, {name}. You are {age}.")
```



The screenshot shows a Jupyter Notebook interface with the title "project1". The notebook contains three cells:

- [1]: `2+5` → Output: 7
- [2]: `print('Hello, World!')` → Output: Hello, World!
- [3]: `# Вывод: 'Hello, Mark. You are 14.'`
`name = "Mark"`
`x = 2`
`y = 3`
`c = 4`
`age = (c+y)*x`
`print(f"Hello, {name}. You are {age}.")`
→ Output: Hello, Mark. You are 14.

Попробуем решить задачу посложнее — нарисовать график. Если ноутбук только установили, графики не получится вывести в его рабочее поле. Чтобы они отображались, нужны библиотеки **pandas** и **Matplotlib**.

pandas используют для анализа данных в форме таблиц, а **Matplotlib** помогает строить графики и диаграммы.

Добавить библиотеки можно с помощью команд в ячейке:

!pip install pandas

!pip install matplotlib

В результате выполнения этих команд Jupyter Notebook или установит на Ваш компьютер (для этого нужен выход Вашего компьютера в интернет), или сообщит Вам, что эти библиотеки уже установлены на Ваш компьютер:



∞ Приходько И.А.

localhost:8888/notebooks/my%20project/project1.ipynb

jupyter project1 Last Checkpoint: 18 minutes ago

File Edit View Run Kernel Settings Help Trusted JupyterLab Python 3 (ipykernel) Code

```
[4]: !pip install pandas
Requirement already satisfied: pandas in c:\users\user\appdata\roaming\python\python312\site-packages (2.2.1)
Requirement already satisfied: numpy<2,>=1.26.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\appdata\roaming\python\python312\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\appdata\roaming\python\python312\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\appdata\roaming\python\python312\site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\roaming\python\python312\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

[5]: !pip install matplotlib
Requirement already satisfied: matplotlib in c:\python312\lib\site-packages (3.8.3)

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyParsing>=2.3.1 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\roaming\python\python312\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

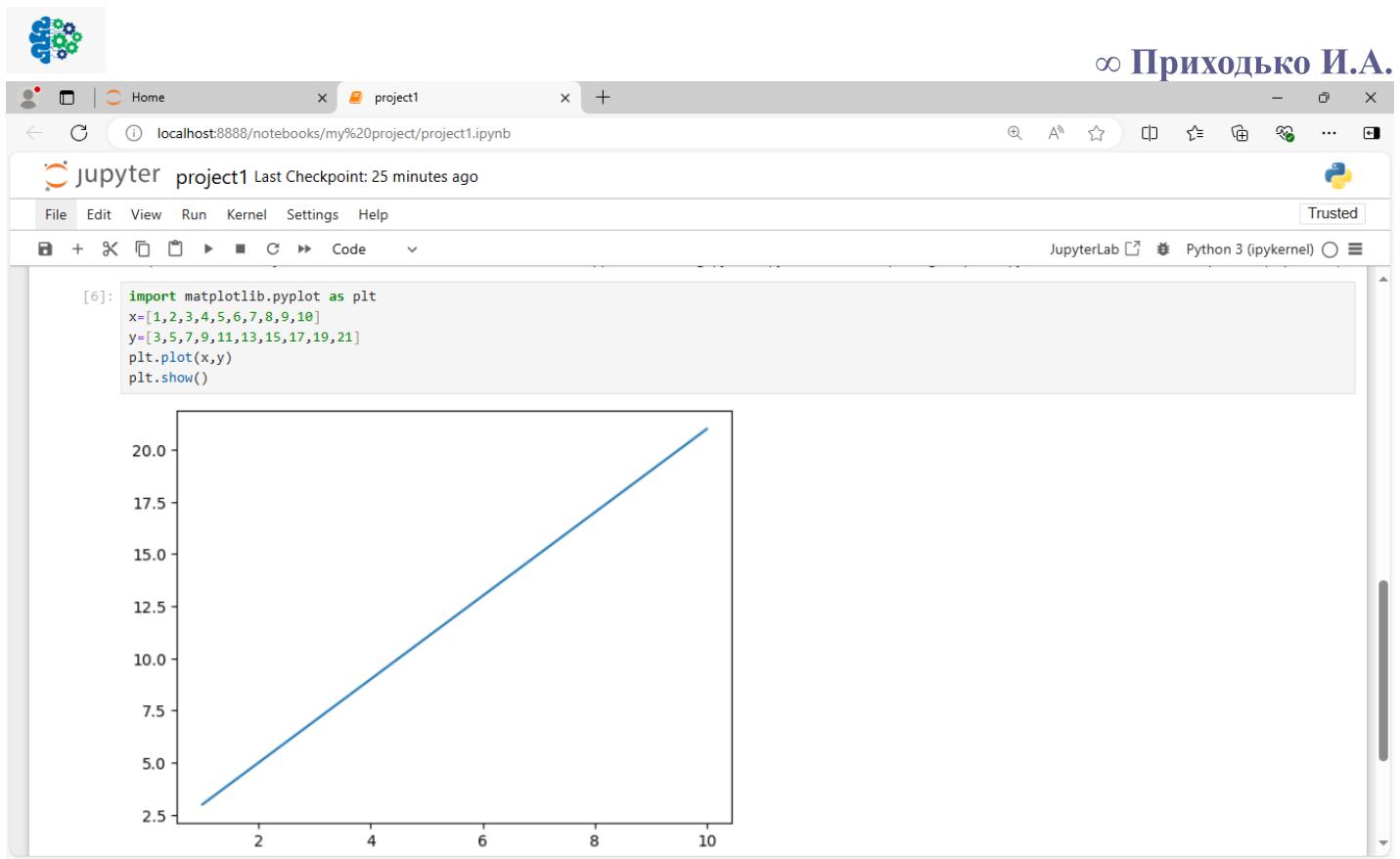
Давайте запустим код для простого линейного графика:

```
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7,8,9,10]
y=[3,5,7,9,11,13,15,17,19,21]
plt.plot(x,y)
plt.show()
```

Первая строка импортирует graphing-библиотеки **pyplot** из **Matplotlib API**. Третья и четвёртая строки — оси x и y. Чтобы построить график, нужно вызвать метод **plot()**, а для отображения — метод **show()**.

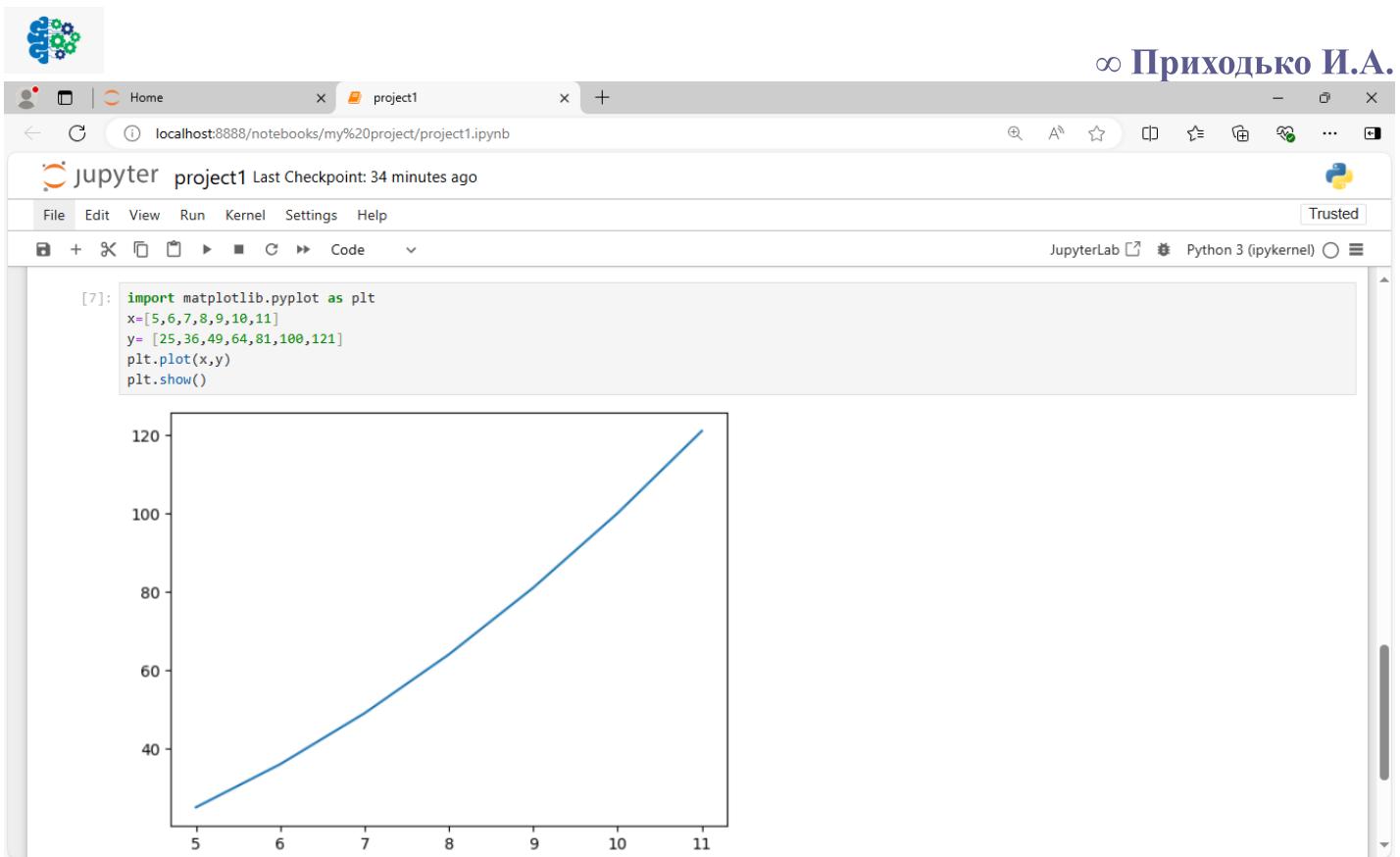
 **API** (аббр. от англ. **application programming interface**, дословно — **программный интерфейс приложения**) — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими.

API упрощает процесс программирования при создании приложений, абстрагируя базовую реализацию и предоставляя только объекты или действия, необходимые разработчику.



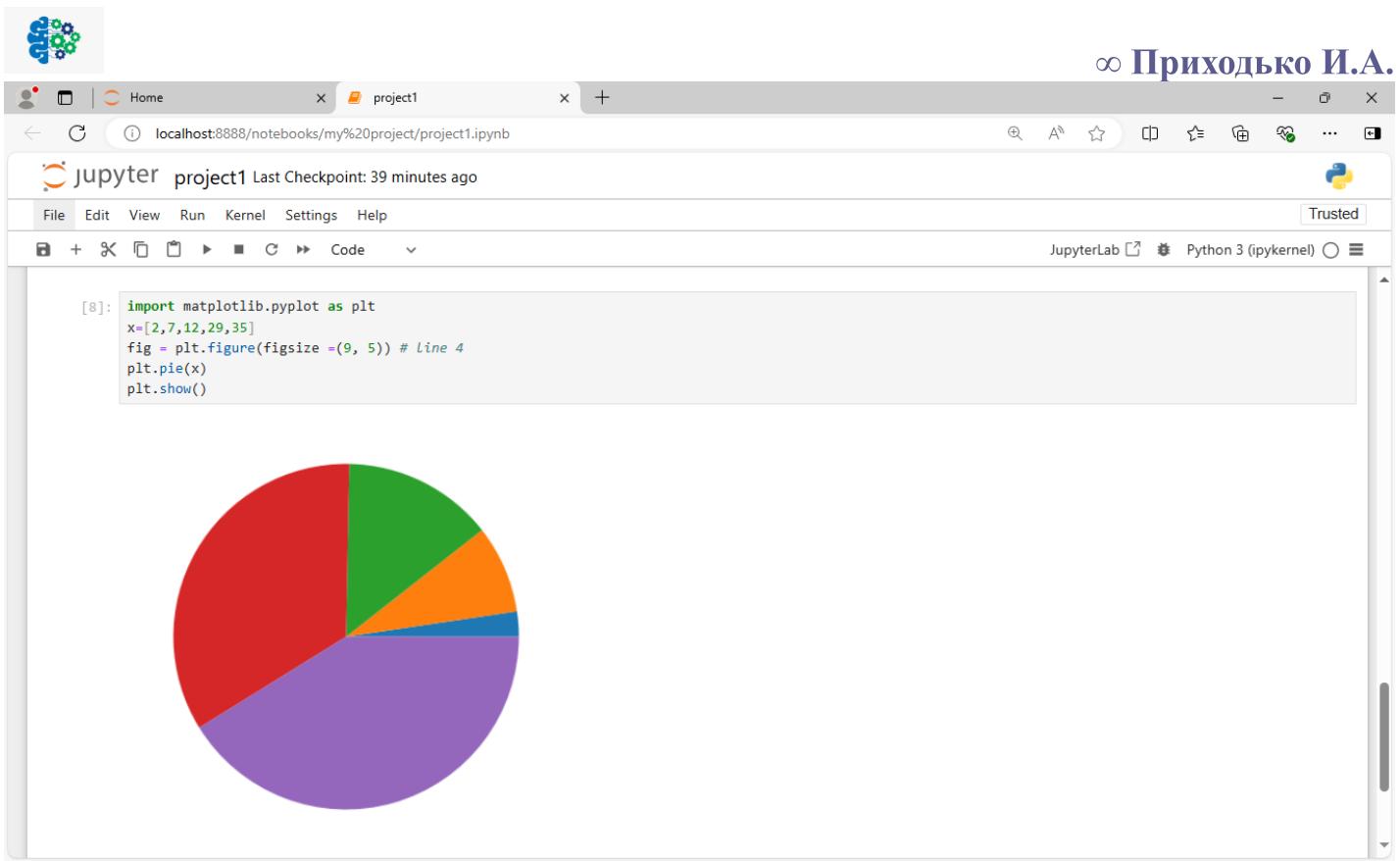
Чтобы из прямой сделать кривую, нужно подставить другие значения для оси у:

```
import matplotlib.pyplot as plt  
x=[5,6,7,8,9,10,11]  
y= [25,36,49,64,81,100,121]  
plt.plot(x,y)  
plt.show()
```



Matplotlib подходит не только для графиков, но и для диаграмм. Здесь используется **figsize** — метод, который позволяет менять размеры графика. Все размеры в графиках на Jupyter-ноутбуке настраивает библиотека, поэтому визуализировать данные в нужном формате бывает сложно. Метод **figsize** пригодится, когда нужно, например, сделать какую-нибудь фигуру больше или меньше, изменив соотношение сторон. Официальные документы **pandas** советуют использовать соотношение сторон 1, но можно устанавливать любые. Например, чтобы построить диаграмму:

```
import matplotlib.pyplot as plt
x=[2,7,12,29,35]
fig = plt.figure(figsize =(9, 5)) # line 4
plt.pie(x)
plt.show()
```

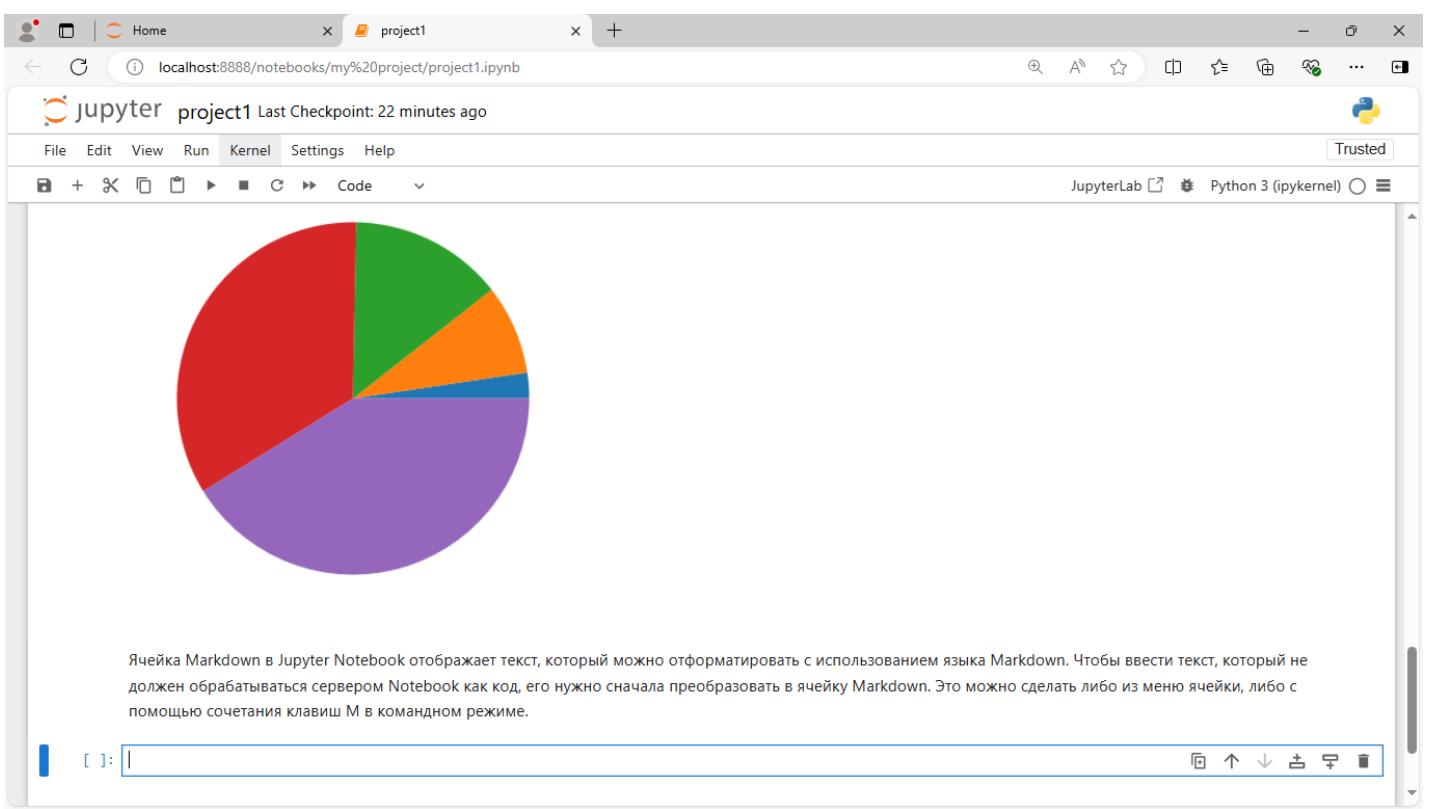
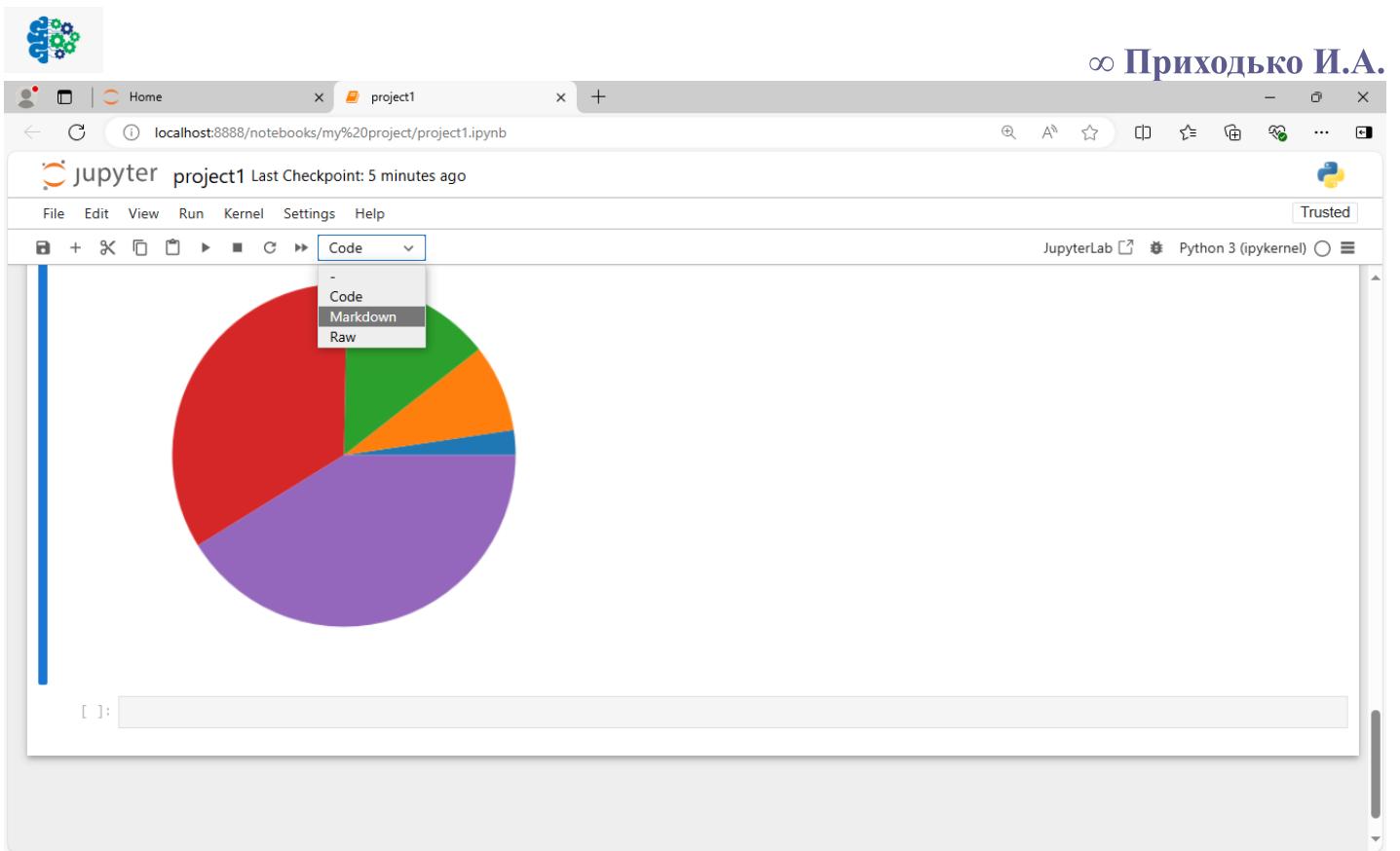


До сих пор мы рассматривали режим работы Jupyter Notebook в режиме **Code**.

Но есть еще режим **Markdown**.

- ❸ Ячейка **Markdown** в Jupyter Notebook отображает текст, который можно отформатировать с использованием языка **Markdown**.

Чтобы ввести текст, который не должен обрабатываться сервером Notebook как код, его нужно сначала преобразовать в ячейку **Markdown**. Это можно сделать либо из меню ячейки, либо с помощью сочетания клавиш **M** в командном режиме.



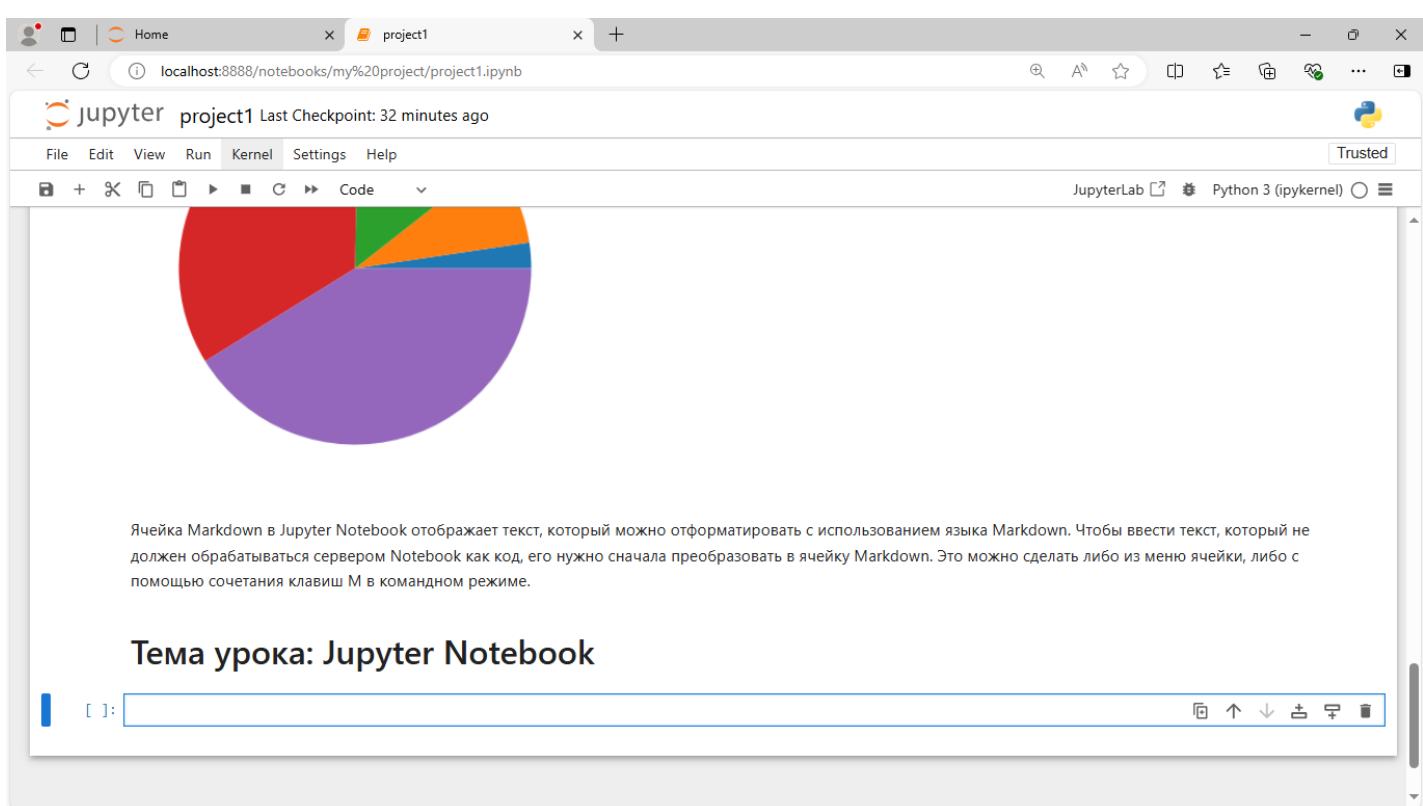
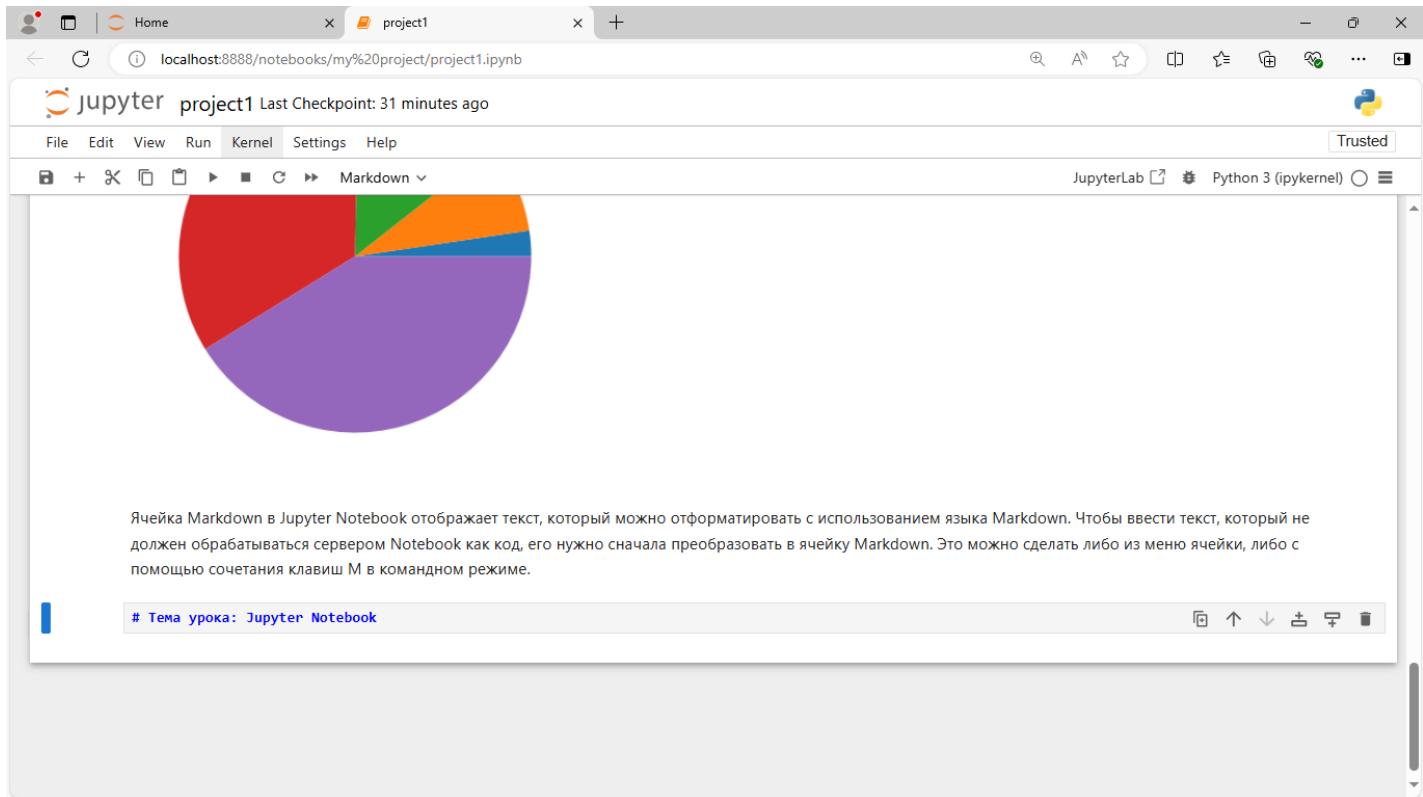
Режим ячейки Markdown предоставляет Вам отличную возможность сопровождать Ваш код и вычисления разъясняющим текстом.

Вы можете текст (или часть текста) превратить в заголовок. Для этого Вам нужно находиться не в той ячейке, где находится текст, а кликнуть левее этой ячейки (у синей



вертикальной полосы) и нажать клавишу 1 (самый большой шрифт заголовка), 2,3,4,5,6 (самый маленький шрифт заголовка):

при нажатии 1:



при нажатии 6:

The screenshot shows a Jupyter Notebook interface. At the top, there's a header bar with a logo, a 'Home' button, and a tab labeled 'project1'. Below the header is a toolbar with various icons. The main area contains a pie chart with four segments (purple, red, orange, green) and a text cell below it. The text cell contains the following text:

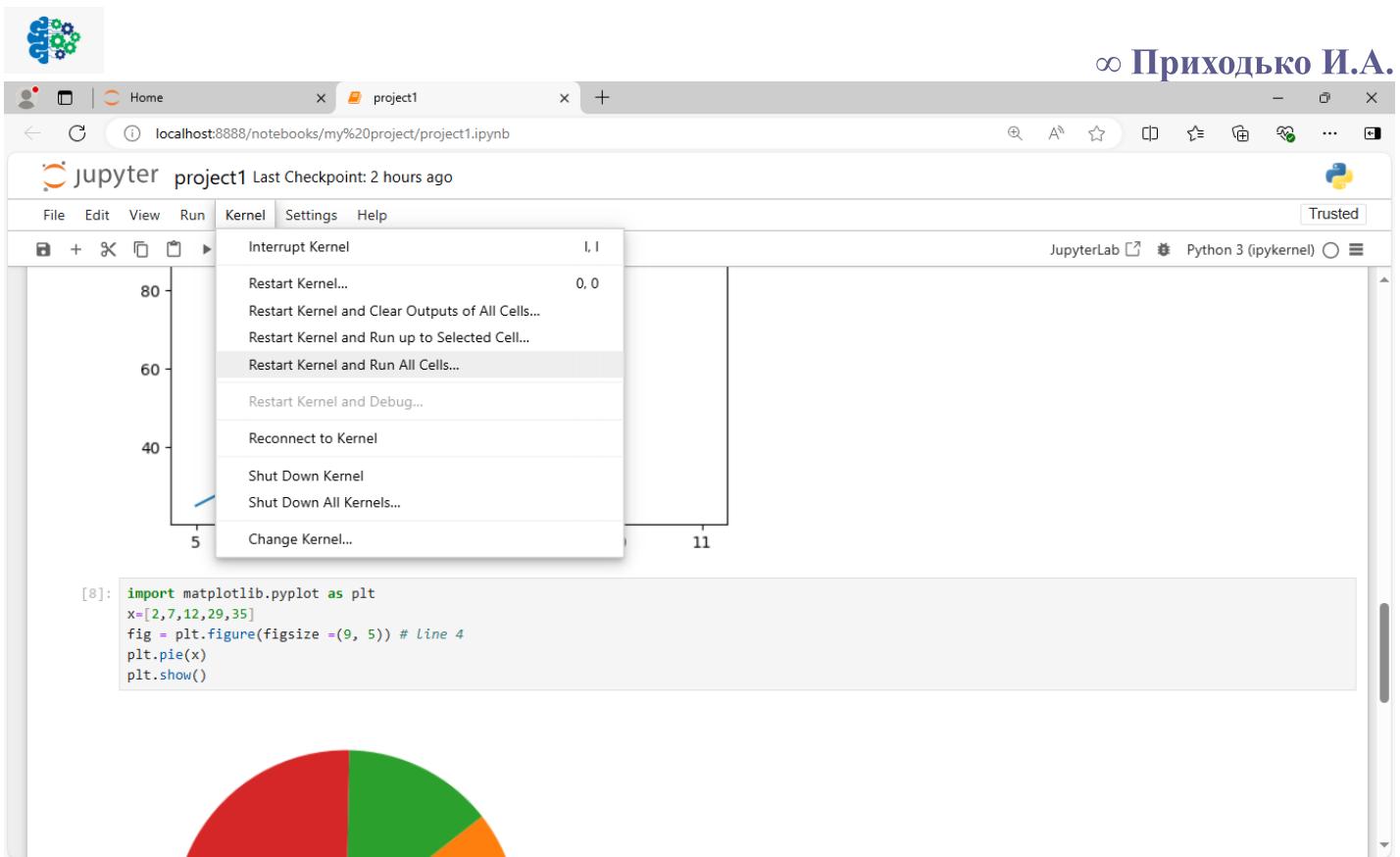
Ячейка Markdown в Jupyter Notebook отображает текст, который можно отформатировать с использованием языка Markdown. Чтобы ввести текст, который не должен обрабатываться сервером Notebook как код, его нужно сначала преобразовать в ячейку Markdown. Это можно сделать либо из меню ячейки, либо с помощью сочетания клавиш M в командном режиме.

Тема урока: Jupyter Notebook

This screenshot shows the same Jupyter Notebook interface as the previous one, but the active cell is now a code cell, indicated by the 'Code' tab in the toolbar. The pie chart remains in the same position. The text cell below it still contains the same explanatory text about Markdown cells.

Тема урока: Jupyter Notebook

Давайте обратим внимание на то, что все ячейки вначале не имеют числового номера слева от себя, но после выполнения этой ячейки она получает номер – это порядковый номер выполнения инструкций в нашем ноутбуке:



5. Shortcuts в Jupyter Notebook (полный список в Help-Keyboard Shortcuts)

<i>Клавиша / сочетание клавиш</i>	<i>Результат</i>
A	добавляет новую ячейку выше выбранной
B	добавляет новую ячейку ниже выбранной
DD (2 раза нажимаем D)	удаляет ячейку
X	удаляет ячейку, копирует содержимое удаляемой ячейки в буфер
C	копирует ячейку
V	вставляет скопированную ячейку под текущей ячейкой
Ctrl+A	копирует несколько выделенных ячеек сразу
Shift	зажимаем и двигаемся вниз, выделяя то количество ячеек, которое Вы хотите объединить
Shift + M	все выделенные при этом ячейки объединяются
Shift + Enter (равносильно нажатию кнопки Run или инструмента ►)	запускаем выделенные ячейки, но при этом мы увидим только последнюю команду выполненную. При этом мы переходим к следующей ячейке
Ctrl + Enter	запускаем выделенные ячейки, но при этом мы увидим только последнюю



	команду выполненную. При этом мы не переходим к следующей ячейке
M	Превращает выбранную ячейку в Markdown
Y	Превращает выбранную ячейку в Code

Получить исчерпывающую информацию по shortcuts можно, нажав **Help – Show Keyboard Shortcuts:**

The screenshot shows a Jupyter Notebook interface with a pie chart in the main area. A context menu is open over the chart, with the 'Help' option highlighted. A sub-menu is displayed under 'Help' with the following options: 'About Jupyter Notebook', 'Show Keyboard Shortcuts...', 'Launch Jupyter Notebook File Browser', 'Jupyter Reference', 'JupyterLab FAQ', 'JupyterLab Reference', 'Markdown Reference', 'About Jupyter', 'Markdown Reference', and 'Documentation'. The 'Show Keyboard Shortcuts...' option is underlined, indicating it is selected. The status bar at the bottom of the window shows the text 'Ячейка Markdown в Jupyter Notebook отображает текст'.

Тема урока: Jupyter Notebook

Ячейка Markdown в Jupyter Notebook отображает текст

The screenshot shows a Jupyter Notebook interface with a pie chart in the main area. A 'Keyboard Shortcuts' pop-up window is open, listing various keyboard shortcuts for navigation and editing. The text in the main notebook cell is:

Ячейка Markdown в Jupyter Notebook отображает текст, который можно отформатировать с использованием языка Markdown. Чтобы ввести текст, который не должен обрабатываться сервером Notebook как код, его нужно сначала преобразовать в ячейку Markdown. Это можно сделать либо из меню ячейки, либо с помощью сочетания клавиш M в командном режиме.

Тема урока: Jupyter Notebook

Ячейка Markdown в Jupyter Notebook отображает текст

Например,

А (нажатие клавиши латинской буквы A) – приводит к созданию ячейки выше выбранной ячейки:

The screenshot shows the same Jupyter Notebook interface after pressing the 'A' key. A new cell is now visible above the original cell, indicated by the number '[]:'.

Ячейка Markdown в Jupyter Notebook отображает текст, который можно отформатировать с использованием языка Markdown. Чтобы ввести текст, который не должен обрабатываться сервером Notebook как код, его нужно сначала преобразовать в ячейку Markdown. Это можно сделать либо из меню ячейки, либо с помощью сочетания клавиш M в командном режиме.

Тема урока: Jupyter Notebook

Ячейка Markdown в Jupyter Notebook отображает текст



В (нажатие клавиши латинской буквы b) – приводит к созданию ячейки ниже выбранной ячейки:

The screenshot shows a Jupyter Notebook interface with a single cell containing the text: "Ячейка Markdown в Jupyter Notebook отображает текст, который можно отформатировать с использованием языка Markdown. Чтобы ввести текст, который не должен обрабатываться сервером Notebook как код, его нужно сначала преобразовать в ячейку Markdown. Это можно сделать либо из меню ячейки, либо с помощью сочетания клавиш M в командном режиме." A new cell below it is highlighted with a blue border, indicating it was created using the 'B' key.

DD (2 раза нажимаем D) – приводит к удалению выбранной ячейки:

The screenshot shows the same Jupyter Notebook interface after the 'D' key was pressed twice on the previously selected cell. The cell has been removed, leaving a blank space in the notebook.



Закладка **Kernel** позволяет нам перезапустить наш ноутбук (причем в различных вариантах):

The screenshot shows a Jupyter Notebook interface with a pie chart plot in cell [8]. A context menu is open over the plot, with the 'Restart Kernel and Run All Cells...' option highlighted. Other options include 'Interrupt Kernel', 'Restart Kernel...', 'Reconnect to Kernel', 'Shutdown Kernel', and 'Change Kernel...'. The code in cell [8] is:

```
[8]: import matplotlib.pyplot as plt
x=[2,7,12,29,35]
fig = plt.figure(figsize =(9, 5)) # Line 4
plt.pie(x)
plt.show()
```

The resulting pie chart is displayed below the code cell.

В конце своей работы с ноутбуком его нужно закрыть – это обязательное условие для корректного завершения работы ноутбука!):

The screenshot shows a Jupyter Notebook interface with a line plot in cell [7]. A context menu is open over the plot, with the 'Shutdown Kernel' option highlighted. Other options include 'Interrupt Kernel', 'Restart Kernel...', 'Reconnect to Kernel', 'Shutdown Kernel', and 'Change Kernel...'. The code in cell [7] is:

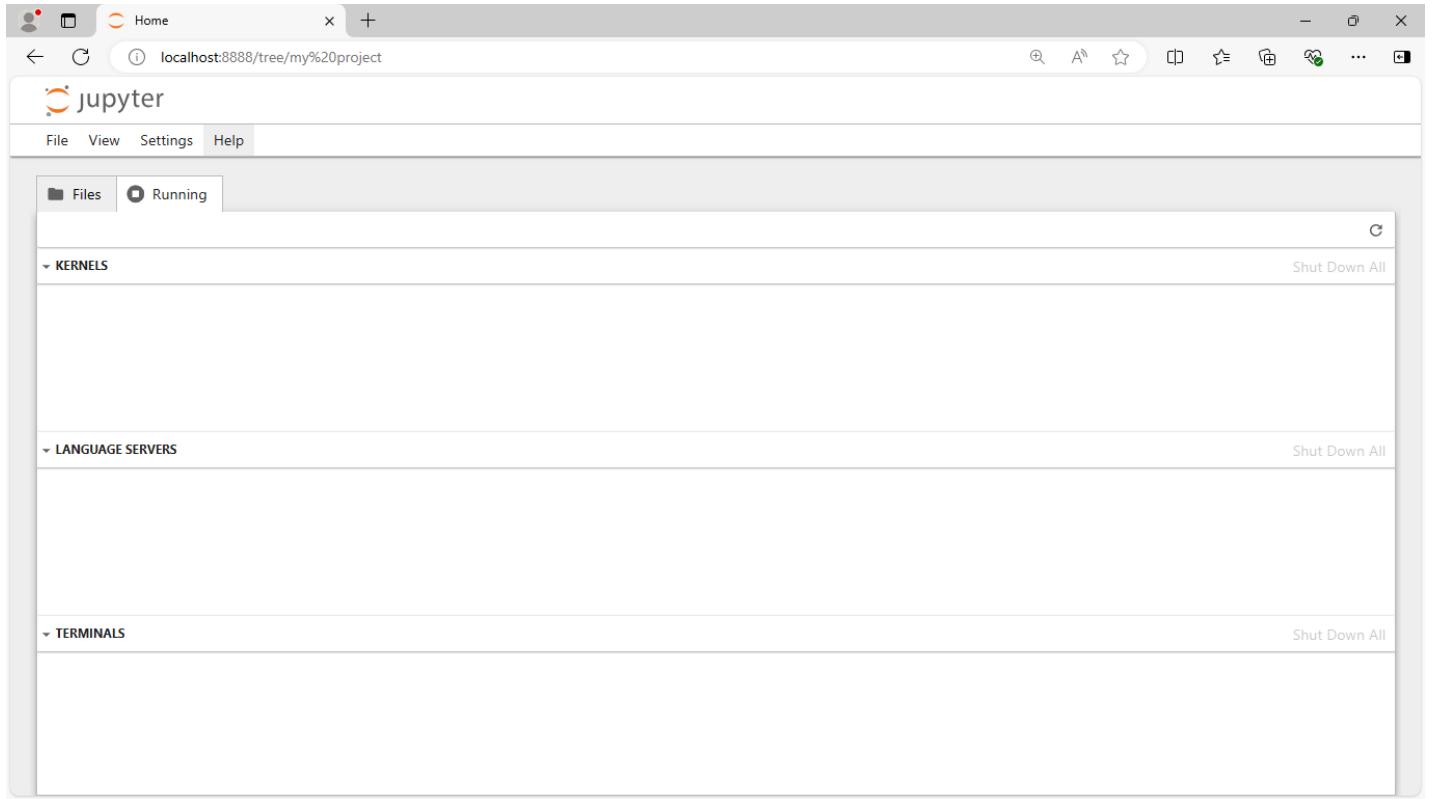
```
[7]: import matplotlib.pyplot as plt
x=[5,6,7,8,
y= [25,36,4
plt.plot(x,y)
plt.show()
```

The resulting line plot is displayed below the code cell.



И только после этого мы можем нажать на символ X и физически закрыть ноутбук.

Убедиться в том, что ноутбук прекратил свою работу, можно, нажав закладку Running (там не должно быть работающих ноутбуков):

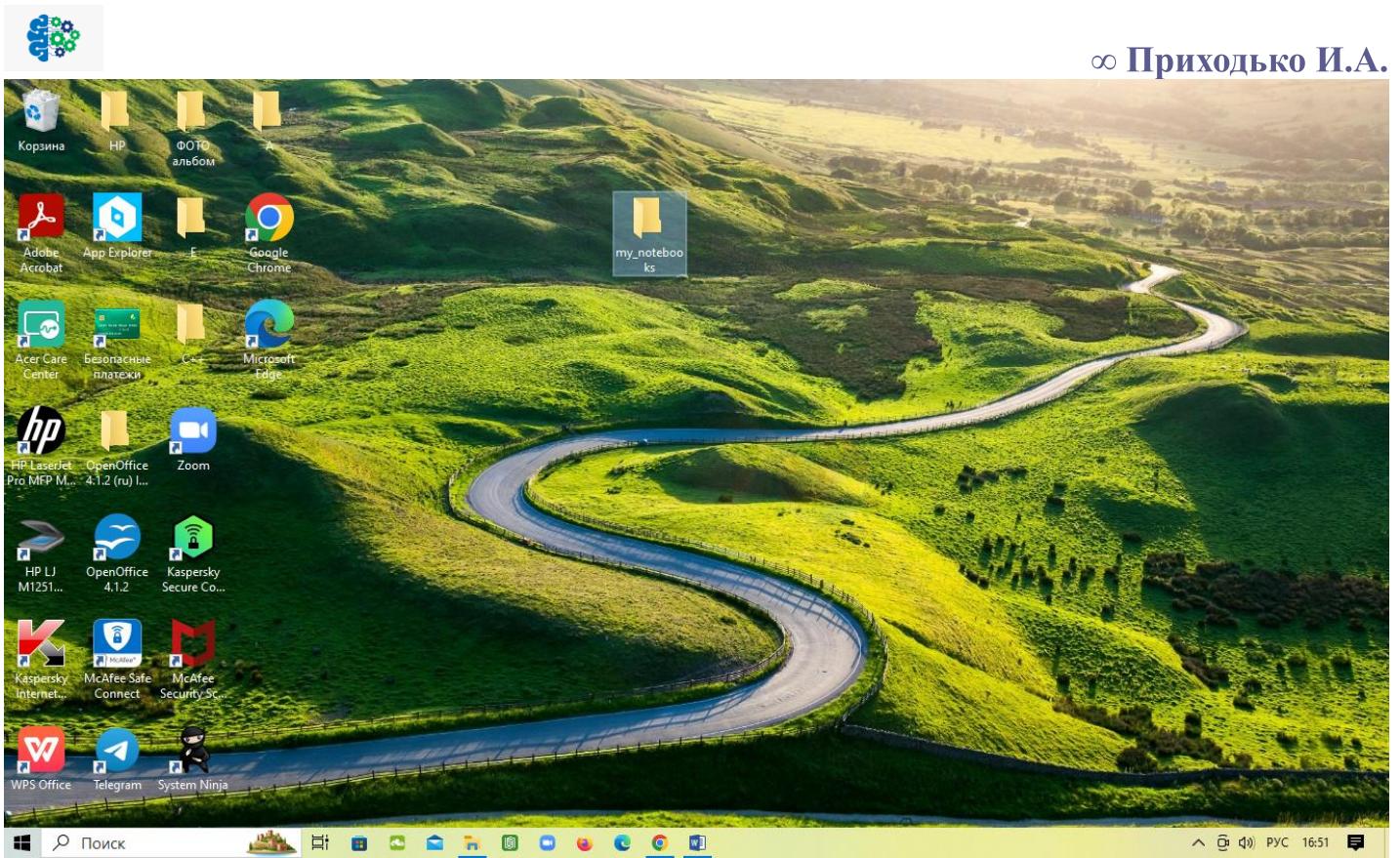


При этом Ваш ноутбук сохранён корректно в папке. И Вы в любой момент можете вернуться к нему и продолжить работу с ним.

Файлы с расширением **.ipynb** могут быть использоваться точно так же, как используются другие файлы (.txt, .docx, .pdf, .ppt и др.): их можно редактировать, копировать, отправлять и получать по электронной почте и пр.

А что если Вы хотели бы хранить свои ноутбуки в какой-то своей папке?

Для примера, давайте создадим новую папку на рабочем столе и назовём её **my_notebooks** :



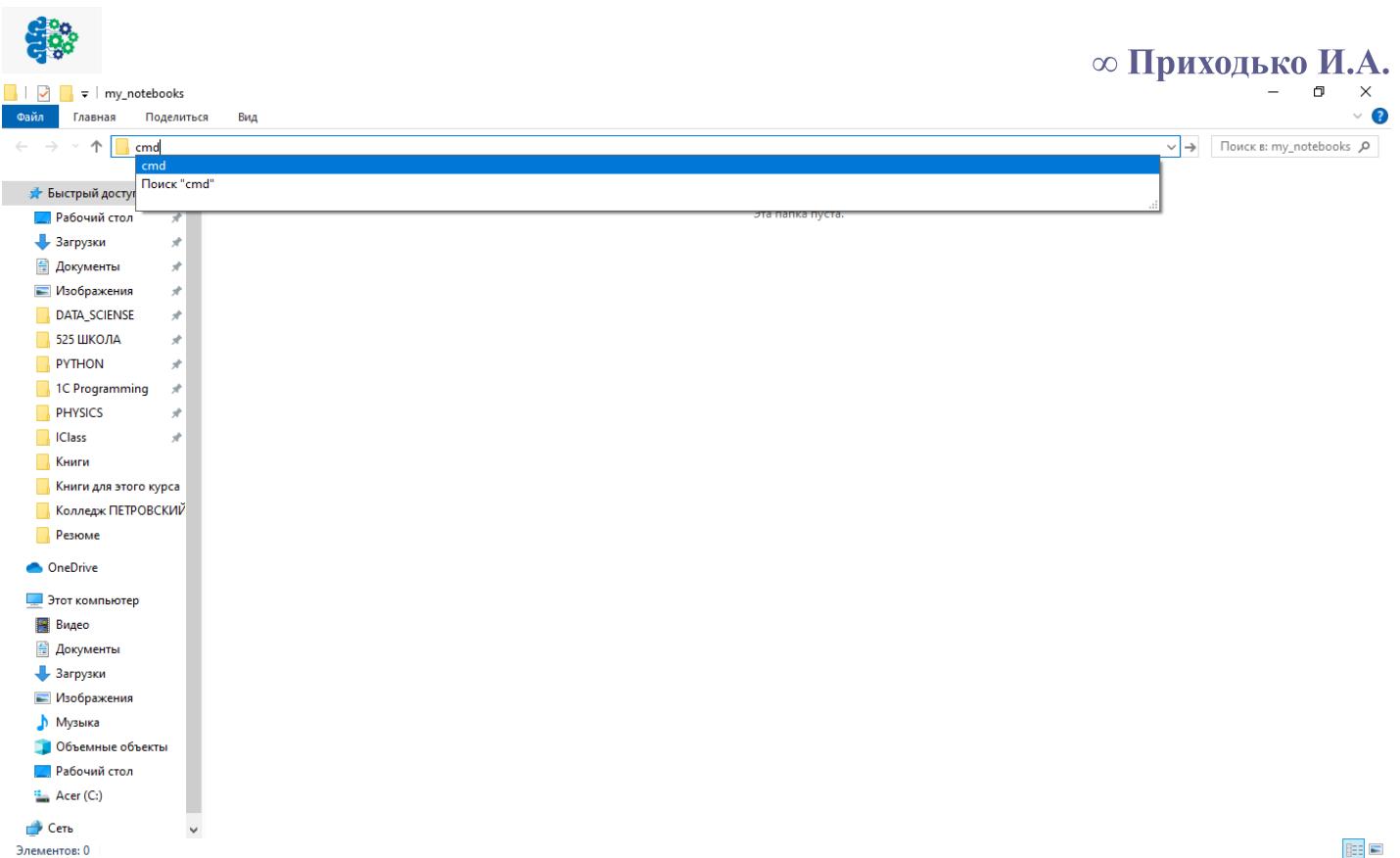
Теперь откроем эту новую папку. В её адресной строке вместо

C:\Users\User\Desktop\my_notebooks

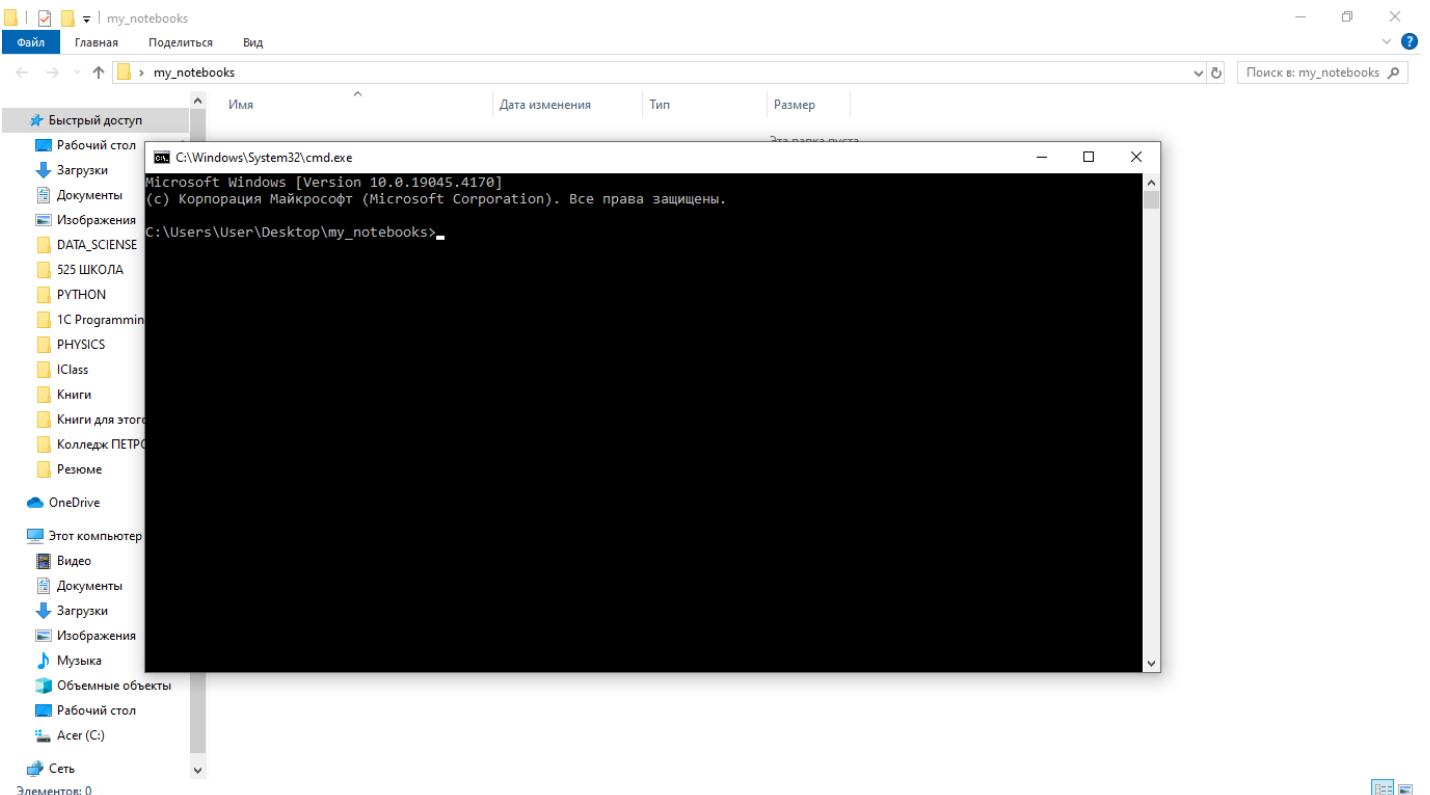
введём

cmd

и нажмём Enter:



В результате должна открыться консоль:



В консоли после приглашения вводим команду:

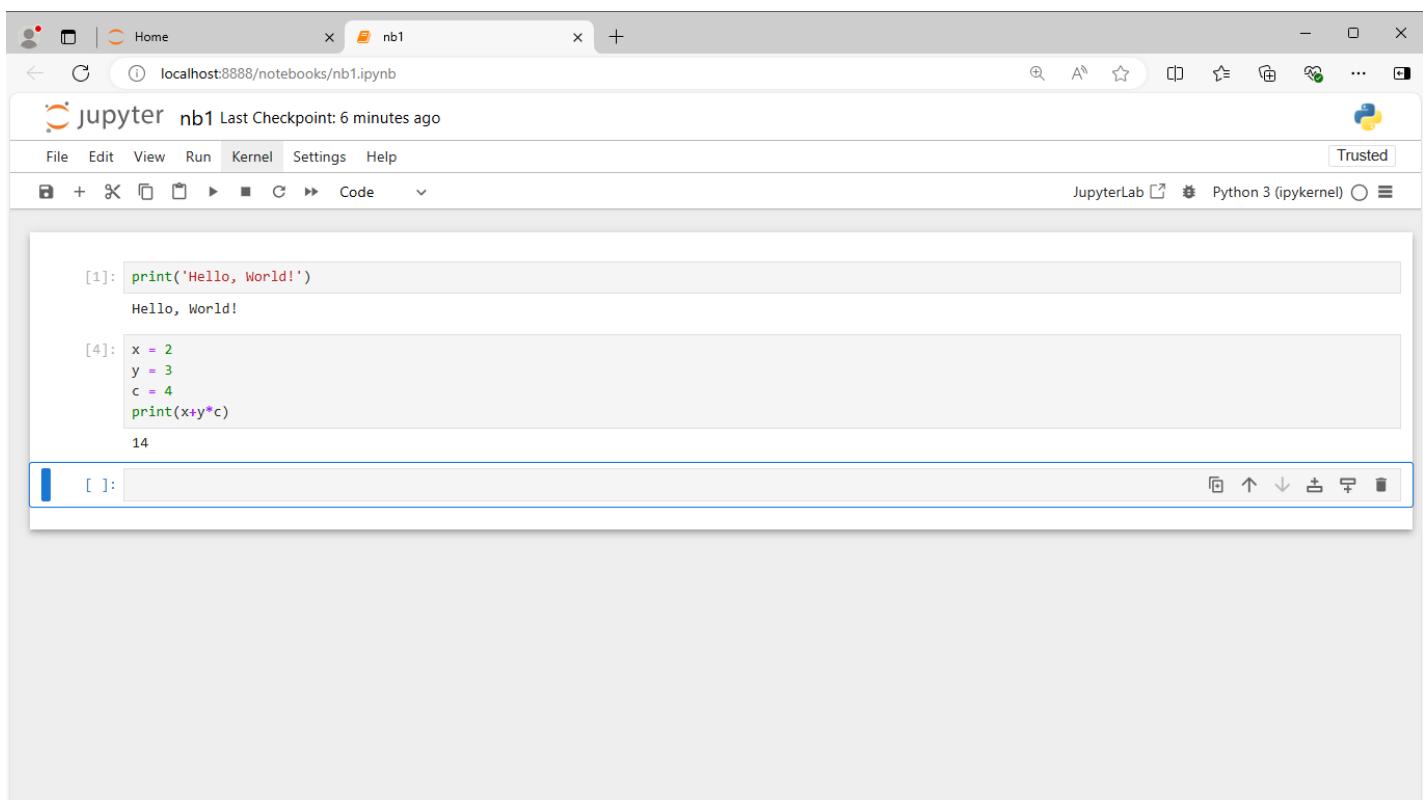
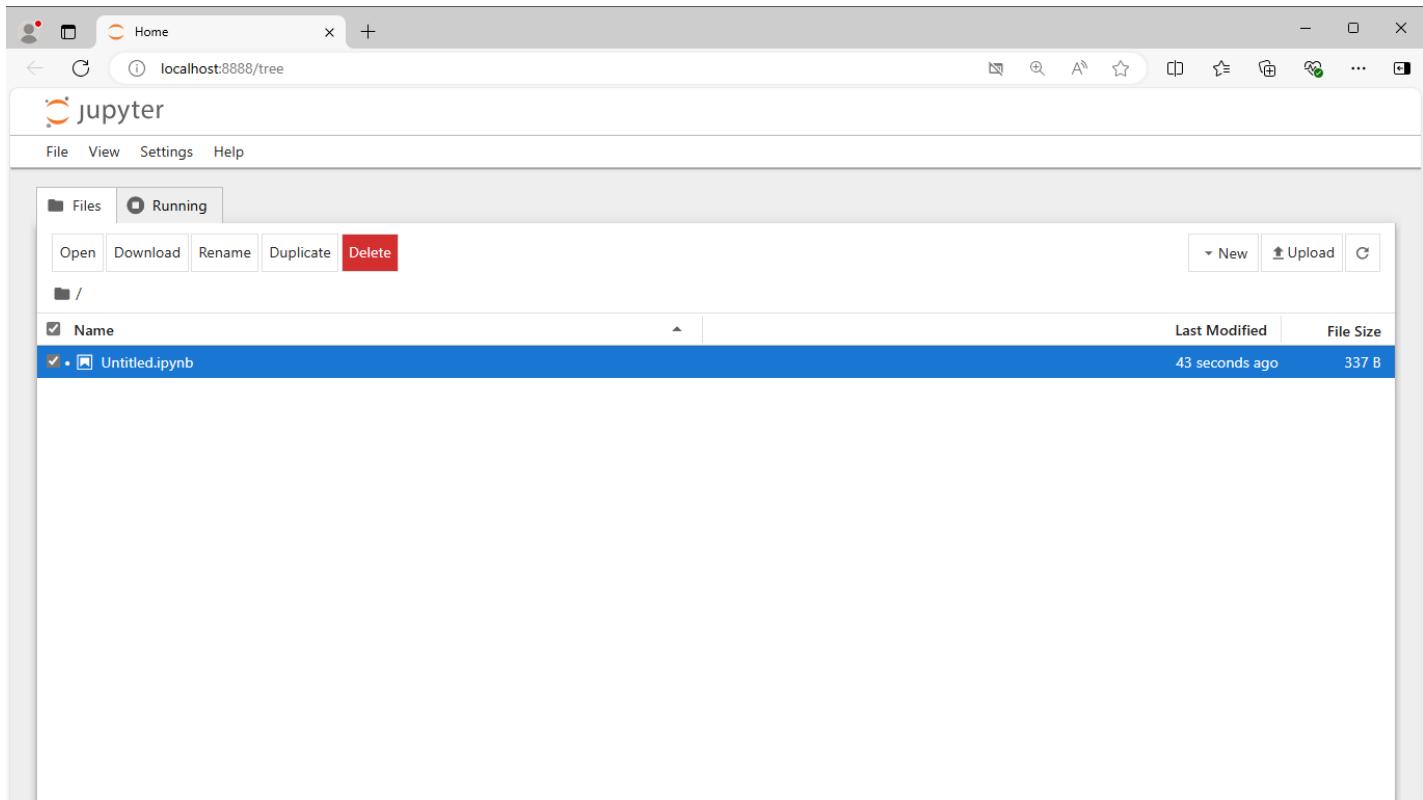
jupyter notebook

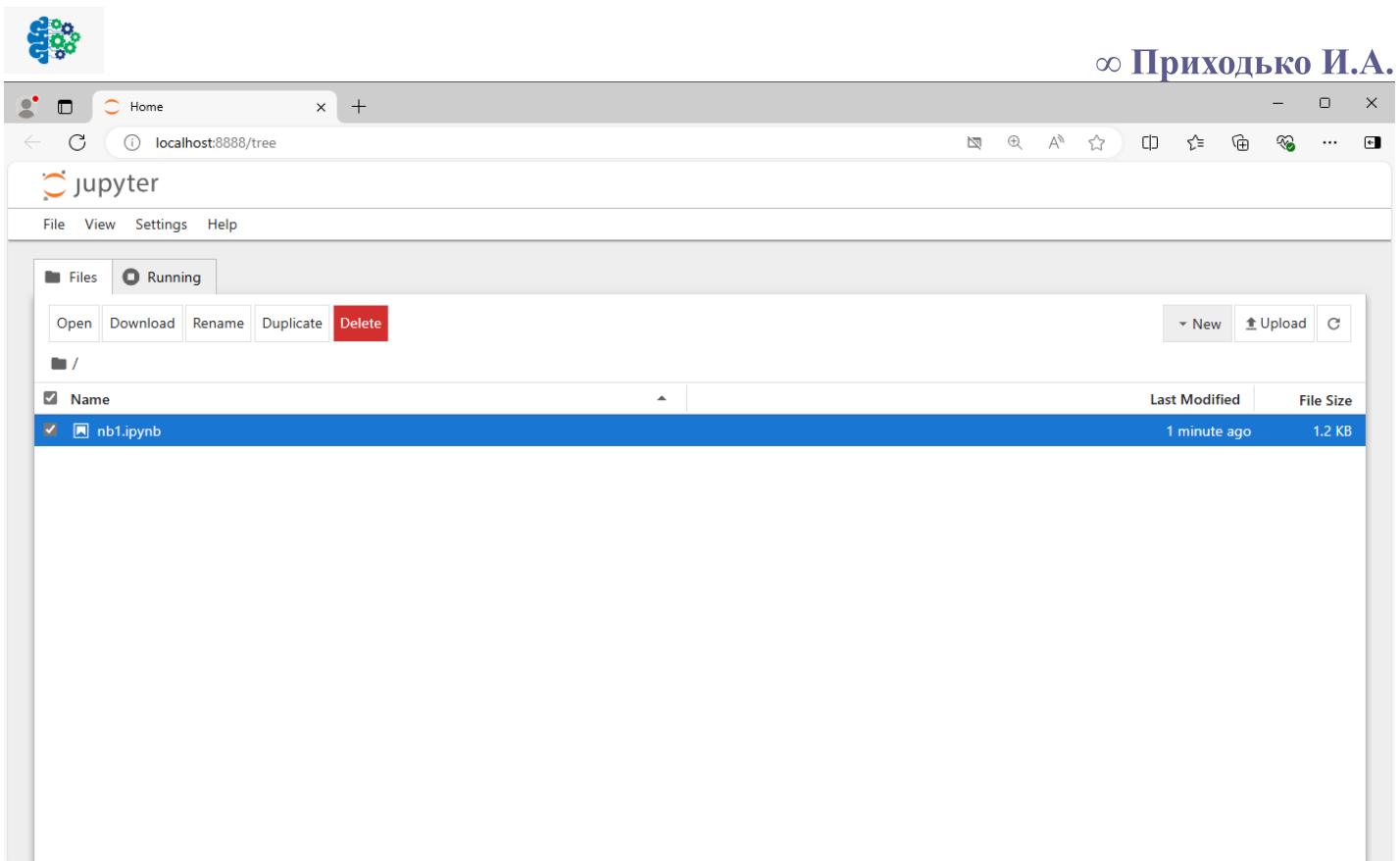


и нажимаем Enter (должен запуститься Jupyter Notebook).

Далее мы уже знаем, что и как можно сделать.

Например: можно создать новый ноутбук в этой папке, дать этому ноутбуку название, открыть этот ноутбук и написать какой-то код, после этого закрыть ноутбук:





В эту папку (если Вам удобно) Вы можете сохранять ноутбуки, полученные из других источников.

На этом мы заканчиваем сегодняшнее задание.

Что мы сегодня узнали:

- 1.Что такое Jupyter Notebook и где применяется;**
- 2.Поддерживаемые языки;**
- 3.Как запустить Jupyter Notebook;**
- 4.Как работать с Jupyter Notebook (режимы Code и Markdown);**
- 5.Shortcuts в Jupyter Notebook.**

Все это нам очень пригодится на последующих занятиях.

Домашнее задание:

- 1.Попробуйте выполнить в Jupyter Notebook на домашнем компьютере (ноутбуке) те же самые упражнения и команды, что мы с Вами проделали на занятии.

Всем спасибо за внимание.

Желаю Вам успехов в освоении анализа больших данных!



∞ Приходько И.А.

До встречи на следующем занятии.