МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ
Институт системной и программной инженерии
и информационных технологий (Институт СПИНТех)

Практикум по курсу
"Эффективные методологии разработки программного обеспечения "

Работа 1. Знакомство с Jupyter Notebook

Jupyter Notebook – это крайне удобный инструмент для интерактивной разработки и создания красивых аналитических отчетов, так как он позволяет хранить вместе код, изображения, комментарии, формулы и графики. Jupyter поддерживает различные языки программирования, однако здесь основное внимание будет уделено *Python*, поскольку он является наиболее распространенный вариантом использования.

Задачи этой практической работы:

1. запускать и сохранять блокноты,
2. знакомство с их структурой и интерфейсом,
3. знакомство с некоторыми основными терминами и возможностями, практическое понимание того, как самостоятельно использовать *Jupyter Notebook*.

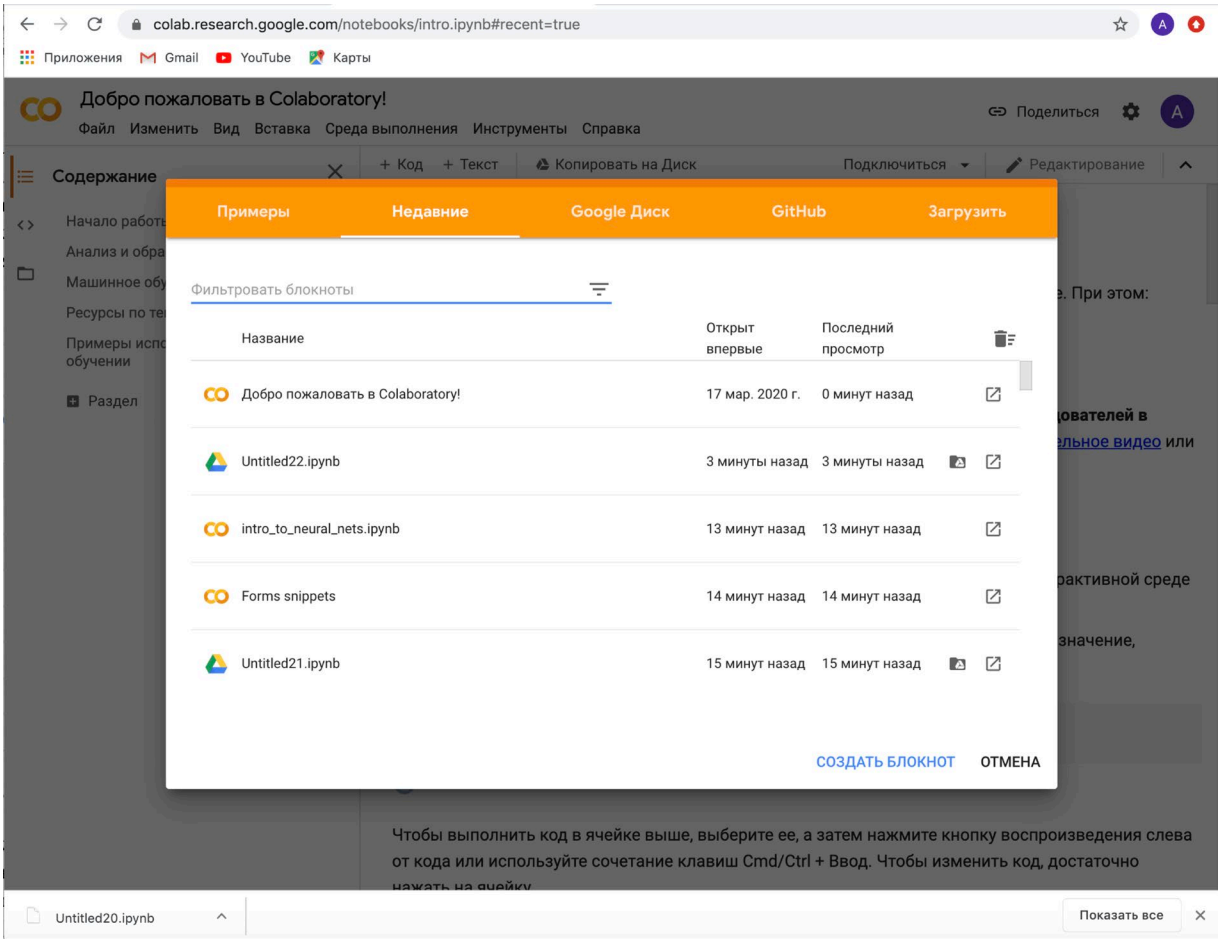
Прежде чем приступить к выполнению лабораторных заданий, рекомендуется ознакомиться с языком программирования Python. Тем не менее, если у вас есть опыт работы с другим языком программирования, Python в этой работе не будет для вас слишком сложным.

1. Запуск Jupyter Notebook

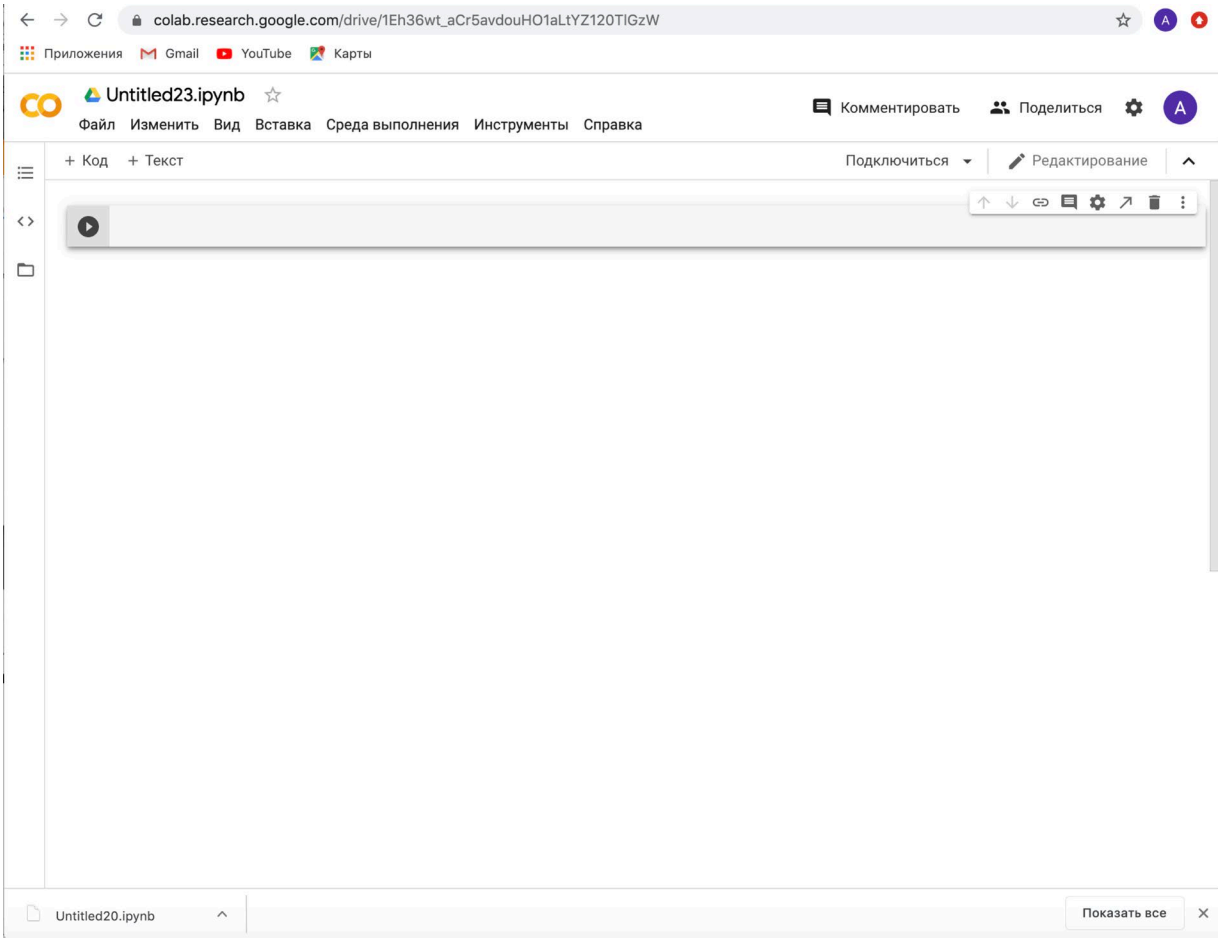
Запуск приложения: https://colab.research.google.com/

После запуска появляется панель управления ноутбуками Jupyter. Она предоставляет вам доступ только к файлам и подпапкам, содержащимся в каталоге запуска Jupyter; однако каталог запуска может быть изменен. Интерфейс панели управления в основном интуитивно понятен.

Если у Вас есть история работы с ноутбуками, она отобразится так:



Ваш первый блокнот Jupyter откроется в новой вкладке – каждый блокнот использует свою вкладку, поэтому вы можете открывать несколько блокнотов одновременно.



Файлы блокнота сщздаются в формате **.ipynb** и представляют собой текстовый файл, который описывает содержимое вашей записной книжки в формате JSON. Каждая ячейка и ее содержимое, включая вложения изображений, которые были преобразованы в строки текста, перечислены в нем вместе с некоторыми метаданными. Если умеете, вы можете отредактировать их самостоятельно, выбрав «Edit> Edit Notebook Metadata» в строке меню в записной книжке. Но ключевое слово здесь – «*можете*»; нет никакой другой причины, кроме любопытства, делать это.

2. Интерфейс Jupyter Notebook

Теперь, когда вы открытый блокнот, давайте осмотримся. Jupyter – это просто продвинутый текстовый процессор. Просмотрите внимательно меню, потратьте несколько минут, чтобы прокрутить список команд.

В Jupyter есть два важных термина: **cells** (ячейки) и **kernels** (ядра). Они являются ключом как к пониманию Jupyter, так и к тому, что делает его не просто текстовым процессором:

- **kernel** (ядро) – это «вычислительный движок», который выполняет код, содержащийся в документе ноутбука.
- **cell** (ячейка) – это контейнер для текста, который будет отображаться в записной книжке, или код, который будет выполняться ядром записной книжки.

3. Ячейки (Cell)

Мы вернемся к ядрам немного позже, сначала разберемся с ячейками. Ячейки образуют структуру ноутбука. На приведенном выше скриншоте это поле с зеленым контуром. Существует два основных типа ячеек:

- **Ячейка кода** содержит код, который должен быть выполнен в ядре, и отображает его вывод ниже.
- **Ячейка Markdown** содержит текст, отформатированный с использованием Markdown, и отображает его вывод на месте при запуске.

Первая ячейка в новой записной книжке всегда является ячейкой **кода**. Давайте проверим это на классическом примере с Hello World.

Упражнение

Введите **print ('Hello World!')** в ячейку кода

```
In [ ]: print("Hello World!")
```

и нажмите кнопку запуска на панели инструментов или нажмите **Ctrl + Enter**.

Результат должен выглядеть так:

```
In [1]: print("Hello World!")
        Hello World!
```

Когда вы запустите ячейку, результат её работы отобразится ниже, а метка слева изменится с **In []** на **In [1]**. Вывод ячейки кода также является частью документа. Вы всегда можете определить разницу между ячейками кода и ячейками Markdown, потому что ячейки кода имеют эту метку слева, а ячейки Markdown – нет.

Часть «**In**» метки – это короткая запись слова «**Input**», а номер метки указывает, когда ячейка была выполнена в ядре – в нашем случае ячейка была выполнена первой. Запустите ячейку снова, и метка изменится на In [2], потому что теперь ячейка была

второй, запущенной в ядре. Чуть позже, когда мы познакомимся с ядрами. станет понятно, почему это так полезно.

Упражнение

В строке меню нажмите **Insert** (*Вставить*) и выберите **Insert Cell Below** (*Вставить ячейку ниже*), чтобы создать новую ячейку кода под первой, или нажмите «+» на палитре команд, затем напечайте следующий код, запустите и посмотрите, что происходит.

```
In [*]: import time
        time.sleep(5)
```

Созданная вами только что ячейка не производит никакого вывода, но для ее выполнения требуется пять секунд. Обратите внимание, как **Jupyter** показывает, что ячейка в данный момент работает, изменив метку на **In [*]**.

Как правило, выходные данные ячейки поступают из любых текстовых данных, специально напечатанных во время выполнения ячеек, а также из значения последней строки в ячейке, будь то переменная-одиночка, вызов функции или что-то еще.

Упражнение

Создайте новую ячейку кода и наберите в ней следующий код:

```
In [ ]: def say_hello(recipient):
            return 'Hello, {}!'. format(recipient)
        say_hello('Tim!')
```

Результат должен выглядеть так:

```
Out[2]: 'Hello, Tim!!'
```

4. Горячие клавиши Jupyter Notebook

Вы, возможно, заметили, что при запуске ячеек, их рамка становится синей, тогда как в процессе редактирования она зелёная. Всегда есть одна «активная» ячейка, выделенная рамкой, цвет которой обозначает ее текущий режим: зеленый *edit mode (режим редактирования)* и синий – *command mode (командный режим)*.

До сих вы только запускали ячейку с помощью горячих клавиш **Ctrl + Enter**, но есть еще много других возможностей. Сочетания клавиш являются очень популярным аспектом среды Jupyter, поскольку они обеспечивают быстрый рабочий процесс на основе ячеек. Многие из этих действий вы можете выполнять в активной ячейке, когда она находится в командном режиме.

Ниже представлен список основных сочетаний клавиш в Jupyter.

- Переключение между режимом редактирования и командным режимом с помощью Esc и Enter соответственно.
- В командном режиме:
  - Прокрутка ячеек вверх и вниз с помощью клавиш «**Вверх**» и «**Вниз**».
  - **A** или **B** вставляют новую ячейку выше или ниже активной ячейки.
  - **M** преобразует активную ячейку в ячейку Markdown.
  - **Y** устанавливает активную ячейку в кодовую ячейку.
  - **D** + **D** (**D** дважды) удаляет активную ячейку.
  - **Z** отменяет удаление ячейки.
  - Удерживание **Shift** + **Вверх** или **Вниз**, выбирает несколько ячеек одновременно.
  - Выделение нескольких ячеек **Shift + M** объединяет выбранные ячейки.
- **Ctrl + Shift + –** в режиме редактирования разделяет активную ячейку по курсору.
- Вы также можете нажать **Shift + клик** на полях слева от ваших ячеек, для их выбора.

Упражнение

Попробуйте все сочетания горячих клавиш. После того, как вы опробуете все команды, создайте с их помощью новую ячейку Markdown, дальше вы научитесь форматировать текст в ваших блокнотах.

5. Markdown

Markdown – это легкий и простой в освоении язык разметки для форматирования текста. Его синтаксис имеет однозначное соответствие с тегами HTML, поэтому некоторые предварительные знания здесь могут быть полезны, но это определенно не является обязательным условием. Давайте рассмотрим основы на следующем примере:

```
# This is a level 1 heading
## This is a level 2 heading
This is sime plain text that forms a paragraph.
Add emphasis via **bold** and __bold__, or *italic* and _italic_.
Paragraphs must be separeted by an empty line.
* Sometimes we want to include lists.
* Which can be indented.
1. Lists can also be numbered.
2. For ordered lists.
[It is possible to include hypelinks](https://www.example.com)
Inline code uses single backticks: 'foo()', and code blocks use triple
backticks:
'''
bar()
'''
Or can be indented by 4 spaces:
foo()
And finally, adding images is easy: ![Alt text](http://web-profy.com/wp-
content/uploads/2013/09/startup.jpg)
```

Результат выполнения этой ячейки:

This is a level 1 heading

This is a level 2 heading

This is sime plain text that forms a paragraph. Add emphasis via **bold** and **bold**, or *italic* and *italic*. Paragraphs must be separeted by an empty line.

- Sometimes we want to include lists.
- Which can be indented.
- Lists can also be numbered.
- For ordered lists. It is possible to include hypelinks Inline code uses single backticks: 'foo()', and code blocks use triple backticks: ''' bar() ''' Or can be indented by 4 spaces: foo() And finally, adding images is easy:

Три варианта прикрепления изображений:

- Используйте URL-адрес для изображения из Интернета.
- Используйте локальный URL-адрес для изображений, которые хранятся рядом с вашим ноутбуком.
- Добавьте вложение через «Edit> Insert Image» - эта команда преобразует изображение в строку и сохранит его в файле .ipynb вашего ноутбука. Однако, обратите внимание, что это сделает ваш файл .ipynb намного больше!

У Markdown есть гораздо больше возможностей, особенно в отношении гиперссылок, а также можно просто включить простой HTML. Если вы захотите узнать больше, вы можете обратиться к официальному руководству от создателя Markdown, Джона Грубера, на его веб-сайте.

Упражнение

Наберите в ячейке Markdown любой текст с использованием основных тегов для его форматирования, также ваш текст должен содержать как минимум одну гиперссылку и одно изображение.

6. Ядра (Kernels)

За каждым ноутбуком работает ядро. Когда вы запускаете ячейку кода, этот код выполняется в ядре, и любой вывод возвращается обратно в ячейку для отображения. Состояние ядра сохраняется во времени и между ячейками – оно относится к документу в целом, а не к отдельным ячейкам.

Например, если вы импортируете библиотеки или объявляете переменные в одной ячейке, они будут доступны в другой. Таким образом, вы можете думать о документе блокнота как о чем-то сравнимом с файлом сценария, за исключением того, что он является мультимедийным. Давайте попробуем.

Упражнение

Создайте новую ячейку кода, импортируйте пакет Python – numpy, и напечатайте следующую функцию:

```
In [3]: import numpy as np
        def square(x):
            return x*x
```

Как только мы выполните ячейку, вы сможете ссылаться на **np** и **square** в любой другой ячейке. Создайте новую ячейку и введите следующий код:

```
In [4]: x=np.random.randint(1,10)
        y=square(x)
        print('%d squared is %d' % (x,y))

        6 squared is 36
```

Это будет работать независимо от порядка ячеек в вашем блокноте. Давайте попробуем, распечатайте снова наши переменные:

```
In [5]: print('%d squared is %d' % (x,y))

        6 squared is 36
```

Здесь нет сюрпризов! Но теперь давайте изменим y:

```
In [ ]: y = 15
```

Как вы думаете, что произойдет, если мы снова запустим ячейку, содержащую наш оператор print? Заново напечайте переменные x,y или просто выполните предыдущую ячейку с кодом, вставьте в отчёт полученный результат и поясните.

Большую часть времени поток в вашем ноутбуке будет идти сверху вниз, но часто приходится возвращаться, чтобы внести изменения. В этом случае важен порядок выполнения, указанный слева от каждой ячейки, например, In [6], позволит вам узнать, имеет ли какая-либо из ваших ячеек устаревший вывод. И если вы когда-

нибудь захотите сбросить настройки, есть несколько невероятно полезных опций из меню на панели инструментов – Kernel:

- **Restart**: перезапускает ядро, таким образом очищая все переменные и т.д., которые были определены.
- **Restart & Clear Output**: то же, что и выше, но также стирает вывод, отображаемый под ячейками кода.
- **Restart & Run All**: то же, что и выше, но также будет запускать все ваши ячейки в порядке от первого до последнего.
- Если ваше ядро зависло в вычислении, и вы хотите остановить его, вы можете выбрать опцию **Interupt**.

7. Выбор ядра

Возможно, вы заметили, что Jupyter дает вам возможность сменить ядро, и на самом деле есть много разных вариантов. Когда вы создавали новую записную книжку на панели инструментов, вы выбирали версию Python, вы фактически выбирали, какое ядро использовать.

Существуют не только ядра для разных версий Python, но и более 100 языков, включая Java, C, Fortran., R, Julia, и даже в imatlab и ядре Calysto MATLAB Kernel для Matlab. Ядро SoS обеспечивает многоязычную поддержку в пределах одного ноутбука. Каждое ядро имеет свои собственные инструкции по установке, и, вероятно, потребует от вас выполнения некоторых команд на вашем компьютере.

8. Графики с помощью библиотеки Matplotlib

Рассмотрим пример использования библиотеки Matplotlib. Для того, чтобы получать графики рядом с ячейками с кодом необходимо выполнить специальную magic команду после того, как импортируете matplotlib:

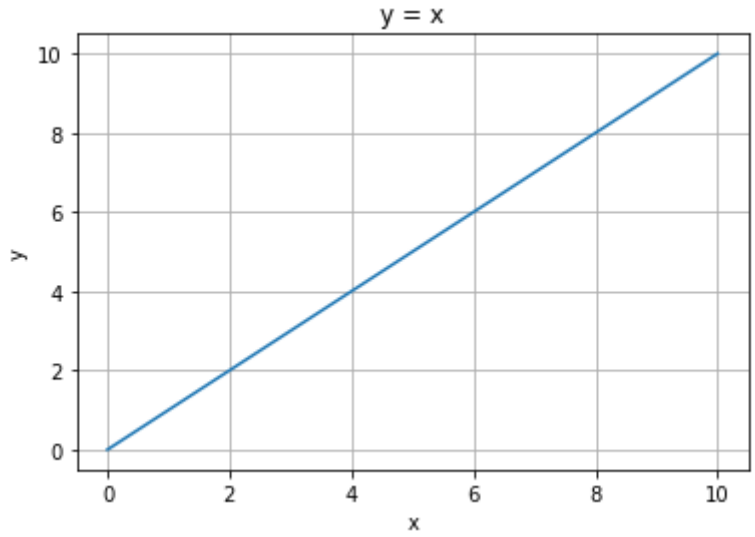
```
In [6]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Теперь перейдем непосредственно к Matplotlib. Наша задача построить разные типы графиков, настроить их внешний вид и освоиться в работе с этим инструментом.

Для начал построим простую линейную зависимость, дадим нашему графику название, подпишем оси и отобразим сетку. Код программы:

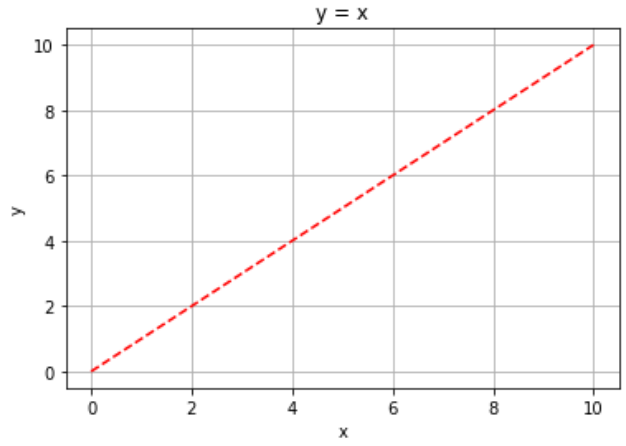
```
In [ ]: x = np.linspace(0, 10, 50)
        y = x;
        plt.title("y = x")
        plt.xlabel("x")
        plt.ylabel("y")
        plt.grid()
        plt.plot(x,y);
```

В результате получим следующий график:



Изменим тип линии и ее цвет, для этого в функцию plot(), в качестве третьего параметра передадим строку, сформированную определенным образом, в нашем случае это “r–”, где “r” означает красный цвет, а “–” – тип линии – пунктирная линия. Более подробно о том, как задавать цвет и какие типы линии можно использовать, вы можете прочитать в дополнительной литературе.

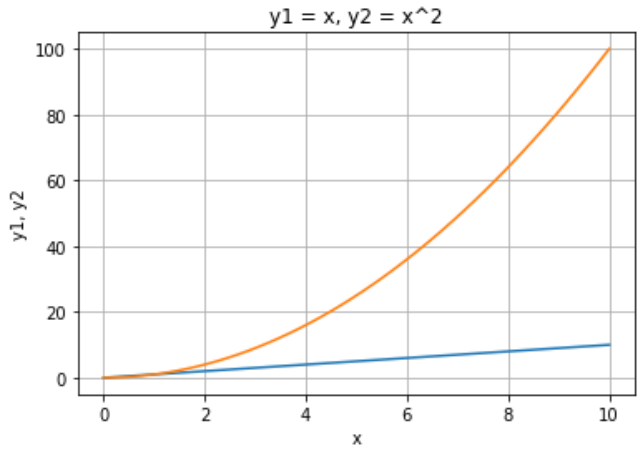
```
In [7]: x = np.linspace(0, 10, 50)
        y = x;
        plt.title("y = x")
        plt.xlabel("x")
        plt.ylabel("y")
        plt.grid()
        plt.plot(x,y, "r--");
```



Несколько графиков на одном поле

Построим несколько графиков на одном поле:

```
In [8]: x = np.linspace(0, 10, 50)
        y1 = x;
        y2 = [i**2 for i in x]
        plt.title("y1 = x, y2 = x^2")
        plt.xlabel("x")
        plt.ylabel("y1, y2")
        plt.grid()
        plt.plot(x,y1,x, y2);
```



В приведенном примере в функцию plot() последовательно передаются два массива для построения первого графика и два массива для построения второго, при этом, как вы можете заметить, для обоих графиков массив значений независимой переменной x один и то же.

Несколько разделенных полей с графиками

Третья, довольно часто встречающаяся задача – это отобразить два или более различных поля, на которых будет отображено по одному или более графику.

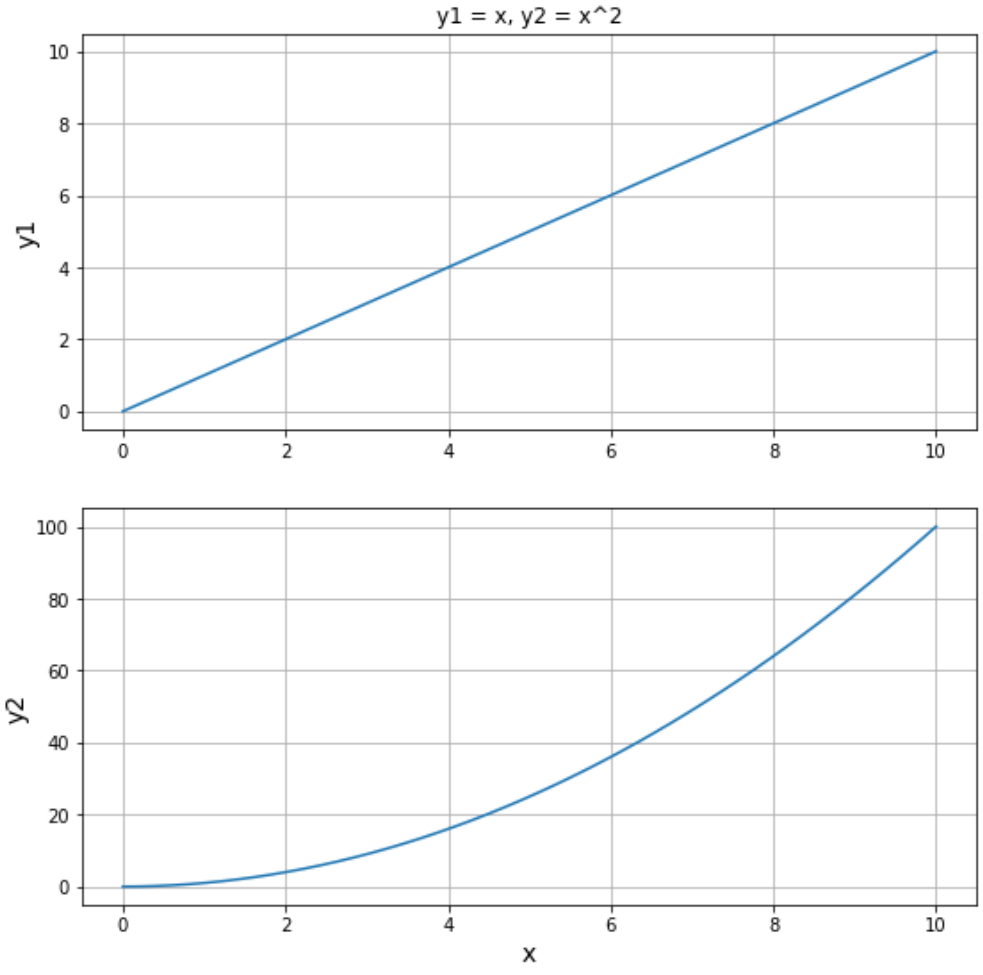
Построим уже известные нам две зависимость на разных полях:

In [9]:
```
x = np.linspace(0, 10, 50)
y1 = x;
y2 = [i**2 for i in x]

plt.figure(figsize=(9,9))

plt.subplot(2, 1, 1)
plt.plot(x, y1)
plt.title("y1 = x, y2 = x^2")
plt.ylabel("y1", fontsize=14)
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(x, y2)
plt.ylabel("y2", fontsize=14)
plt.xlabel("x", fontsize=14)
plt.grid(True)
```



Здесь мы воспользовались новыми функциями:

- **figure()** – функция для задания глобальных параметров отображения графиков. В нее, в качестве аргумента, мы передаем кортеж, определяющий размер общего поля.
- **subplot()** – функция для задания местоположения поля с графиком. Существует несколько способов задания областей для вывода через функцию subplot() мы воспользовались следующим: первый аргумент – количество строк, второй – столбцов в формируемом поле, третий – индекс (номер поля, считаем сверху вниз, слева направо).

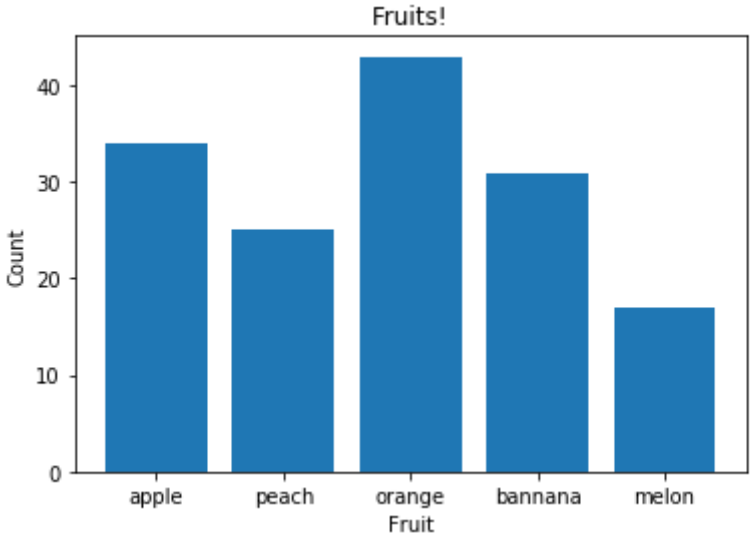
Остальные функции уже вам знакомы, дополнительно мы использовали параметр **fontsize** для функций **xlabel()** и **ylabel()**, для задания размера шрифта.

Построение диаграммы для категориальных данных

До этого мы строили графики по численным данным, т.е. зависимая и независимая переменные имели числовой тип. На практике довольно часто приходится работать с данными нечисловой природы – имена людей, название фруктов, и т.п.

Построим диаграмму, на которой будет отображаться количество фруктов в магазине:

```
In [ ]: fruits = ["apple", "peach", "orange", "bannana", "melon"]
        counts = [34, 25, 43, 31, 17]
        plt.bar(fruits, counts)
        plt.title("Fruits!")
        plt.xlabel("Fruit")
        plt.ylabel("Count")
```

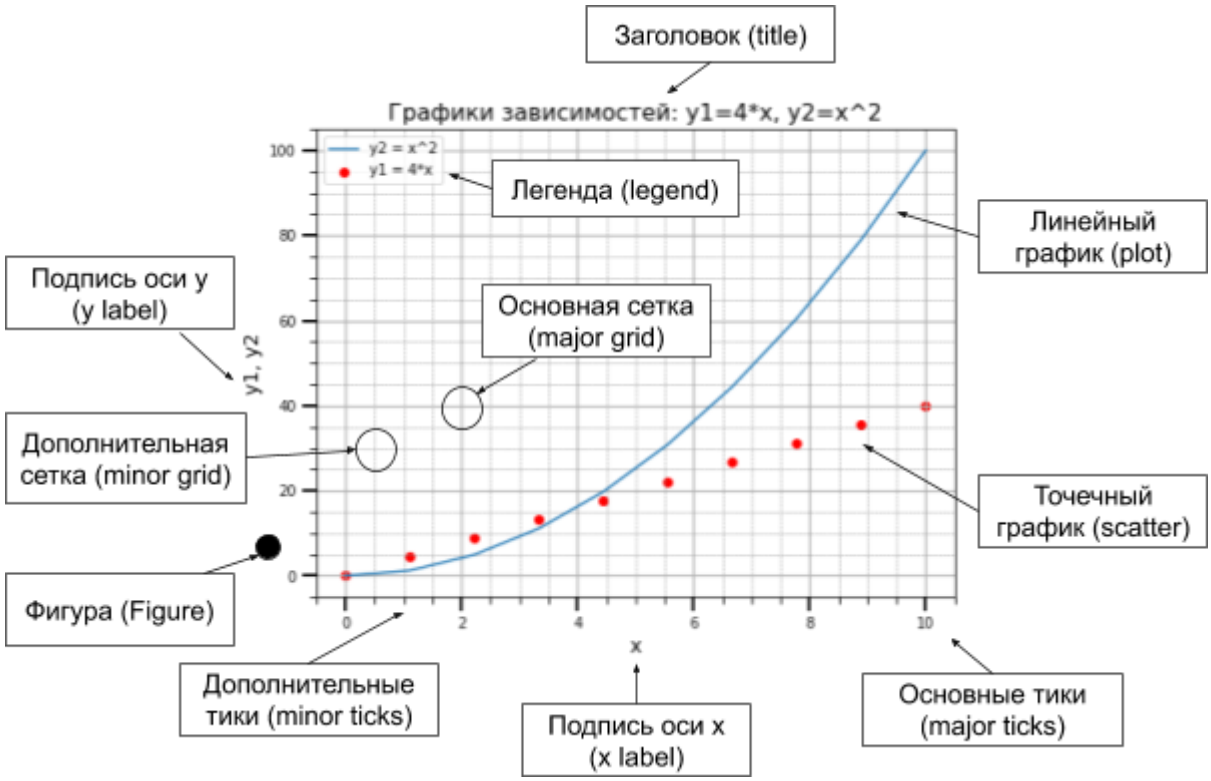


Для вывода диаграммы мы использовали функцию bar().

Основные элементы графика

Рассмотрим основные термины и понятия, касающиеся изображения графика, с которыми вам необходимо будет познакомиться, для того, чтобы в дальнейшем у вас

не было трудностей при выполнении данного курса лабораторных работ и документации по библиотеке matplotlib.



Корневым элементом при построения графиков в системе Matplotlib является Фигура (Figure). Все, что нарисовано на рисунке выше является элементами фигуры.

Упражнение

Постройте все приведённые выше примеры графиков с такими же или своими данными.

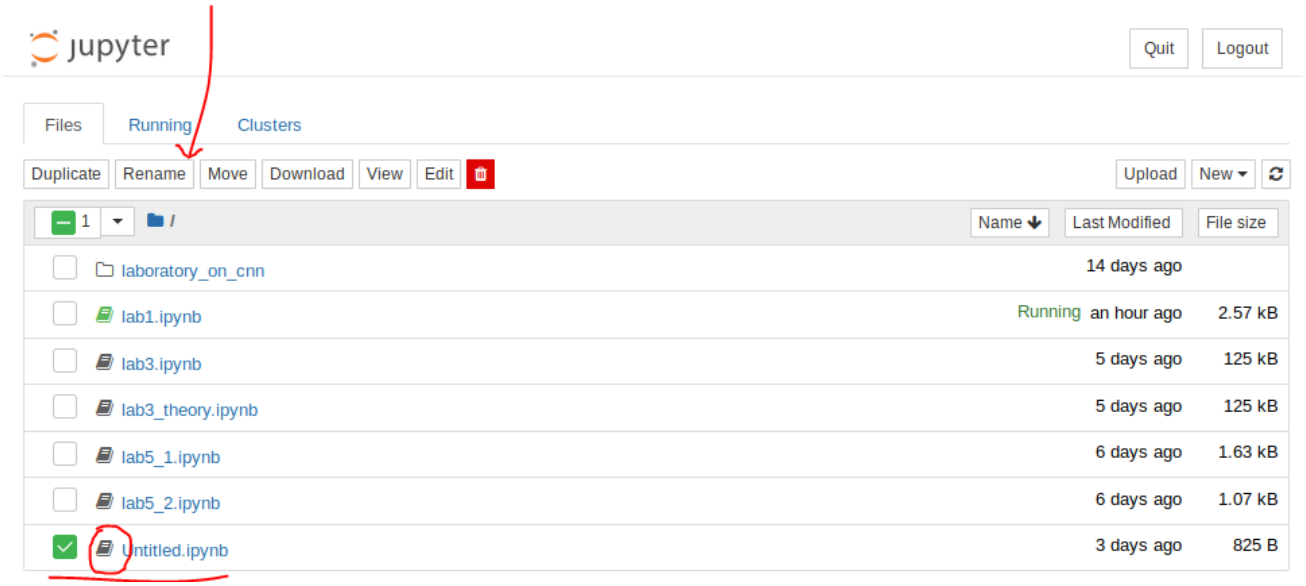
9. Названия для ноутбуков

Прежде чем начать создавать свой проект, вы, вероятно, захотите дать ему осмысленное имя. Возможно, это несколько сбивает с толку: но вы не можете назвать или переименовать свои записные книжки из самого приложения для записной книжки, а для переименования файла **.ipynb** необходимо использовать панель управления или файловый браузер. Мы вернемся к панели управления, чтобы переименовать созданный ранее файл, который имеет имя файла по-умолчанию для записной книжки – **Untitled.ipynb**.

Вы не можете переименовать ноутбук во время его работы, потому что его сначала нужно выключить. Самый простой способ сделать это – выбрать «File> Close and Halt» на панели инструментов ноутбука. Однако вы также можете выключить

ядро, перейдя в «Kernel> Shutdown» из ноутбука или выбрав ноутбук на панели управления и нажав «Shutdown» (см. Изображение ниже).

Затем вы можете выбрать свой блокнот и нажать «Rename» на панели управления.



Обратите внимание, что закрытие вкладки «notebook» в вашем браузере не «закроет» вашу записную книжку так же, как закрытие документа в традиционном приложении. Ядро ноутбука будет продолжать работать в фоновом режиме и должно быть отключено, прежде чем оно действительно «закроется». Это очень удобно, если вы случайно закрыли вкладку или браузер! Если ядро закрыто, вы можете закрыть вкладку, не беспокоясь о том, работает ли оно по-прежнему или нет.

Упражнение

Отключите ядро и переименуйте ваш ноутбук из панели управления.

Задание к лабораторной работе

1) Выполните все упражнения из данного документа.
2) Ниже приведена таблица, строки котрой соответствуют различным функциям $f(n)$, а столбцы значениям времени t. Заполните таблицу максимальными значениями n, для которых задача может быть решена за время t. Предполагается, что время работы алгоритма, необходимое для решения задачи равно $f(n)$ микросекунд.

	Секунда	Минута	Час	День	Месяц	Год	Век
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

С помощью библиотеки Matplotlib создайте графики функций из приведенной таблицы.

3) Представьте отчёт в виде файла **.ipynb** с результатами вашей работы.
4) Расскажите основные особенности и преимущества Jupyter Notebook.