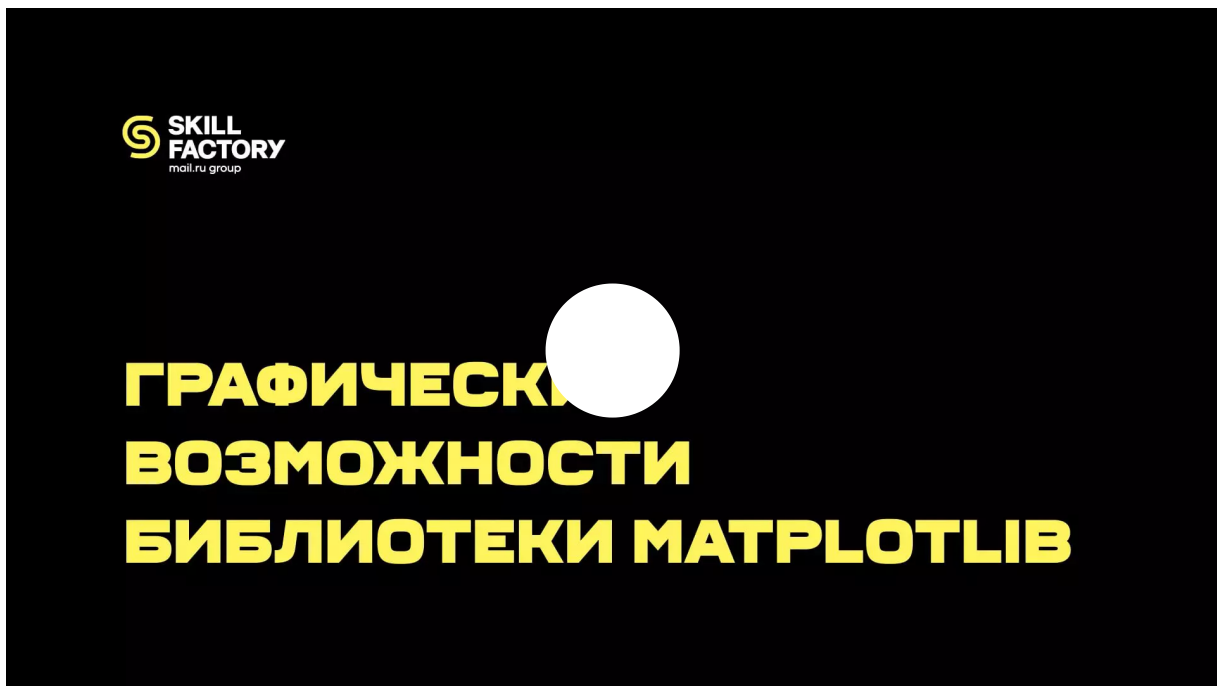




Курс > Блок 1... > PYTHON... > 5. Граф...

## 5. Графические возможности библиотеки Matplotlib



→ [Скачать ноутбук из скринкаста](#)



Настало время сделать визуальный анализ качественнее — переходим к библиотеке **Matplotlib**.

### НЕМНОГО О БИБЛИОТЕКЕ

Matplotlib — это библиотека *Python*, обладающая большим количеством возможностей для визуализации и настройки отображения графиков и диаграмм.

Для установки библиотеки введите в командную строку (или командную строку *Anaconda*) следующее:

```
pip install matplotlib
```

→ На самом деле мы с вами уже использовали *Matplotlib* — встроенная визуализация в *Pandas* полностью основана на данной библиотеке. Однако визуализация в *Matplotlib* не ограничивается только *DataFrame*: с помощью *Matplotlib* можно визуализировать любые последовательности (списки, словари, *NumPy*-массивы).

→ К тому же инструментарий библиотеки поможет вам расширить ваши возможности визуализации, управляя параметрами настройки графиков вручную.

За визуализацию графиков в *Matplotlib* отвечает модуль **pyplot**. Традиционно он импортируется под псевдонимом `plt`. Для более корректного отображения графиков в ноутбуках используется команда `%matplotlib inline`.

```
import matplotlib.pyplot as plt  
%matplotlib inline
```

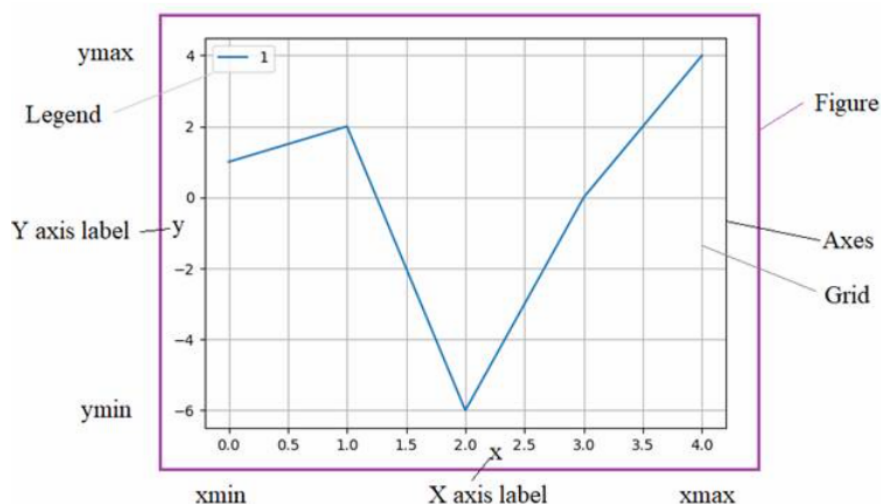
**Примечание.** Если вы используете тёмную тему в *VS Code*, то для корректного отображения графиков на тёмном фоне выполните следующую команду по установке стиля отображения:

```
plt.style.use('default')
```

## ОСНОВНЫЕ ОБЪЕКТЫ MATPLOTLIB

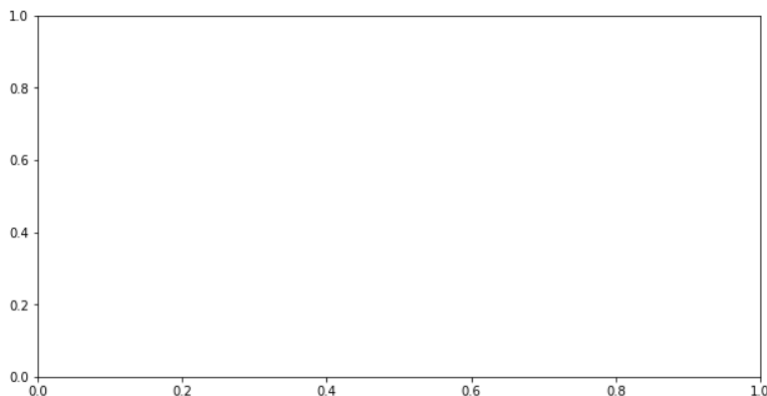
Библиотека *Matplotlib* позволяет работать в нескольких режимах. Самый распространённый и мощный по функционалу — **объектно-ориентированный режим**. Он основан на работе с объектами фигур (*figure*, их ещё называют **канвасами** или **холстами**) и координатных плоскостей (*axes*, или системы координат).

На рисунке ниже представлена визуализация основных компонентов графика в *Matplotlib*:



Процесс работы над графиком максимально прозрачен: сначала создаётся объект фигуры (*fig*), содержащий необходимую информацию и настройки, например размер в дюймах (*figsize*, восемь дюймов в ширину, четыре — в высоту). К этому объекту с помощью метода `add_axes()` добавляется координатная плоскость, а на ней располагаются графические объекты. Для создания координатной плоскости необходимо указать её расположение на фигуре в виде списка из координат. В нашем случае она начинается в левом нижнем углу без отступов (координаты 0, 0) и занимает всё отведённое место в области (100%, ширина и высота равны 1).

```
fig = plt.figure(figsize=(8, 4))
axes = fig.add_axes([0, 0, 1, 1])
```



Теперь на созданной системе координат мы можем построить график.

Утверждается, что коллективная вакцинация позволяет минимизировать риск заражения коронавирусной инфекцией (но только после второго компонента). Давайте проверим это на примере США: построим диаграмму рассеяния, которая покажет зависимость числа ежедневно обнаруживаемых случаев заражения (*daily\_confirmed*) от общего количества привитых вторым компонентом вакцины (*people\_fully\_vaccinated*) в США.

У объекта координатной плоскости `axes` вызовем метод `scatter()`.

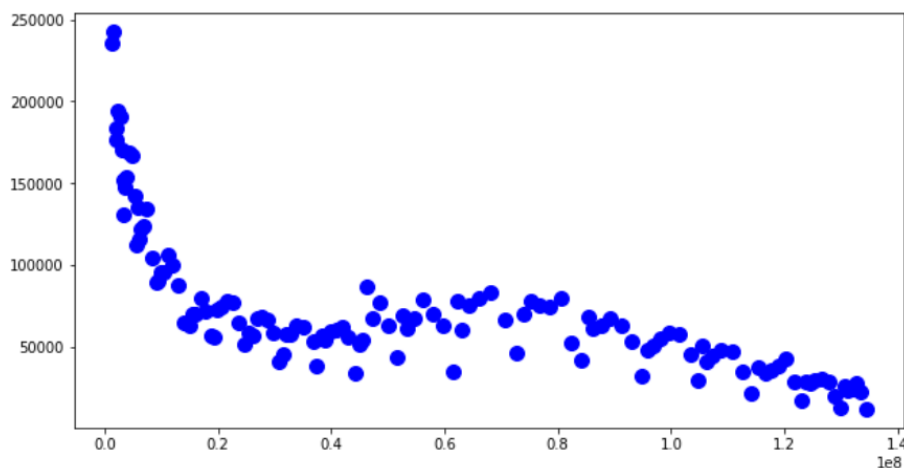
*Кликните на плашку, чтобы увидеть информацию ↓*

### Основные параметры метода `scatter()`

На **диаграмме рассеяния** по оси абсцисс откладываем суммарное число поставленных вакцин, а по оси ординат — ежедневный прирост заболевших:

```
us_data = covid_df[covid_df['country'] == 'United States']

fig = plt.figure(figsize=(8, 4))
axes = fig.add_axes([0, 0, 1, 1])
axes.scatter(
    x=us_data['people_fully_vaccinated'],
    y=us_data['daily_confirmed'],
    s=100,
    marker='o',
    c = 'blue'
);
```



Из диаграммы видно, что, в основном, с ростом числа привитых вторым компонентом людей заболеваемость падает, замедляясь на уровне около 50 тысяч заболевших в день, и продолжает снижение дальше.

Построим **круговую диаграмму**, чтобы отобразить ТОП-10 комбинаций вакцин в мире.

**Примечание.** Обратите внимание, что под «распространённостью» вакцины здесь подразумевается не количество введённых доз (таких данных у нас в таблице нет), а количество стран, в которых она используется. При этом подсчёт ведётся не по каждой отдельной вакцине, а по их комбинациям, и одна и та же вакцина учитывается несколько раз в сочетаниях с другими.

Для построения круговых диаграмм в *Matplotlib* используется метод [pie\(\)](#).

Кликните на плашку, чтобы увидеть информацию ↓

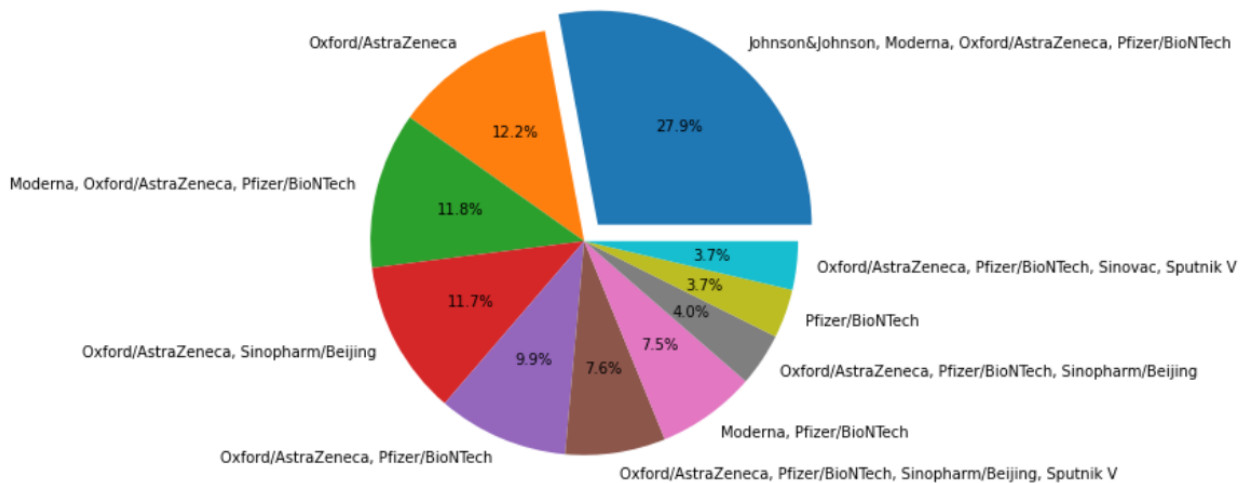
### Основные параметры метода `pie()`

ТОП-10 комбинаций вакцин (vaccines) по распространённости мы находим с помощью метода `value_counts()`. Круговую диаграмму строим на полученных значениях, метки для каждого значения — индексы промежуточной таблицы. Будем отображать доли в процентах и округлять их до одного знака после запятой. Самую распространённую вакцину сместим на 10 % от центра:

```

vaccine_combinations = covid_df['vaccines'].value_counts()[:10]
fig = plt.figure(figsize=(5, 5))
axes = fig.add_axes([0, 0, 1, 1])
axes.pie(
    vaccine_combinations,
    labels=vaccine_combinations.index,
    autopct='%1f%%',
    explode = [0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
);

```



Самой распространённой комбинацией вакцин является комбинация: *Johnson&Johnson, Moderna, Oxford/AstraZeneca, Pfizer/BioNTech*. Причём можно заметить, что в большинстве популярных комбинаций присутствует вакцина *Oxford/AstraZeneca*.

► Примечание (для внимательных и любознательных)

## ДОБАВЛЕНИЕ ИНФОРМАТИВНОСТИ В ГРАФИКИ

Вы, наверное, заметили, что до этого мы не подписывали графики. График не имеет смысла, если без лишних слов непонятно, что на нём изображено. Управлять информативностью графика можно с помощью методов координатной плоскости `axes`. Перечислим основные из них (не пугайтесь, запоминать их не обязательно — вы всегда сможете подсмотреть их в документации):

- `axes.set_title()` — заголовок
- `axes.xaxis.set_tick_params()`

диаграммы, а также его настройки (например, параметр `fontsize` отвечает за размер шрифта);

- `axes.set_xlabel()` — название оси абсцисс;
- `axes.set_ylabel()` — название оси ординат;
- `axes.set_xticks()` — установка отметок на оси абсцисс;
- `axes.set_yticks()` — установка отметок на оси ординат;

— управление параметрами отметок на оси абсцисс (например, параметр `rotation` отвечает за поворот отметок в градусах);

- `axes.yaxis.set_tick_params()` — управление параметрами отметок на оси ординат;
- `axes.legend()` — отображение легенды;
- `axes.grid()` — установка сетки.

Например, изобразим на одном графике, как росла общая заболеваемость (*confirmed*), число зафиксированных смертей (*deaths*), выздоровевших пациентов (*recovered*) и активных случаев (*active*) в Китае.

Для построения линейных графиков в *Matplotlib* используется метод `plot()` (*не путайте с методом `plot()` в *Pandas*!*). При вызове метода без параметров по оси ординат откладываются значения столбца таблицы, по оси абсцисс — индексы (в нашем случае это будут даты).

Дополнительно в параметрах метода указываем параметр `label` — название графика, которое будет отображаться на легенде, а также `lw` — ширину линии графика. Добавим к графику заголовок, названия осей, установим метки по оси `y` с частотой в 10 000, повернём метки по оси `x` на 30 градусов, а также добавим легенду (метки для легенды выставляются в параметре `label` метода `plot()`):

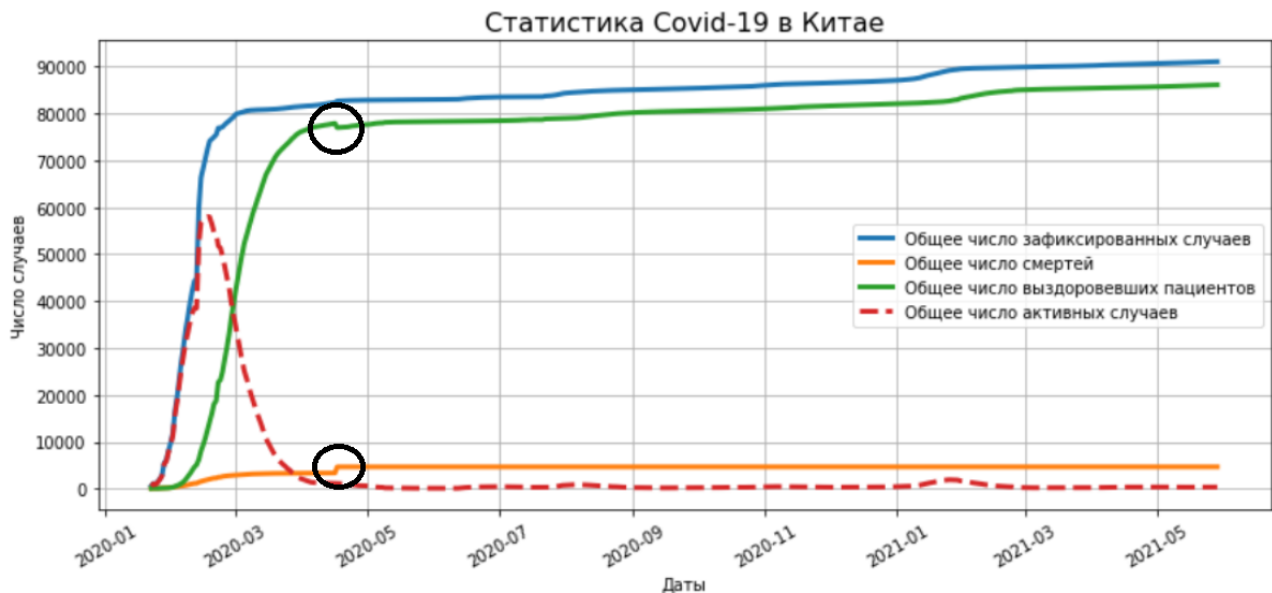
```

china_data = covid_df[covid_df['country'] == 'China']
china_grouped = china_data.groupby(['date'])[['confirmed', 'active',
'deaths', 'recovered']].sum()

#визуализация графиков
fig = plt.figure(figsize=(10, 4))
axes = fig.add_axes([0, 0, 1, 1])
axes.plot(china_grouped['confirmed'], label='Общее число
зафиксированных случаев', lw=3)
axes.plot(china_grouped['deaths'], label='Общее число смертей', lw=3)
axes.plot(china_grouped['recovered'], label='Общее число выздоровевших
пациентов', lw=3)
axes.plot(china_grouped['active'], label='Общее число активных случаев',
lw=3, linestyle='dashed')

#установка параметров отображения
axes.set_title('Статистика Covid-19 в Китае', fontsize=16)
axes.set_xlabel('Даты')
axes.set_ylabel('Число случаев')
axes.set_yticks(range(0, 100000, 10000))
axes.xaxis.set_tick_params(rotation=30)
axes.grid()
axes.legend();

```



На графике наблюдается резкий рост заболеваемости на начальном периоде до середины февраля 2020 года, после чего видно резкое падение числа активных случаев (примерно в этот период власти Китая ввели тотальный локдаун и прекратили транспортное сообщение со всем миром). Далее число активных



случаев только падает, а темп прироста числа заболевших снижается. При этом общее число смертей в Китае остаётся практически на одном уровне (около 5 тысяч), уже начиная с мая 2020 года (новые пациенты умирают крайне редко).

Стоит обратить внимание на выделенные **чёрным** маркером сдвиги в графиках. Они являются свидетельством противоречия в данных: общее число выздоровевших пациентов почему-то резко упало, а число умерших возросло. Почему так произошло? Произошёл пересчёт числа заболевших? Данные были утеряны или искажены? Если вы в своей практике столкнётесь с такими же противоречиями, вам необходимо будет обратиться к первоисточнику, чтобы выяснить причину.

## ИСПОЛЬЗОВАНИЕ НЕСКОЛЬКИХ СИСТЕМ КООРДИНАТ

При использовании библиотеки *Matplotlib* вовсе не обязательно ограничиваться одной системой координат.

→ Вы можете размещать несколько систем координат на одной фигуре, что позволит нам отображать вспомогательную информацию на основном графике.

Для добавления второй системы координат необходимо повторно применить к объекту `fig` метод `add_axes`, указав новое имя для второй системы координат.

Например, отобразим ТОП-5 стран по общему числу привитых вторым компонентом людей (*people\_fully\_vaccinated*), а также ТОП-5 стран по числу полностью привитых на 100 человек населения (*people\_fully\_vaccinated\_per\_hundred*).

Для этого построим столбчатые диаграммы с помощью метода `bar()`.

Кликните на плашку, чтобы увидеть информацию ↓

### Основные параметры метода `bar()`

Группируем таблицу по странам, находим последний по дате зафиксированный показатель с помощью метода `last()` и выбираем ТОП-5 стран с использованием метода `nlargest()`.

При отображении графиков создаём две координатные плоскости `main_axes` и `insert_axes`, на каждой из них отдельно строим столбчатые диаграммы.

```
vacc_country = covid_df.groupby('country')
['people_fully_vaccinated'].last().nlargest(5)
vacc_country_per_hundred = covid_df.groupby('country')
['people_fully_vaccinated_per_hundred'].last().nlargest(5)

#визуализация главного графика
fig = plt.figure(figsize=(13, 4))
main_axes = fig.add_axes([0, 0, 1, 1])
main_axes.bar(x = vacc_country.index, height = vacc_country);
main_axes.set_ylabel('Число вакцинированных (2 компонент)')
main_axes.set_title('Топ 5 стран по числу полностью привитых людей')

#визуализация вспомогательного графика
insert_axes = fig.add_axes([0.6, 0.6, 0.38, 0.38])
insert_axes.bar(x = vacc_country_per_hundred.index, height =
vacc_country_per_hundred, width=0.5);
insert_axes.set_ylabel('На 100 человек')
insert_axes.xaxis.set_tick_params(rotation=45)
```



#### ► Открыть примечание

Нетрудно заметить, что два представленных рейтинга отличаются: лидером по числу полностью привитых является США, а вот по числу вакцин на 100 человек населения — Сейшель. В первый список попали страны с большим количеством населения, которое они активно прививают. Во второй список попали маленькие страны, которые проще всего обеспечить вакциной.

В наш ТОП не попали страны, которые пользуются однокомпонентной вакциной, например Китай.

## SUBPLOTS

Создание дополнительных систем координат с помощью метода `add_axes()` полезно, однако используется не так часто.

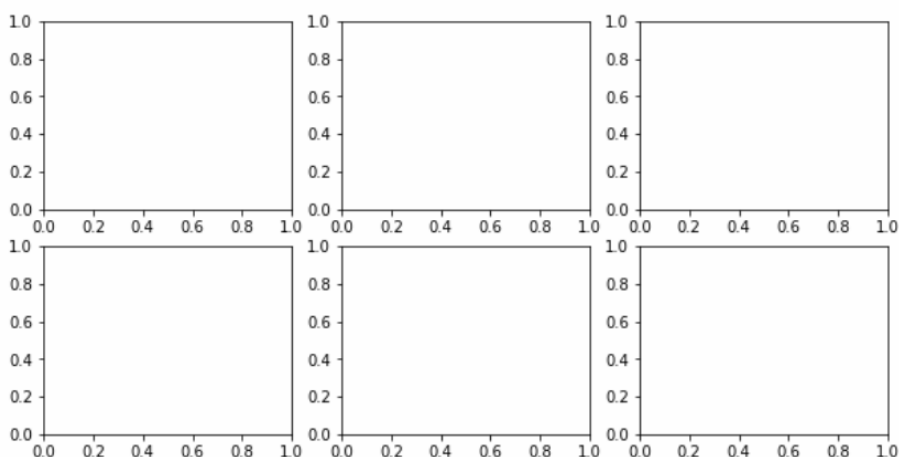
В большинстве случаев для отображения нескольких систем координат используется функция `subplots()`. Она создаёт целую таблицу из систем координат на одной фигуре. Функция возвращает новую фигуру, а также список координатных плоскостей.

*Кликните на плашку, чтобы увидеть информацию ↓*

### Основные параметры метода `subplots()`

Например, следующий код создаст шесть координатных плоскостей, сведённых в таблицу размера 2x3:

```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10, 5))
```



Теперь, обладая знаниями о методе `subplots()`, построим три графика:

1. Столбчатую диаграмму, которая покажет динамику ежедневной вакцинации в России.
2. Линейный график изменения ежедневной заболеваемости в стране.

### 3. Гистограмму ежедневной заболеваемости в стране.

За построение гистограмм в библиотеке *Matplotlib* отвечает метод hist().

*Кликните на плашку, чтобы увидеть информацию ↓*

#### Основные параметры метода hist()

Фильтруем таблицу `covid_df` по признаку страны и выбираем записи только для России.

Для того чтобы отобразить график в соответствующей координатной плоскости, нужно обратиться к списку `axes` по индексу (от 0 до 2). Дальнейшая настройка графиков вам уже известна.

```

russia_data = covid_df[covid_df["country"] == "Russia"]

# визуализация систем координат
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 4))

# столбчатая диаграмма
axes[0].bar(
    x=russia_data["date"],
    height=russia_data["daily_vaccinations"],
    label="Число вакцинированных",
)
axes[0].set_title("Ежедневная вакцинация в России")
axes[0].xaxis.set_tick_params(rotation=45)

# линейный график
axes[1].plot(
    russia_data["date"],
    russia_data["daily_confirmed"],
    label="Число заболевших",
    color="tomato",
    lw=2,
)
axes[1].set_title("Ежедневная заболеваемость в России")
axes[1].xaxis.set_tick_params(rotation=45)

# гистограмма
axes[2].hist(
    x=russia_data["daily_confirmed"], label=["Число заболевших"],
    color="lime", bins=20
)
axes[2].set_title("Гистограмма заболеваемости в России")
axes[2].xaxis.set_tick_params(rotation=30)

```



- На первом графике можно наблюдать колеблющийся рост числа ежедневно вакцинированных людей. Особенно в глаза бросается «пенёк» в период с конца января до начала марта 2021 года. Это период, когда данные о процессе вакцинации людей не обновлялись.
- На втором графике мы видим две волны коронавируса в России. Первая — в середине марта 2020 года, которая достигла максимума в 13 тысяч заболевших за сутки. Вторая волна, судя по графику, началась в октябре 2020 года и достигла своего пика почти в 30 тысяч заболевших за сутки в конце декабря этого же года (точные данные: 24 декабря было зафиксировано рекордное число подтверждённых случаев: 29935).

Далее с ростом показателей вакцинации и введением новых карантинных мер заболеваемость снова постепенно снижается.

- На третьем графике можно увидеть, что большая часть наблюдений ежедневной заболеваемости находится в интервале от 5 до 10 тысяч человек в день. Ещё один пик гистограммы находится около 0 — это случаи, зафиксированные на начальных этапах эпидемии (в Россию *Covid-19* пришёл позже, чем во многие другие страны).

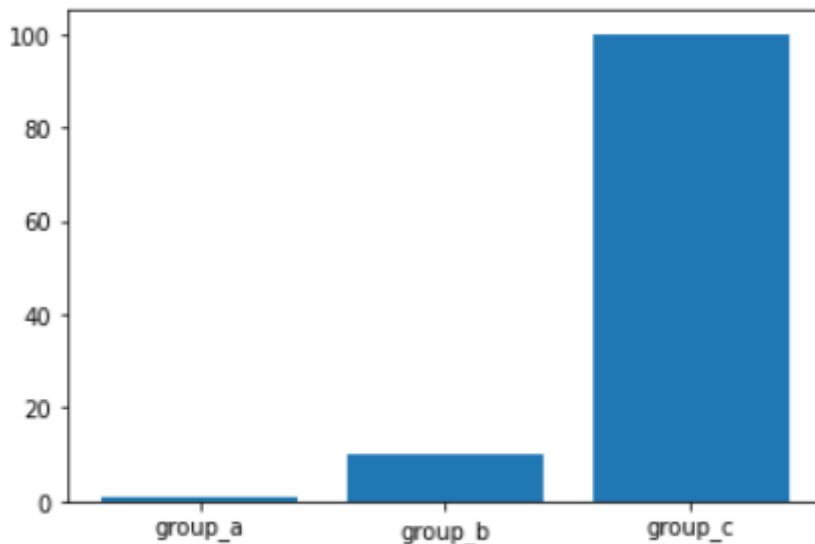
👉 Мы рассмотрели лишь основные графики и их настройки в библиотеке *Matplotlib*. На самом деле библиотека имеет гораздо больший спектр возможностей, на изучение которого не хватит даже целого курса по визуализации, — от добавления текста на диаграмму до визуализации изображений и 3D-графиков.

Если вам вдруг понадобится какая-то особенная функциональность *Matplotlib*, которую мы не рассматривали, рекомендуем поискать её в [документации по библиотеке](#).

Также стоит отметить, что, помимо объектно-ориентированного подхода в работе с библиотекой *Matplotlib*, вы можете встретить и **модульный подход**.

Модульный подход основан на обращении к модулю *pyplot* (*plt*) напрямую, а не средствами объектов фигур и плоскостей. Например, следующий код строит столбчатую диаграмму: по оси *x* откладываются элементы списка *names* (названия групп), а высоту столбцов определяет список *values*.

```
names = ['group_a', 'group_b', 'group_c']  
values = [1, 10, 100]  
plt.bar(names, values)  
plt.show()
```



И модульный, и объектно-ориентированный подходы имеют одинаковое право на существование. Ознакомьтесь со [статьёй](#), где используется преимущественно модульный подход, и проведите параллель с изученным материалом.

А пока предлагаем вам закрепить знания, ответив на **несколько вопросов**  
↓

## Задание 5.1

1/1 point (graded)

Выберите основные объекты в библиотеке *Matplotlib* при использовании объектно-ориентированного режима:

- ☐ Координатные плоскости и заголовки графиков

☐ Плоскости и поверхности


☒ Фигуры (холсты) и координатные плоскости ✓

☐ Выпуклые и невыпуклые фигуры

Отправить

## Задание 5.2

1/1 балл (оценивается)

 Справка по клавиатуре

Соотнесите методы объекта axes и графики, которые они строят (при возникновении затруднений обратитесь к [документации](#)).

axes.plot()

**Линейный график**

axes.boxplot()

**Коробчатая диаграмма**

axes.scatter()

**Диаграмма рассеяния**

axes.bar()

**Столбчатая диаграмма**


axes.hist()

**Гистограмма**




Сбросить

## РЕЗУЛЬТАТ

 Хорошо! Вы справились с этим заданием.

## Задание 5.3

1/1 балл (оценивается)

 Справка по клавиатуре

Соотнесите методы объекта axes и параметры, которые эти методы регулируют



`axes.set_xlabel()`**Название оси  
абсцисс**`axes.set_ylabel()`**Название оси  
ординат**`axes.legend()`**Легенда**`axes.xaxis.set_tick_  
params()`**Параметры  
отметок на оси  
абсцисс**`axes.grid()`**Сетка на  
графике**`axes.set_title()`**Название  
графика**

Сбросить

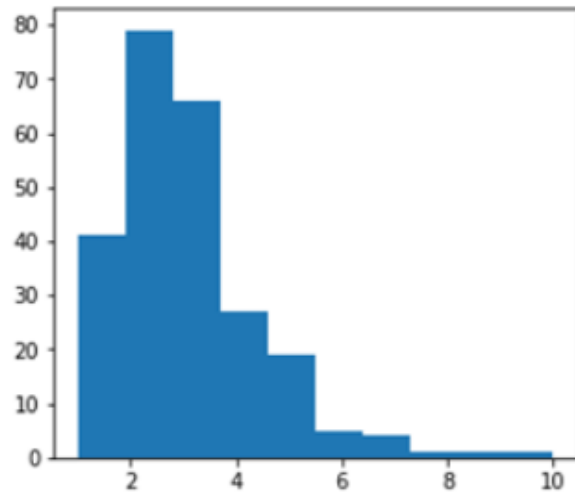
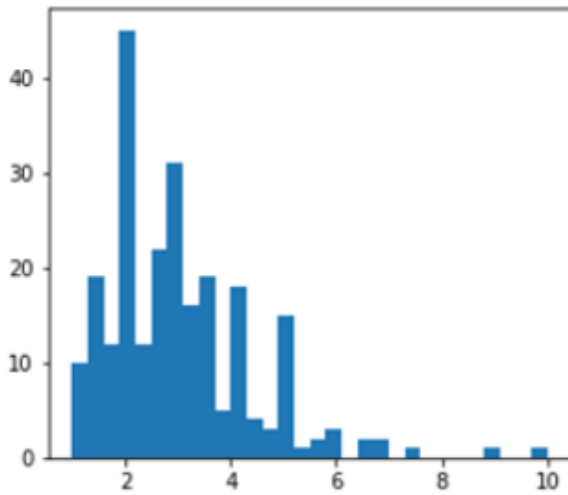
## РЕЗУЛЬТАТ

**i** Хорошо! Вы справились с этим заданием.

## Задание 5.4

1/1 point (graded)

По какому параметру отличаются друг от друга представленные ниже графики?

☐ xlim☐ Источник данных☒ bins ✓☐ ncolumns

### Ответ

Верно:

Гистограммы имеют одинаковый диапазон по осям x и y и отличаются только числом столбцов. За число столбцов в гистограмме отвечает именно параметр `bins`.

Отправить

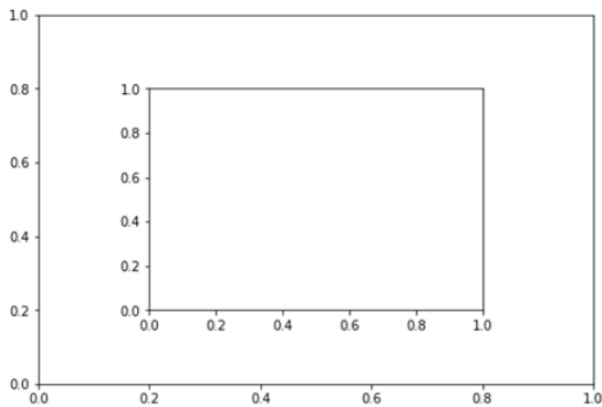
## Задание 5.5

1/1 point (graded)

Дан код для отрисовки основной системы координат:

```
fig = plt.figure(figsize=(13, 4))  
main_axes = fig.add_axes([0, 0, 1, 1])
```

Выберите правильный набор параметров, позволяющих разместить вспомогательную ось так, как показано на заготовке графика ниже.



Прежде чем заносить варианты кода в ноутбук, попробуйте определить верный вариант ответа логически.



```
insert_axes = fig.add_axes([0.5, 0.5, 0.5, 0.5])
```



```
insert_axes = fig.add_axes([0.2, 0.2, 0.6, 0.6])
```



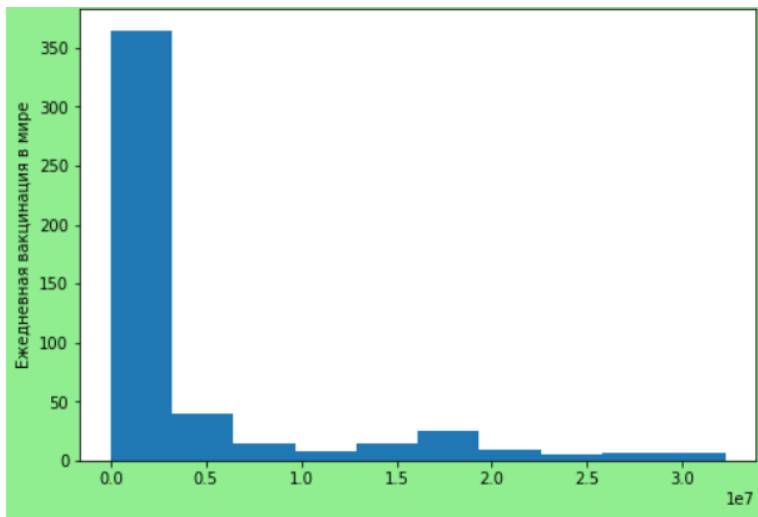
```
insert_axes = fig.add_axes([0.6, 0.6, 0.2, 0.2])
```

Отправить

## Задание 5.6

1/1 point (graded)

Выберите вариант кода, который позволит построить приведённый ниже график (посмотрите в документации параметры объекта [figure](#), которые позволяют раскрашивать части фигуры).



Прежде чем заносить варианты кода в ноутбук, попробуйте определить верный вариант ответа логически.



```
fig = plt.figure(facecolor='lightgreen')
axes = fig.add_axes([0, 0, 1, 1])
axes.hist(covid_df.groupby('date')['daily_vaccinations'].sum())
axes.set_ylabel('Ежедневная вакцинация в мире');
```



```
fig = plt.figure(edgecolor='lightgreen')
axes = fig.add_axes([0, 0, 1, 1])
axes.hist(covid_df.groupby('date')['daily_vaccinations'].sum())
axes.set_ylabel('Ежедневная вакцинация в мире');
```



```
fig = plt.figure(facecolor='lightgreen')
axes = fig.add_axes([0, 0, 1, 1])
axes.hist(covid_df.groupby('date')['total_vaccinations'].sum())
axes.set_xlabel('Ежедневная вакцинация в мире');
```



```
fig = plt.figure(facecolor='lightgreen')
axes = fig.add_axes([0, 0, 1, 1])
axes.hist(covid_df.groupby('date')['daily_vaccinations'].mean())
axes.set_xlabel('Ежедневная вакцинация в мире');
```

Отправить

## Задание 5.7

1/1 point (graded)

Какое название лучше всего подойдёт графику, который строит код ниже?

```
v = covid_df.groupby(['country'])['total_vaccinations'].last().nlargest(5)
fig = plt.figure(figsize=(12, 4))
axes = fig.add_axes([0, 0, 1, 1])
axes.bar(x=v.index,height=v)
```

☐ ТОП-5 стран по числу заболевших на 1 млн человек

☐ ТОП-5 стран с наименьшими показателями летальности

☒ ТОП-5 стран по общему числу сделанных прививок ✓

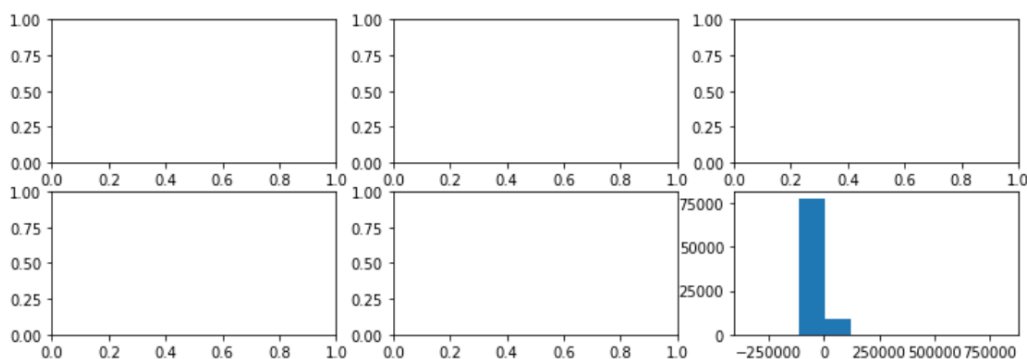
☐ ТОП-5 стран по средней ежедневной вакцинации на 1 млн человек

Отправить

## Задание 5.8

1/1 point (graded)

Подумайте (или посмотрите в документации), по каким индексам надо обратиться к списку осей `axes`, полученному с помощью метода `subplots()`, чтобы построить такой график:



☐ `axes[3][2]`☒ `axes[1, 2]` ✓☐ `axes[2][1]`☐ `axes[1][2]`

© Все права защищены

[Help center](#) [Политика конфиденциальности](#) [Пользовательское соглашение](#)Built on 