РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Основы работы с пакетом matplotlib»

Отчет по лабораторной работе № 3.4

по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-21-1

Толубаев Рамиль Ахметович

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности библиотеки matplotlib языка программирования Python.

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

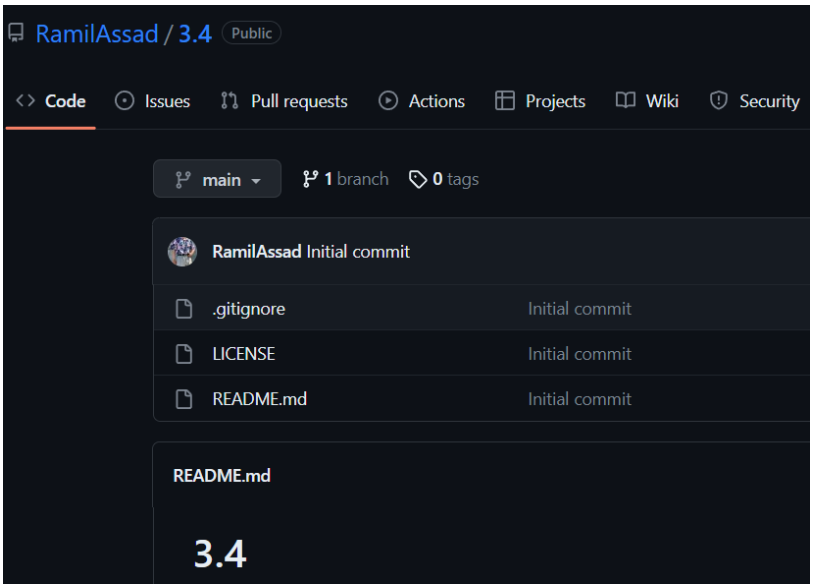


Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
Cloning into 'lw_3.4'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 4.13 KiB | 704.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.4>git checkout -b develop
Switched to a new branch 'develop'

C:\Users\Asus\Desktop\Учеба\4 семестр\Анализ данных\lw_3.4>
```

Рисунок 3 - Ветвление по модели git-flow

4. Проработать примеры лабораторной работы.

Пример 1.

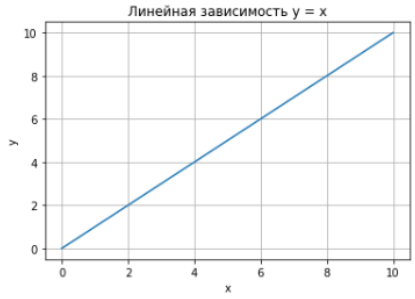
Построение графика

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
# Независимая (x) и зависимая (y) переменные
x = np.linspace(0, 10, 50)
y = x
```

In [5]:
```
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y) # построение графика
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x1ccdeefcfd0>]
```



In [6]:
```
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y, "r--") # построение графика
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1ccdf00f730>]
```

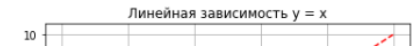


Рисунок 4 - Результат выполнения примера 1

Пример 2.

Несколько графиков на одном поле

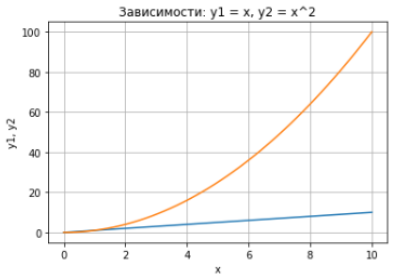
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
```

In [4]:
```
# Квадратичная зависимость
y2 = [i**2 for i in x]
```

In [5]:
```
# Построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y1, y2") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y1, x, y2) # построение графика
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x2763c2173d0>,
 <matplotlib.lines.Line2D at 0x2763c217400>]
```



In [ ]:

Рисунок 5 - Результат выполнения примера 2

Пример 3.

Несколько разделенных полей с графиками

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
```

In [3]:
```
# Квадратичная зависимость
y2 = [i**2 for i in x]
```

In [4]:
```
# Построение графиков
plt.figure(figsize=(9, 9))

plt.subplot(2, 1, 1)
plt.plot(x, y1) # построение графика

plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.ylabel("y1", fontsize=14) # ось ординат
plt.grid(True) # включение отображение сетки

plt.subplot(2, 1, 2)
plt.plot(x, y2) # построение графика

plt.xlabel("x", fontsize=14) # ось абсцисс
plt.ylabel("y2", fontsize=14) # ось ординат
plt.grid(True)
```

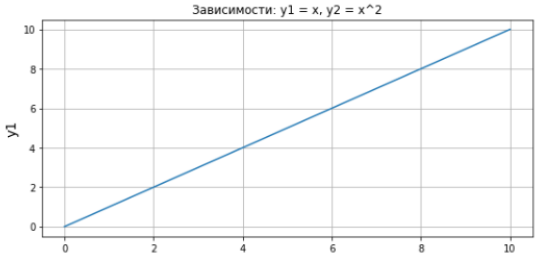


Рисунок 6 - Результат выполнения примера 3

Пример 4.

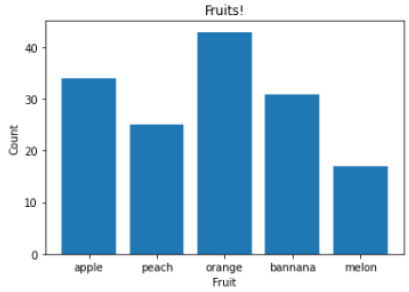
Построение диаграммы для категориальных данных

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
fruits = ["apple", "peach", "orange", "bannana", "melon"]
counts = [34, 25, 43, 31, 17]
```

In [3]:
```
plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
```

Out[3]:
```
Text(0, 0.5, 'Count')
```



In [ ]:

Рисунок 7 - Результат выполнения примера 4

Пример 5.

Основные элементы графика

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
                               AutoMinorLocator)
```

In [4]:
```
x = np.linspace(0, 10, 10)
y1 = 4*x
y2 = [i**2 for i in x]
```

In [7]:
```
fig, ax = plt.subplots(figsize=(8, 6))

ax.set_title("Графики зависимостей: y1=4*x, y2=x^2", fontsize=16)
ax.set_xlabel("x", fontsize=14)
ax.set_ylabel("y1, y2", fontsize=14)
ax.grid(which="major", linewidth=1.2)
ax.grid(which="minor", linestyle="--", color="gray", linewidth=0.5)

ax.scatter(x, y1, c="red", label="y1 = 4*x")
ax.plot(x, y2, label="y2 = x^2")

ax.legend()

ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())

ax.tick_params(which="major", length=10, width=2)
ax.tick_params(which="minor", length=5, width=1)

plt.show()
```

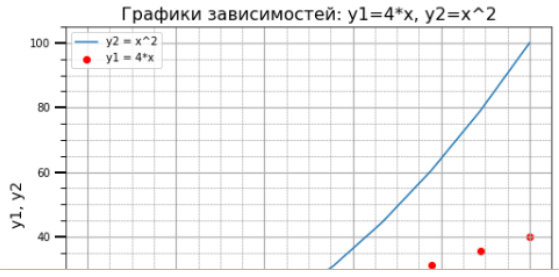


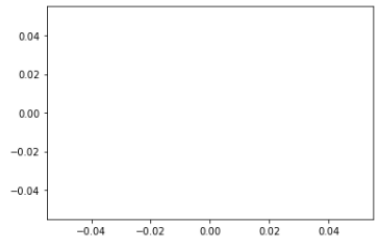
Рисунок 8 - Результат выполнения примера 5

Пример 6.

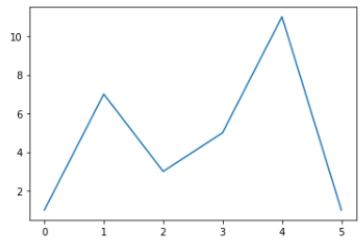
Работа с инструментом pyplot

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: plt.plot()
Out[2]: []
```




```
In [3]: plt.plot([1, 7, 3, 5, 11, 1])
Out[3]: [<matplotlib.lines.Line2D at 0x208c4653b80>]
```




```
In [4]: plt.plot([1, 5, 10, 15, 20], [1, 7, 3, 5, 11])
Out[4]: [<matplotlib.lines.Line2D at 0x208c46cd280>]
```

Рисунок 9 - Результат выполнения примера 6

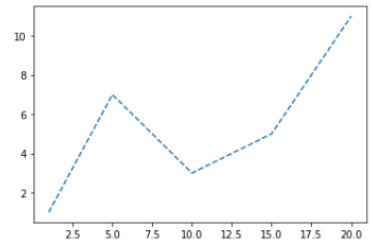
Пример 7

Работа с линейным графиком

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: x = [1, 5, 10, 15, 20]
        y = [1, 7, 3, 5, 11]

In [3]: line = plt.plot(x, y)
        plt.setp(line, linestyle='--')
Out[3]: [None]
```




```
In [4]: x = [1, 5, 10, 15, 20]
        y1 = [1, 7, 3, 5, 11]
        y2 = [i*1.2 + 1 for i in y1]
        y3 = [i*1.2 + 1 for i in y2]
        y4 = [i*1.2 + 1 for i in y3]

In [5]: plt.plot(x, y1, '-', x, y2, '--', x, y3, '-.', x, y4, ':')
Out[5]: [<matplotlib.lines.Line2D at 0x27e90af3280>,
         <matplotlib.lines.Line2D at 0x27e90af33d0>,
         <matplotlib.lines.Line2D at 0x27e90af33a0>,
         <matplotlib.lines.Line2D at 0x27e90af35e0>]
```

Рисунок 10 - Результат выполнения примера 7

Пример 8

Размещение графиков на разных полях

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Исходный набор данных
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
```

In [3]:
```
# Настройка размеров подложки
plt.figure(figsize=(12, 7))

# Вывод графиков
plt.subplot(2, 2, 1)
plt.plot(x, y1, '-')

plt.subplot(2, 2, 2)
plt.plot(x, y2, '--')

plt.subplot(2, 2, 3)
plt.plot(x, y3, '-.')

plt.subplot(2, 2, 4)
plt.plot(x, y4, ":")
```

Out[3]: `[<matplotlib.lines.Line2D at 0x190dbfe2a90>]`

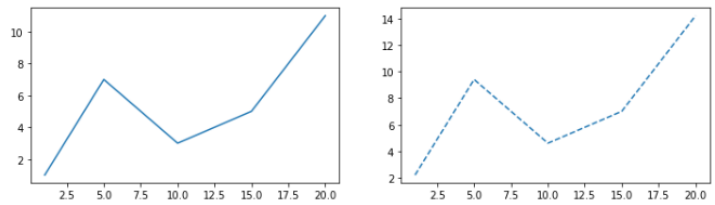


Рисунок 11 - Результат выполнения примера 8

Контрольные вопросы:

1. Как осуществляется установка пакета matplotlib?

Существует два основных варианта установки этой библиотеки: в первом случае вы устанавливаете пакет Anaconda, в состав которого входит большое количество различных инструментов для работы в области машинного обучения и анализа данных (и не только).

Второй вариант – это воспользоваться менеджером pip и установить Matplotlib самостоятельно, для этого введите в командной строке вашей операционной системы следующие команды:

$ python -m pip install -U pip

$ python -m pip install -U matplotlib

2. Какая "магическая" команда должна присутствовать в ноутбуках Jupyter для корректного отображения графиков matplotlib?

%matplotlib inline

3. Как отобразить график с помощью функции plot ?

Передать в качестве параметров значения x и y.

x = np.linspace(0, 10, 50)

y = x

plt.plot(x, y)

4. Как отобразить несколько графиков на одном поле?

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.

5. Какой метод Вам известен для построения диаграмм категориальных данных?

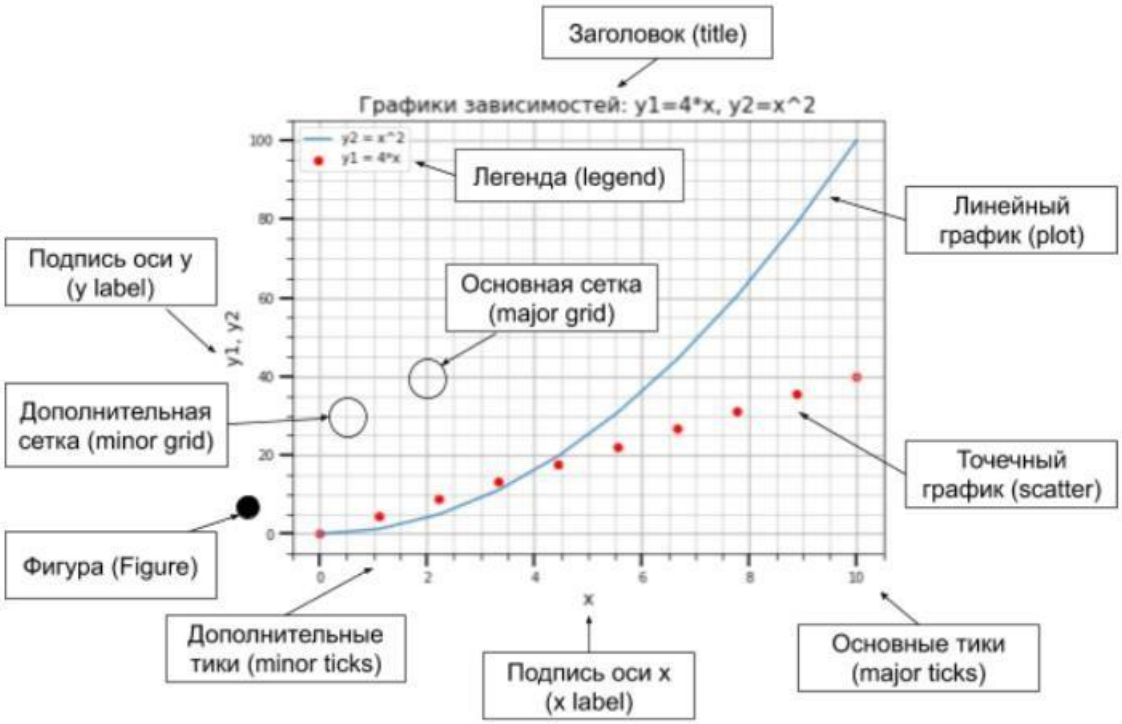
Использование функции bar().

fruits = ["apple", "peach", "orange", "bannana", "melon"]

counts = [34, 25, 43, 31, 17]

plt.bar(fruits, counts)

6. Какие основные элементы графика Вам известны?



7. Как осуществляется управление текстовыми надписями на графике?

Для задания подписи **оси** x используется функция xlabel(), оси y – ylabel().

Функции xlabel()/ylabel() принимают в качестве аргументов параметры конструктора класса matplotlib.text.Text.

plt.xlabel('Day', fontsize=15, color='blue')

Для задания **заголовка** графика используется функция title():

plt.title('Chart price', fontsize=17)

За размещение **текста** на поле графика отвечает функция text(), которой вначале передаются координаты позиции надписи, после этого – текст самой надписи.

plt.text(1, 1, 'type: Steel')

8. Как осуществляется управление легендой графика?

Легенда будет размещена на графике, если вызвать функцию legend().

9. Как задать цвет и стиль линий графика?

Параметры, которые отвечают за отображение графика можно задать непосредственно в самой функции plot():

plt.plot(x, y, color='red')

Либо воспользоваться функцией setp(), через которую можно модифицировать нужные параметры:

plt.setp(color='red', linewidth=1)

Стиль линии графика задается через параметр linestyle, который может принимать значения из приведенной ниже таблицы.

Значение параметра	Описание
'-' или 'solid'	Непрерывная линия
'–' или 'dashed'	Штриховая линия
'-.' или 'dashdot'	Штрихпунктирная линия
':' или 'dotted'	Пунктирная линия
'None' или ' ' или ''	Не отображать линию

Стиль линии можно передать сразу после указания списков с координатами без указания, что это параметр linewidth.

plt.plot(x, y, '--')

Задание цвета линии графика производится через параметр color (или c, если использовать сокращенный вариант).

10. Как выполнить размещение графика в разных полях?

Работа с функцией subplot()

Самый простой способ представить графики в отдельных полях – это использовать функцию supplot() для задания их мест размещения. До этого момента мы не работали с Фигурой (Figure) напрямую, значения ее параметров, задаваемые по умолчанию, нас устраивали. Для решения текущей задачи придется один из параметров – размер подложки, задать вручную. За это отвечает аргумент figsize функции figure(), которому присваивается кортеж из двух float элементов, определяющих высоту и ширину подложки.

После задания размера, указывается местоположение, куда будет установлено поле с графиком с помощью функции subplot(). Чаще всего используют следующие варианты вызова subplot:

subplot(nrows, ncols, index)

subplot(pos)

plt.figure(figsize=(12, 7))

Вывод графиков

```
plt.subplot(221)
plt.plot(x, y1, '-')

plt.subplot(222)
plt.plot(x, y2, '--')

plt.subplot(223)
plt.plot(x, y3, '-.')

plt.subplot(224)
plt.plot(x, y4, ':')
```

Работа с функцией subplots()

Одно из неудобств использования последовательного вызова функций subplot() заключается в том, что каждый раз приходится указывать количество строк и столбцов сетки. Для того, чтобы этого избежать, можно воспользоваться функцией subplots(), из всех ее параметров, нас пока интересуют только первые два, через них передается количество строк и столбцов сетки.

Функция subplots() возвращает два объекта, первый – это Figure, подложка, на которой будут размещены поля с графиками, второй – объект или массив объектов Axes, через которые можно получить полных доступ к настройке внешнего вида отображаемых элементов.

```
fig, axs = plt.subplots(2, 2, figsize=(12, 7))
axs[0, 0].plot(x, y1, '-')
axs[0, 1].plot(x, y2, '--')
axs[1, 0].plot(x, y3, '-.')
axs[1, 1].plot(x, y4, ':')
```

Вывод: были исследованы базовые возможности библиотеки matplotlib языка программирования Python.