МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе № 3.5 Визуализация данных с помощью matplotlib

Выполнил:
Шальнев Владимир Сергеевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

Проработанные примеры:

Проверка установки

Ввод [1]:
```
import matplotlib
matplotlib.__version__
```

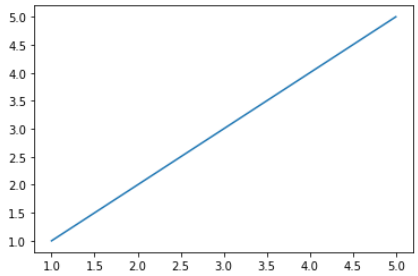
Out[1]:
```
'3.3.4'
```

Быстрый старт

Ввод [2]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Ввод [3]:
```
plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])
```

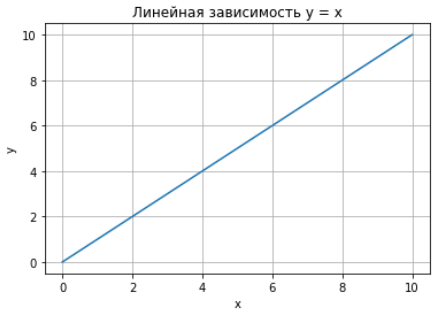
Out[3]:
```
[<matplotlib.lines.Line2D at 0x2ad062bda00>]
```



Построение графика

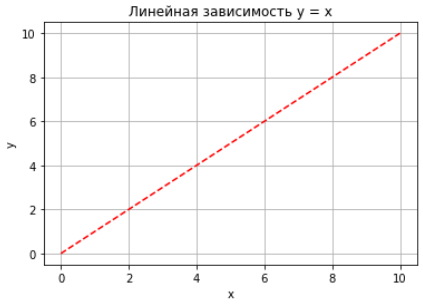
Ввод [4]:
```
import numpy as np
# Независимая (x) и зависимая (y) переменные
x = np.linspace(0, 10, 50)
y = x
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y) # построение графика
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x2ad06a42b20>]
```



Ввод [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, "r--") # построение графика
```

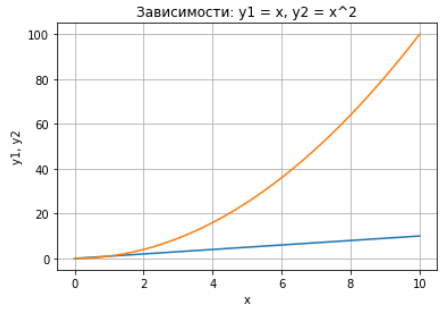
Out[5]:
```
[<matplotlib.lines.Line2D at 0x2ad06aa9850>]
```



Несколько графиков на одном поле

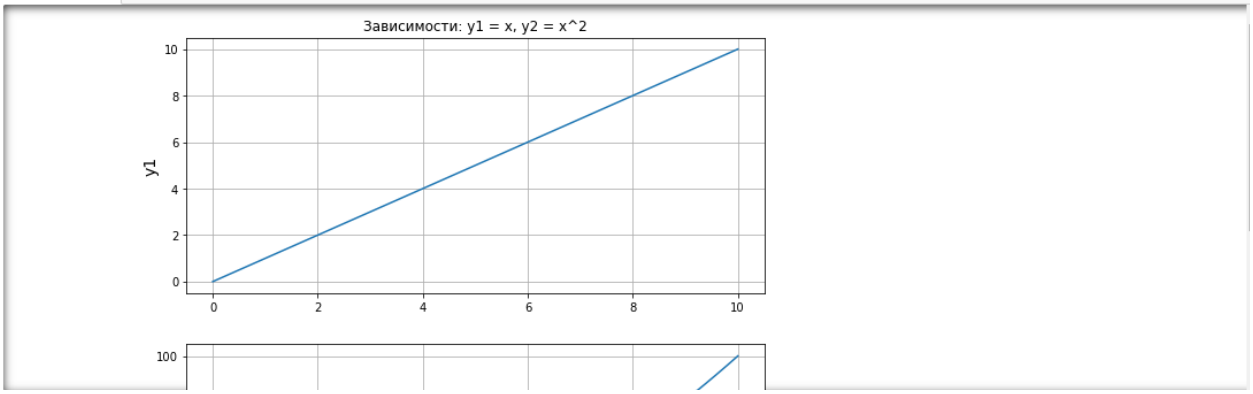
Ввод [6]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# Построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y1, y2") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y1, x, y2) # построение графика
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x2ad06b0e610>,
 <matplotlib.lines.Line2D at 0x2ad06b0e6d0>]
```



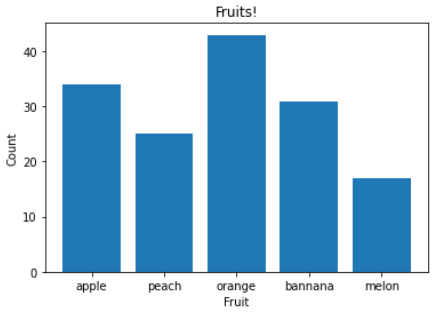
Несколько разделенных полей с графиками

Ввод [28]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# 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)
```

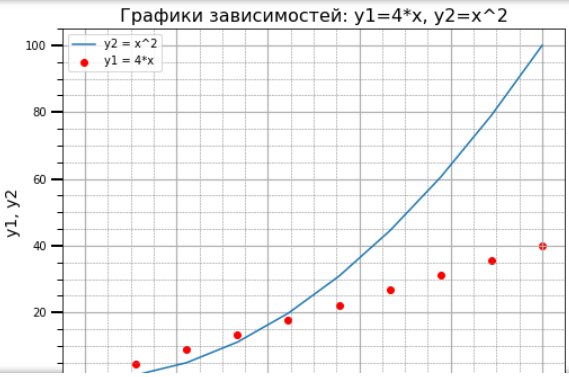


Ввод [8]:
```
fruits = ["apple", "peach", "orange", "bannana", "melon"]
counts = [34, 25, 43, 31, 17]
plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
```

Out[8]:
```
Text(0, 0.5, 'Count')
```



Ввод [9]:
```
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter, AutoMinorLocator)
x = np.linspace(0, 10, 10)
y1 = 4*x
y2 = [i**2 for i in x]
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()
```

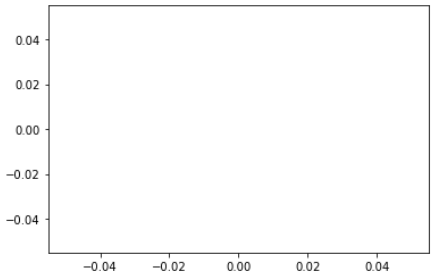


Работа с инструментом pyplot

Построение графиков

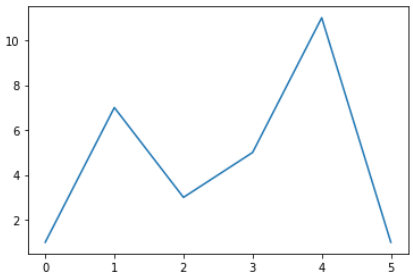
Ввод [10]:
```
plt.plot()
```

Out[10]:
```
[]
```



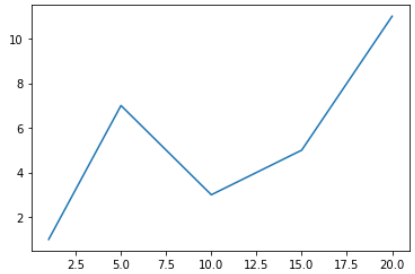
Ввод [11]:
```
plt.plot([1, 7, 3, 5, 11, 1])
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x2ad06f1bc70>]
```



Ввод [12]:
```
plt.plot([1, 5, 10, 15, 20], [1, 7, 3, 5, 11])
```

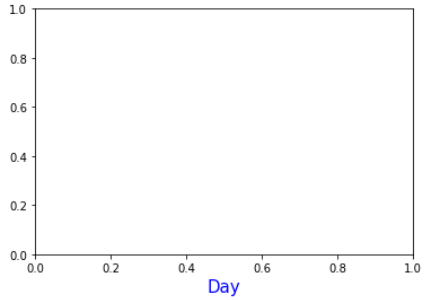
Out[12]:
```
[<matplotlib.lines.Line2D at 0x2ad06f7f1f0>]
```



Текстовые надписи на графике

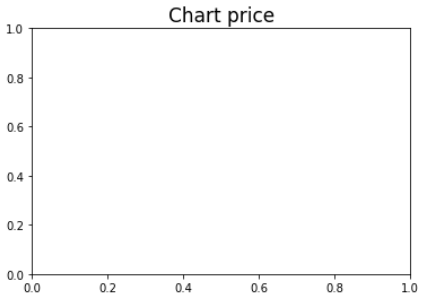
Ввод [13]:
```
plt.xlabel('Day', fontsize=15, color='blue')
```

Out[13]:
```
Text(0.5, 0, 'Day')
```



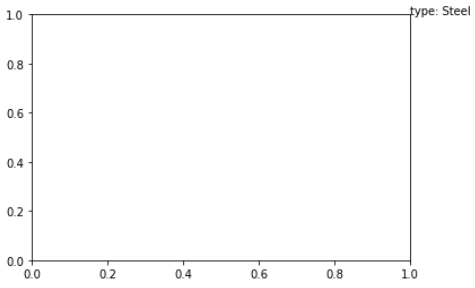
Ввод [14]:
```
plt.title('Chart price', fontsize=17)
```

Out[14]:
```
Text(0.5, 1.0, 'Chart price')
```



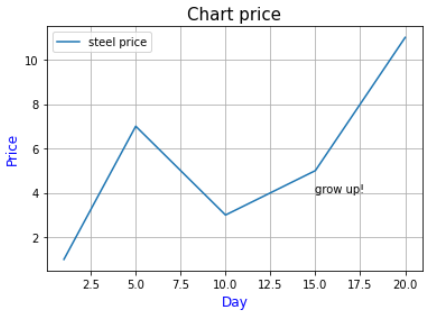
Ввод [15]:
```
plt.text(1, 1, 'type: Steel')
```

Out[15]:
```
Text(1, 1, 'type: Steel')
```



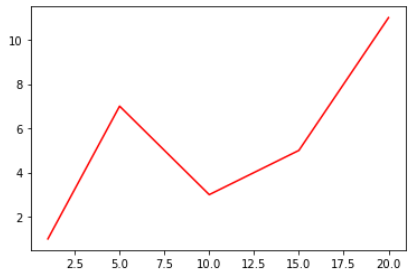
Ввод [16]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, label='steel price')
plt.title('Chart price', fontsize=15)
plt.xlabel('Day', fontsize=12, color='blue')
plt.ylabel('Price', fontsize=12, color='blue')
plt.legend()
plt.grid(True)
plt.text(15, 4, 'grow up!')
```

Out[16]:
```
Text(15, 4, 'grow up!')
```



Ввод [17]:
```
plt.plot(x, y, color='red')
```

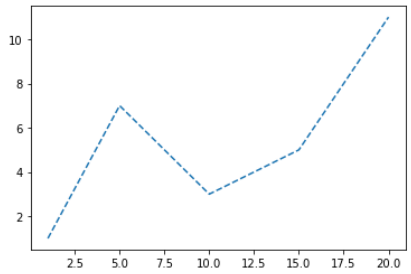
Out[17]:
```
[<matplotlib.lines.Line2D at 0x2ad06bf6880>]
```



Работа с линейным графиком

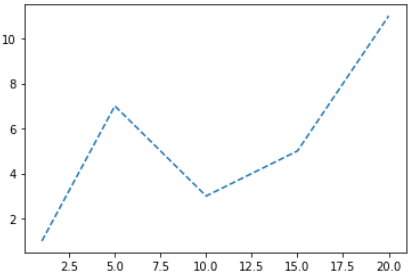
Ввод [18]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--')
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x2ad06db6280>]
```



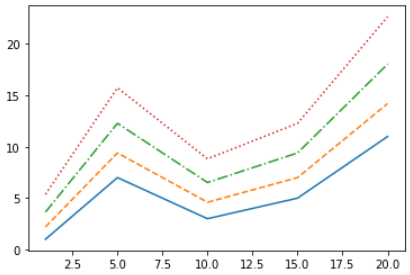
Ввод [19]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
line = plt.plot(x, y)
plt.setp(line, linestyle='--')
```

Out[19]:
```
[None]
```



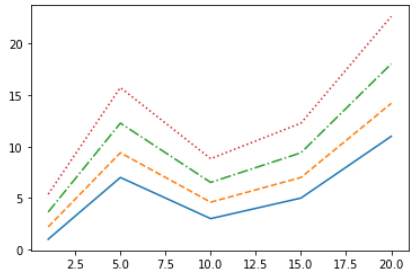
Ввод [20]:
```
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
plt.plot(x, y1, '-', x, y2, '--', x, y3, '-.', x, y4, ':')
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x2ad06e6d790>,
 <matplotlib.lines.Line2D at 0x2ad06e6d880>,
 <matplotlib.lines.Line2D at 0x2ad06e6d730>,
 <matplotlib.lines.Line2D at 0x2ad06e6d8b0>]
```



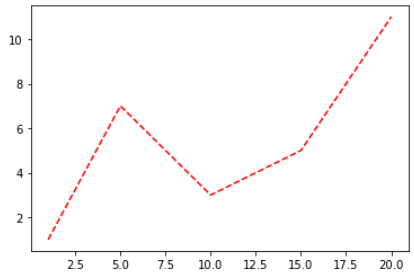
Ввод [21]:
```
plt.plot(x, y1, '-')
plt.plot(x, y2, '--')
plt.plot(x, y3, '-.')
plt.plot(x, y4, ':')
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x2ad0801a5b0>]
```



Ввод [22]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--r')
```

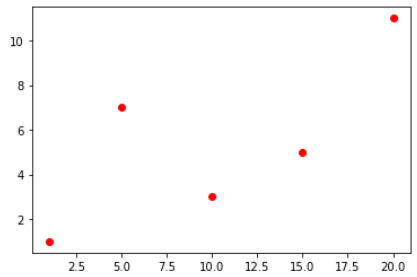
Out[22]:
```
[<matplotlib.lines.Line2D at 0x2ad08076580>]
```



Тип графика

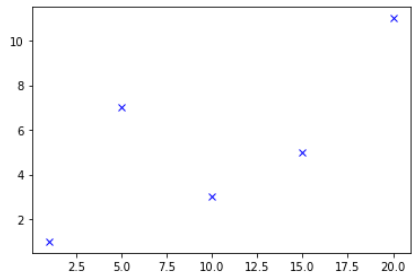
Ввод [23]:
```
plt.plot(x, y, 'ro')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x2ad080c3910>]
```



Ввод [24]:
```
plt.plot(x, y, 'bx')
```

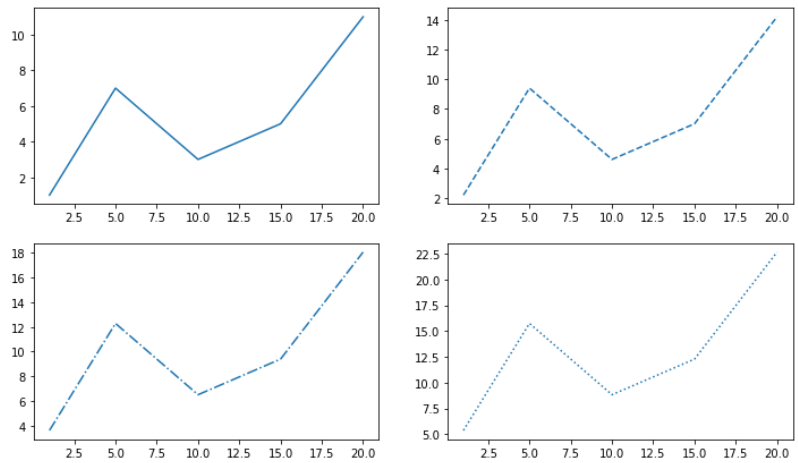
Out[24]:
```
[<matplotlib.lines.Line2D at 0x2ad08120460>]
```



Размещение графиков на разных полях

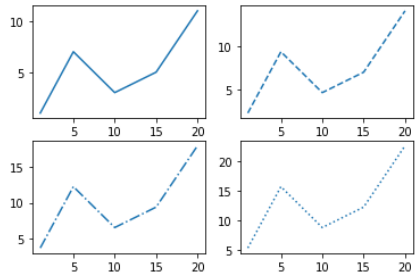
Ввод [25]:
```
# Исходный набор данных
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
# Настройка размеров подложки
plt.figure(figsize=(12, 7))
# Вывод графиков
plt.subplot(2, 2, 1)
plt.plot(x, y1, '-')
plt.subplot(2, 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[25]:
```
[<matplotlib.lines.Line2D at 0x2ad081ff6a0>]
```



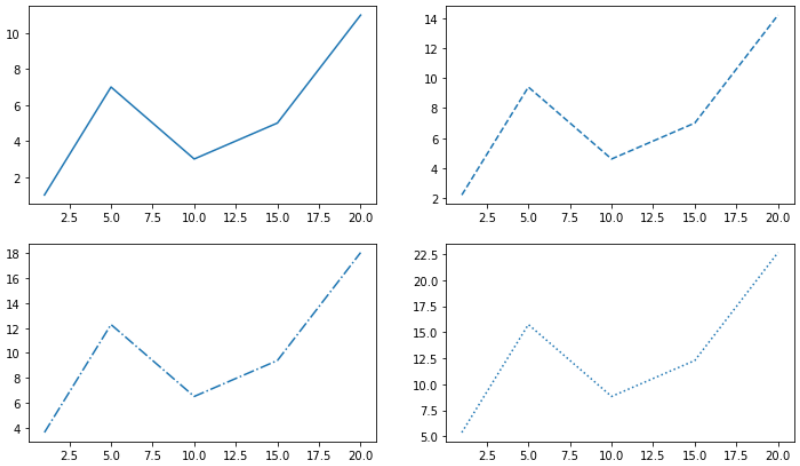
Ввод [26]:
```
# Вывод графиков
plt.subplot(221)
plt.plot(x, y1, '-')
plt.subplot(222)
plt.plot(x, y2, '--')
plt.subplot(223)
plt.plot(x, y3, '-.')
plt.subplot(224)
plt.plot(x, y4, ':')
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x2ad08325850>]
```



Ввод [27]:
```
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, ':')
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x2ad085e69a0>]
```



Ноутбук с линейным графиком:

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения линейного графика.

Была выбрана задача симуляции полета тела, из точки с координатами $(0, 0)$, брошенная под углом α к горизонту, с начальной скоростью v_0

Ввод [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Уравнение, траектории движения тела по оси y:

$$y = x \tan \alpha - \frac{gx^2}{2v_0^2 \cos^2 \alpha}$$

Ввод [2]:
```
def calc_y(x, v0, a, g):
    return (x * np.tan(a)) - (g * x * x) / (2*(v0**2)*(np.cos(a)**2))
```

Ввод [3]:
```
a = float(input("Угол от 0 до 90: "))
v0 = float(input("Начальная скорость: "))
```

```
Угол от 0 до 90: 45
Начальная скорость: 20
```

Устанавливаем значение ускорения свободного падения.

Перводим значения α из градусов в радианы.

А также выситываем длину полета:

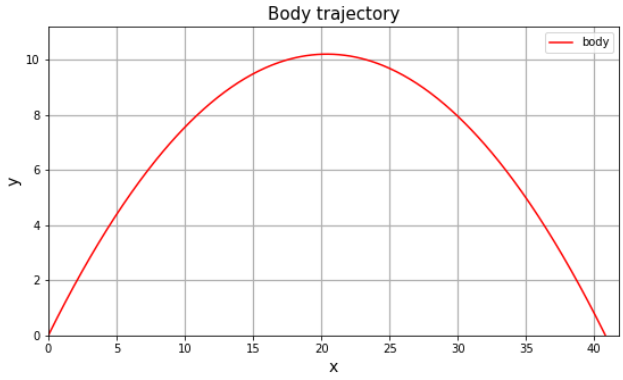
$$l = \frac{2v_0^2 \sin \alpha \cos \alpha}{g}$$

И высоту:

$$h = \frac{v_0^2 \sin^2 \alpha}{2g}$$

Ввод [4]:
```
g = 9.8
a = np.radians(a)

l = (2 * (v0**2) * np.sin(a) * np.cos(a)) / g
h = (v0**2 * (np.sin(a)**2)) / (2 * g)
```

Ввод [5]:
```
x = np.arange(0, np.ceil(l) + 1, 0.1)
vfunc = np.vectorize(calc_y)
y = vfunc(x, v0, a, g)

x = x[y >= 0]
y = y[y >= 0]

fig, ax = plt.subplots(figsize=(9, 5))

ax.set_xlim(0, l + 1)
ax.set_ylim(0, h + 1)
ax.set_xlabel("x", fontsize=14)
ax.set_ylabel("y", fontsize=14)
ax.set_title('Body trajectory', fontsize=15)
ax.grid(which="major", linewidth=1.2)
ax.grid(which="minor", linestyle="--", color="gray", linewidth=0.5)

ax.plot(x, np.abs(y), c="r", label="body",)
ax.legend()
plt.show()
```



Ноутбук со столбчатой диаграммой:

> **Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения круговой диаграммы, условие которой предварительно необходимо согласовать с преподавателем.**

Была выбрана задача по анализу датасета страховок.

Подключение не обходимых библиотек и открытие файла с данными:

Ввод [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv ('insurance.csv')
```

Вывод начала датасета:

Ввод [2]:
```
df.head()
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Выбираем столбцы, в которых содержится информация о поле человека и том, курит ли он:

Ввод [3]:
```
df = df[['sex', 'smoker']]
df.describe()
```

Out[3]:

	sex	smoker
count	1338	1338
unique	2	2
top	male	no
freq	676	1064

Выбираем из данных курящих и некурящих мужчин и женщин:

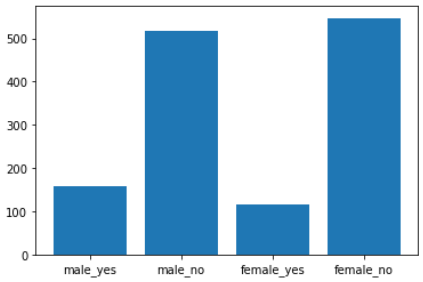
Ввод [4]:
```
m_y = df.loc[(df.sex == 'male') & (df.smoker == 'yes')].value_counts()
m_n = df.loc[(df.sex == 'male') & (df.smoker == 'no')].value_counts()
f_y = df.loc[(df.sex == 'female') & (df.smoker == 'yes')].value_counts()
f_n = df.loc[(df.sex == 'female') & (df.smoker == 'no')].value_counts()

keys_m_y = list(dict(m_y).keys())
values_m_y = list(dict(m_y).values())[0]

keys_m_n = list(dict(m_n).keys())
values_m_n = list(dict(m_n).values())[0]

keys_f_y = list(dict(f_y).keys())
values_f_y = list(dict(f_y).values())[0]

keys_f_n = list(dict(f_n).keys())
values_f_n = list(dict(f_n).values())[0]

keys_m_y = str(keys_m_y[0][0]) + "_" + str(keys_m_y[0][1])
keys_m_n = str(keys_m_n[0][0]) + "_" + str(keys_m_n[0][1])
keys_f_y = str(keys_f_y[0][0]) + "_" + str(keys_f_y[0][1])
keys_f_n = str(keys_f_n[0][0]) + "_" + str(keys_f_n[0][1])

groups = [keys_m_y, keys_m_n, keys_f_y, keys_f_n]
counts = [values_m_y, values_m_n, values_f_y, values_f_n]

print(groups)
print(counts)
```

```
['male_yes', 'male_no', 'female_yes', 'female_no']
[159, 517, 115, 547]
```

Выводим столбчатые диаграммы:

Ввод [5]:
```
plt.bar(groups, counts)
```

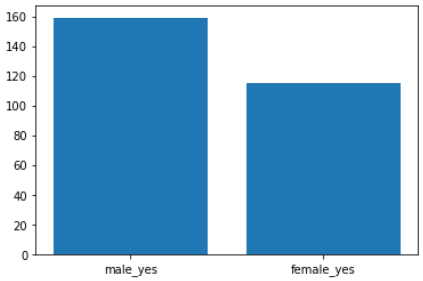
Out[5]:
```
<BarContainer object of 4 artists>
```



Выбираем только курящих мужчин и женщин и строим диаграммы:

Ввод [6]:
```
smokers_groups = list(groups[0::2])
smokers_counts = list(counts[0::2])
plt.bar(smokers_groups, smokers_counts)
```

Out[6]:
```
<BarContainer object of 2 artists>
```

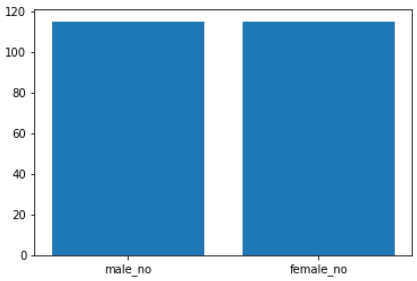


Из диаграмм видно, что количество курящих мужчин больше

Выбираем только некурящих мужчин и женщин и строим диаграммы:

Ввод [7]:
```
no_smokers_groups = list(groups[1::2])
no_smokers_counts = list(counts[2::2])
plt.bar(no_smokers_groups, no_smokers_counts)
```

Out[7]:
```
<BarContainer object of 2 artists>
```



Из диаграмм видно, что количество некурящих мужчин и женщин одинаково

Ноутбук с круговой диаграммой:

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения круговой диаграммы.

Подключение не обходимых библиотек и открытие файла с данными:

Ввод [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv ('insurance.csv')
```

Вывод начала датасета:

Ввод [2]:
```
df.head()
```

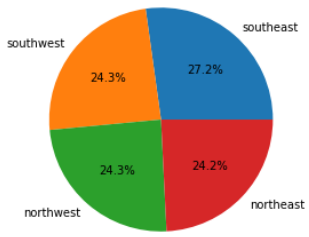
Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Выбираем данные по регионам, в котрых люди получили страховки. Строим диаграмму:

Ввод [3]:
```
test = df['region'].value_counts()
keys = list(dict(test).keys())
values = list(dict(test).values())

fig, ax = plt.subplots()
ax.pie(values, labels=keys, autopct='%1.1f%%')
ax.axis("equal")
```

Out[3]:
```
(-1.1207194321058473,
 1.100986639624088,
 -1.1125533933009824,
 1.1112460774223782)
```

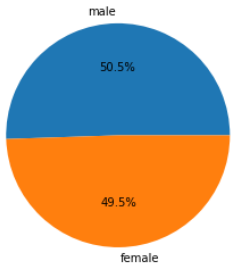


Из диаграммы видно, что количество регионов в которых люди получили страховки примерно равно.

Выбираем данные по полу, людей которые получили страховки. Строим диаграмму:

Ввод [4]:
```
test = df['sex'].value_counts()

keys = list(dict(test).keys())
values = list(dict(test).values())

fig, ax = plt.subplots()
ax.pie(values, labels=keys, autopct='%1.1f%%')
ax.axis("equal")
```

Out[4]:
```
(-1.1040121656056667,
 1.1001910555050318,
 -1.104486483140735,
 1.1023569618236218)
```



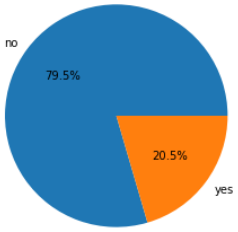
Из диаграммы видно, что количество людей которые получили страховки примерно равно.

Выбираем данные по курящим людям, которые получили страховки. Строим диаграмму:

Выбираем данные по курящим людям, которые получили страховки. Строим диаграмму:

Ввод [5]:
```
test = df['smoker'].value_counts()

keys = list(dict(test).keys())
values = list(dict(test).values())

fig, ax = plt.subplots()
ax.pie(values, labels=keys, autopct='%1.1f%%')
ax.axis("equal")
```

Out[5]:
```
(-1.1018718496698334,
 1.1000891356985636,
 -1.1027154146109623,
 1.1011093550812814)
```



Из диаграммы видно, что количество некурящих людей, которые получили страхову больше.

Ноутбук с изображением:

Найти какое-либо изображение в сети Интернет. Создать ноутбук, в котором будет отображено выбранное изображение средствами библиотеки matplotlib по URL из сети Интернет.

Ввод [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from PIL import Image
import requests
from io import BytesIO
```

Ввод [2]:
```
plt.figure(figsize=(10, 10))
response_stone = requests.get('https://img.gazeta.ru/files3/397/14400397/chmonya-pic_32ratio_900x600-900x600-7396.jpg')
img = Image.open(BytesIO(response_stone.content))
plt.imshow(img)
```

Out[2]:
```
<matplotlib.image.AxesImage at 0x1ed59461ca0>
```

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Как выполнить построение линейного графика с помощью matplotlib?

plot([x], y, [fmt], *, data=None, **kwargs)

plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

- x, x2, ...: array - набор данных для оси абсцисс первого, второго и т.д. графика.
- y, y2, ...: array - набор данных для оси ординат первого, второго и т.д. графика.
- fmt: str - формат графика, задается в виде строки: [marker][line][color] .
- **kwargs – свойства класса Line2D (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.line
- s.Line2D.html#matplotlib.lines.Line2D), которые предоставляют доступ к большому количеству настроек внешнего вида графика, отметим наиболее полезные:

2. Как выполнить заливку области между графиком и осью? Между двумя графиками?

fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, *,data=None, **kwargs)

Основные параметры функции:

- x : массив длины N`- набор данных для оси абсцисс.
- y1 : массив длины N или скалярное значение - набор данных для оси ординат – первая кривая.
- y2 : массив длины N или скалярное значение - набор данных для оси ординат – вторая кривая.
- where : массив bool элементов (длины N), optional, значение по умолчанию: None – задает заливаемый цветом регион, который определяется координатами x[where]: интервал будет залит между x[i] и x[i+1], если where[i] и where[i+1] равны True.
- step : {‘pre’, ‘post’, ‘mid’}, optional - определяет шаг, если используется step-функция для отображения графика.
- **kwargs - свойства класса Polygon (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.patches.Polygon.html#matplotlib.patches.Polygon)

3. Как выполнить выборочную заливку, которая удовлетворяет некоторому условию?

where : массив bool элементов (длины N), optional, значение по умолчанию: None – задает заливаемый цветом регион, который определяется координатами x[where]: интервал будет залит между x[i] и x[i+1], если where[i] и where[i+1] равны True.

4. Как выполнить двухцветную заливку?

plt.fill_between(x, y, where=y>=0, color="g", alpha=0.3)

plt.fill_between(x, y, where=y<=0, color="r", alpha=0.3)

5. Как выполнить маркировку графиков?

'o'	Окружность (*circle marker*)
'v'	Треугольник, направленный вниз (*triangle_down marker*)
'^'	Треугольник, направленный вверх(*triangle_up marker*)
'<'	Треугольник, направленный влево (*triangle_left marker*)
'>'	Треугольник, направленный вправо (*triangle_right marker*)
'1'	Треугольник, направленный вниз (*tri_down marker*)
'2'	Треугольник, направленный вверх(*tri_up marker*)
'3'	Треугольник, направленный влево (*tri_left marker*)
'4'	Треугольник, направленный вправо (*tri_right marker*)
's'	Квадрат (*square marker*)
'p'	Пятиугольник (*pentagon marker*)
'*'	Звезда (*star marker*)
'h'	Шестиугольник (*hexagon1 marker*)
'H'	Шестиугольник (*hexagon2 marker*)
'+'	Плюс (*plus marker*)
'x'	X-образный маркер (*x marker*)
'D'	Ромб (*diamond marker*)
'd'	Ромб (*thin_diamond marker*)
'|'	Вертикальная линия (*vline marker*)
'_'	Горизонтальная линия (*hline marker*)

6. Как выполнить обрезку графиков?

y_masked = np.ma.masked_where(y < -0.5, y)

plt.plot(x, y_masked, linewidth=3)

7. Как построить ступенчатый график? В чем особенность ступенчатого графика?

Такой график строится с помощью функции step(),

которая принимает следующий набор параметров:

- x: array_like - набор данных для оси абсцисс
- y: array_like - набор данных для оси ординат
- fmt: str, optional - задает отображение линии (см. функцию plot()).
- data: indexable object, optional - метки.
- where : {'pre', 'post', 'mid'}, optional , по умолчанию 'pre' - определяет место, где будет установлен шаг.

- ‘pre’: значение y ставится слева от значения x, т.е. значение y[i] определяется для интервала (x[i-1]; x[i]).
- ‘post’: значение y ставится справа от значения x, т.е. значение y[i] определяется для интервала (x[i]; x[i+1]).
- ‘mid’: значение y ставится в середине интервала.

8. Как построить стековый график? В чем особенность стекового графика?

Для построения стекового графика используется функция stackplot(). Суть его в том, что графики отображаются друг над другом, и каждый следующий является суммой предыдущего и заданного набора данных:

```
x = np.arange(0, 11, 1)
y1 = np.array([(-0.2)*i**2+2*i for i in x])
y2 = np.array([(-0.4)*i**2+4*i for i in x])
y3 = np.array([2*i for i in x])
labels = ["y1", "y2", "y3"]
fig, ax = plt.subplots()
ax.stackplot(x, y1, y2, y3, labels=labels)
ax.legend(loc='upper left')
```

9. Как построить stem-график? В чем особенность stem-графика?

Визуально этот график выглядит как набор линий от точки с координатами (x, y) до базовой линии, в верхней точке ставится маркер:

```
x = np.arange(0, 10.5, 0.5)
y = np.array([(-0.2)*i**2+2*i for i in x])
plt.stem(x, y)
```

10. Как построить точечный график? В чем особенность точечного графика?

Для отображения точечного графика предназначена функция scatter(). В простейшем виде точечный график можно получить передав функции scatter() наборы точек для x, y координат:

```
x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y)
```

11. Как осуществляется построение столбчатых диаграмм с помощью matplotlib?

bar() – для построения вертикальной диаграммы

barh() – для построения горизонтальной диаграммы.

```
np.random.seed(123)
groups = [f"P{i}" for i in range(7)]
counts = np.random.randint(3, 10, len(groups))
plt.bar(groups, counts)

plt.barh(groups, counts)
```

12. Что такое групповая столбчатая диаграмма? Что такое столбчатая диаграмма с errorbar элементом?

Используя определенным образом подготовленные данные можно строить групповые диаграммы:

```
width = 0.3
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, g1, width)
rects2 = ax.bar(x + width/2, g2, width)
```

Errorbar элемент позволяет задать величину ошибки для каждого элемента графика. Для этого используются параметры xerr, yerr и ecolor (для задания цвета)

13. Как выполнить построение круговой диаграммы средствами matplotlib?

Для построения круговых диаграмм в Matplotlib используется функция pie().

```
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BMV", "AUDI", "Jaguar"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis("equal")
```

14. Что такое цветовая карта? Как осуществляется работа с цветовыми картами в matplotlib?

Цветовая карта представляет собой подготовленный набор цветов, который хорошо подходит для визуализации того или иного набора данных.

```
cmap='название цветовой карты'
```

15. Как отобразить изображение средствами matplotlib?

Основное назначение функции imshow() состоит в представлении 2d растров. Это могут быть картинки, двумерные массивы данных, матрицы и т.п.

```
from PIL import Image
import requests
from io import BytesIO
response = requests.get('https://matplotlib.org/_static/logo2.png')
img = Image.open(BytesIO(response.content))
plt.imshow(img)
```

16. Как отобразить тепловую карту средствами matplotlib?

pcolormesh()

Функция pcolormesh() похожа по своим возможностям на imshow(), но есть и отличия.

- C : массив - 2D массив скалярных значений
- cmap : str или Colormap, optional - см. cmap в imshow()
- norm : Normalize, optional - см. norm в imshow()
- vmin , vmax : scalar, optional, значение по умолчанию: None - см. vmin, vmax в imshow()
- edgecolors : {‘none’, None, ‘face’, color, color sequence}, optional - цвет границы, по умолчанию: ‘none’, возможны следующие варианты:

    - ‘none’ or ”: без отображения границы.
    - None: черный цвет.
    - ‘face’: используется цвет ячейки.
    - Можно выбрать цвет из доступных наборов.
- alpha : scalar, optional, значение по умолчанию: None - см. alpha в imshow().
- shading : {‘flat’, ‘gouraud’}, optional - стиль заливки, доступные значения:
    - ‘flat’: сплошной цвет заливки для каждого квадрата.
    - ‘gouraud‘: для каждого квадрата будет использован метод затенения Gouraud.
- snap : bool, optional, значение по умолчанию: False - привязка сетки к границам пикселей.