ЛАБОРАТОРНАЯ РАБОТА №3
ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ *MATPLOTLIB* ДЛЯ ПОСТРОЕНИЯ ГРАФИКОВ

Matplotlib одна из самых распространенных библиотек для построения *2D* графиков. Она обеспечивает как быстрое создание графиков, так и сохранение результатов в различные форматы. В лабораторной работе будут рассмотрены наиболее общие случаи ее применения.

Pyplot позволяет создавать графики с использование процедурного интерфейса, который с *Matlab*. Поэтому вызов команд в них схож. Для того, чтобы импортировать модуль *pyplot* из библиотеки *Matplotlib* используется команда *import*.

Самый простой график.

Для построения графика первое, что необходимо начать – это рассчитать значения графиков синуса и косинусов на интервале *X* от $-\pi$ to $+\pi$ (включительно), сохранив их в массивы *C* и *S*.

Matplotlib обладает рядом настроек, задавая которые можно изменять внешний вид графиков: размер изображения, разрешение, ширина линий, оси и подписи к ним, количество графиков на изображении, текст, стиль и т.д:

plot – построение графиков. Аргумент *color* задает цвет линии. *linewidth* –ширина линии, *linestyle* – стиль линии, *label* – метка линии.

spines – функция для работы с осями изображения. По умолчанию, оси изображения, находятся по краям изображения. В пример показан способ их перемещения в центр графика.

xlim, ylim – границы по оси *x* и *y*.

xticks, *yticks* – насечки по оси *x* и *y*

legend – добавляет подписи к графику

show – отображение графика

annotate – аннотирование графиков.

scatter – строит диаграмму рассеяния

Для добавления легенды используется функция *legend*.

Задание 1. Реализуйте код, приведенный в примере. Закомментируйте код.

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5), dpi=80)
plt.subplot(111)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-",  label="sine")

ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data', 0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data', 0))

plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.xticks([-np.pi, -np.pi / 2, 0, np.pi / 2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.yticks([-1, 1],
          [r'$-1$', r'$+1$'])

plt.legend(loc='upper left')

t = 2*np.pi/3
plt.plot([t, t], [0, np.cos(t)],
        color='blue', linewidth=1.5, linestyle="--")
plt.scatter([t, ], [np.cos(t), ], 50, color='blue')
plt.annotate(r'$sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
            xy=(t, np.sin(t)), xycoords='data',
            xytext=(10, 30), textcoords='offset points', fontsize=16,
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

plt.plot([t, t], [0, np.sin(t)],
        color='red', linewidth=1.5, linestyle="--")
plt.scatter([t, ], [np.sin(t), ], 50, color ='red')
plt.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$', xy=(t, np.cos(t)),
            xycoords='data', xytext=(-90, -50),
            textcoords='offset points', fontsize=16,
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
```

Построение нескольких графиков.

Понятие «изображение» (“*figure*”) означает целиком окно. Внутри изображения может быть несколько графиков (“*subplots*”). В предыдущем примере показано не явное использование, что удобно использовать при быстром построении графиков. Использование функции subplots позволяет задавать местоположение графиков и перемещать их. При построении графика необходимо использовать команду *gca*, которая возвращает индекс текущих осей для работы, эта команда в свою очередь вызывает команду *gcf* для того, чтобы определить с какой фигурой происходит работа. Если в команду *figure* не были переданы параметры, то по умолчанию используется subplot(111).

Изображение – это окно в *GUI*, которое имеет заголовок “*Figure #*”. Нумерация изображений начинается с 1 в отличие от классической нумерации в *Python*, которая начинается с 0. Такая нумерация соответствует классической нумерации в *MatLab*. Существует ряд параметров для настройки фигуры:

Аргумент	Значение по умолчанию	Описание
num	*1*	Номер изображения
figsize	*figure.figsize*	Размер фигуры в дюймах (ширина, высота)
dpi	*figure.dpi*	Разрешение в пикселях на дюйм
facecolor	*figure.facecolor*	Цвет фона
edgecolor	*figure.edgecolor*	Цвет за границей фона
frameon	*True*	Отображать контур или нет

Параметры можно задать в отдельном файле и использовать много раз. Чаще всего меняется такой параметр как номер. Для того, чтобы установить параметры можно использовать стандартные функции *setp* или *set_something*. При работе с *GUI* для того, чтобы закрыть фигуру достаточно нажать на «x» в правом верхнем угле. Однако закрыть окно можно и программно, воспользовавшись функцией *close*. В зависимости от переданного аргумента можно закрыть текущее окно (по умолчанию), все окно (значение аргумента «*all*») или окно с определенным номером.

```
plt.close(1)
```

Функция *subplots* позволяет построить несколько графиков на одном изображении. Для этого необходимо задать количество строк и столбцов и количество графиков. Однако команда *gridspec* является более мощным средством для создания графиков.

```
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

plt.figure(figsize=(6, 4))
G = gridspec.GridSpec(3, 3)

axes_1 = plt.subplot(G[0, :])
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'Axes 1', ha='center', va='center', size=24, alpha=.5)

axes_2 = plt.subplot(G[1, :-1])
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'Axes 2', ha='center', va='center', size=24, alpha=.5)

axes_3 = plt.subplot(G[1:, -1])
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'Axes 3', ha='center', va='center', size=24, alpha=.5)

axes_4 = plt.subplot(G[-1, 0])
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'Axes 4', ha='center', va='center', size=24, alpha=.5)

axes_5 = plt.subplot(G[-1, -2])
plt.xticks(())
plt.yticks(())
plt.text(0.5, 0.5, 'Axes 5', ha='center', va='center', size=24, alpha=.5)

plt.tight_layout()
plt.show()
```

Задание 2. Реализуйте код, приведенный в примере.

Функция *axes* по своему назначению схожа с функцией *subplots*, однако она позволяет размещать графики в любом месте изображения. Поэтому ее использует, если

необходимо выполнить, наложение одного графика на другой, например, маленького на большой.

```
import matplotlib.pyplot as plt

plt.axes([.1, .1, .8, .8])
plt.xticks(())
plt.yticks(())
plt.text(.6, .6, 'axes([0.1, 0.1, .8, .8])', ha='center', va='center',
        size=20, alpha=.5)

plt.axes([.2, .2, .3, .3])
plt.xticks(())
plt.yticks(())
plt.text(.5, .5, 'axes([0.2, 0.2, .3, .3])', ha='center', va='center',
        size=16, alpha=.5)

plt.show()
```

Для построения насечек на оси графика используется функция *set_major_locator*. Все параметры насечек находятся в объекте *matplotlib.ticker.Locator*. Управление датами в качестве подписи осей далеко не тривиальная задача, поэтому для управления им в *Python* есть специальная функция *matplotlib.dates*.

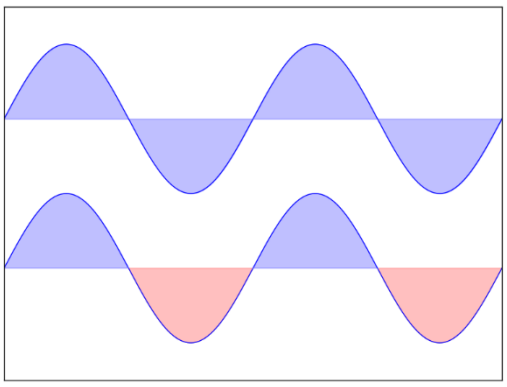
Задание 3. Реализуйте код приведенный в примере.

Для выполнения дальнейших заданий Вам необходимо будет предварительно импортировать библиотеку *numpy*, дав ей сокращенной название *np*.

ЗАДАНИЕ 4. ПОСТРОЕНИЕ КЛАССИЧЕСКОГО ГРАФИКА

Используя команду *fill_between* и базовый код, реализуйте график показанный на рисунке.

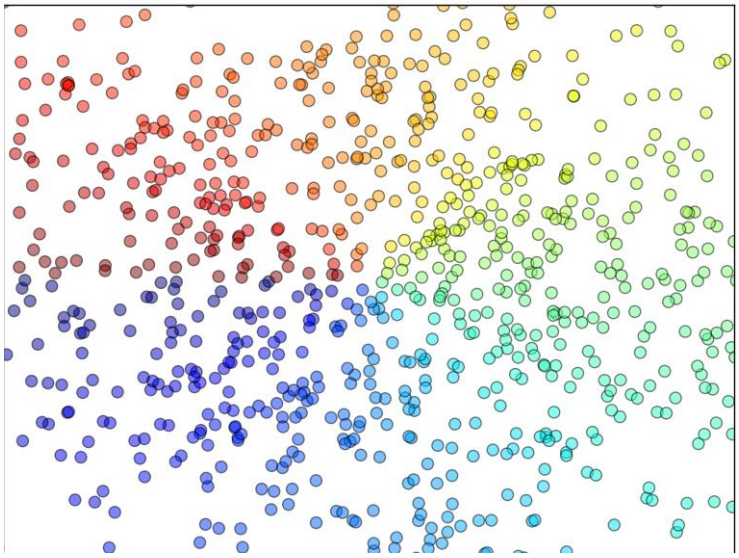
```
import numpy as np
n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)

plt.plot(X, Y + 1, color='blue', alpha=1.00)
plt.plot(X, Y - 1, color='blue', alpha=1.00)
```



ЗАДАНИЕ 5. ПОСТРОЕНИЕ ДИАГРАММЫ РАССЕЯНИЯ

Используя приведенный ниже код постройте график, показанный на рисунке: цвет и прозрачность задаются углом от оси *OX*.

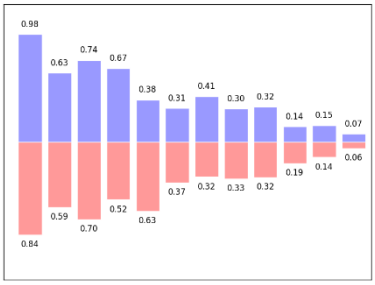
```
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)

plt.scatter(X,Y)
```



ЗАДАНИЕ 6. ГИСТОГРАММЫ ИЗОБРАЖЕНИЙ

Используя приведённый код, постройте график показанный на рисунке: обратите внимание на подписи осей.

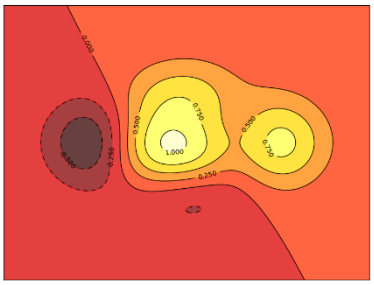
```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')

for x, y in zip(X, Y1):
    plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va='bottom')

plt.ylim(-1.25, +1.25)
```



ЗАДАНИЕ 7. ИЗОЛИНИИ.

Используя приведённый код, постройте график показанный на рисунке: для этого необходима команда *clabel*

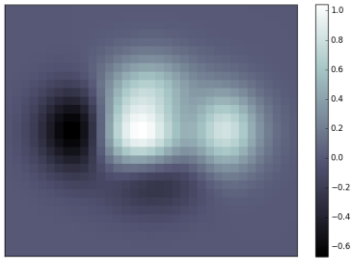
```
def f(x, y):
    return (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 -y ** 2)

n = 256
x = np.linspace(-3, 3, n)
y = np.linspace(-3, 3, n)
X, Y = np.meshgrid(x, y)

plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap='jet')
C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)
```



ЗАДАНИЕ 8. РИСУНКИ

Используя приведённый код, постройте график показанный на рисунке. Обратите внимание на цветовую карту, интерполяцию изображения. Для выполнения Вам понадобится функция *colobar*.

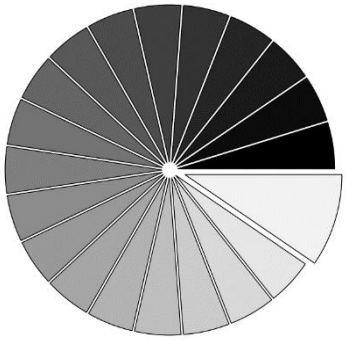
```
def f(x, y):
    return (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)

n = 10
x = np.linspace(-3, 3, 4 * n)
y = np.linspace(-3, 3, 3 * n)
X, Y = np.meshgrid(x, y)
plt.imshow(f(X, Y))
```



ЗАДАНИЕ 9. КРУГОВЫЕ ДИАГРАММЫ.

Используя приведённый код, постройте график показанный на рисунке: необходимо изменить *Z*.

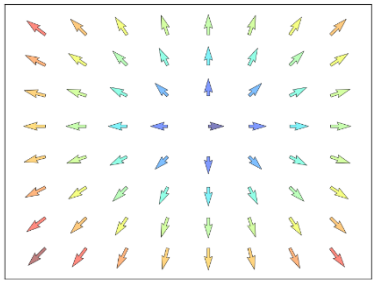
```
Z = np.random.uniform(0, 1, 20)
plt.pie(Z)
```



ЗАДАНИЕ 10. ГРАДИЕНТНЫЕ ГРАФИКИ

Используя приведённый код, постройте график показанный на рисунке: стрелочки необходимо нарисовать будет дважды.

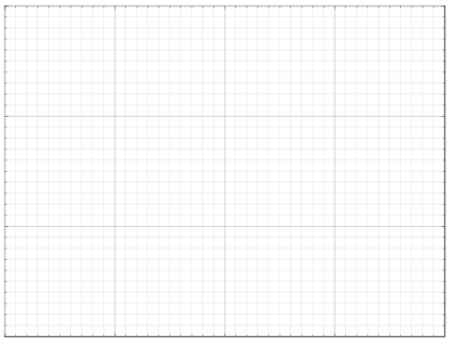
```
n = 8
X, Y = np.mgrid[0:n, 0:n]
plt.quiver(X, Y)
```



ЗАДАНИЕ 11. СЕТКА.

Используя приведённый код, постройте график показанный на рисунке.

```
axes = plt.gca()
axes.set_xlim(0, 4)
axes.set_ylim(0, 3)
axes.set_xticklabels([])
axes.set_yticklabels([])
```



ЗАДАНИЕ 12. ПОСТРОЕНИЕ НЕСКОЛЬКИХ ГРАФИКОВ.

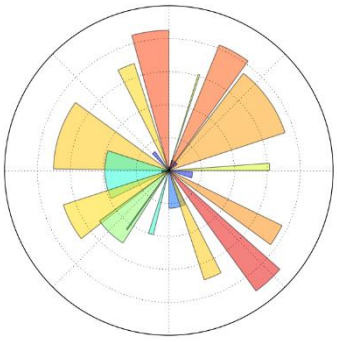
Используя приведённый код, постройте график показанный на рисунке.

```
plt.subplot(2, 2, 1)
plt.subplot(2, 2, 3)
plt.subplot(2, 2, 4)
```

ЗАДАНИЕ 13. ПОЛЯРНЫЕ КООРДИНАТЫ

Для выполнения следующего задания необходимо изменить только axes. Используя приведённый код, постройте график показанный на рисунке.

```
plt.axes([0, 0, 1, 1])

N = 20
theta = np.arange(0., 2 * np.pi, 2 * np.pi / N)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
bars = plt.bar(theta, radii, width=width, bottom=0.0)

for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.jet(r / 10.))
    bar.set_alpha(0.5)
```



ЗАДАНИЕ 14. 3D ГРАФИК.

Используя *contourf* и приведенный код, постройте график показанный на рисунке.

```
plt.axes([0, 0, 1, 1])

N = 20
theta = np.arange(0., 2 * np.pi, 2 * np.pi / N)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
bars = plt.bar(theta, radii, width=width, bottom=0.0)

for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.jet(r / 10.))
    bar.set_alpha(0.5)
```