ЛАБОРАТОРНАЯ РАБОТА №2: ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ MATPLOTLIB ДЛЯ ГРАФИЧЕСКОГО ОТОБРАЖЕНИЯ

Matplotlib – одна из самых распространенных библиотек для 2D-графиков. Она позволяет как быстро создавать графики, так и сохранять результаты в различных форматах. В лабораторной работе мы рассмотрим наиболее распространенные случаи ее применения. *Pyplot* позволяет создавать графики, используя процедурный интерфейс, который аналогичен *Matlab*. Поэтому вызов команд в нем похож. Для импорта модуля *pyplot* из библиотеки *Matplotlib* используется команда import.

Простейший график. Для построения графика сначала нужно рассчитать значения функций синуса и косинуса на интервале *X* от -π до +π (включительно), сохраняя их в массивах *C* и *S*. *Matplotlib* имеет ряд настроек, которые можно использовать для изменения внешнего вида графиков: размер изображения, разрешение, ширина линии, оси и подписи, количество графиков на изображении, текст, стиль и т.д. Аргумент *color* указывает цвет линии, *linewidth* определяет ширину линии, *linestyle* используется для стиля линии, label задает подпись линии. *spines* – это функция для работы с осями изображения. По умолчанию оси изображения расположены по краям изображения. В примере показано, как переместить их в центр графика. *xlim*, *ylim* используются для границ осей *x* и *y*. *xticks*, *yticks* – это метки на осях *x* и *y*, *legend* добавляет подписи к графику, *show* отображает график, *annotate* аннотирует графики, *scatter* строит диаграмму рассеяния, а функция *legend* используется для добавления легенды.

Задание 1. Реализуйте код, приведенный в примере (Рисунок 1). Прокомментируйте код.

Рисование нескольких графиков. Термин «фигура» означает все окно. Внутри изображения может быть несколько графиков (*subplots*). В предыдущем примере показано неявное использование, которое удобно для быстрого построения графиков. Использование функции *subplots* позволяет указать расположение 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 определения, с какой фигурой работает. Если параметры не переданы в команду *figure*, по умолчанию используется *subplot*(111).

Изображение – это окно в графическом интерфейсе пользователя, которое помечено как *Figure #*. Нумерация изображений начинается с 1, в отличие от классической нумерации в Python, которая начинается с 0. Эта нумерация соответствует классической нумерации в *MatLab*. Существуют различные параметры для настройки формы (Таблица 1).

Таблица 1 – Параметры фигуры.

Аргумент	Значение по умолчанию	Описание
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 границы вокруг фона
frameon	True	Отображать контур или нет

```
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"))
```

Рисунок 1 – Пример построения графика.

Параметры могут быть установлены в отдельном файле и использоваться многократно. Для установки параметров можно использовать стандартные функции *setp* или *set_something*. Для закрытия фигуры достаточно кликнуть на x в верхнем правом углу. Однако также можно закрыть окно программно, используя функцию *close*. В зависимости от переданного аргумента, можно закрыть текущее окно (по умолчанию), все окна (значение аргумента *all*) или окно с определенным номером.

Функция *subplots* позволяет построить несколько графиков на одном изображении. Для этого необходимо указать количество строк и столбцов, а также количество графиков. Однако команда *gridspec* является более мощным инструментом для создания графиков.

Задание 2. Реализуйте код, показанный на рисунке 2.

```
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()
```

Рисунок 2 – Рисование нескольких графиков.

Функция *axes* по своему назначению аналогична функции *subplots*, но она позволяет размещать графики в любом месте изображения. Поэтому ее используют для наложения одного графика на другой, например, маленького графика на большом.

Функция *set_major_locator* используется для построения меток на оси графика. Все параметры меток находятся в объекте *matplotlib.ticker.locator*. Управление датами в качестве подписей осей – не тривиальная задача, поэтому в *Python* существует специальная функция *matplotlib.dates* для ее решения.

Задание 3. Реализуйте код, приведенный в примере (Рисунок 3).

```
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()
```

Рисунок 3 – Размещение графиков в любом месте изображения.

Для последующих задач вам потребуется предварительно импортировать библиотеку *numpy*, присвоив ей сокращенное имя *np*.

Задание 4. Построение графика. Используя команду *fill_between* и базовый код, реализуйте график, показанный на рисунке 4.

А Б

```
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)
```

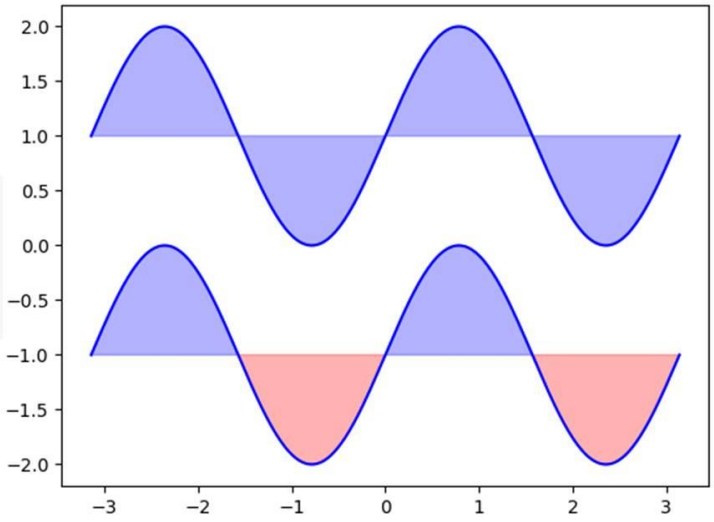

Рисунок 4 – Простой график. А) Пример кода Б) Итоговое изображение.

Задание 5. Диаграмма рассеяния. Используя приведенный ниже код, постройте график, показанный на рисунке: цвет и прозрачность задаются углом от оси *OX* (Рисунок 5).

А Б

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

plt.scatter(X,  Y)
```

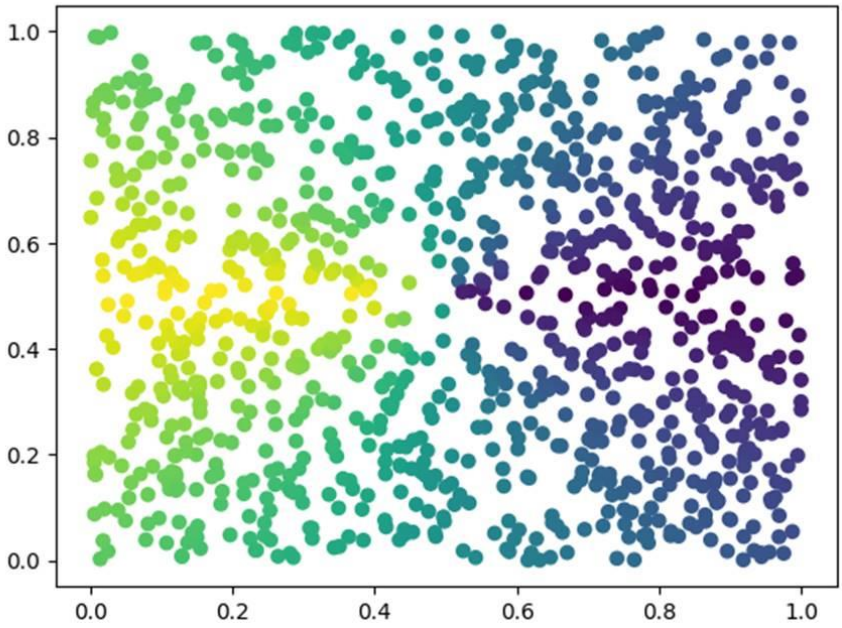

Рисунок 5 – Диаграмма рассеяния. А) Пример кода Б) Итоговое изображение.

Задание 6. Гистограммы. Используя данный код, постройте график, показанный на рисунке: обратите внимание на подписи осей (Рисунок 6).

А

```
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+0.5, '%.2f' % y, ha='center', va= 'bottom')

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

Б

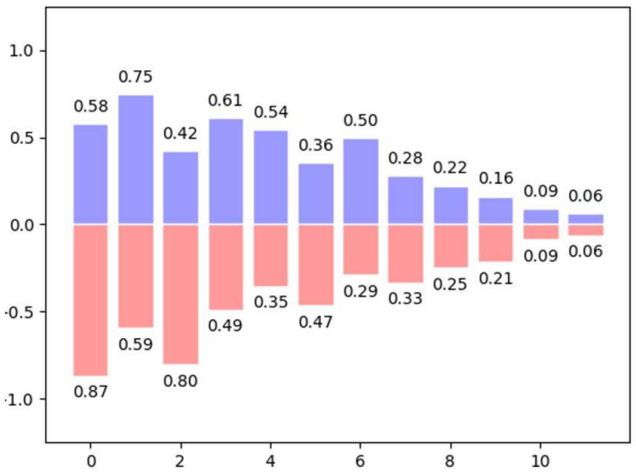

Рисунок 6 – Гистограммы. А) Пример кода Б) Итоговое изображение.

Задание 7. Контурные линии. Используя приведенный выше код, постройте график, показанный на рисунке 7: для этого потребуется команда *clabel*.

А

```
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)
```

Б

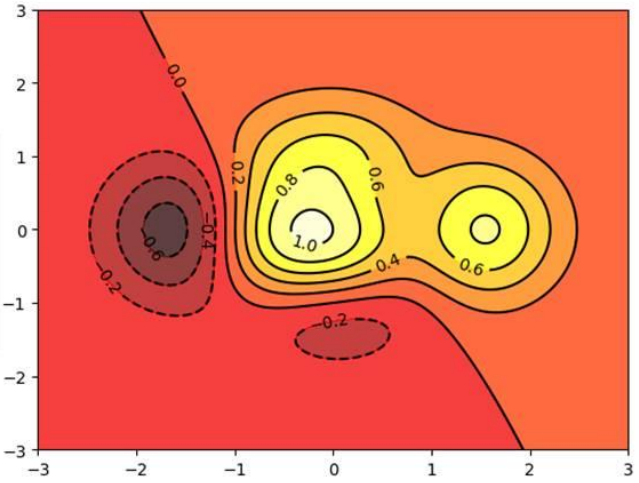

Рисунок 7 – Контурные линии. А) Пример кода
Б) Результирующее изображение.

Задание 8. Цветовая карта. Используя данный код, постройте график, показанный на рисунке 8. Обратите внимание на цветовую карту и интерполяцию изображения. Вам потребуется функция *colorbar*.

А

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

Б

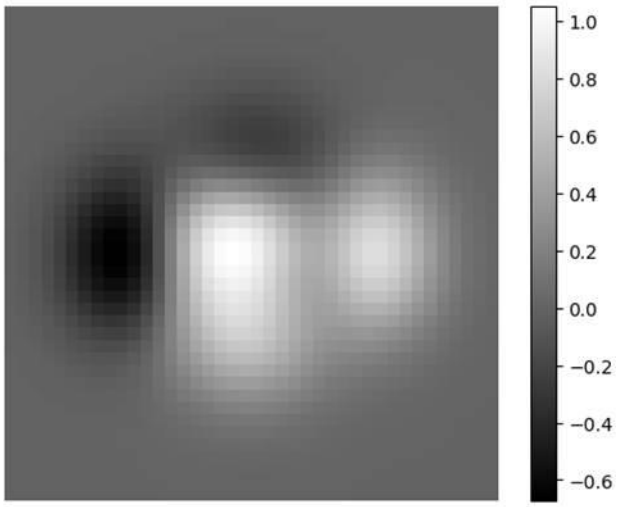

Рисунок 8 – Цветовая карта. А) Пример кода
Б) Результирующее изображение.

Задание 9. Круговые диаграммы. Используя данный код, постройте график, показанный на рисунке 9: вам нужно изменить *Z*.

А

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

Б

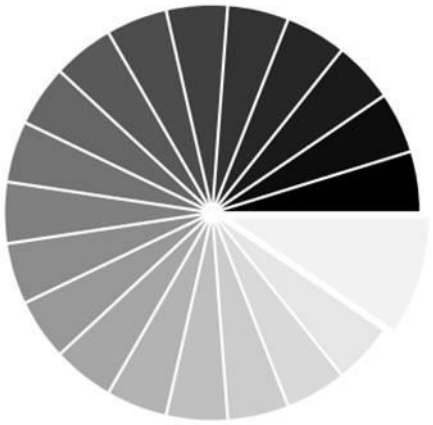

Рисунок 9 – Круговая диаграмма. А) Пример кода
Б) Результирующее изображение.

Задание 10. Стрелки. Используя данный код, постройте график, показанный на рисунке 10: стрелки должны быть нарисованы дважды.

А

```
n = 8
X, Y = np.mgrid[0:5, 0:5]
T = np.arctan2(Y - n/ 2., X - n / 2.)
R = 10 + np.sqrt((Y - n / 2.) ** 2 + (X - n / 2.) ** 2)
U, V = R * np.cos(T), R * np.sin(T)
plt.quiver(X, Y)
```

Б

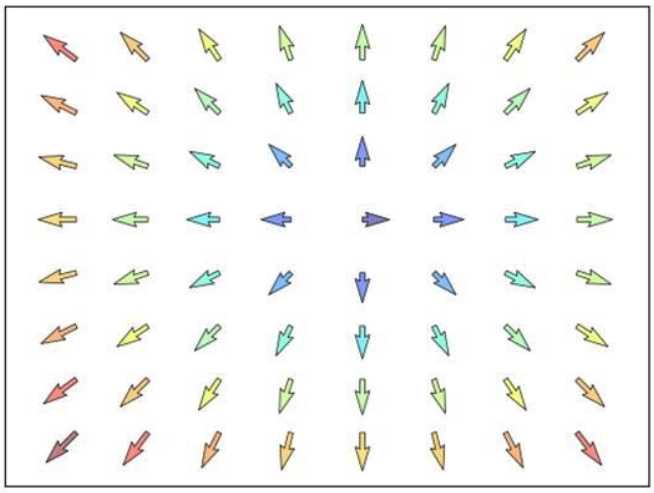

Рисунок 10 – Стрелки. А) Пример кода Б) Результирующее изображение.

Задание 11. Сетка. Используя данный код, постройте график, показанный на рисунке 11.

А

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

Б

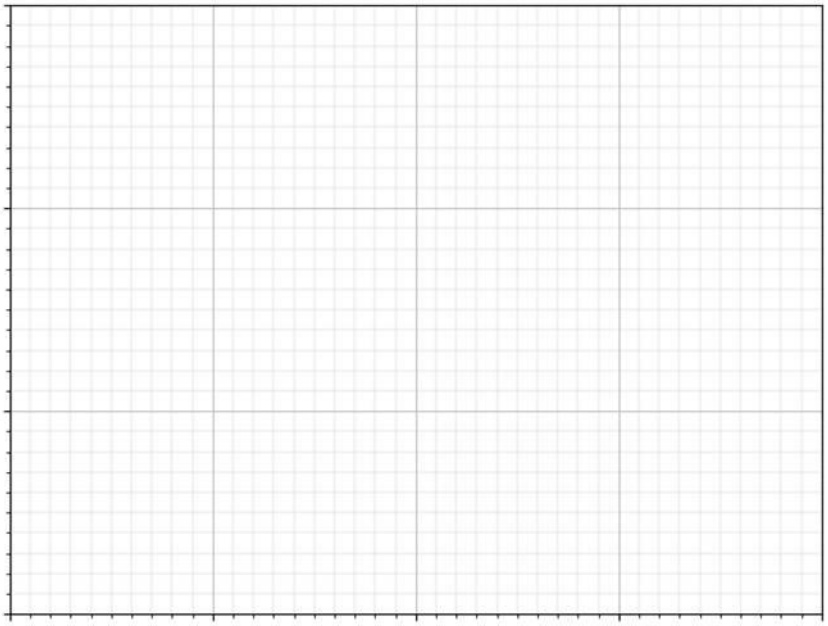

Рисунок 11 – Сетка. А) Пример кода Б) Результирующее изображение.

Задание 12. Построение нескольких графиков. Используя данный код, постройте график, показанный на рисунке 12.

А

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

Б

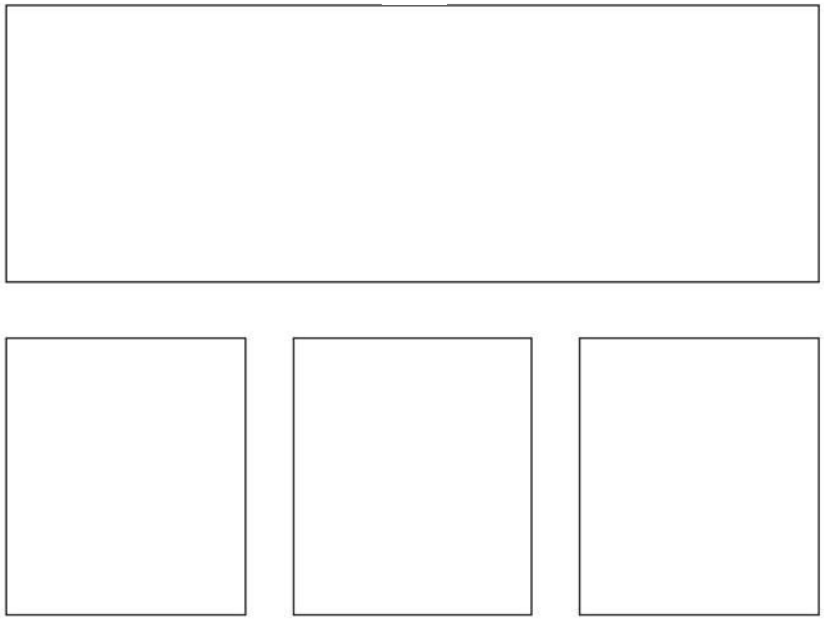

Рисунок 12 – Построение нескольких графиков. А) Пример кода Б) Результирующее изображение.

Задание 13. Полярные координаты. Для выполнения следующей задачи необходимо изменить только оси. Используя данный код, постройте график, показанный на рисунке 13.

А

```
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)
```

Б

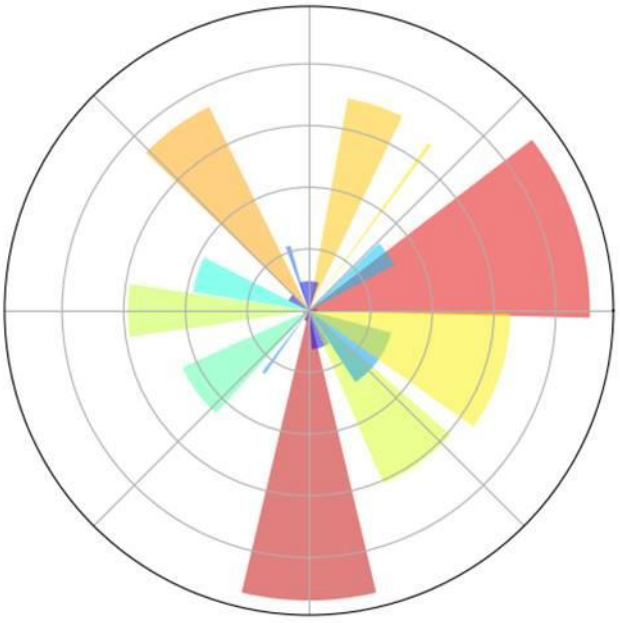

Рисунок 13 – Полярные координаты. А) Пример кода Б) Результирующее изображение

Задачи для закрепления теоретического материала

Выполните задания по вариантам.

Таблица 2–Задания по вариантам

	Вариант				
	1	2	3	4	5
Задание	1	1	1	1	1
	2	2	2	2	2
	3	3	3	3	3
	4	5	6	7	8
	13	12	11	10	9