# Matplotlib. Часть 1

Библиотека matplotlib требует предварительной установки, это можно сделать с помощью менеджера пакетов **pip**

In [ ]:
```
# pip install matplotlib
```

## Графики в декартовой системе координат

Импортируем необходимые библиотеки

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

Зададим массив "иксов" и "игреков"

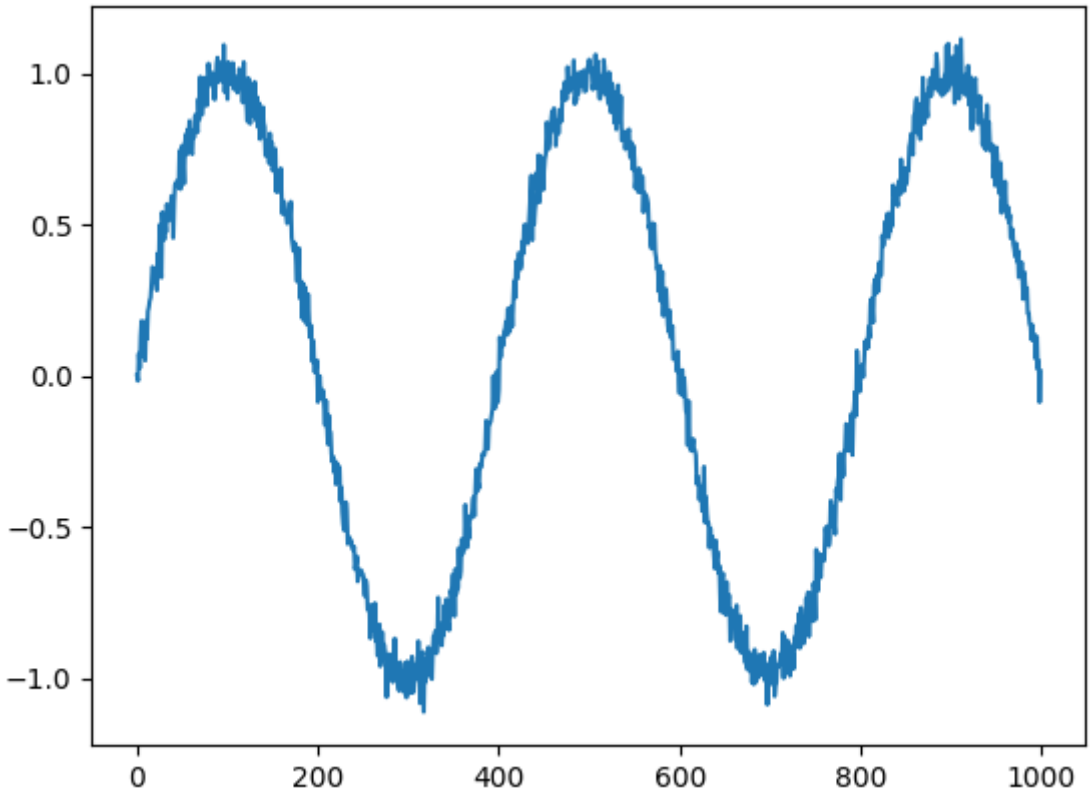
In [ ]:
```
x = np.linspace(0, 5*np.pi, 1000)
y = np.sin(x)+0.05*np.random.randn(1000)
```

За отображение графика отвечает команда **plot()**.

Если подать в функцию один массив, тогда в качестве значений по оси абсцисс будут отсчёты (номера точек).

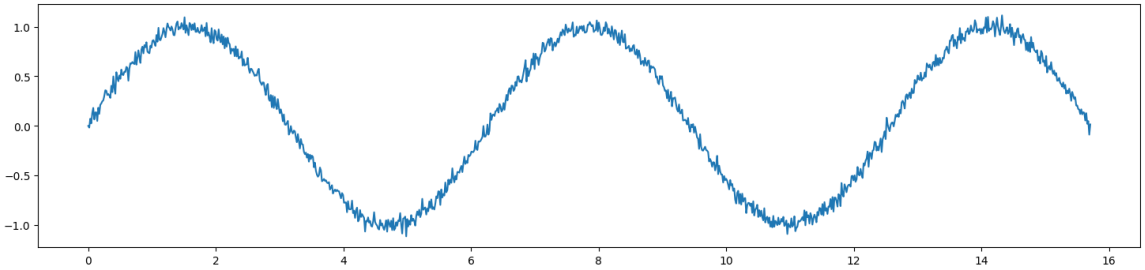
Для отображения графика используется команда **plt.show()**

In [ ]:
```
plt.plot(y)

plt.show()
```




Изменим размеры графика с помощью параметра *figsize* (размеры в дюймах), а также выведем истинные значения на оси абсцисс

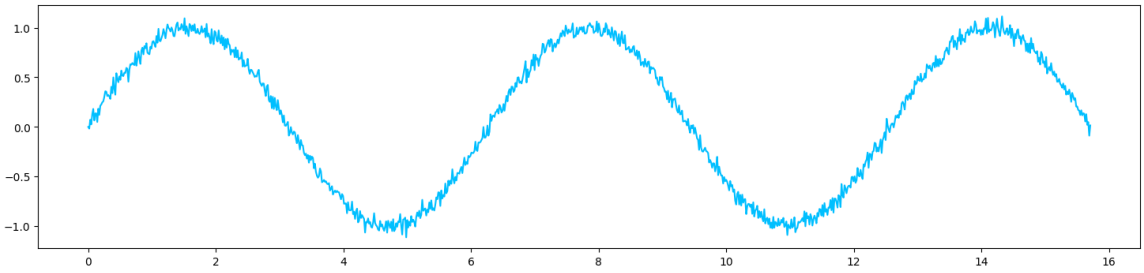
In [ ]:
```
plt.figure(figsize=(18, 4))

plt.plot(x, y)

plt.show()
```




Поменяем цвет на "глубоконебесноголубой"="deepskyblue" с помощью параметра *color* (*c*).

Толщину и стиль линии можно настроить с помощью параметров *linewidth* (*lw*) и *linestyle* (*ls*) соответственно

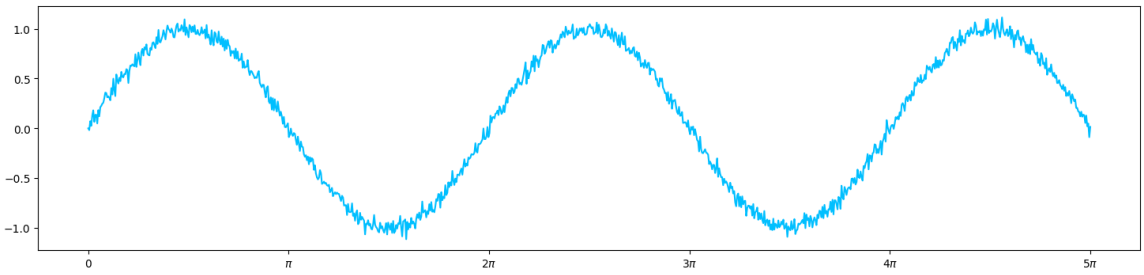
In [ ]:
```
plt.figure(figsize=(18, 4))

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

plt.show()
```




Хотелось бы, чтобы значения на оси абсцисс были в радианах. Для того, чтобы настроить метки оси создадим два массива:

- points - точки, в которых нужны метки
- labels - значения меток в этих точках

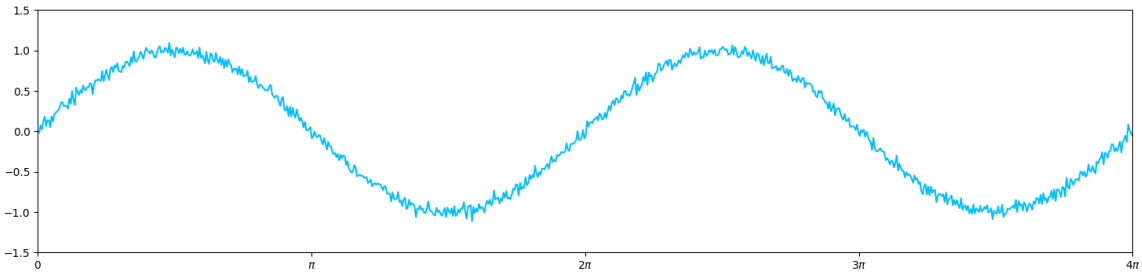
In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

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

plt.show()
```




Можно настроить пределы отображений по осям с помощью **plt.xlim()** и **plt.ylim**.

В качестве параметров указываются минимальное и максимальное значения отрезка

In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

plt.xlim(0, 4*np.pi)
plt.ylim(-1.5, 1.5)

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

plt.show()
```

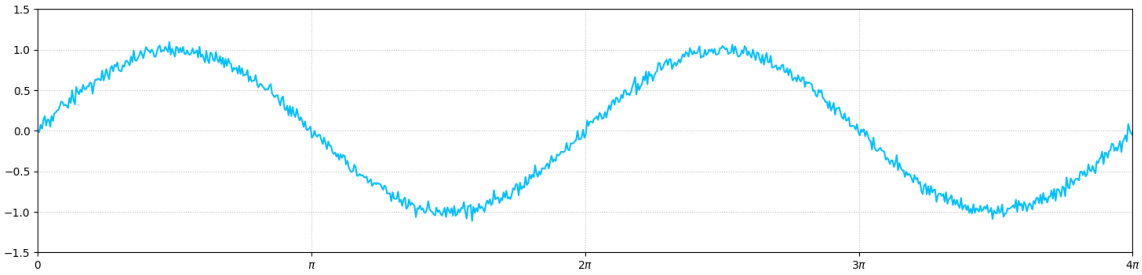



Добавим сетку с помощью **plt.grid()**.

В данном примере подаются необязательные параметры:

- alpha - настраивает прозрачность
- linestyle - стиль линии

In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

plt.xlim(0, 4*np.pi)
plt.ylim(-1.5, 1.5)

plt.grid(alpha=0.75, linestyle=':')

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

plt.show()
```



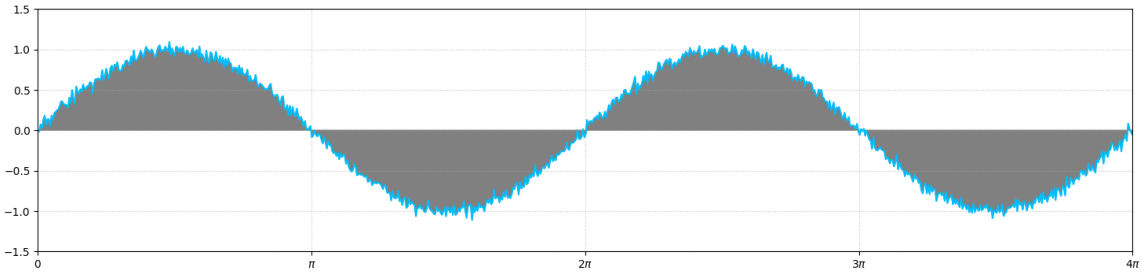

Добавим заливку серым цветом с помощью **plt.fill_between()**.

В данном примере параметры:

- x - массив значений иксов
- 0 - от какого значения заливать
- y - до какого значения заливать
- color - каким цветом

In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

plt.xlim(0, 4*np.pi)
plt.ylim(-1.5, 1.5)
```

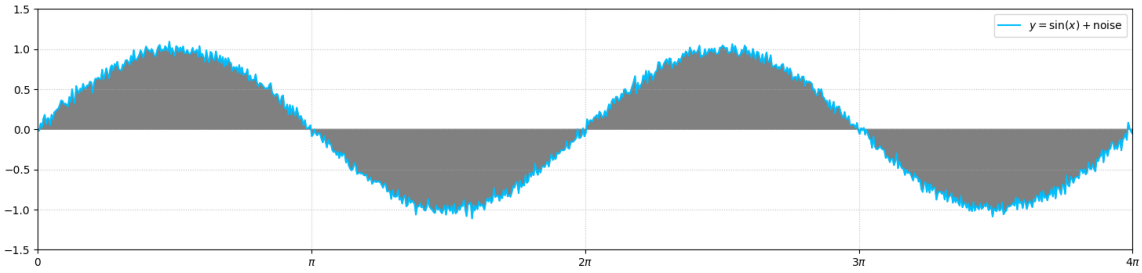
```
plt.grid(alpha=0.75, linestyle=':')

plt.plot(x, y, color='deepskyblue')
plt.fill_between(x, 0, y, color='gray')

plt.show()
```




Добавим легенду с помощью **plt.legend()**.

Легенда собирается автоматически, поэтому добавим параметр *label* в **plt.plot()**

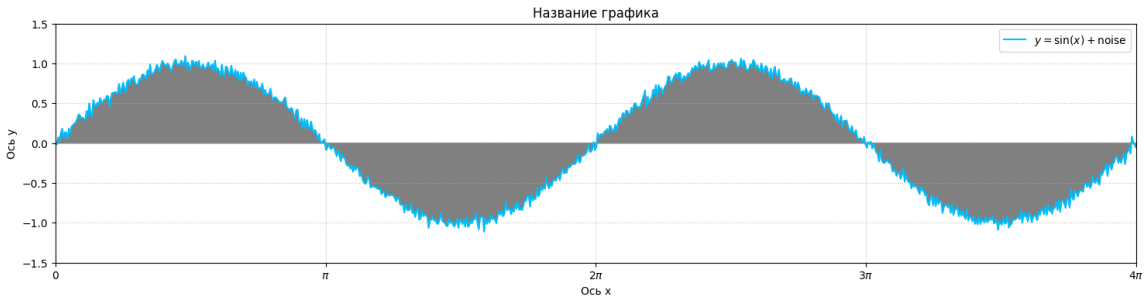
In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

plt.xlim(0, 4*np.pi)
plt.ylim(-1.5, 1.5)

plt.grid(alpha=0.75, linestyle=':')

plt.plot(x, y, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.fill_between(x, 0, y, color='gray')

plt.legend()

plt.show()
```




Добавим заголовок графика и подписи осей

In [ ]:
```
plt.figure(figsize=(18, 4))

points = np.array([0,  np.pi, 2*np.pi,  3*np.pi, 4*np.pi, 5*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$', r'$3\pi$', r'$4\pi$', r'$5\pi$']
plt.xticks(points, labels)

plt.xlim(0, 4*np.pi)
plt.ylim(-1.5, 1.5)
```

```
plt.grid(alpha=0.75, linestyle=':')

plt.plot(x, y, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.fill_between(x, 0, y, color='gray')

plt.legend()

plt.title('Название графика')
plt.xlabel('Ось x')
plt.ylabel('Ось y')

plt.show()
```

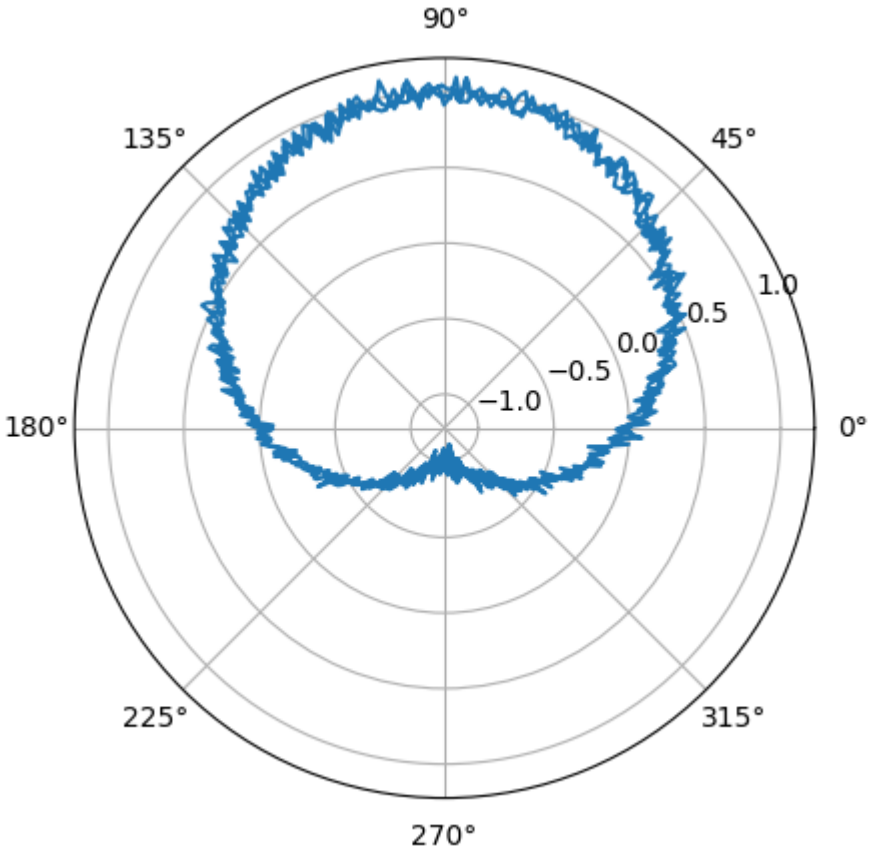



---

# Графики в полярной системе координат

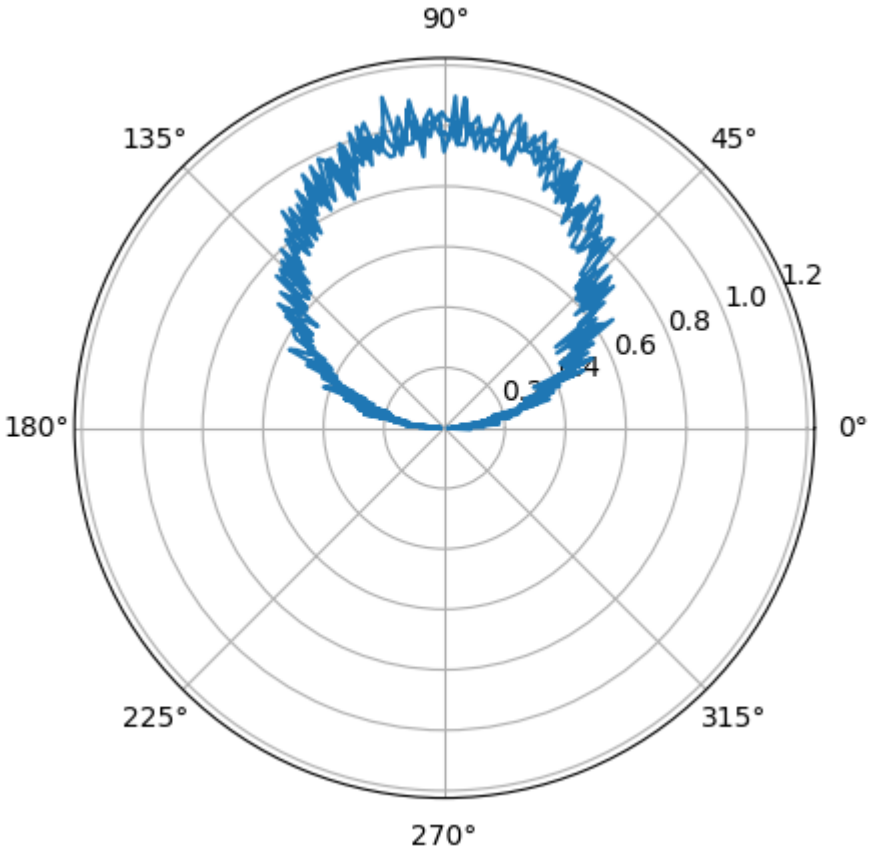
В самом простейшем случае полярный график можем вывести с помощью команды **plt.polar**, однако при этом полученное изображение будет выглядеть непривычно из-за того, что в начале координат вместо нуля будет минимальное значение функции

In [ ]:
```
plt.polar(x, y)

plt.show()
```



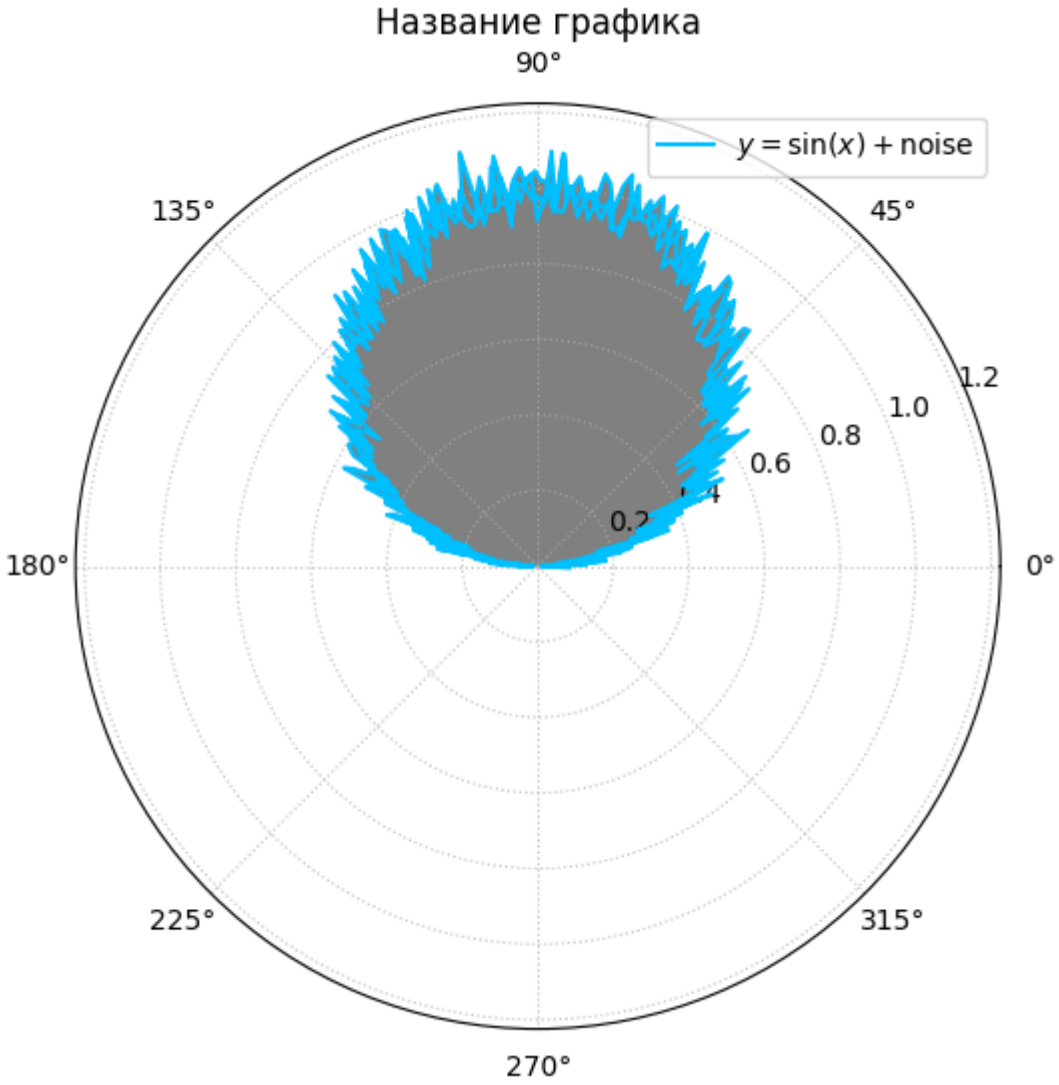

Настроим его с помощью **plt.ylim()**

In [ ]:
```
plt.polar(x, y)

plt.ylim(bottom=0)

plt.show()
```




Применим некоторые настройки отображения из предыдущего раздела

In [ ]:
```
plt.figure(figsize=(6, 6))

plt.polar(x, y, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')

plt.fill_between(x, 0, y, color='gray')
plt.title('Название графика')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')
plt.legend()

plt.show()
```

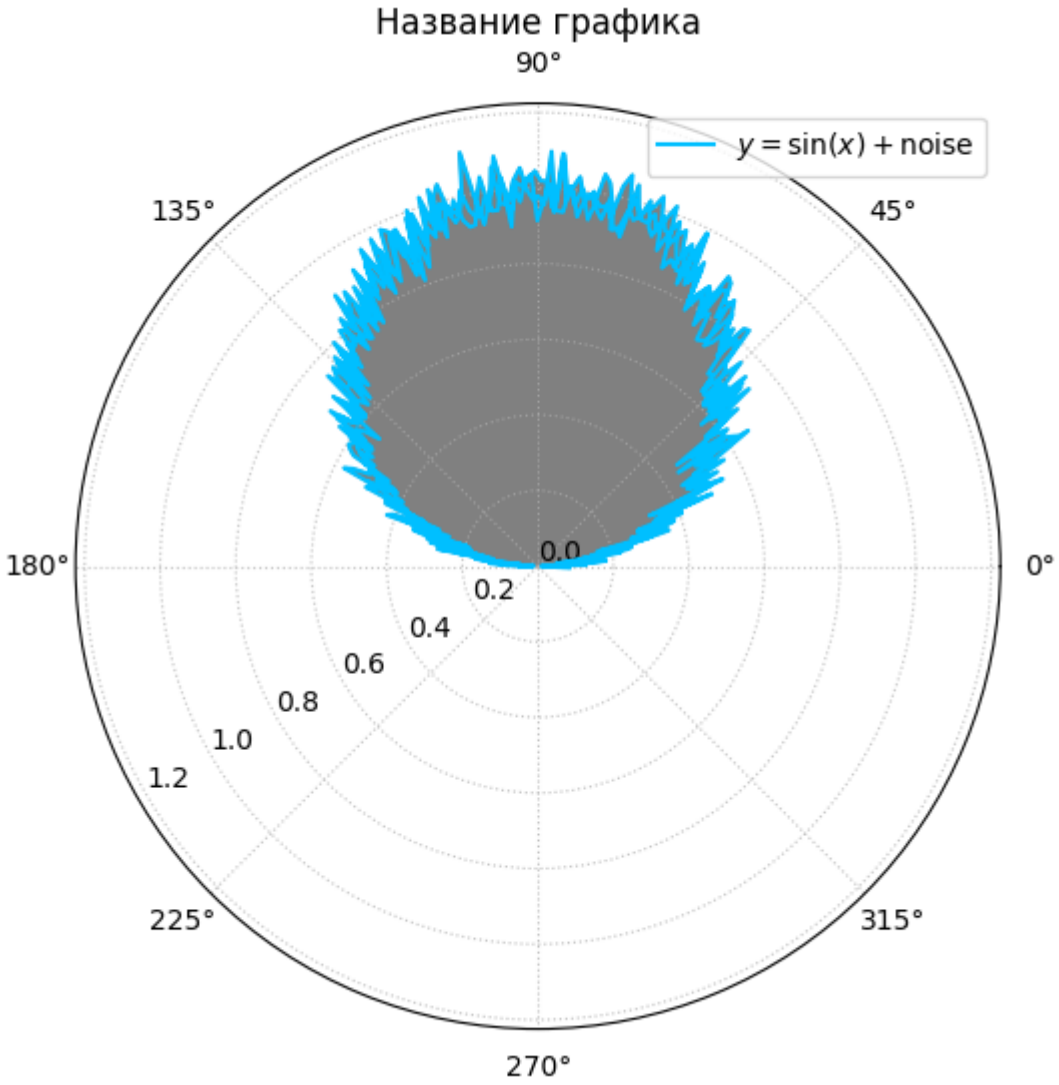



Дополнительно настроим расположение меток оси с помощью обращения к зоне Axes командой plt.gca().

gca = get current axes

plt

In [ ]:
```
plt.figure(figsize=(6, 6))

plt.polar(x, y, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')

plt.fill_between(x, 0, y, color='gray')
plt.title('Название графика')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')
plt.legend()

ax = plt.gca()
ax.set_rgrids([0.2*i for i in range(7)])
ax.set_rlabel_position(210)

plt.show()
```

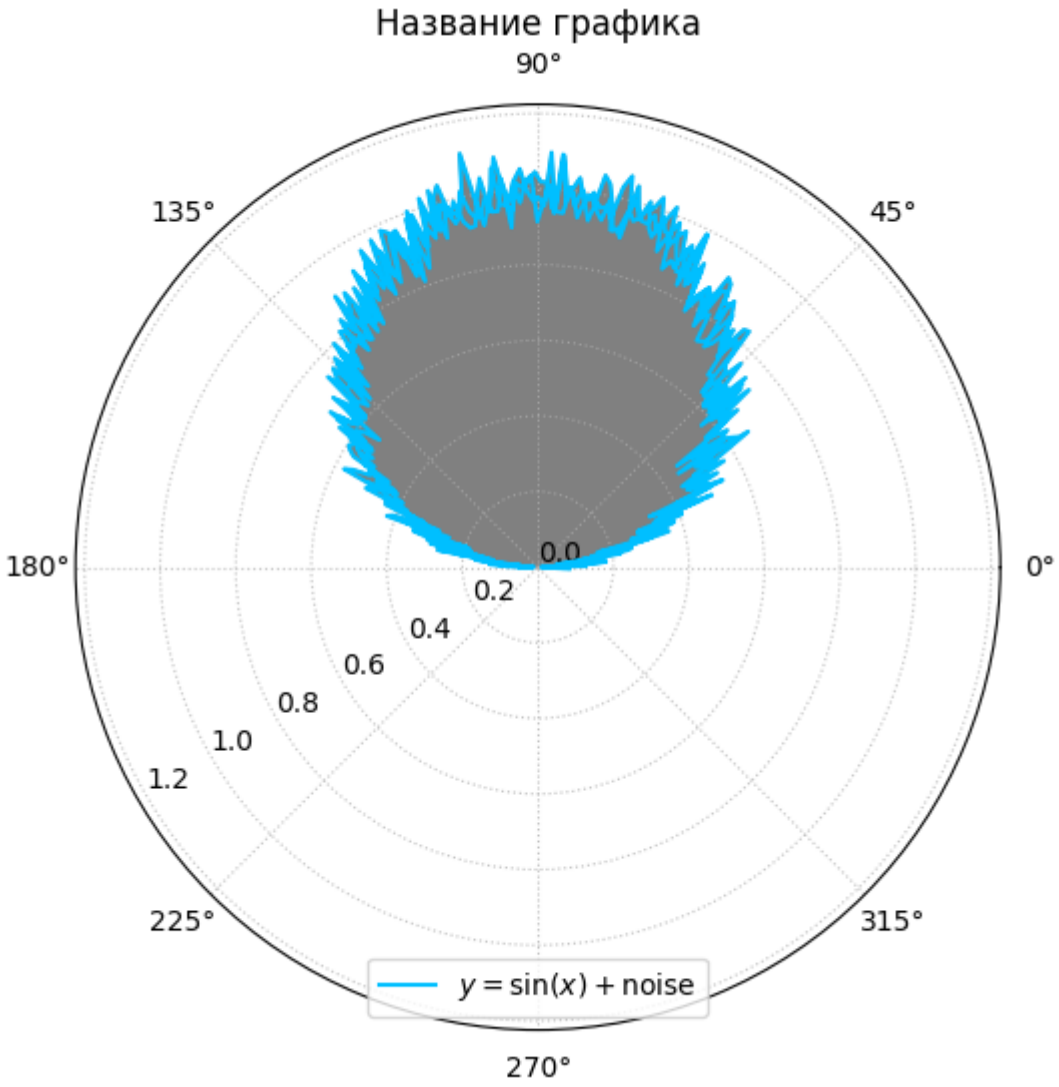



Настроим расположение легенды с помощью параметра loc

| Location String | Location Code |
|---|---|
| 'best' (Axes only) | 0 |
| 'upper right' | 1 |
| 'upper left' | 2 |
| 'lower left' | 3 |
| 'lower right' | 4 |
| 'right' | 5 |
| 'center left' | 6 |
| 'center right' | 7 |
| 'lower center' | 8 |
| 'upper center' | 9 |
| 'center' | 10 |

In [ ]:
```
plt.figure(figsize=(6, 6))

plt.polar(x, y, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
```

```
plt.fill_between(x, 0, y, color='gray')
plt.title('Название графика')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')
plt.legend(loc=8)

ax = plt.gca()
ax.set_rgrids([0.2*i for i in range(7)])
ax.set_rlabel_position(210)

plt.show()
```

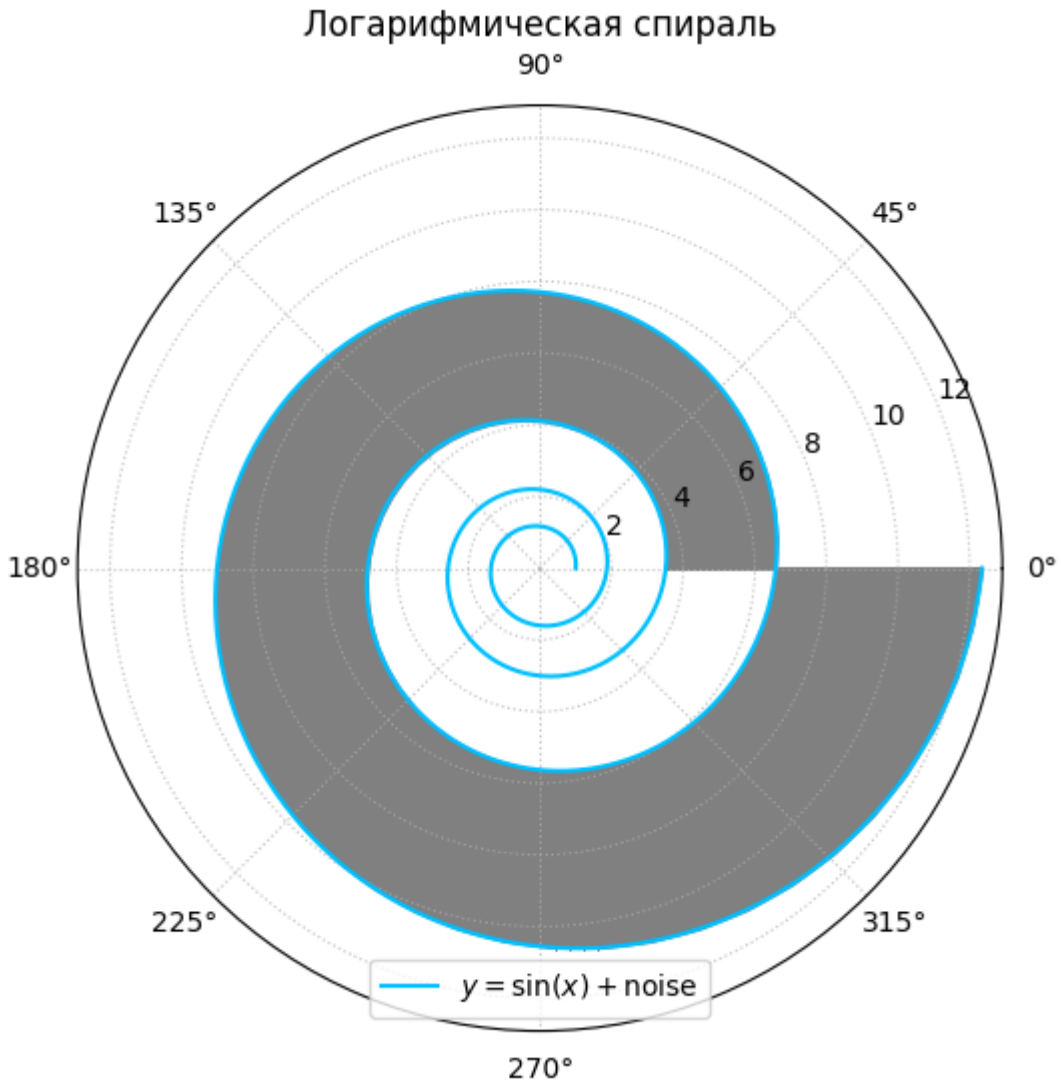



В качестве дополнительного примера выведем график логарифмической спирали и закрасим область одного витка

In [ ]:
```
def spiral(x):
    return np.exp(0.1*x)


x_spiral = np.linspace(0, 8*np.pi, 1000)
y_spiral = spiral(x_spiral)
```

In [ ]:
```
plt.figure(figsize=(6, 6))
```

```
plt.polar(x_spiral, y_spiral, color='deepskyblue',
          label=r'$y=\sin(x)+\mathrm{noise}$')

x_bot = np.linspace(4*np.pi, 6*np.pi)
x_top = np.linspace(6*np.pi, 8*np.pi)
plt.fill_between(x_bot, spiral(x_bot), spiral(x_top), color='gray')

plt.legend(loc=8)
plt.title('Логарифмическая спираль')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')

plt.show()
```




---

# Отображение нескольких графиков на одном рисунке

Зададим массивы значений

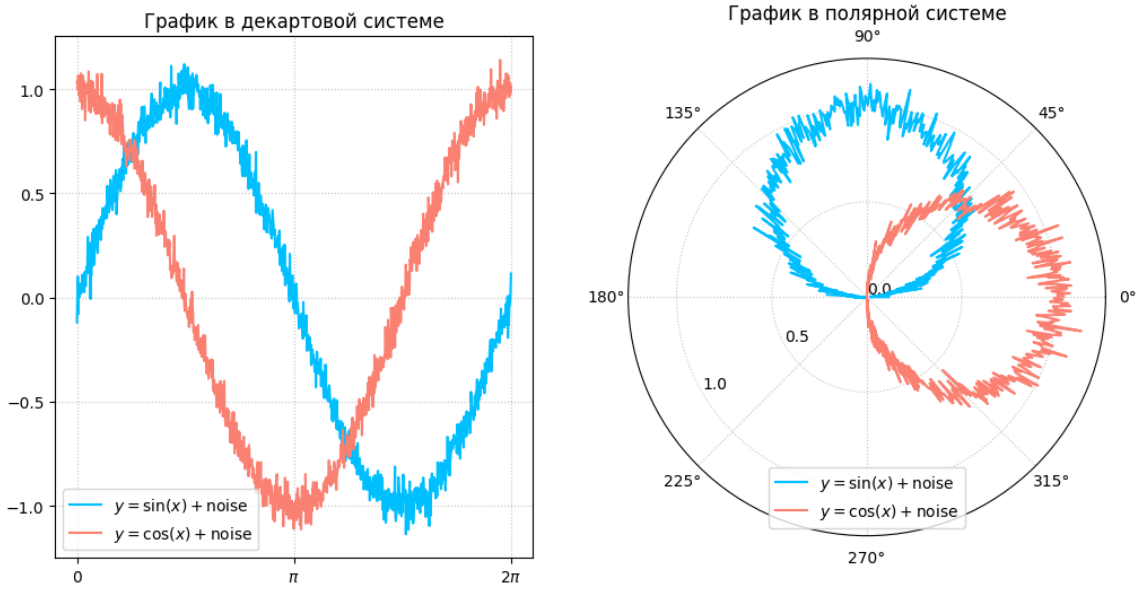
In [ ]:
```
t = np.linspace(0, 2*np.pi, 1000)
y_sin = np.sin(t)+0.05*np.random.randn(1000)
y_cos = np.cos(t)+0.05*np.random.randn(1000)
```

Подграфик создаётся с помощью команды **plt.subplot()**.

В качестве параметра вводится трёхзначное число или последовательность из трёх значений:

1. Число строк
2. Число столбцов
3. Номер ячейки

In [ ]:
```
plt.figure(figsize=(12, 6))

plt.subplot(121)

plt.plot(t, y_sin, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.plot(t, y_cos, color='salmon', label=r'$y=\cos(x)+\mathrm{noise}$')
plt.grid(alpha=0.75, linestyle=':')
points = np.array([0, np.pi, 2*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$']
plt.xticks(points, labels)
plt.title('График в декартовой системе')
plt.legend(loc=3)

plt.subplot(122, projection='polar')

ax = plt.gca()
ax.set_rgrids(np.arange(0, 1.5, 0.5))
ax.set_rlabel_position(210)
plt.polar(t, y_sin, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.polar(t, y_cos, color='salmon', label=r'$y=\cos(x)+\mathrm{noise}$')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')
plt.title('График в полярной системе')
plt.legend(loc=8)

plt.show()
```

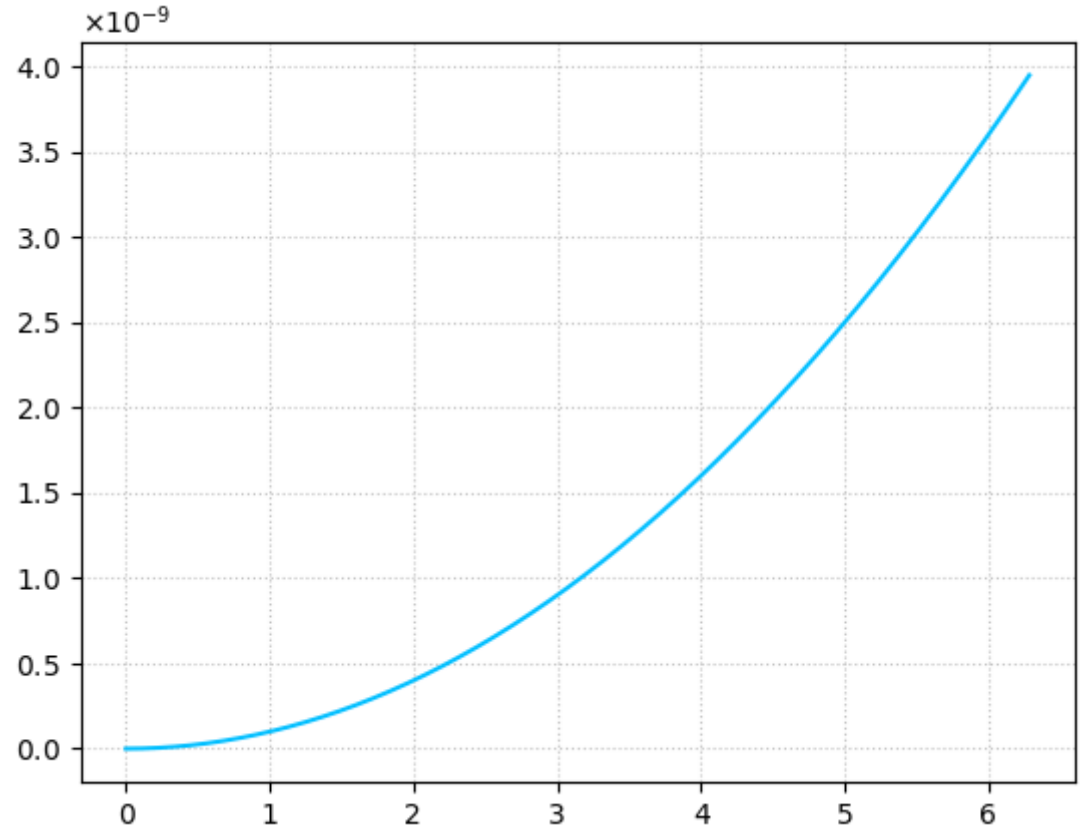


# Настройка меток осей

In [ ]:
```
from matplotlib import ticker
```

In [ ]:
```
formatter = ticker.ScalarFormatter(useMathText=True)
formatter.set_scientific(True)
formatter.set_powerlimits((0,10))
ax.yaxis.set_major_formatter(formatter)
```

In [ ]:
```
y_small=t**2*1e-10

plt.plot(t, y_small, color='deepskyblue')
plt.grid(alpha=0.75, linestyle=':')

ax=plt.gca()
ax.yaxis.set_major_formatter(formatter)
```




---

# Сохранение графиков

Реализуется с помощью команды **plt.savefig()**.

Чаще всего требуется два варианта:

- png
- eps - векторный формат

Эти строки кода добавляются в самом конце построения графика, но перед **plt.show()**

In [ ]:
```
# plt.savefig('name.png', dpi=300)
# plt.savefig('name.eps', format='eps')
```

In [ ]:
```
plt.figure(figsize=(12, 6))

plt.subplot(121)

plt.plot(t, y_sin, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.plot(t, y_cos, color='salmon', label=r'$y=\cos(x)+\mathrm{noise}$')
plt.grid(alpha=0.75, linestyle=':')
points = np.array([0, np.pi, 2*np.pi])
labels = [r'$0$', r'$\pi$', r'$2\pi$']
plt.xticks(points, labels)
plt.title('График в декартовой системе')
plt.legend(loc=3)

plt.subplot(122, projection='polar')

ax = plt.gca()
ax.set_rgrids(np.arange(0, 1.5, 0.5))
ax.set_rlabel_position(210)
plt.polar(t, y_sin, color='deepskyblue', label=r'$y=\sin(x)+\mathrm{noise}$')
plt.polar(t, y_cos, color='salmon', label=r'$y=\cos(x)+\mathrm{noise}$')
plt.ylim(bottom=0)
plt.grid(alpha=0.75, linestyle=':')
plt.title('График в полярной системе')
plt.legend(loc=8)

plt.savefig('example_plot.png', dpi=300)

plt.show()
```