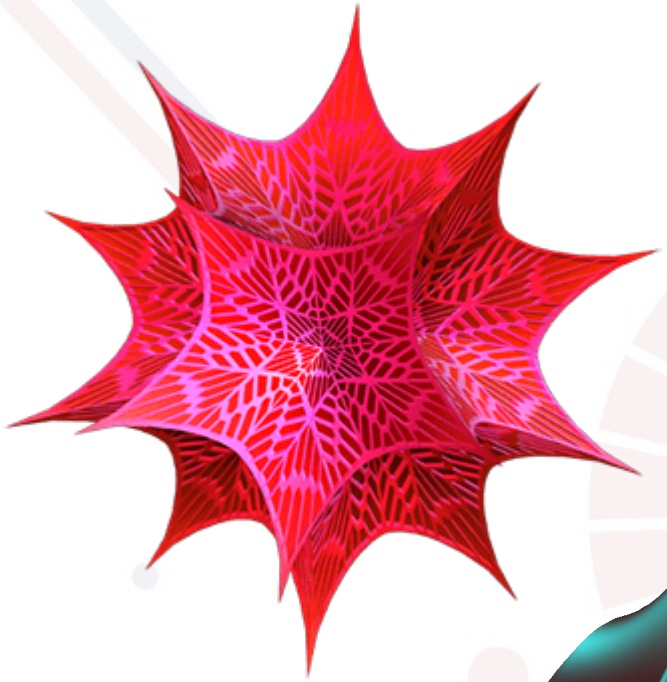
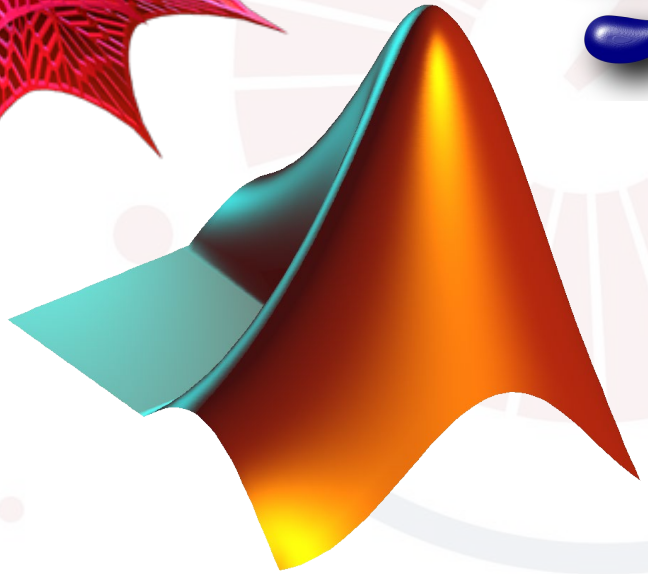
Применение Python в инженерных и научных расчетах

Мишин А.А., кафедра ВМСС





python™

Почему Python?

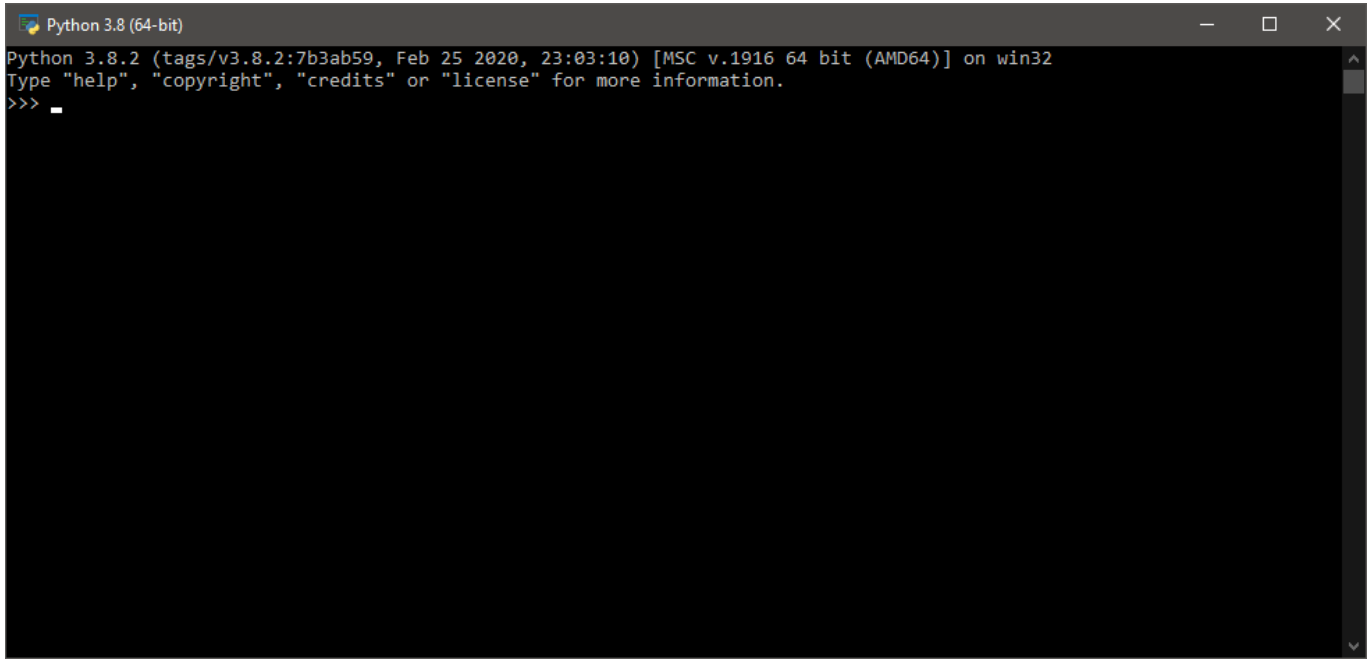
- Простой
- Популярный
- Открытый
- Мультипарадигменный
- Общеприменимый
- Расширяемый

Почему иногда все же не Python?

- Медленный
- Непостоянный
- Не всегда есть готовое решение

REPL

REPL (Readevaluate-print loop)



Пример программы

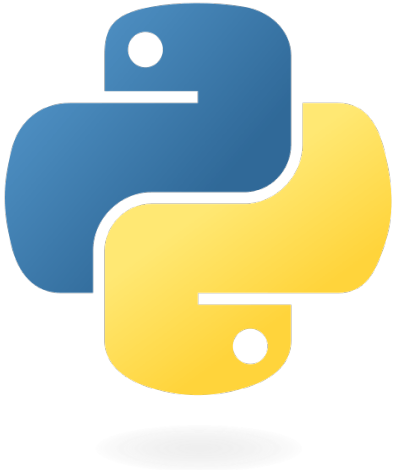
```
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

Стандартные модули

- numbers — Числовые абстрактные базовые классы
- math — Математические функции
- cmath — Математические функции для комплексных чисел
- decimal — Десятичная арифметика с фиксированной и плавающей запятой
- fractions — Рациональные числа
- random — Генерация псевдослучайных чисел
- statistics — Функции математической статистики

…

Нестандартные модули



PyPi

NumPy

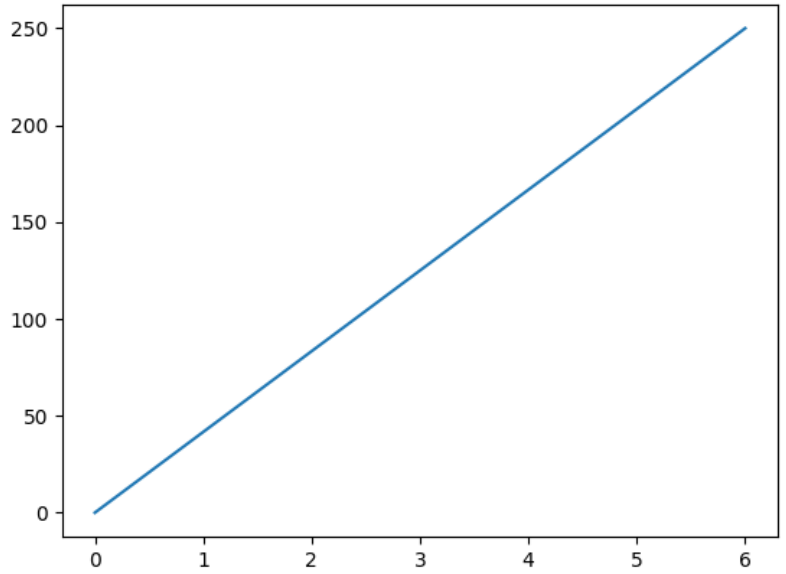
Пример : вычислить среднеквадратичную ошибку

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

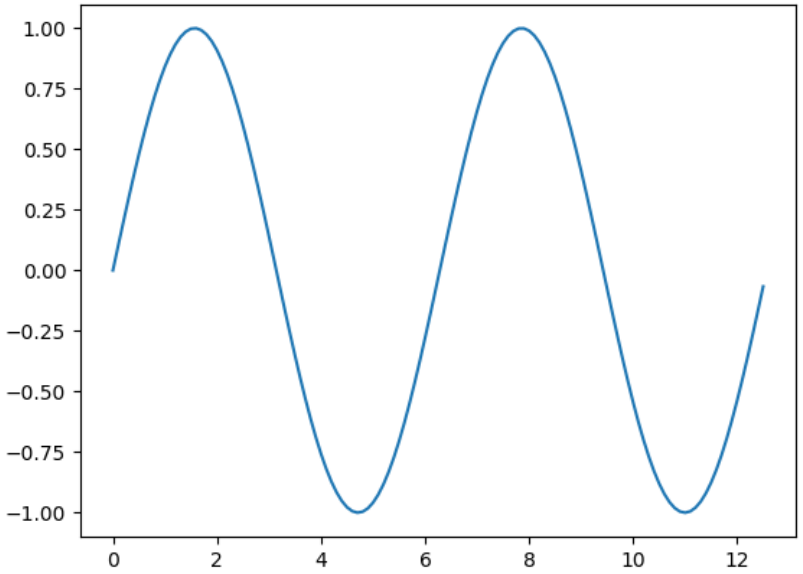
```
y1  = np.array([1, 2, 3, 4, 5, 6])
y2  = np.array([[2, 2, 3], [4, 5, 5]]).flatten()
mse = (1/n) * np.sum(np.square(y1-y2))
```

MatPlotLib

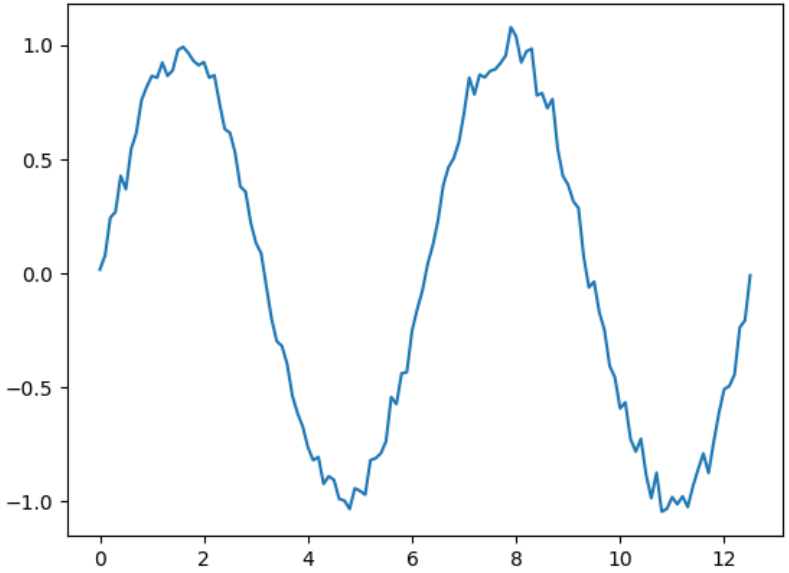
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0, 6])
y = np.array([0, 250])

plt.plot(x, y)
plt.show()
```



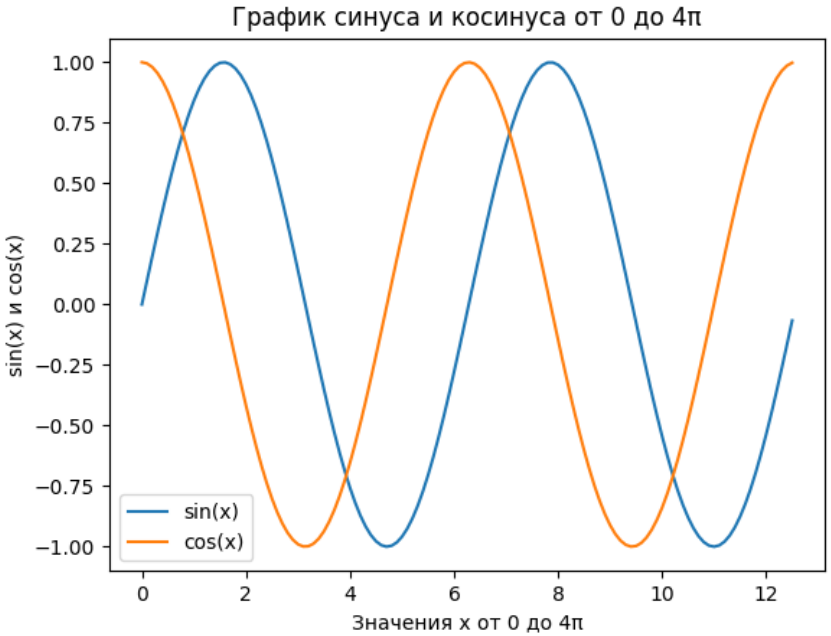
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,4*np.pi,0.1)
y = np.sin(x)

plt.plot(x, y)
plt.show()
```



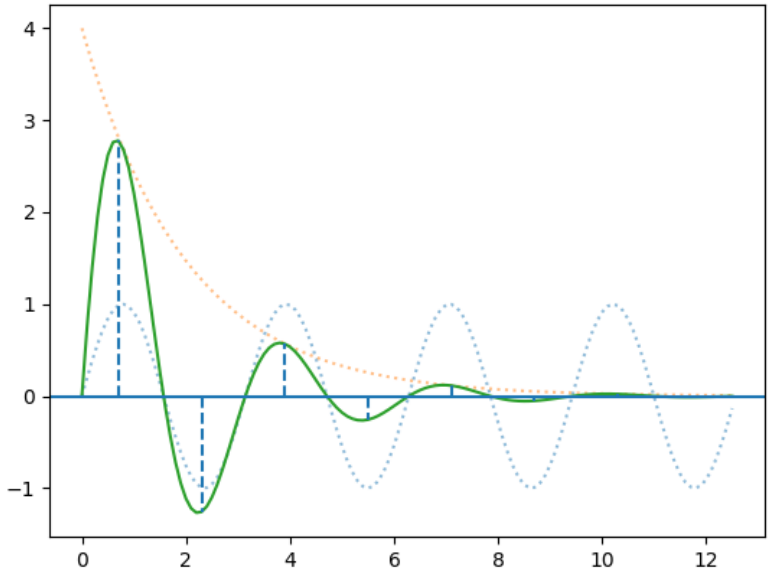
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,4*np.pi,0.1)
y = np.sin(x)
noise = np.random.normal(0,0.05,y.size)
y += noise

plt.plot(x, y)
plt.show()
```



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

x = np.arange(0,4*np.pi,0.1)
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x,y1,x,y2)
plt.xlabel('Значения x от 0 до 4π')
plt.ylabel('sin(x) и cos(x)')
plt.title('График синуса и косинуса
от 0 до 4π')
plt.legend(['sin(x)', 'cos(x)'])
plt.show()
```



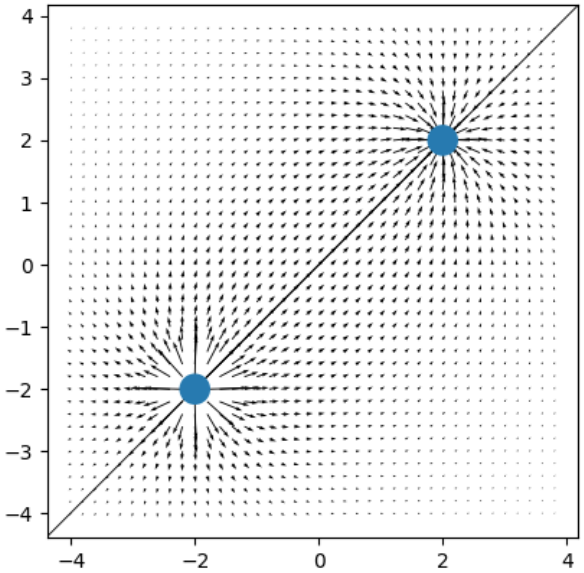
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,4*np.pi,0.1)
y1 = np.sin(2*x)
y2 = np.exp(-x/2) * 4
y3 = y1 * y2

plt.plot(x,y1,':',alpha=0.5)
plt.plot(x,y2,':',alpha=0.5)
plt.plot(x,y3)

plt.vlines(x=x[7::16],ymin=0,yma
x=y3[7::16],linestyles='dashed')
plt.axhline(y=0)
plt.show()
```

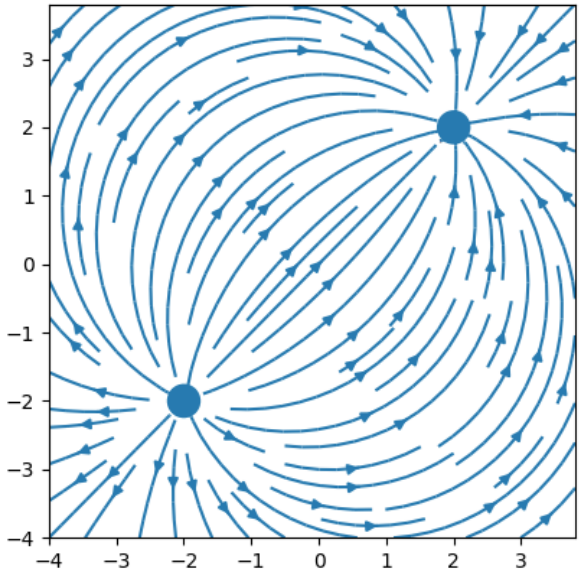


Пример : изобразить модель диполя

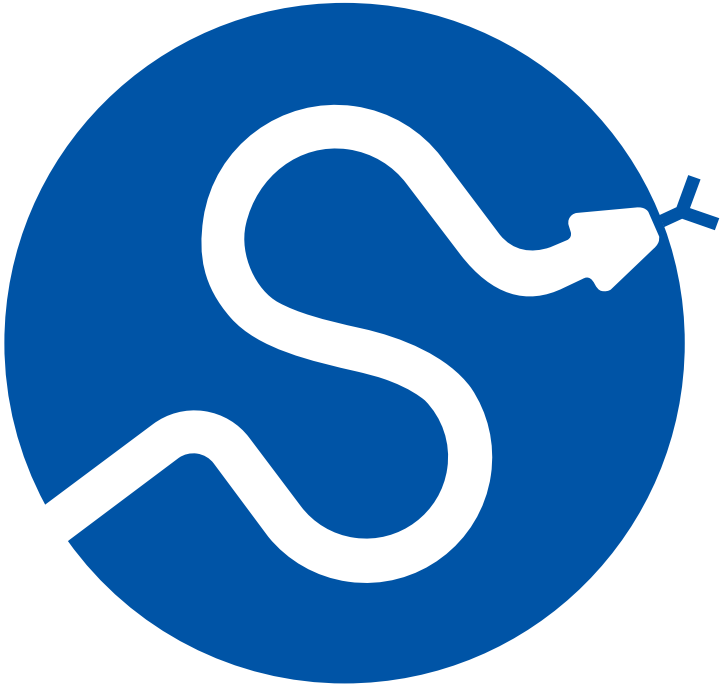
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

Ch = np.array([2,2])
pcircle = Circle(Ch, .25)
ncircle = Circle(-1*Ch, .25)
X,Y = np.meshgrid(np.arange(-10,10,.2), np.arange(-10,10,.2) )
U = (X + Ch[0])/((X+Ch[0])**2 + (Y+Ch[1])**2) - (X - Ch[0])/((X-Ch[0])**2 + (Y-Ch[1])**2)
V = (Y+Ch[1])/((X+Ch[0])**2 + (Y+Ch[1])**2) - (Y-Ch[1])/((X-Ch[0])**2 + (Y-Ch[1])**2)

plt.figure()
plt.streamplot(X,Y,U,V)

#plt.figure()
#plt.quiver(X,Y,U,V,scale=50)

plt.gca().add_patch(pcircle)
plt.gca().add_patch(ncircle)

plt.show()
```

Пример : изобразить модель диполя



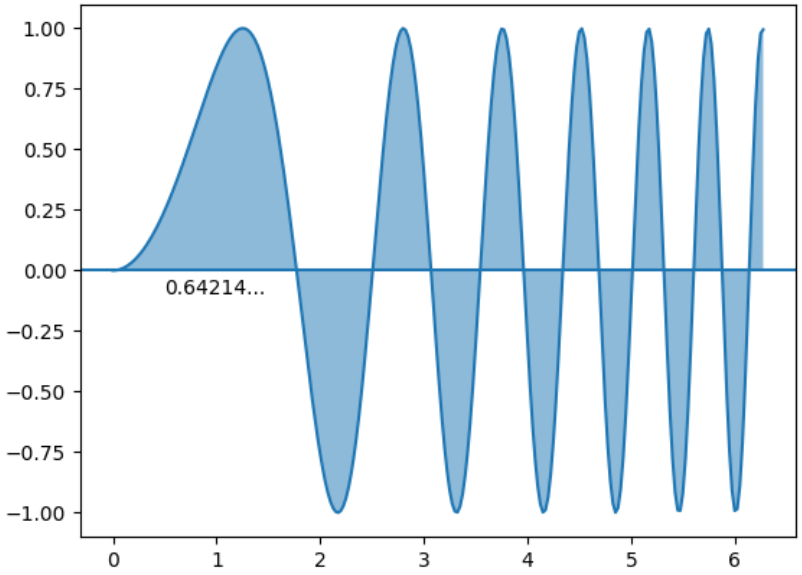


SciPy



$$\mathrm{I} = \int_0^{2\pi} \sin(x^2)\, dx$$

```
import matplotlib.pyplot as plt
import scipy.integrate as intgr
import numpy as np

def f(x):
    return np.sin(x**2)

x = np.arange(0,2*np.pi,0.025)
y = f(x)

res, err = intgr.quad(f, 0, 2*np.pi)

plt.axhline(y=0)
plt.fill_between(x,y, alpha=0.5)
plt.plot(x, y)
plt.text(0.5, -0.1, "%.5f..."%res)
plt.show()
```

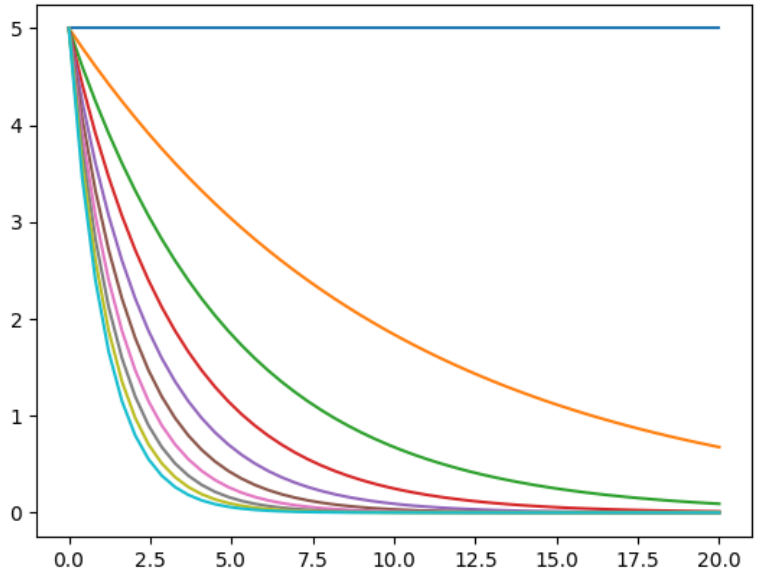


$$\frac{dy(t)}{dx} = -k\, y(t)\,,$$

$$y_0 = 5$$

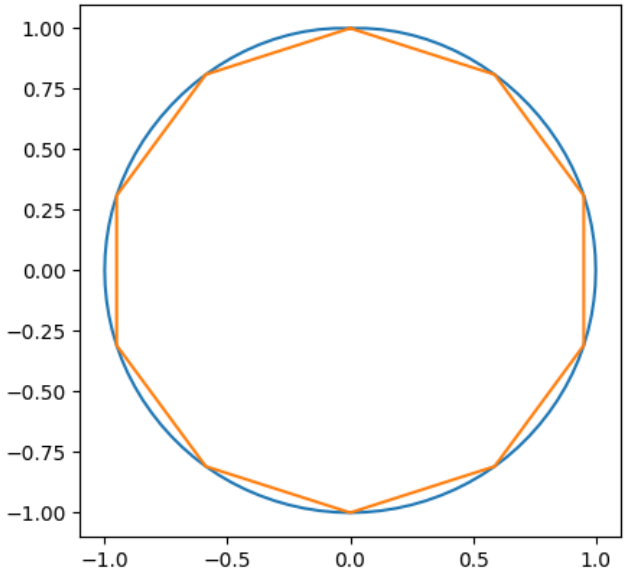
$$k = [0; 1]$$

```
import matplotlib.pyplot as plt
import scipy.integrate as intgr
import numpy as np

def f(y,t,k):
    dydt = -k * y
    return dydt

y0 = 5
t = np.linspace(0,20)
k = np.arange(0,1,0.1)

for z in k:
    y = intgr.odeint(f,y0,t,args=(z,))
    plt.plot(t,y)
plt.show()
```



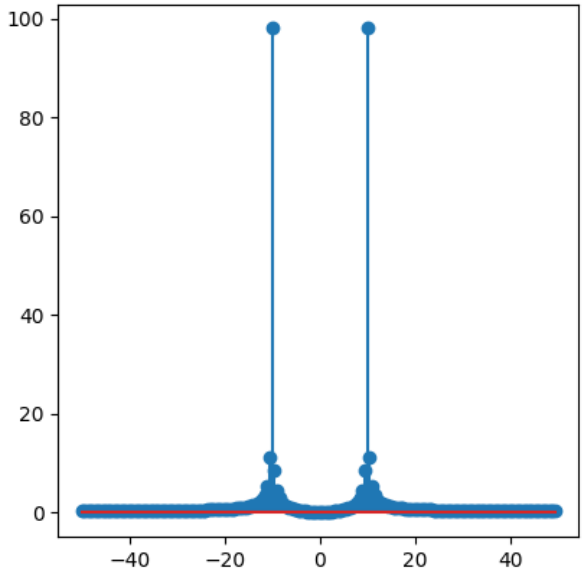
$$x = \sin(2\pi t)$$
$$x = \cos(2\pi t)$$
$$t = [0; 1]$$

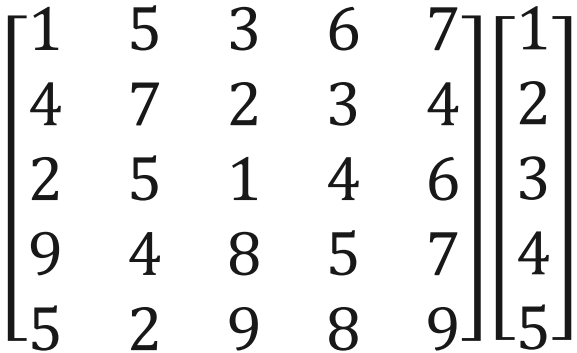
```
import matplotlib.pyplot as plt
import scipy.interpolate as inter
import numpy as np

t = np.arange(0, 1.1, .1)
x = np.sin(2*np.pi*t)
y = np.cos(2*np.pi*t)
tck, u = inter.splprep([x, y], s=0)
unew = np.arange(0, 1.01, 0.01)
out = inter.splev(unew, tck)
plt.plot(out[0], out[1], x, y)
plt.show()
```



$$DFT[10Hz\ sine]$$

```
import matplotlib.pyplot as plt
import scipy.fft as ff
import numpy as np

freq = 10
samprate = 100

t = np.linspace(0,2,2*samprate)
x = np.sin(freq*2*np.pi*t)

fx = ff.fft(x)
freqs = ff.fftfreq(len(x))*samprate

plt.stem(freqs, np.abs(fx))
plt.show()
```

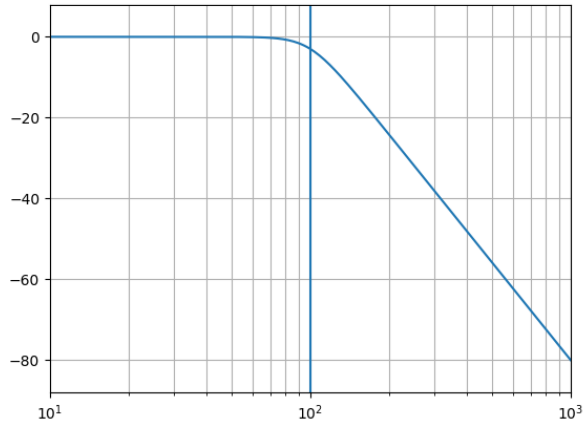




```
import scipy.linalg as la
import numpy as np

A = np.array([[1,5,3,6,7],
              [4,7,2,3,4],
              [2,5,1,4,6],
              [9,4,8,5,7],
              [5,2,9,8,9]])
b = np.arange(1,6)

print(la.det(A))
print(la.solve(A, b))
```

ФНЧ

```
import matplotlib.pyplot as plt
import scipy.signal as sig
import numpy as np

b, a = sig.butter(4, 100, 'low', analog=True)
w, h = sig.freqs(b, a)
plt.semilogx(w, 20 * np.log10(abs(h)))
plt.margins(0, 0.1)
plt.grid(which='both', axis='both')
plt.axvline(100)
plt.show()
```



SymPy

```
import math
import sympy

math.sqrt(9)  # 3.0
math.sqrt(8)  # 2.82842712475
sympy.sqrt(3) # sqrt(3)
sympy.sqrt(8) # 2*sqrt(2)
```

```
from sympy import symbols

x, y = symbols('x y')
expr = x + 2*y
expr           # x + 2*y
expr + 1       # x + 2*y + 1
expr - x       # 2*y
```

```
from sympy import symbols, expand, factor


x, y = symbols('x y')
expr = x + 2*y
expanded_expr = expand(x*expr)
expanded_expr          # x**2 + 2*x*y
factor(expanded_expr) # x*(x + 2*y)
```

```
from sympy import *

x = symbols('x’)
diff(sin(x)*exp(x), x) # exp(x)*sin(x) + exp(x)*cos(x)
integrate(exp(x)*sin(x) + exp(x)*cos(x), x) # exp(x)*sin(x)
integrate(sin(x**2), (x, -oo, oo)) # sqrt(2)*sqrt(pi)/2
limit(sin(x)/x, x, 0) # 1
solve(x**2 - 2, x) # [-sqrt(2), sqrt(2)]
```