Практическое занятие 2

Математический анализ

https://docs.sympy.org/latest/tutorial/intro.html (https://docs.sympy.org/latest/tutorial/intro.html)

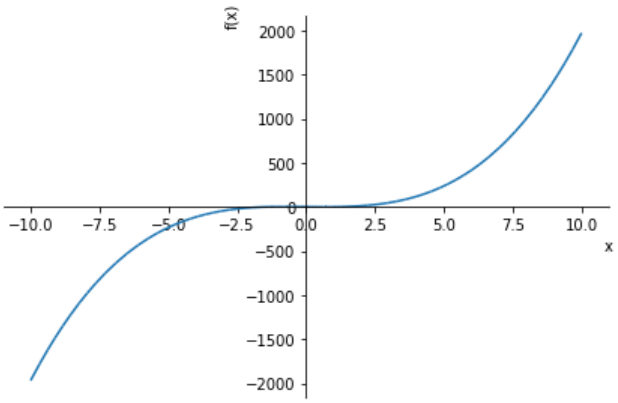
In [1]:
```
# Вначале для простоты будем подключать модуль sympy целиком
from sympy import *
# А это уже магия!
%matplotlib inline
```

Графики в Sympy

Пусть есть функция одной переменной, равная выражению, содержащему эту переменную, например, $f(x) = 2x^3 - 3x + 1$.

Построить график этой функции можно с помощью метода $plot$.

In [2]:
```
x = Symbol('x')
f = 2*x**3 - 3*x + 1
plot(f)
```



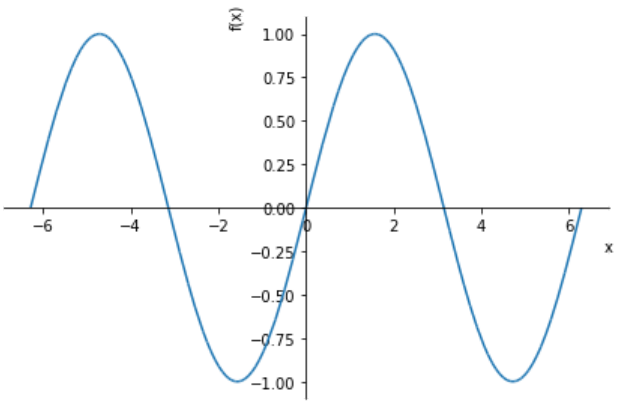
Out[2]:
```
<sympy.plotting.plot.Plot at 0x236540a5630>
```

Можно задать интервал значений переменной, для которого требуется построить график. Можно передавать выражение для функции в качестве аргумента plot. Для того, чтобы задать интервал значений переменной, нужно указать значение необязательного параметра, представляющего собой tuple из имени переменной, левого и правого конца интервала.

Пример 1

Построим $f(x) = \sin(x)$ на $[-2\pi, 2\pi]$.

In [3]:
```
plot(sin(x), (x, -2*pi, 2*pi))
```



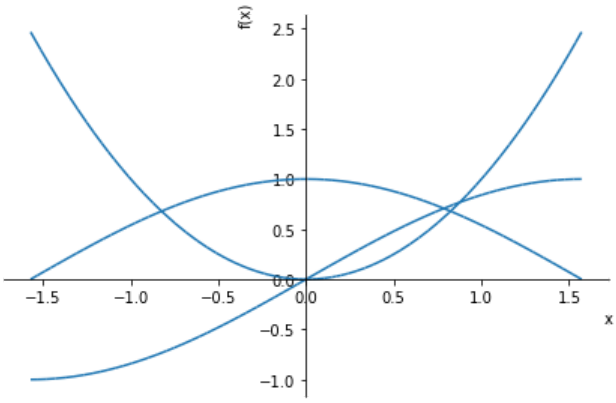
Out[3]:
```
<sympy.plotting.plot.Plot at 0x18f3a981748>
```

На одном графике можно изобразить несколько функций, они перечисляются через запятую при вызове plot. При это можно (но необязательно!) указать интервал значений переменной.

Пример 2

Построим на одном графике несколько функций: $\sin(x)$, $\cos(x)$ и x^2 на $[-\pi/2, \pi/2]$.

In [4]:
```
plot(sin(x), cos(x), x**2,
     (x, -pi/2, pi/2))
```



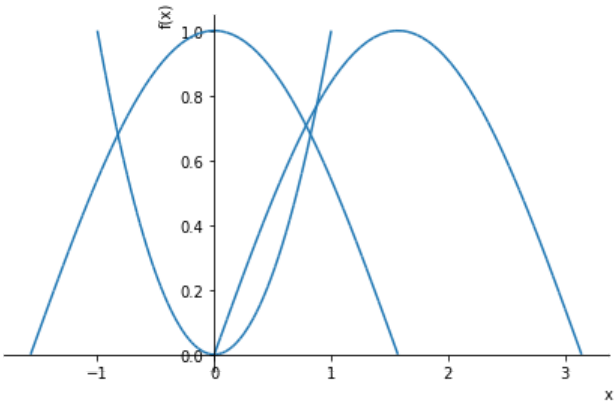
Out[4]:
```
<sympy.plotting.plot.Plot at 0x18f3adb98d0>
```

Если на одном графике нужно изобразить несколько функций, причем каждую на своем интервале, то при вызове plot перечисляются через запятую tuple, состоящие из выражения для функции (или имени переменной, в которую это выражение записано) и tuple, описывающей интервал значений функции (этот tuple состоит из имени переменной, левого и правого конца интервала).

Пример 3

Построим на одном графике функцию $\sin(x)$ на $[0, \pi]$, $\cos(x)$ на $[-\pi/2, \pi/2]$ и x^2 на $[-1, 1]$.

In [5]:
```
plot((sin(x), (x, 0, pi)),
     (cos(x), (x, -pi/2, pi/2)),
     (x**2, (x, -1, 1)))
```



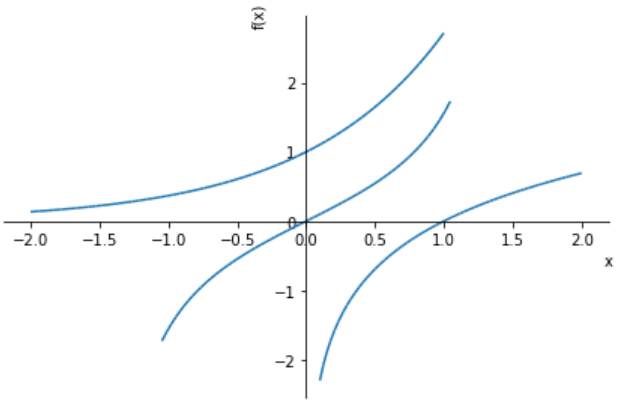
Out[5]:
```
<sympy.plotting.plot.Plot at 0x18f3b1eca20>
```

Пример 4

Построим на одном графике функции $\tan(x)$ на $[-\pi/3, \pi/3]$, $\exp(x)$ на $[-2, 1]$ и $\log(x)$ на $[0.1, 2]$.

Составим список из функций и список из интервалов, на основе этих списков составим tuple, состоящий из tuple, содержащих выражение для функции и tuple из имени переменной, левого и правого конца интервала.

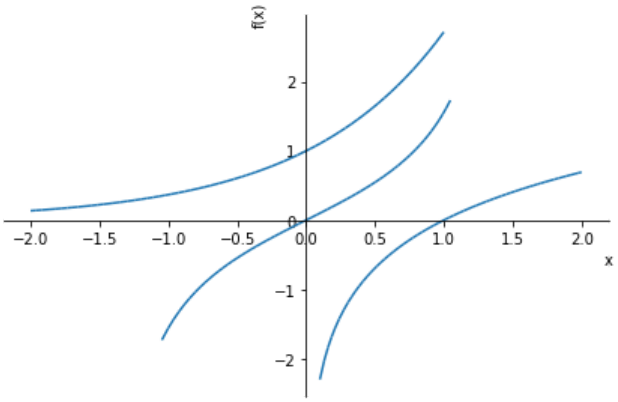
In [6]:
```
funcs = [tan(x), exp(x), log(x)]
intervals = [(-pi/3, pi/3), (-2, 1), (0.1, 2)]
n = len(funcs)
func_to_plot = tuple((funcs[i], (x, intervals[i][0], intervals[i][1])) for i in range(n))
plot(*func_to_plot)
```



Out[6]: `<sympy.plotting.plot.Plot at 0x18f3b155f28>`

Обратите внимание на * перед funcs и на tuple в предпоследней строчке кода. Без tuple получится генератор, его нельзя использовать в plot. Можно вместо этого создать список list:

In [7]:
```
func_to_plot = [(funcs[i], (x, intervals[i][0], intervals[i][1])) for i in range(n)]
plot(*func_to_plot)
```



Out[7]: `<sympy.plotting.plot.Plot at 0x18f3aea20b8>`

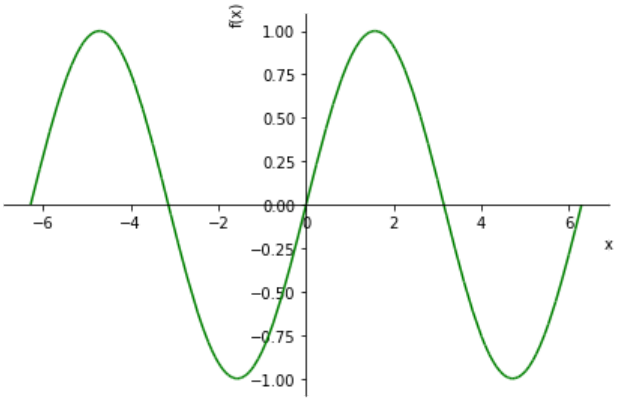
Цвет линии графика

Цвет линии определяется параметром line_color, по умолчанию blue.

Пример 5

Построим зеленый график $\sin(x)$ на $[-2\pi, 2\pi]$.

In [8]:
```
plot(sin(x), (x, -2*pi, 2*pi), line_color='green')
```



Out[8]: `<sympy.plotting.plot.Plot at 0x18f3b5ea630>`

Действия со списками. Append.

Для добавления элемента в конец списка можно воспользоваться методом append.

Пример 6

Создадим пустой список my_list и добавим к нему сначала 1, потом π, затем x^i, $i = 0, \ldots, 4$.

In [9]:
```
my_list = []
my_list.append(1)
display(my_list)
my_list.append(pi)
display(my_list)
for i in range(5):
    my_list.append(x**i)
display(*my_list)
```

```
[1]
```

```
[1, pi]
```

1

π

1

x

x^2

x^3

x^4

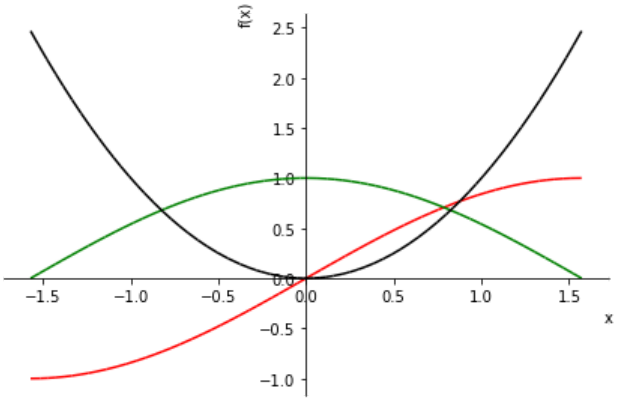
Разноцветные графики в одной плоскости.

Для того, чтобы на одном графике вывести несколько функций разных цветов нужно создать несколько графиков с помощью plot, но не изображать их, а собрать в список с помощью append как в Примере 6. Для того, чтобы график не изображался, изменим параметр show, равный по умолчанию True.

Пример 7

Выведем на одном графике функции из Примера 2: $\sin(x)$, $\cos(x)$ и x^2 на $[-\pi/2, \pi/2]$. Цвета для графиков будем брать из списка colors.

In [10]:
```
interval = (x, -pi/2, pi/2)
p = plot(sin(x), interval, line_color='red', show=False)
p.append(plot(cos(x), interval, line_color='green', show=False)[0])
p.append(plot(x**2, interval, line_color='black', show=False)[0])
p.show()
```



Параметры plot, их значения по умолчанию. Название, пределы по осям, легенда и т.п.

Полный список аргументов plot:

plot(*args, title=None, xlabel=None, ylabel=None, aspect_ratio='auto', xlim=None, ylim=None, axis_center='auto', axis=True, xscale='linear', yscale='linear', legend=False, autoscale=True, margin=0, annotations=None, markers=None, rectangles=None, fill=None, backend='default',* *kwargs)

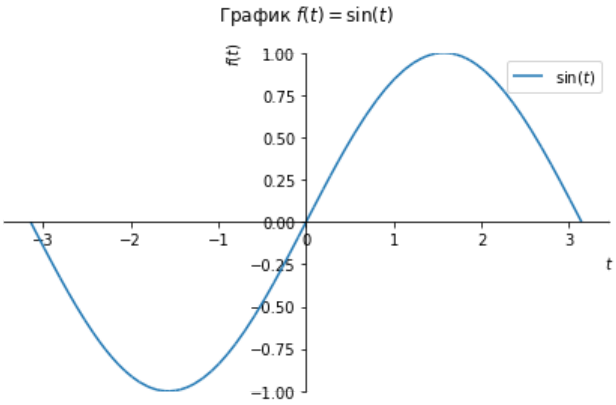
Подробности в документации

https://docs.sympy.org/latest/modules/plotting.html?highlight=plot#module-sympy.plotting.plot (https://docs.sympy.org/latest/modules/plotting.html?highlight=plot#module-sympy.plotting.plot)

Пример 8

Построим график функции $\sin(t)$, название графика "График $f(t) = \sin(t)$", подписи к осям t и $f(t)$, пределы по вертикальной оси (-1, 1), обозначение графика в легенде $\sin(t)$.

In [11]:
```
t = Symbol('t')
plot(sin(t), (t, -pi, pi), title='График $f(t) = \sin(t)$\n', xlabel='$t$', ylabel='$f(t)$',
     ylim=(-1, 1), legend=True, label='$\sin(t)$')
```



Out[11]: <sympy.plotting.plot.Plot at 0x18f3c6235c0>

График неявно заданной функции

Изобразить график уравнения, например, окружности, удобнее с помощью функции plot_implicit. Уравнение неявно заданной функции передается plot_implicit как Equality, сокращенно Eq. Аргументами Eq выступают левая и правая части уравнения (lhs и rhs соответственно).

Уравнение окружности:

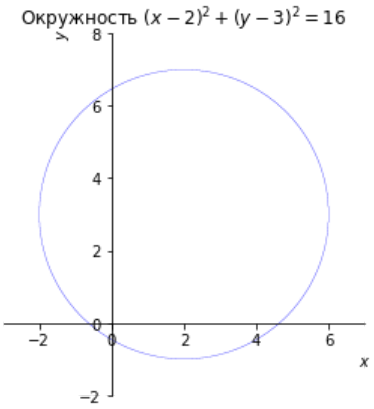
$$(x - x_0)^2 + (y - y_0)^2 = R^2,$$

где x_0 и y_0 - координаты центра окружности, R - радиус.

Пример 9

Нарисуем окружность с центром в точке $A(2, 3)$ радиуса 4. Используем параметр aspect_ratio для того, чтобы окружность была круглой, а не похожей на эллипс. Параметр aspect_ratio='auto' по умолчанию, а если нужно задать какое-то другое значение, то нужно присвоить aspect_ratio значение, равное tuple из двух чисел float, отношение этих чисел определяет соотношение масштабов по осям. Нам нужно, чтобы масштаб по каждой оси был одинаковым, поэтому передаем aspect_ratio=(1, 1). Название графика 'Окружность $(x - 2)^2 + (y - 3)^2 = 16$'.

In [12]:
```
x, y =symbols('x y')
plot_implicit(Eq((x - 2)**2 + (y - 3)**2, 16),
              (x, -3, 7),
              (y, -2, 8),
              aspect_ratio=(1, 1),
              title='Окружность $(x - 2)^2 + (y - 3)^2 = 16$',
              xlabel='$x$',
              ylabel='$y$')
```



Out[12]: <sympy.plotting.plot.Plot at 0x18f3c78d198>

График параметрически заданной функции

Параметрически заданная функция от одной переменной определяется как совокупность зависимостей двух переменных от одной общей переменной, называемой параметром. Например, параметрическое уравнение окружности записывается так:

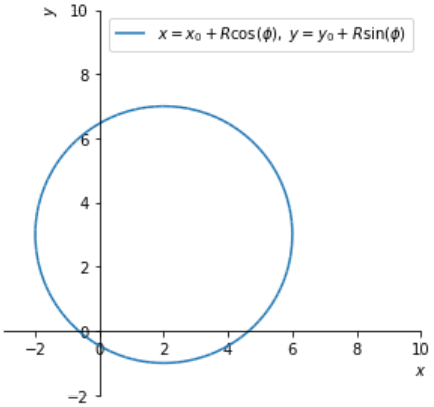
$$\begin{cases} x = x_0 + R\cos(\phi) \\ y = y_0 + R\sin(\phi), \end{cases} \quad \phi \in (0, 2\pi]$$

Изобразить график параметрически заданной функции можно с помощью функции plot_parametric, передавая ей в качестве аргументов выражения для x и y.

Пример 10

Нарисуем окружность с центром в точке $A(2, 3)$ радиуса 4 из примера 9 с помощью функции plot_parametric. Для того, чтобы оси координат проходили через начало координат задаем axis_center=(0, 0), а для того, чтобы пределы по осям были как в Примере 9, полагаем xlim=(-3, 10), ylim=(-2, 10). В легенду включим формулы уравнений парамерически заданной окружности, записанные в строчку.

In [13]:
```
plot_parametric(2 + 4*cos(t), 3 + 4*sin(t), (t, 0, 2*pi),
                aspect_ratio=(1, 1),
                axis_center=(0, 0),
                xlim=(-3, 10),
                ylim=(-2, 10),
                xlabel='$x$', ylabel='$y$',
                legend=True,
                label='$x = x_0 + R\cos(\phi),\ y = y_0 + R\sin(\phi)$')
```



Out[13]: <sympy.plotting.plot.Plot at 0x18f3c8c41d0>