Решение задач на Python, Производная

Вычисление производных

Для вычисления производных используется библиотека sympy. Вычисление производной - функция diff(f,x,k). Вычисляет производную /<-го порядка функции y = f(x) по переменной x. Параметры: f-функция; x- переменная, по которой берется производная, k - необязательный параметр, порядок производной.

Ввод [1]:

```
from sympy import Symbol, limit, oo, sin, sqrt, solve, factorial, symbols, cos, exp, asi
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

Пример 1

Ввод [2]:

```
x = symbols('x')
y = x*cos(x)
diff(x*cos(x), x)
```

Out[2]:

$$-x\sin(x)+\cos(x)$$

Пример 2

Ввод [3]:

```
x = symbols('x')
diff(log(x), x, 3)
```

Out[3]:

$$\frac{2}{x^3}$$

Пример 3

Ввод [4]:

```
y = log(x**3,10)**3
diff(y,x,2).subs(x,10)
```

Out[4]:

$$-\frac{9\,(-6+\log(1000))\log(1000)}{100\log(10)^3}$$

Ввод [5]:

```
diff(y,x,2).subs(x,10).simplify()
```

Out[5]:

$$\frac{81\cdot(2-\log(10))}{100\log(10)^2}$$

Пример 4

Ввод [6]:

```
y = (x**2+x-6)/(x**2-10*x+25)
z = diff(y,x)
z
```

Out[6]:

$$\frac{9\log\left(x^3\right)^2}{x\log(10)^3}$$

Ввод [7]:

```
solve(z, x)
```

Out[7]:

```
[1, -1/2 - sqrt(3)*I/2, -1/2 + sqrt(3)*I/2]
```

Пример 5

Ввод [8]:

```
x = symbols('x')
y = symbols('y')
f = x**2 + y**2 - 4
idiff(f, y, x)
```

Out[8]:

$$-\frac{x}{y}$$

Ввод [9]:

```
f = x**2 + y**2 - 4
idiff(f, y, x, 2)
```

Out[9]:

$$-\frac{\frac{x^2}{y^2}+1}{y}$$

Ввод [10]:

```
idiff(f, y, x, 2).simplify()
```

Out[10]:

$$\frac{-x^2-y^2}{y^3}$$

Пример 6

Ввод [11]:

```
t = symbols('t')
x = t - sin(t)
y = 1 - cos(t)
y_diff = diff(y,t)/diff(x,t)
y_diff
```

Out[11]:

$$\frac{\sin(t)}{1-\cos(t)}$$

Ввод [12]:

```
y_2diff = diff(y_diff,t)/diff(x,t)
y_2diff.simplify()
```

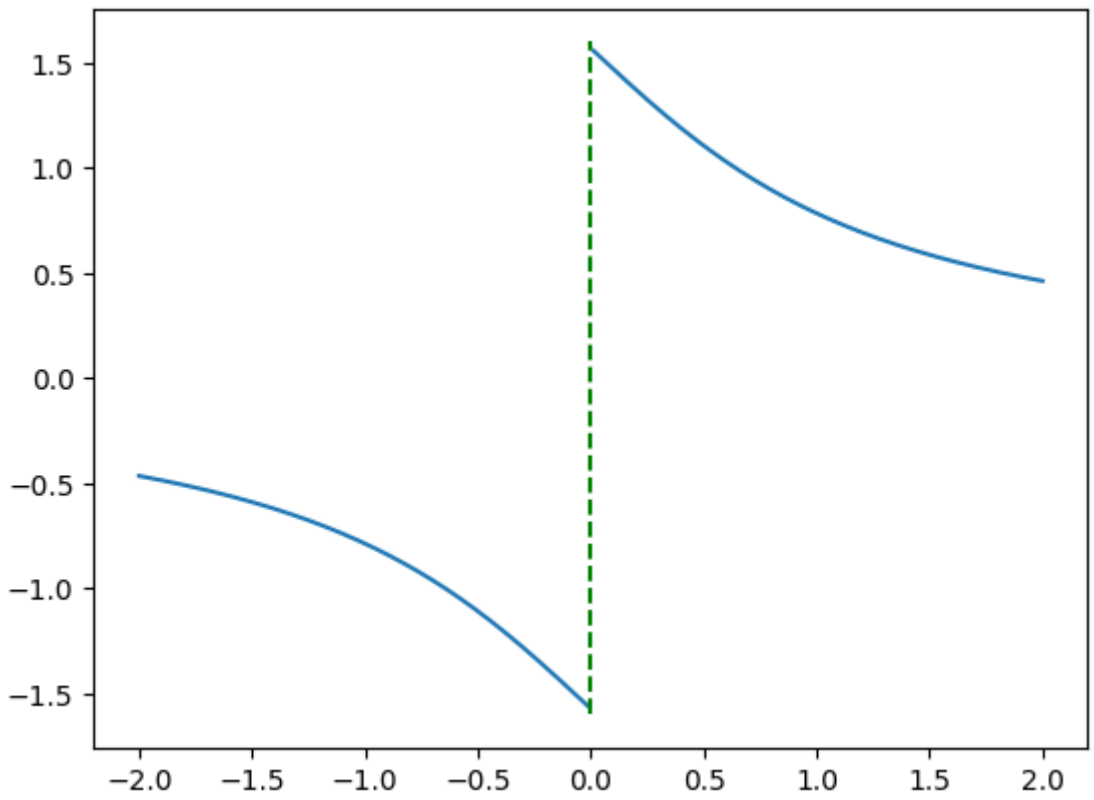
Out[12]:

$$-\frac{1}{(\cos(t)-1)^2}$$

Пример 7

Ввод [13]:

```
x = np.linspace(-2,2,500)
x[(x>-0.01) & (x < 0.01)] = np.nan
y = np.arctan(1/x)
plt.plot(x,y)
plt.vlines(0, -1.6, 1.6, color='g', linestyles='dashed')
plt.show()
```



Ввод [14]:

```
x = symbols('x')
y = atan(1/x)
z = diff(y,x)
limit(z, x, 0, dir='+')
```

Out[14]:

$$-1$$

Ввод [15]:

```
limit(z, x, 0, dir='-')
```

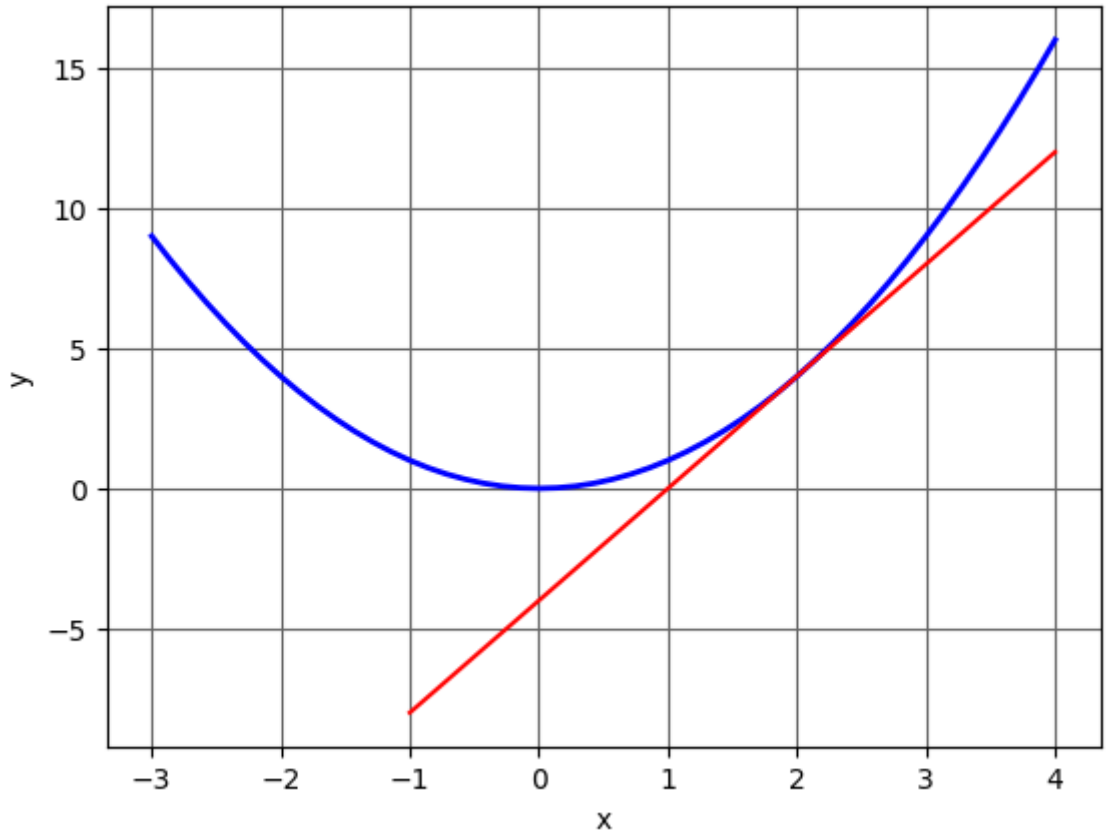
Out[15]:

$$-1$$

Пример 8

Ввод [16]:

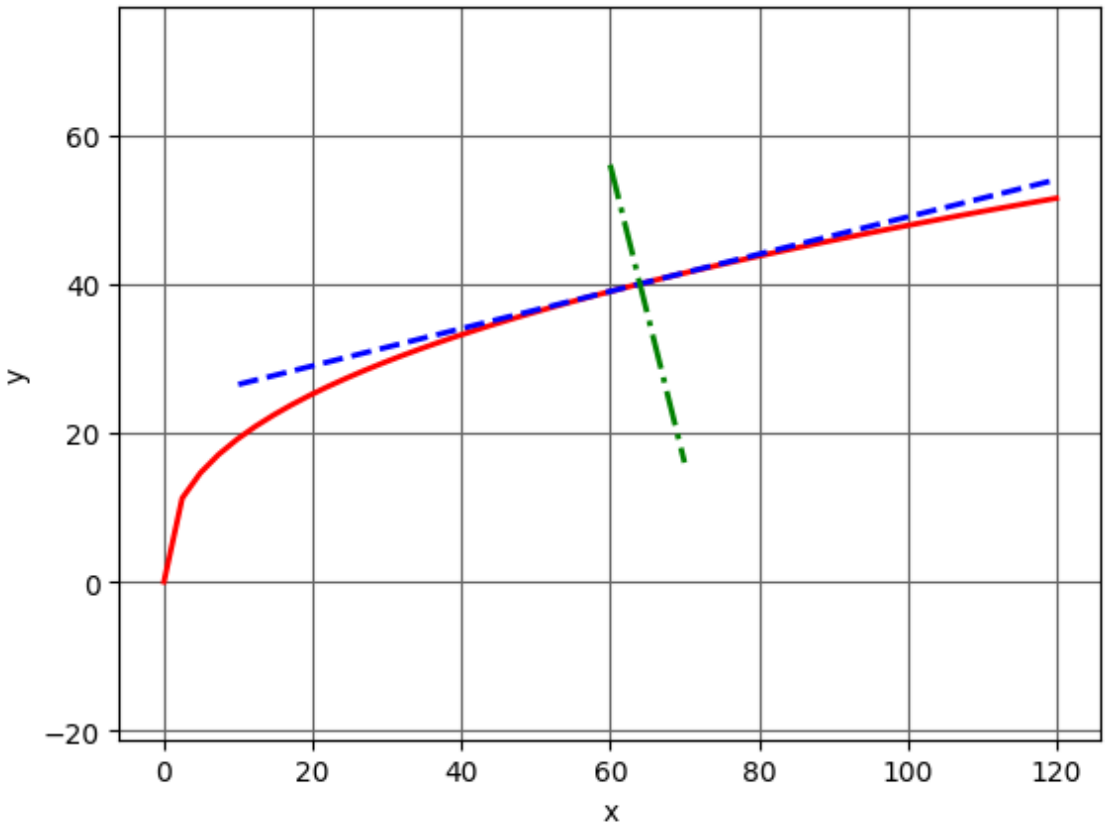
```
x = np.linspace(-3,4,50)
y1 = x**2
plt.plot(x,y1,lw=2,c='b')
x = np.linspace(-1,4,50)
y2 = 4*x - 4
plt.plot(x,y2,c='r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 9

Ввод [17]:

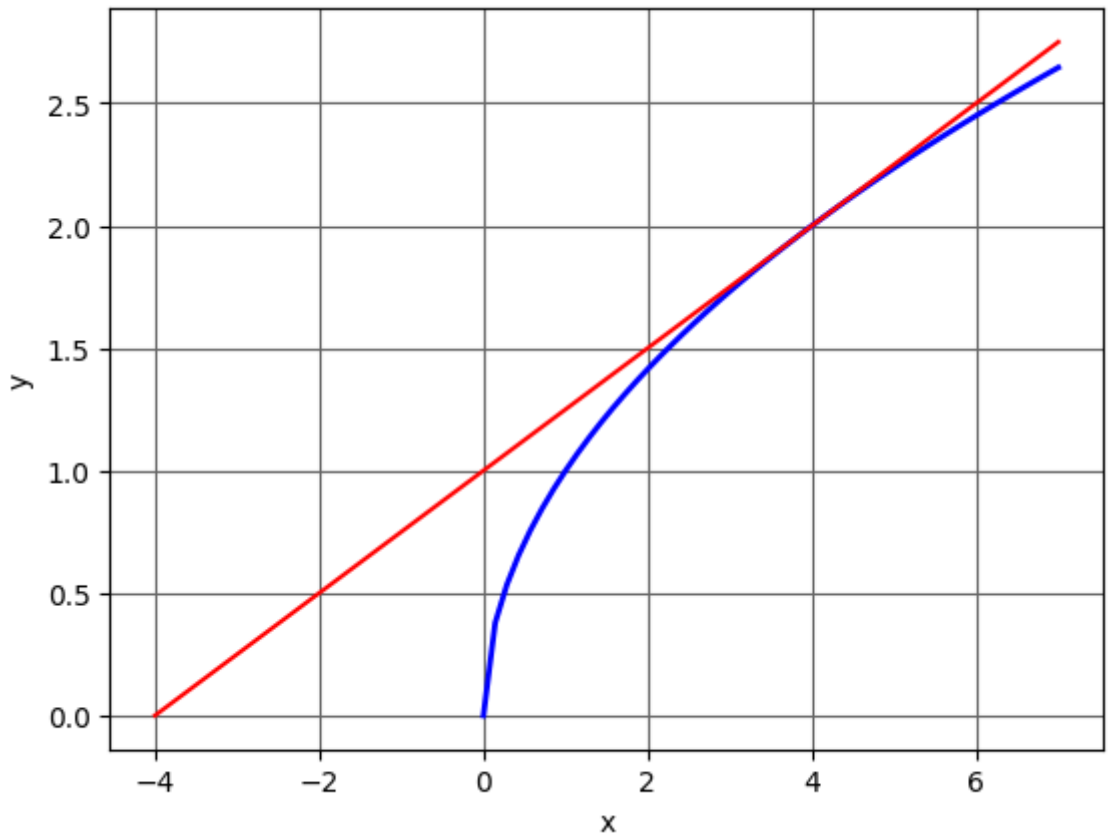
```
x = np.linspace(0,120,50)
y1 = 6*x**(1/3) + 2*x**(1/2)
plt.plot(x,y1,lw=2,c='r')
x = np.linspace(10,120,50)
y2 = x/4 + 24
plt.plot(x,y2,'--',lw=2,c='b')
x = np.linspace(60,70,50)
y3 = 296 - 4*x
plt.plot(x,y3,'-.',lw=2,c='g')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.axis('equal')
plt.show()
```



Пример 10

Ввод [18]:

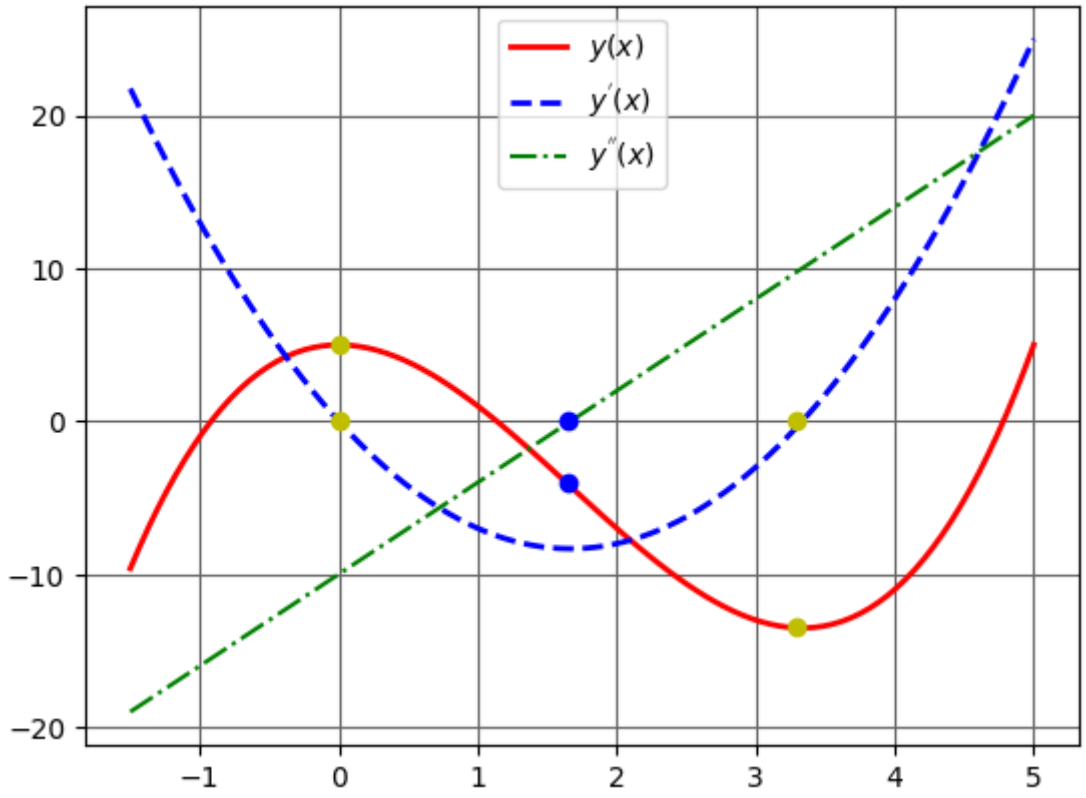
```
x = np.linspace(0,7,50)
y1 = np.sqrt(x)
plt.plot(x,y1,lw=2,c='b')
x = np.linspace(-4,7,50)
y2 = x/4 + 1
plt.plot(x,y2,c='r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid (True, linestyle='-', color='0.4')
plt.show()
```



Исследование функции

Ввод [19]:

```
t = np.linspace(-1.5, 5, 100)
f = t**3 - 5*t**2 + 5
fd = 3*t**2 - 10*t
fdd = 6*t - 10
plt.plot(t,f,lw=2,color='red',label = "$y(x)$")
plt.plot(t,fd,'--',lw=2,color='b',label = "$y^{'}(x)$")
plt.plot(t,fdd, '-.', color='g',label = "$y^{''}(x)$")
plt.plot([0], [0], 'o', color='y')
plt.plot([0], [5], 'o', color='y')
plt.plot([3.3], [0], 'o', color='y')
plt.plot([3.3], [-13.4], 'o', color='y')
plt.plot([1.65], [0], 'o', color='b')
plt.plot([1.65], [-4], 'o', color='b')
plt.grid(True, linestyle='-', color='0.4')
plt.legend()
plt.show()
```



Пример 11

Ввод [20]:

```
x, y = symbols('x y')
solve(x**2 < 3)
```

Out[20]:

$$-\sqrt{3} < x \wedge x < \sqrt{3}$$

Пример 12

Ввод [21]:

```
solve(x**2 - y**2, x)
```

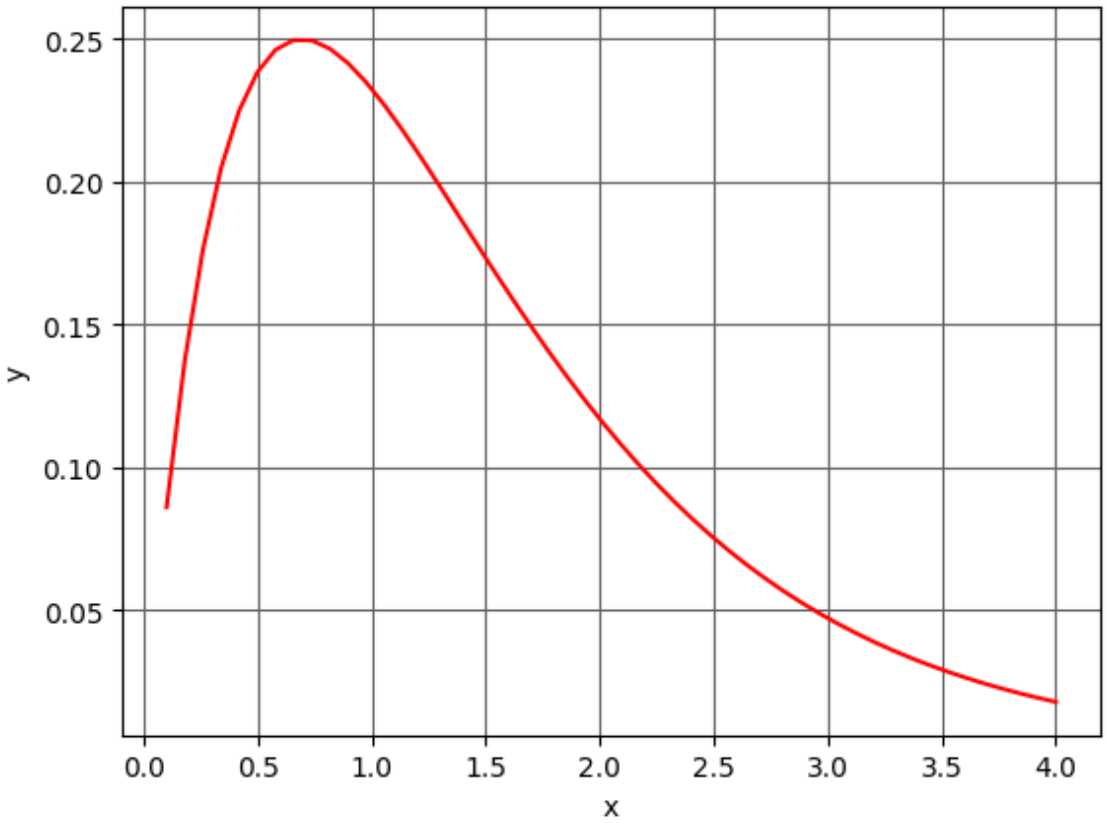
Out[21]:

```
[-y, y]
```

Пример 13

Ввод [22]:

```
f = lambda x: np.exp(-x) - np.exp(-2*x)
x = np.linspace(0.1,4,50)
plt.plot(x, f(x), 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [23]:

```
from scipy.optimize import minimize

f_max = lambda x: -(np.exp(-x) - np.exp(-2*x))
res = minimize(f_max, -2)
res
#print('x_max: %.3f f_max: %.3f' % (res.x, f(res.x)))
```

Out[23]:

```
  message: Optimization terminated successfully.
  success: True
   status: 0
      fun: -0.249999999999441
        x: [ 6.931e-01]
      nit: 12
      jac: [-7.413e-07]
 hess_inv: [[ 1.986e+00]]
     nfev: 26
     njev: 13
```

Пример 14

Ввод [24]:

```
x = symbols('x')
y = x**3
x0 = solve(diff(y,x))[0]
print('x0: %.3f y(x0): %.3f' % (x0, y.subs(x, x0)))
```

```
x0: 0.000 y(x0): 0.000
```

Ввод [25]:

```
diff(y,x,2).subs(x,x0)
```

Out[25]:

$$0$$

Ввод [26]:

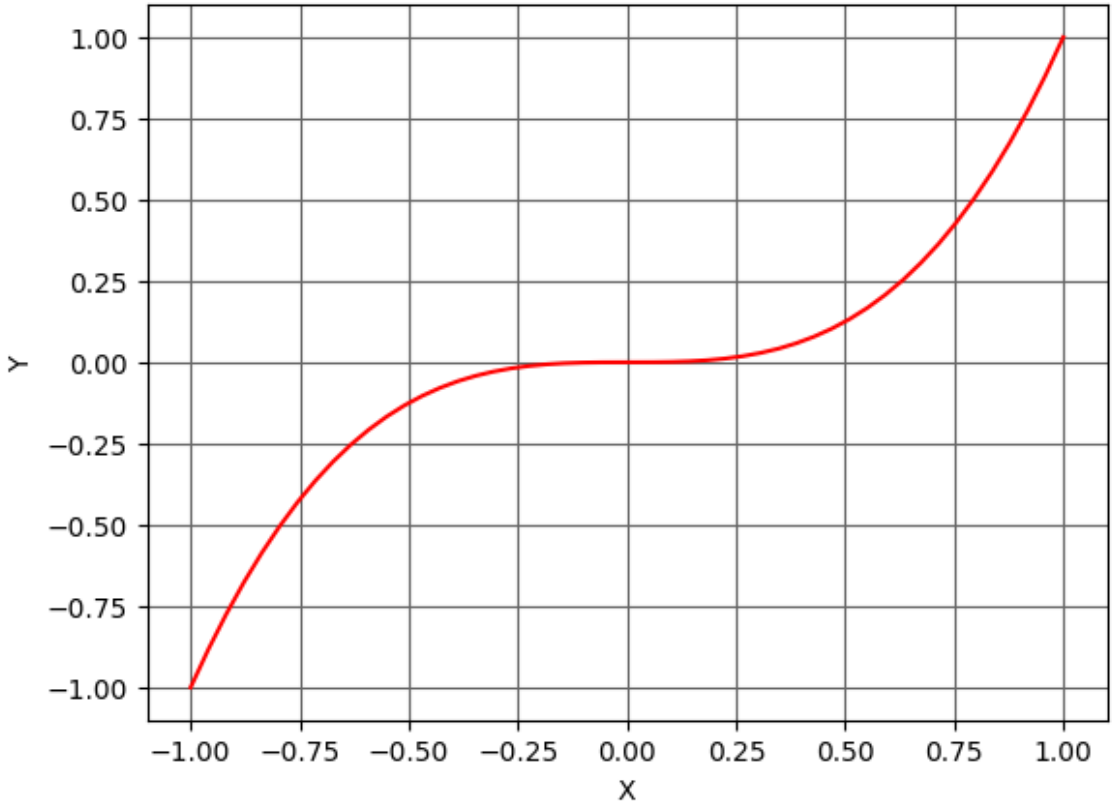
```
diff(y,x,3).subs(x,x0)
```

Out[26]:

$$\frac{246}{3125}$$

Ввод [27]:

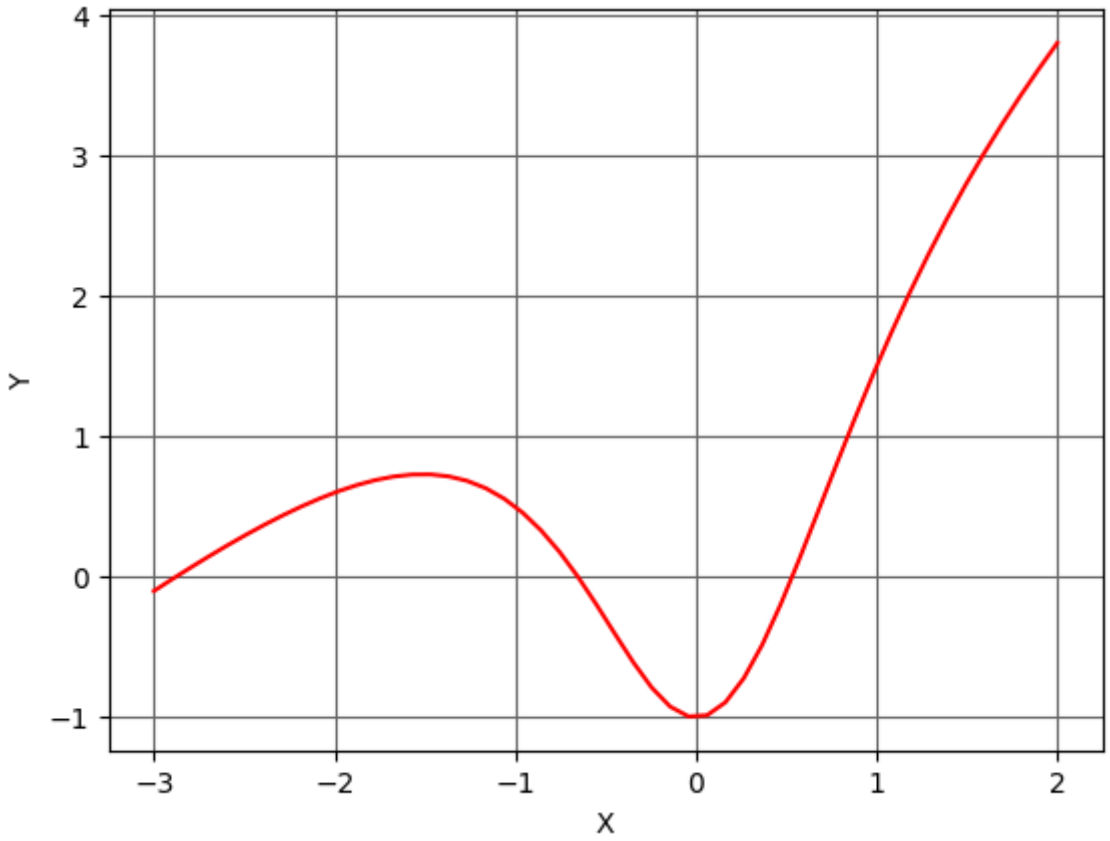
```
x = np.linspace(-1,1,50)
plt.plot(x, x**3, 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 15

Ввод [28]:

```
f = lambda x: (x**3+3*x**2-1) / (x**2+1)
x = np.linspace(-3,2,50)
plt.plot(x, f(x), 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [29]:

```
res = minimize(f, 1)
print('x_min: %.3f fmin: %.3f' % (res.x, f(res.x)))
```

```
x_min: 0.000 fmin: -1.000
```

Ввод [30]:

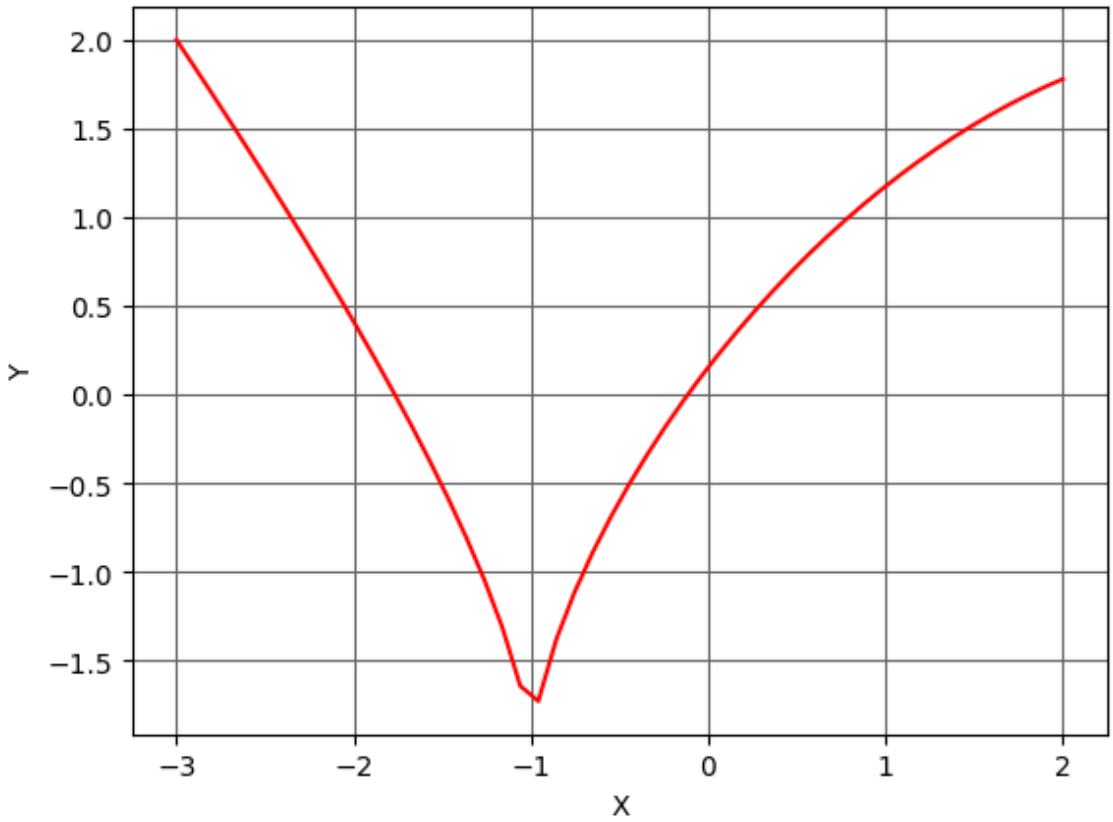
```
f_max = lambda x: -(x**3+3*x**2-1) / (x**2+1)
res = minimize(f_max, -2)
print('x_max: %.3f f max: %.3f' % (res.x, f(res.x)))
```

```
x_max: -1.513 f max: 0.731
```

Пример 16

Ввод [31]:

```
fun = lambda x: np.cbrt(2*(x+1)**2*(5-x)) - 2
x = np.linspace(-3, 3, 100)
plt.xlabel('X')
plt.ylabel('Y')
plt.plot(x, fun(x), 'r')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [32]:

```
res = minimize(fun, -1.5)
print('x_min: %.3f' % res.x)
```

```
x_min: -0.490
```

Ввод [33]:

```
res = minimize(fun, -1.001)
print('xmin: %.3f' % res.x)
```

```
xmin: -1.001
```

Ввод [34]:

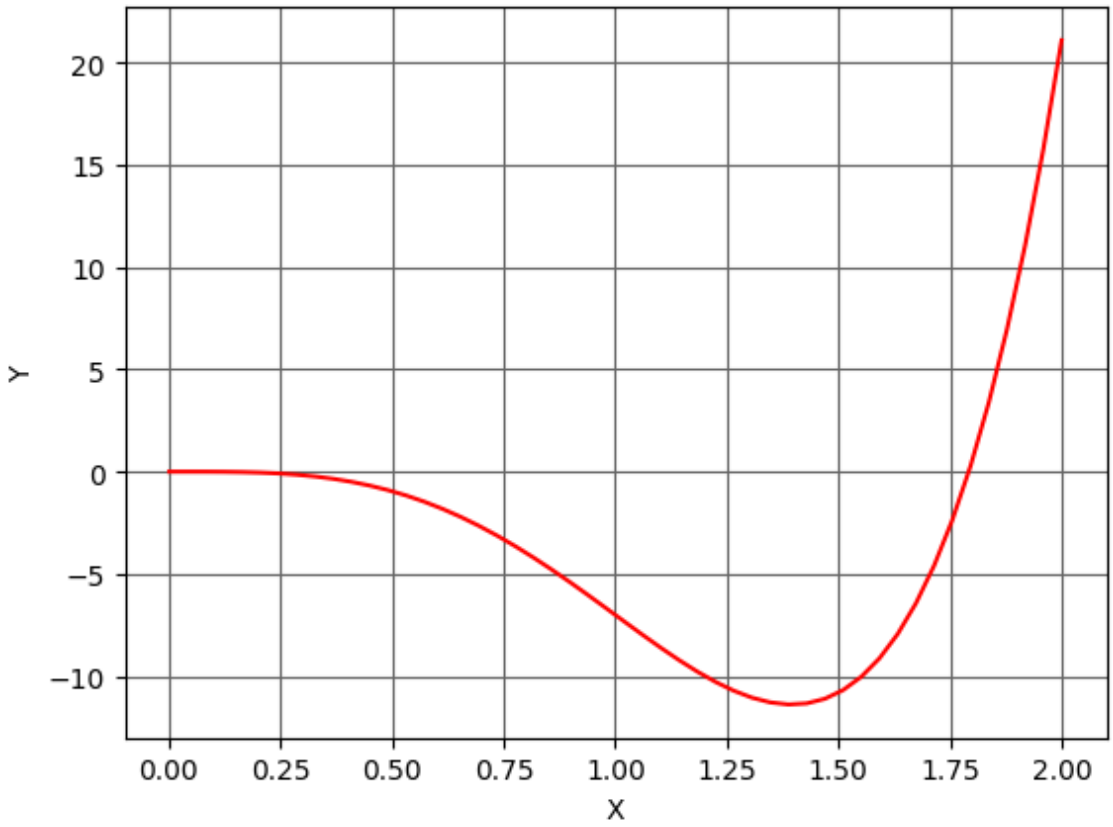
```
print('y(-3): %.3f y(3): %.3f' % (fun(-3), fun(3)))
```

```
y(-3): 2.000 y(3): 2.000
```

Пример 17

Ввод [35]:

```
f = lambda x: x**4 * (12*np.log(x) - 7)
x = np.linspace(0.001,2,50)
plt.plot(x, f(x), 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [36]:

```
x = symbols('x')
y = x**4 * (12*log(x) - 7)
y_2deriv = diff(y,x,2)
y_2deriv
```

Out[36]:

$$144x^2 \log(x)$$

Ввод [37]:

```
x_inflex = solve(y_2deriv, x)
x_inflex
```

Out[37]:

[0, 1]

Ввод [38]:

```
diff(y,x,3).subs(x, 1)
```

Out[38]:

144

Ввод [39]:

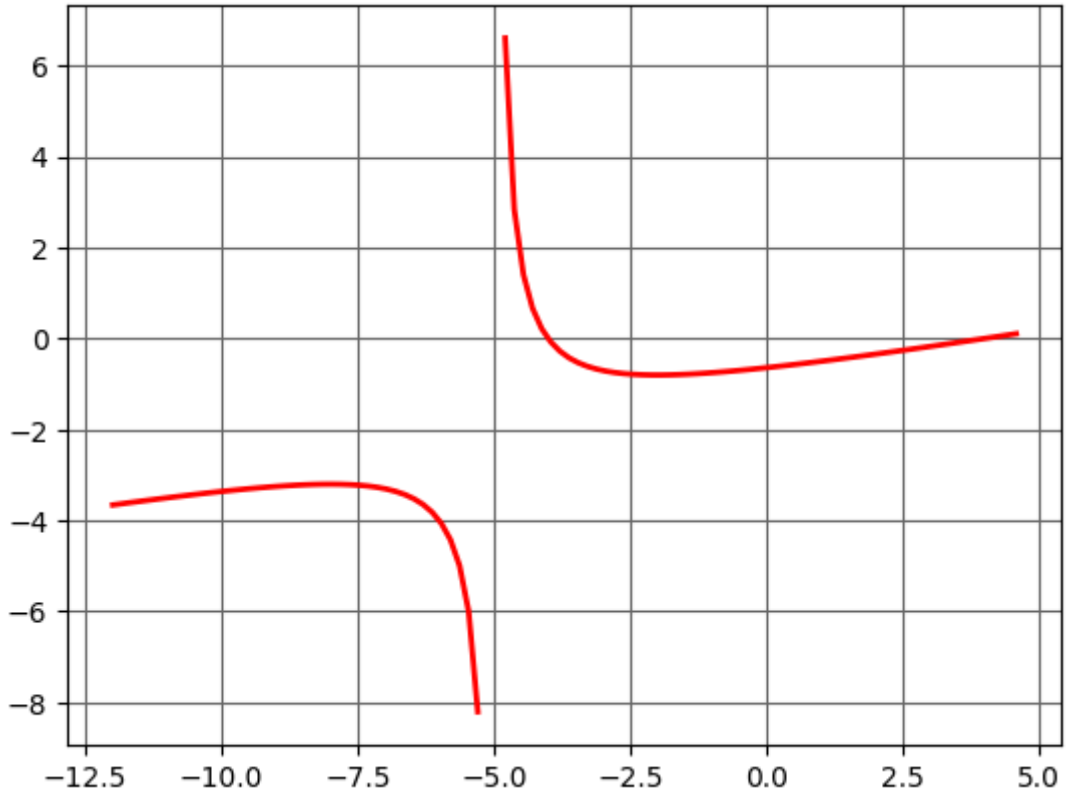
```
print('слева: %.1f справа: %.1f' % (y_2deriv.subs(x,0.9), y_2deriv.subs(x,1.1)))
```

```
слева: -12.3 справа: 16.6
```

Пример 18

Ввод [40]:

```
x = symbols('x')
y = (x**2-16)/(5*(x+5))
f = lambda x: (x**2-16)/(5*(x+5))
x = np.linspace(-12,4.6,100)
x[(x>-5.2) & (x < -4.8)] = np.nan
y = f(x)
plt.plot(x,y,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [41]:

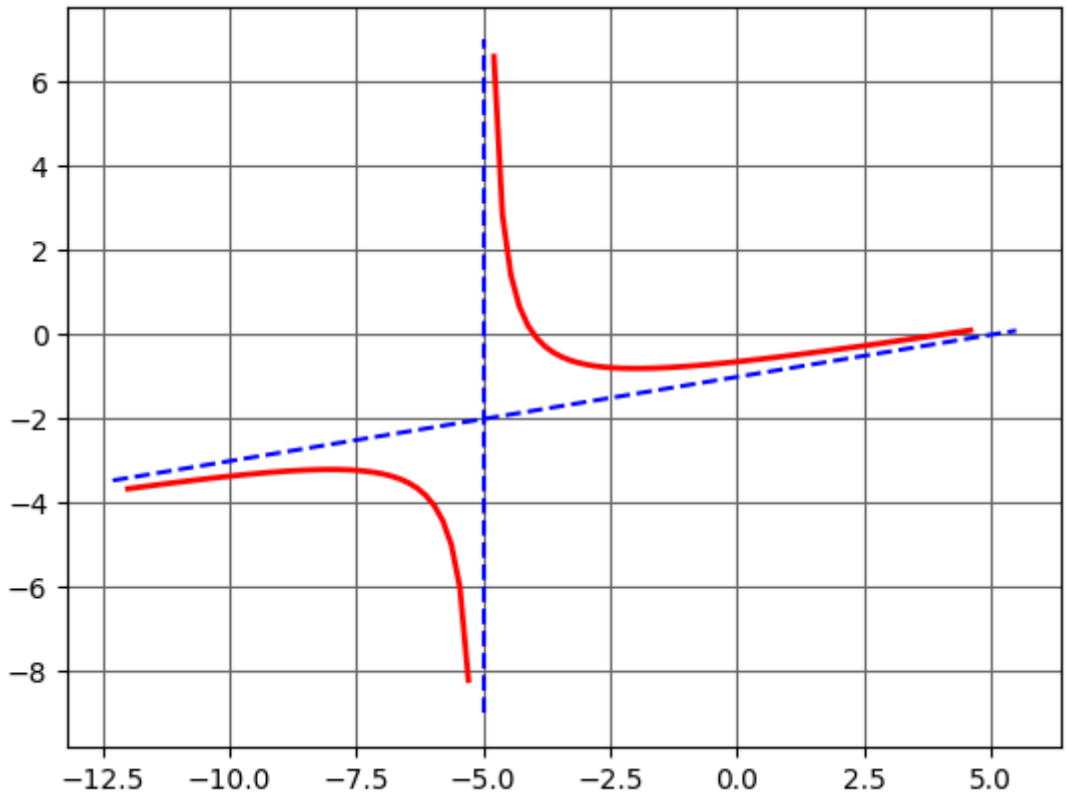
```
k = limit(y/x, x, oo)
k
```

Out[41]:

$$\left[0.304761904761905 \quad 0.306779839677906 \quad 0.308883012171189 \quad 0.311077290\right.$$

Ввод [42]:

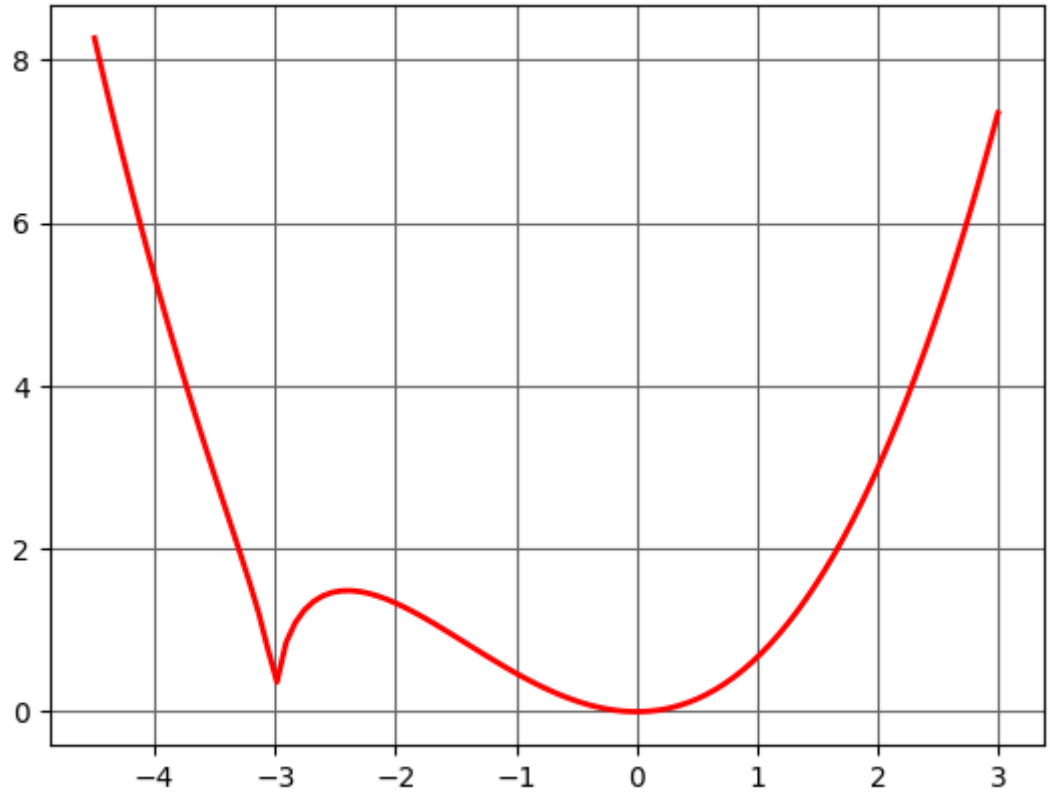
```
f = lambda x: (x**2-16)/(5*(x+5))
x = np.linspace(-12,4.6,100)
x[(x>-5.2) & (x < -4.8)] = np.nan
y = f(x)
plt.plot(x,y,lw=2,color='red')
x = np.linspace(-12.3,5.5,100)
y = x/5 - 1
plt.plot(x,y,'--',color='b')
plt.plot([-5,-5],[-9,7],'--',color='b')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 19

Ввод [43]:

```
x = symbols('x')
y = x**2*sqrt(abs(x+3))/3
f = lambda x: x**2*np.sqrt(abs(x+3))/3
x = np.linspace(-4.5,3,100)
f_x = f(x)
plt.plot(x,f_x,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [44]:

```
y.subs(x,0)
```

Out[44]:

$$\frac{x^2\sqrt{|x+3|}}{3}$$

Ввод [45]:

```
limit(y, x, oo)
```

Out[45]:

$$\frac{x^2\sqrt{|x+3|}}{3}$$

Ввод [46]:

```
k = limit(y/x, x, oo)
k
```

Out[46]:

$$\left[-0.0740740740740741x^2\sqrt{|x+3|} \quad -0.0753424657534247x^2\sqrt{|x+3|} \quad -0.076\right.$$

Ввод [47]:

```
x = symbols('x')
y1 = x**2*sqrt(-x-3)/3
y1_ = diff(y1,x). simplify()
y1_
```

Out[47]:

$$\frac{x\left(-5x-12\right)}{6\sqrt{-x-3}}$$

Ввод [48]:

```
y2 = x**2*sqrt(x+3)/3
y12_ = diff(y1,x,2).simplify()
y12_
```

Out[48]:

$$\frac{\sqrt{-x-3}\cdot\left(5x^2+24x+24\right)}{4\left(x^2+6x+9\right)}$$

Ввод [49]:

```
y22_ = diff(y2,x,2).simplify()
y22_
```

Out[49]:

$$\frac{5x^2+24x+24}{4(x+3)^{\frac{3}{2}}}$$

Ввод [50]:

```
xp1 = solve(y12_)
xp1
```

Out[50]:

```
[-12/5 - 2*sqrt(6)/5, -12/5 + 2*sqrt(6)/5]
```

Ввод [51]:

```
diff(y1, x, 3).subs(x,xp1[0]).evalf(5)
```

Out[51]:

$$-10.465$$

Ввод [52]:

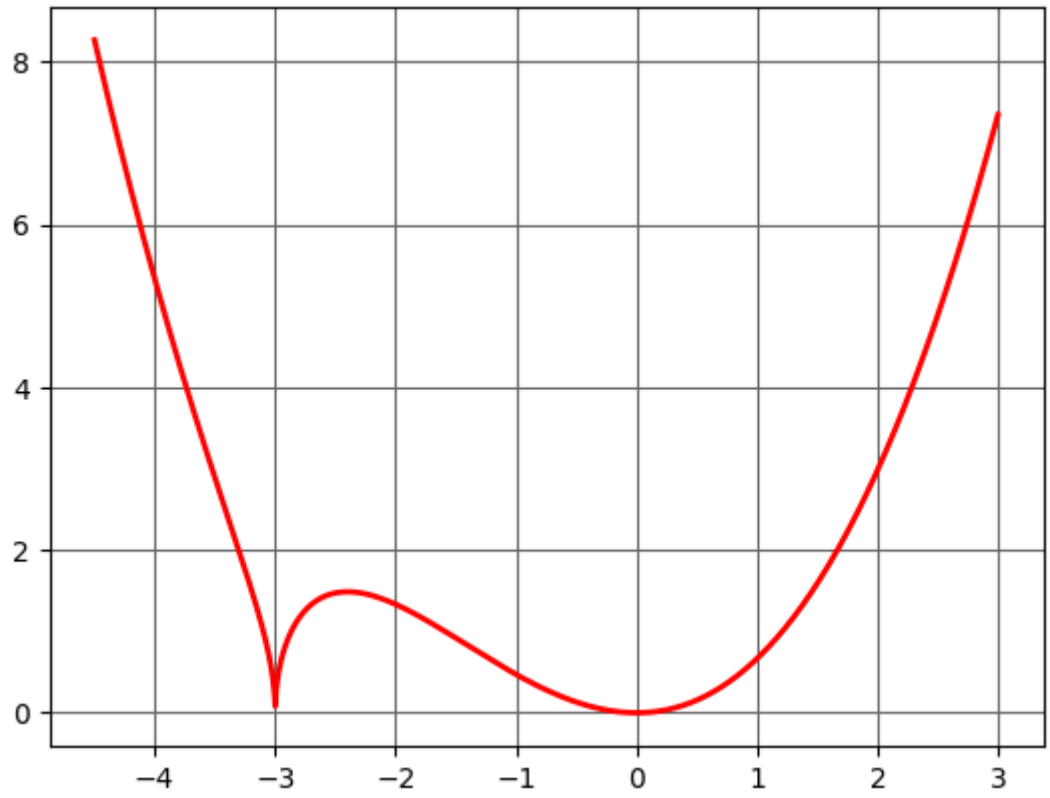
```
diff(y1, x, 3).subs(x,xp1[1]).evalf(4)
```

Out[52]:

$$1.234i$$

Ввод [53]:

```
f = lambda x: x**2*np.sqrt(abs(x+3))/3
x = np.linspace(-4.5,3,2000)
f_x = f(x)
plt.plot(x,f_x,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 20

Ввод [54]:

```
x, y = symbols("x y")
z = x*y**2 + exp(-x)
```

Ввод [55]:

```
diff(z, x, 2)
```

Out[55]:

$$e^{-x}$$

Ввод [56]:

```
diff(z, y, 2)
```

Out[56]:

$$2x$$

Пример 21

Ввод [57]:

```
x, y = symbols('x y')
z = sin(x)*cos(y)
diff(z, x, 2, y)
```

Out[57]:

$$\sin(y)\sin(x)$$

Пример 22

Ввод [58]:

```
x,y = symbols('x y')
z = 5*log(x**2 + y**2)
z_x = diff(z,x).subs({x:1, y:2})
z_y = diff(z,y).subs({x:1, y:2})
grad_f = (z_x, z_y)
grad_f
```

Out[58]:

```
(2, 4)
```

Пример 23

Ввод [59]:

```
x,y = symbols('x y')
z = x**2 + x*y +7
z_x = diff(z,x).subs({x:1, y:-1})
z_y = diff(z,y).subs({x:1, y:-1})
grad_f = (z_x, z_y)
grad_f
```

Out[59]:

```
(0, 0)
```

Пример 24

Ввод [60]:

```
l = Point(3,4)
l_n = l.distance(Point(0,0))
cos_a = l.x/l_n
cos_b = l.y/l_n
x,y = symbols('x y')
z = x**2 + y**2
z_x = diff(z,x).subs({x:1, y:1})
z_y = diff(z,y).subs({x:1, y:1})
z_l = z_x*cos_a + z_y*cos_b
z_l
```

Out[60]:

$$\frac{14}{5}$$

Пример 25

Ввод [61]:

```
def tangent_plane(F,M):
    F_diff_x = diff(F,x).subs({x:M.x,y:M.y,z:M.z})
    F_diff_y = diff(F,y).subs({x:M.x,y:M.y,z:M.z})
    F_diff_z = diff(F,z).subs({x:M.x,y:M.y,z:M.z})

    n = Point(F_diff_x, F_diff_y, F_diff_z)

    p = Plane(M, normal_vector=n).equation()

    K = Point(M.x+n.x, M.y+n.y, M.z+n.z)
    l_n = Line(M, K).arbitrary_point()
    return p, In
```

Ввод [62]:

```
x, y, z = symbols('x y z')
F = x**2 + y**2 + z**2 - 9
M = Point(1,1,1)
p, l_n = tangent_plane(F,M)
```

Ввод [63]:

```
p
```

Out[63]:

$$2x + 2y + 2z - 6$$

Пример 26

Ввод [64]:

```
z = lambda w: (w[0]-1)**2 + (w[1]-3)**4
res = minimize(z, (0, 0))
res.x
```

Out[64]:

```
array([0.99999999, 2.98725136])
```

Ввод [65]:

```
z((1,3)) < z((0.999,3.001))
```

Out[65]:

```
True
```

Ввод [66]:

```
z((1,3))
```

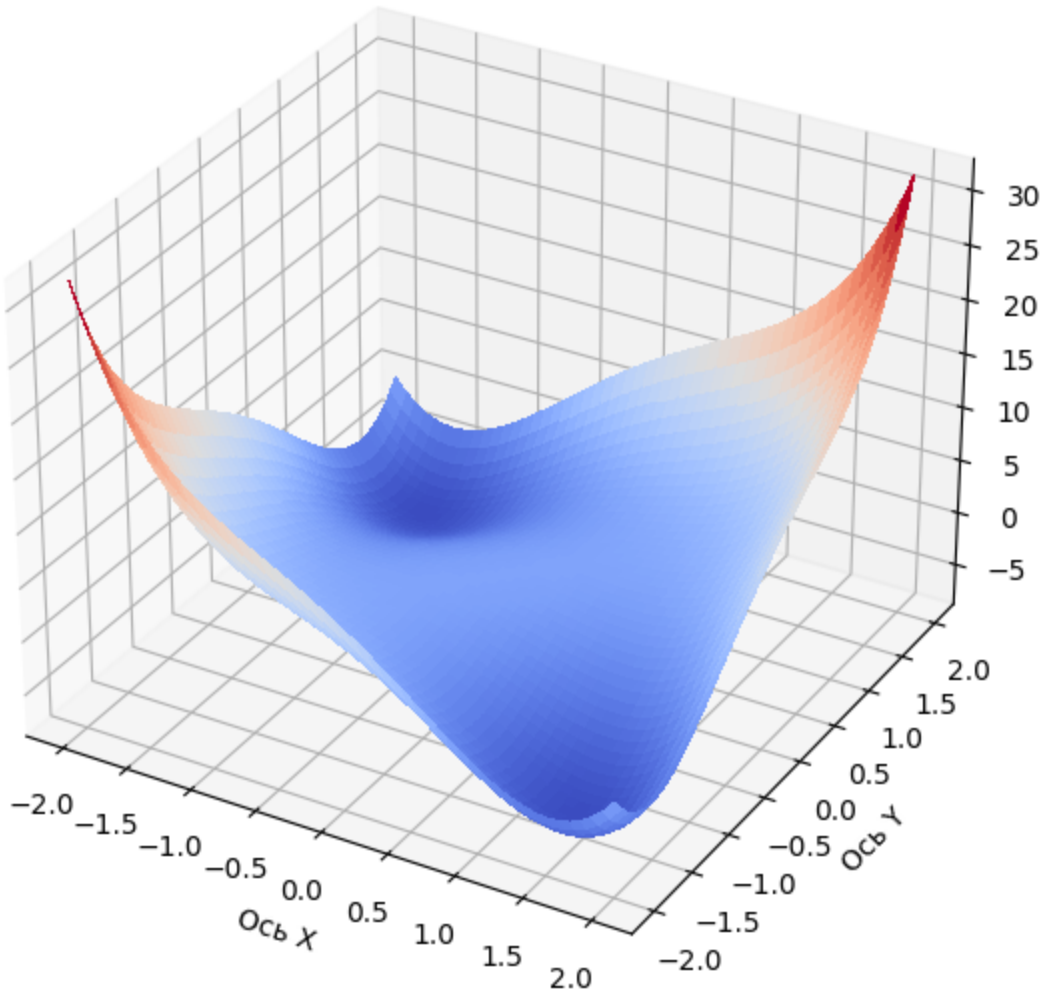
Out[66]:

```
0
```

Пример 27

Ввод [67]:

```
z = lambda w: w[0]**4 + w[1]**4 - 2*w[0]**2 +4*w[0]*w[1] - 2*w[1]**2

fig = plt.figure(figsize=(7,7))
axes = fig.add_subplot(projection='3d')
y = x = np.linspace(-2, 2, 50)
x, y = np.meshgrid(x, y)
Z = z((x,y))
surf = axes.plot_surface(x, y, Z, cmap='coolwarm',linewidth=0, antialiased=False)

axes.set_xlabel('Ось X')
axes.set_ylabel('Ось Y')
axes.set_zlabel('Ось Z')
plt.show()
```



Ввод [68]:

```
res = minimize(z, (1, -1))
res.x
```

Out[68]:

```
array([ 1.41421356, -1.41421356])
```

Ввод [69]:

```
z(res.x)
```

Out[69]:

```
-8.0
```

Ввод [70]:

```
res = minimize(z, (-1, 1))
res.x
```

Out[70]:

```
array([-1.41421357,  1.41421357])
```

Ввод [71]:

```
z(res.x)
```

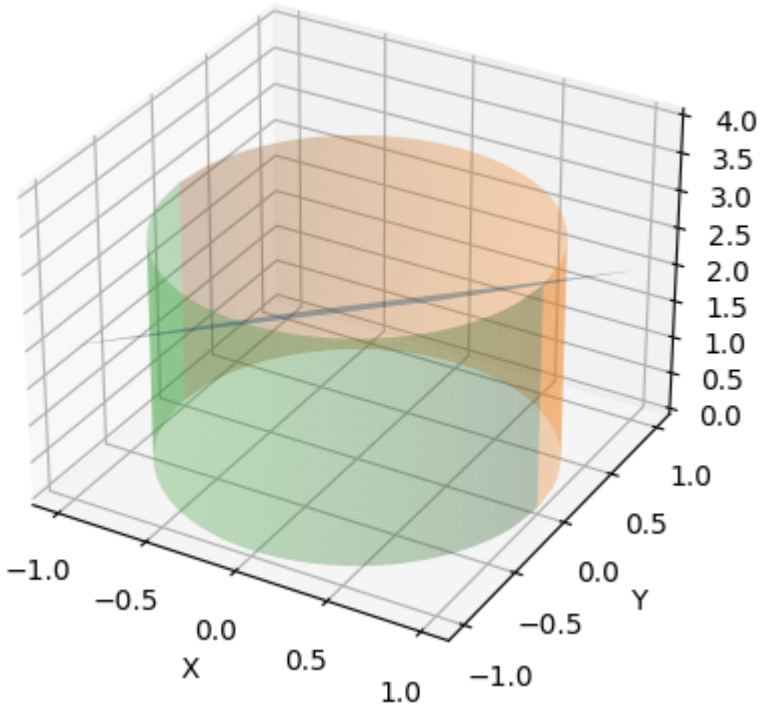
Out[71]:

```
-7.999999999999997
```

Пример 28

Ввод [72]:

```
f = lambda w: w[0] - w[1] + 2
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(-1, 1, 50)
y = np.linspace(-1, 1, 50)

x, y = np.meshgrid(x, y)
z1 = f((x,y))
ax.plot_surface(x, y, z1, alpha=0.4)

x = np.linspace(-1, 1, 100)
z = np.linspace(0, 3, 100)
xc, zc = np.meshgrid(x, z)
yc = np.sqrt(1-xc**2)
ax.plot_surface(xc, yc, zc, alpha=0.3)
ax.plot_surface(xc, -yc, zc, alpha=0.3)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```



Ввод [73]:

```
cons = ({'type': 'eq', 'fun': lambda w: w[0]**2 + w[1]**2 - 1})

bnds = ((None, None), (None, None))

res = minimize(f, (-0.5, 0.5), bounds=bnds, constraints=cons)

res.x
```

Out[73]:

```
array([-0.70710679,  0.70710677])
```

Ввод [74]:

```
f_max = lambda w: -(1.5*w[0] - w[1] + 1)
cons = ({'type': 'eq', 'fun': lambda w: w[0]**2 + w[1]**2 - 1})
bnds = ((None, None), (None, None))
res = minimize(f_max, (0.5, -0.5), bounds=bnds,constraints=cons)

res.x
```

Out[74]:

```
array([ 0.83205051, -0.55469991])
```

Пример 29

Ввод [75]:

```
x,y = symbols('x y')
z = 4.5*x**(0.33) * y**(0.66)
z_x = diff(z, x)
z_y = diff(z, y)

E_x = (x/z)*z_x
E_y = (y/z)*z_y
print('E_x: %.2f E_y: %.2f' % (E_x, E_y))
```

```
E_x: 0.33 E_y: 0.66
```

Пример 30

Ввод [76]:

```
K,V0 = symbols('K V0')
V = V0*log(5+K**2)

Vprim2 = diff(V,K,2)

Vprim3 = diff(V,K,3)

s = solve(Vprim2,K)
s
```

Out[76]:

```
[-sqrt(5), sqrt(5)]
```

Ввод [77]:

```
Vprim3.subs(K,s[1])
```

Out[77]:

$$-\frac{\sqrt{5}V_0}{25}$$

Примеры решения задач

Вычислить y' для функции $xcos(lnx) + sin(lnx))$

Ввод [78]:

```
x = symbols('x')
y = x*(cos(log(x))+sin(log(x)))
diff(y,x)
```

Out[78]:

$$\mathrm{x}\left(-\frac{\sin\left(\log\left(\mathrm{x}\right)\right)}{\mathrm{x}}+\frac{\cos\left(\log\left(\mathrm{x}\right)\right)}{\mathrm{x}}\right)+\sin\left(\log\left(\mathrm{x}\right)\right)+\cos\left(\log\left(\mathrm{x}\right)\right)$$

Ввод [79]:

```
diff(y,x).simplify()
```

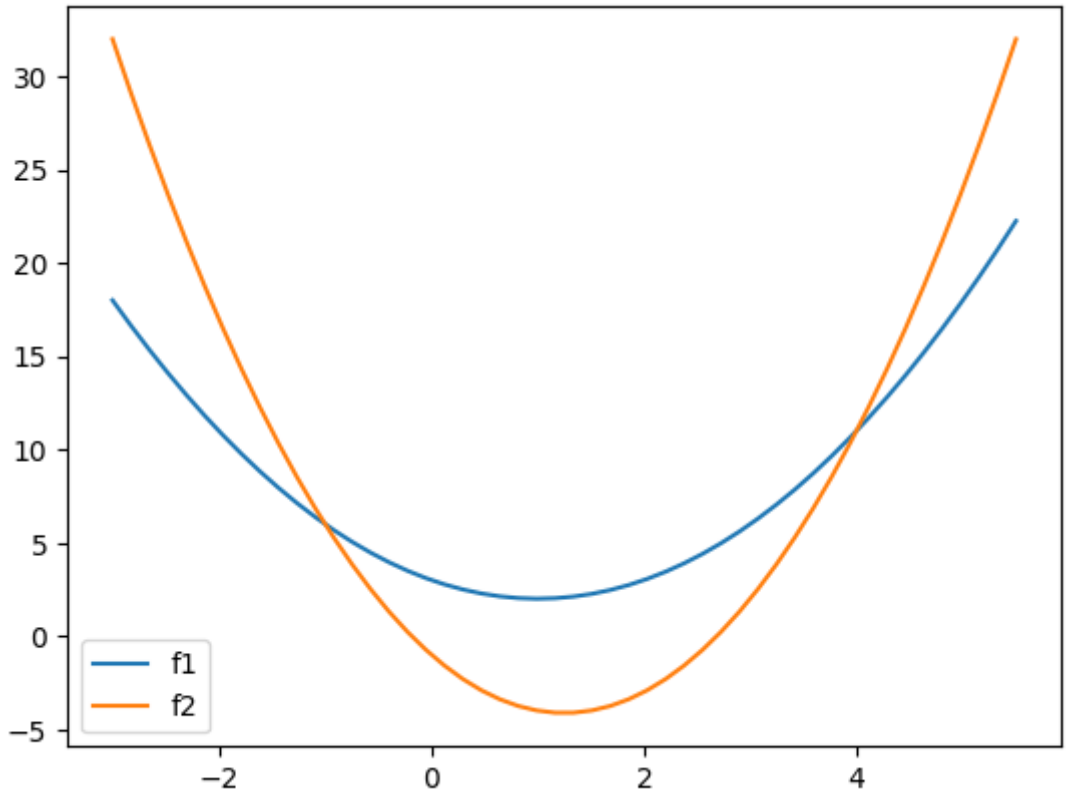
Out[79]:

$$2\cos\left(\log\left(\mathrm{x}\right)\right)$$

Решить уравнение $y'(x) = 0$, где $y(x) = max\{x^2 - 2x + 3; 2x^2 - 5x - 1\}$

Ввод [80]:

```
f1 = lambda x: x**2-2*x+3
f2 = lambda x: 2*x**2-5*x-1
x = np.linspace(-3, 5.5, 50)
y1 = f1(x)
plt.plot(x,y1, label = "f1")
y2 = f2(x)
plt.plot(x,y2, label = "f2")
plt.legend()
plt.show()
```



Ввод [81]:

```
f1(-1) == f2(-1)
```

Out[81]:

```
True
```

Ввод [82]:

```
f1(4) == f2(4)
```

Out[82]:

```
True
```

Ввод [83]:

```
x = symbols('x')
f1 = x**2-2*x+3
f2 = 2*x**2-5*x-1

y_diff1 = diff(f2,x)
y_diff1
```

Out[83]:

$$4x - 5$$

Ввод [84]:

```
y_diff2 = diff(f1,x)
y_diff2
```

Out[84]:

$$2x - 2$$

Показать, что функция $y = xsinx$ удовлетворяет уравнению $\frac{y'}{cosx} - x = tgx$

Ввод [85]:

```
x, y = symbols('x y')
y = x*sin(x)
yprim = diff(y, x)
f = yprim/cos(x) - x
f.simplify()
```

Out[85]:

$$\tan(x)$$

Написать уравнения касательных к графику функции $y = (x^2 + 1)(x - 2)$ в точках её пересечения с осями координат.

Ввод [86]:

```
x = symbols('x', real=True)
y = (x**2 + 1)*(x - 2)
```

Ввод [87]:

```
def tangent(y, x0):
 y0 = y.subs(x, x0)
 x1 = x0 + 1
 k = diff(y,x).subs(x,x0)
 y1 = y0 + k
 return Line((x0,y0), (x1,y1))
```

Ввод [88]:

```
tangent(y, 0).equation()
```

Out[88]:

$$-x + y + 2$$

Ввод [89]:

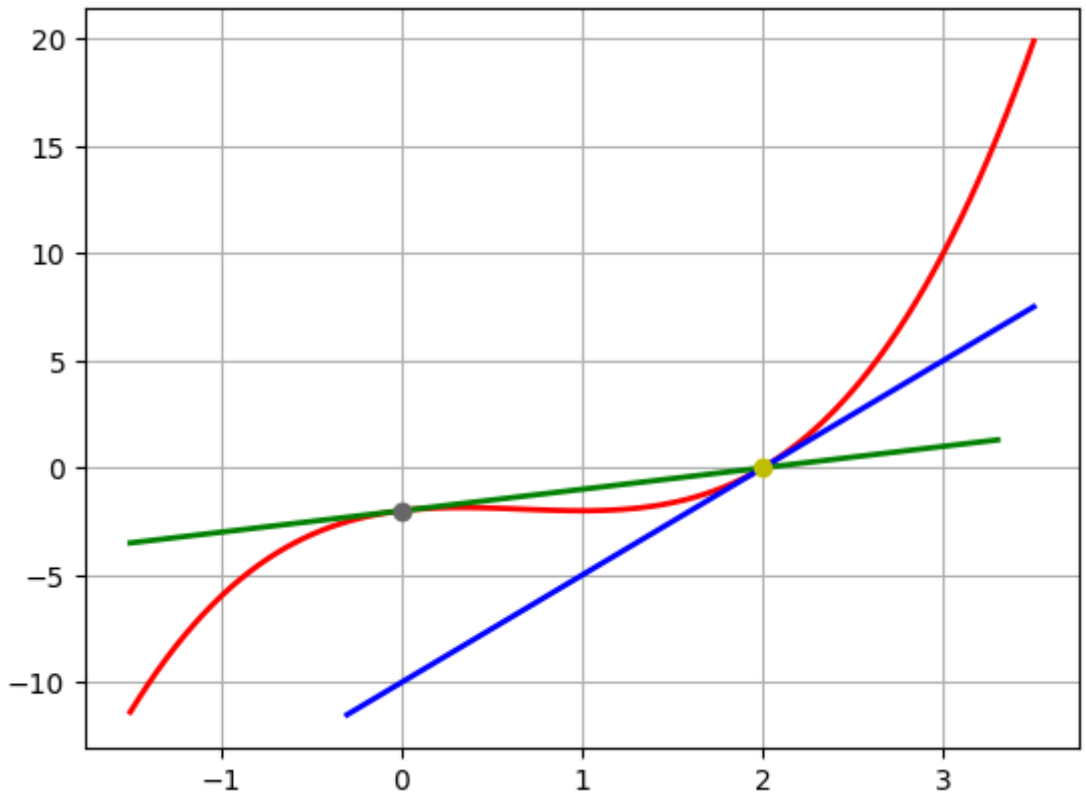
```
xp = solve(y, x)
tangent(y, xp[0]).equation()
```

Out[89]:

$$-5x + y + 10$$

Ввод [90]:

```
x = np.linspace(-1.5,3.5,100)
y = (x**2 + 1)*(x - 2)
plt.plot(x, y, lw=2, color='red')
x = np.linspace(-1.5,3.3,100)
y1 = x - 2
plt.plot(x, y1, lw=2, color='green')

x = np.linspace(-0.3,3.5,100)
y2 = 5*x - 10
plt.plot(x, y2, lw=2, color='blue')
plt.plot([0], [-2], 'o', color='0.4')
plt.plot([2], [0], 'o', color='y')
plt.grid(True)
plt.show()
```



При каком значении параметра a парабола $y = ax^2$ касается кривой $y = lnx$?

Ввод [91]:

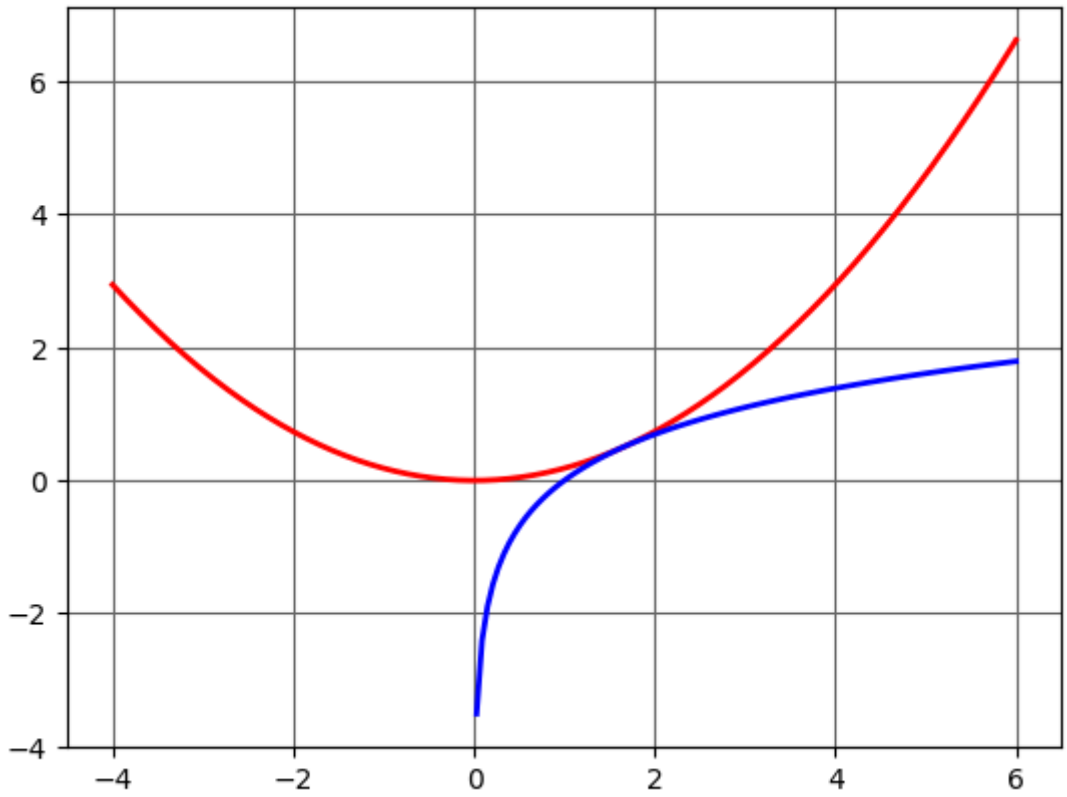
```
x, a, x0 = symbols('x a x0')

y1 = a*x**2
y2 = log(x)

y1_diff = diff(y1,x).subs(x,x0)
y2_diff = diff(y2,x).subs(x,x0)

y1_0 = y1.subs(x,x0)
y2_0 = y2.subs(x,x0)

solve([y1_0-y2_0, y1_diff-y2_diff], [x0, a])
```

Out[91]:

```
[(exp(1/2), exp(-1)/2)]
```

Ввод [92]:

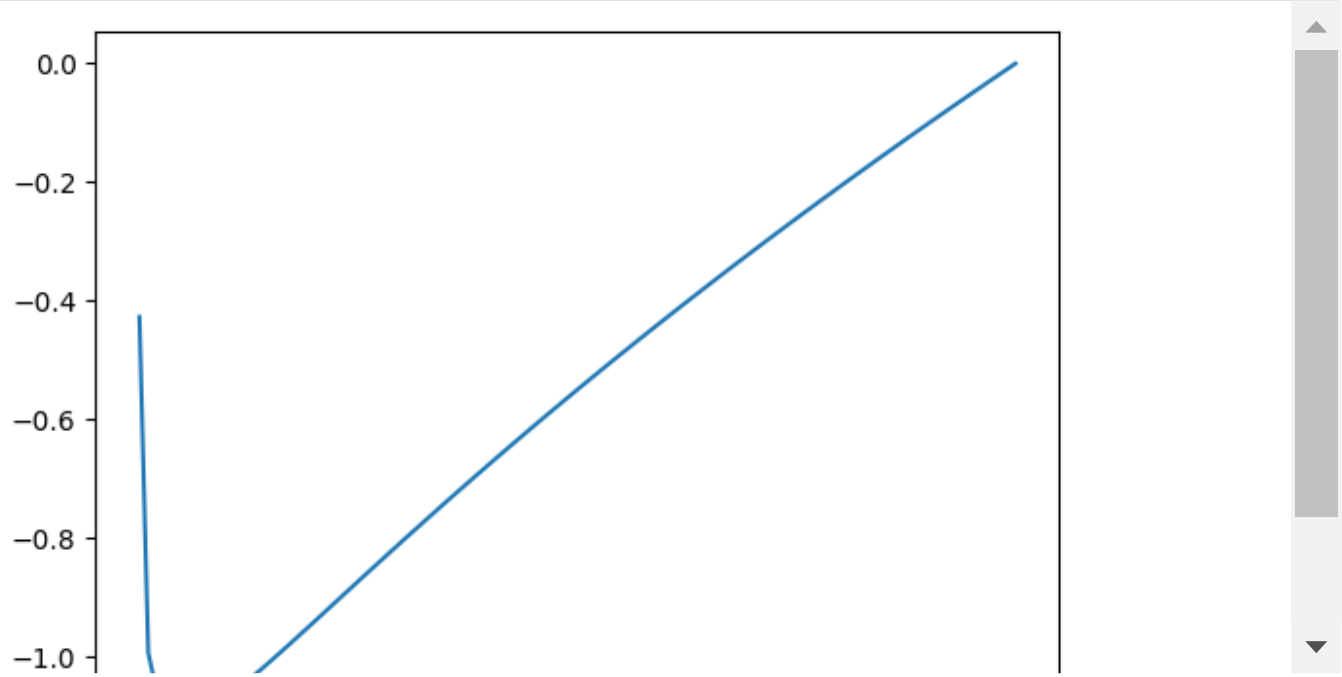
```
x = np.linspace(-4, 6, 500)
y = x**2/(2*np.exp(1))
plt.plot(x, y, lw=2, c='r')
x = np.linspace(0.03, 6, 100)
y = np.log(x)
plt.plot(x, y, lw=2, c='b')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Исследовать на экстремум функцию $y = \sqrt[3]{x} \times lnx$.

Ввод [93]:

```
f = lambda x: (x**(1/3))*np.log(x)

x = np.linspace(0.0001,1,100)
y = f(x)
plt.plot(x, y)
plt.show()
```



Ввод [94]:

```
res = minimize(f, 0.01)
print('xmin: %.4f y(x_min): %.3f' % (res.x, f(res.x)))
```

```
xmin: 0.0498 y(x_min): -1.104
```

Ввод [95]:

```
x = symbols('x')
y = x**(1/3) * log(x)
x_min = solve(diff(y,x))[0]
print('x_min: %.4f y(x_min) %.3f' % (x_min, y.subs(x, x_min)))
```

```
x_min: 0.0498 y(x_min) -1.104
```

Найти производную функции $w = \frac{x^2}{2} + \frac{y^2}{9} - z^2$ в точке $A(1;2)$ по направлению радиус-вектора этой точки.

Ввод [96]:

```
l = Point(2,3,1)

l_n = l.distance(Point(0,0,0))

cos_a = l.x/l_n
cos_b = l.y/l_n
cos_c = l.z/l_n
```

Ввод [97]:

```
x,y,z = symbols('x y z')
w = x**2/2 + y**2/9 - z**2

w_x = diff(w,x).subs({x:2, y:3, z:1})
w_y = diff(w,y).subs({x:2, y:3, z:1})
w_z = diff(w,z).subs({x:2, y:3, z:1})
```

Ввод [98]:

```
w_x = diff(w,x).subs([(x,2),(y,3),(z,1)])
w_y = diff(w,y).subs([(x,2),(y,3),(z,1)])
w_z = diff(w,z).subs([(x,2),(y,3),(z,1)])

w_l = w_x*cos_a + w_y*cos_b + w_z*cos_c
w_l
```

Out[98]:

$$\frac{2\sqrt{14}}{7}$$

Найти экстремумы функции $z = x^2 - 4xy - 2y^2 + 8x$

Ввод [99]:

```
def critical_points(z):

    z_x = diff(z,x)
    z_y = diff(z,y)

    cr_point = solve([z_x, z_y], [x, y], dict=True)

    A = diff(z,x,2)
    B = diff(z,x,y)
    C = diff(z,y,2)

    D = A*C - B**2
    return cr_point, A, D
```

Ввод [100]:

```
def suff_indic(A, D, cr_point):
    A0 = A.subs(cr_point)
    D0 = D.subs(cr_point)
    return D0, A0
```

Ввод [101]:

```
x, y = symbols('x y')
z = x**2 - 4*x*y - 2*y**2 + 8*x
cr_point, A, D = critical_points(z)
cr_point
```

Out[101]:

```
[{x: -4/3, y: 4/3}]
```

Ввод [102]:

```
D0, A0 = suff_indic(A, D, cr_point[0])
D0, A0
```

Out[102]:

```
(-24, 2)
```

Зависимость между себестоимостью продукции C и объёмом её производства Q выражается формулой $C(Q) = 80 - 0,38Q$. Определить эластичность себестоимости при выпуске продукции $Q = 20$ ден. ед.

Ввод [103]:

```
Q = symbols('Q')
c = 80 - 0.38*Q
Dprim = diff(c,Q)
E = (Q*Dprim/c).subs(Q,20)
S(E).n(3)
```

Out[103]:

$$-0.105$$

Функция спроса D и предложения S от цены p имеют вид: $D(p) = 40 - 1,3p$, $S(p) = 20 + 1,2p$. Найти эластичность спроса в точке равновесной цены.

Ввод [104]:

```
p = symbols('p')
D = 40 - 1.3*p
S = 20 + 1.2*p
p0 = solve(D-S,p)
p0[0].n(2)
```

Out[104]:

8.0

Ввод [105]:

```
Dprim = diff(D,p)
E = (p*Dprim/D).subs(p,p0[0])
E.n(3)
```

Out[105]:

-0.351

Индивидуальное задание

Производитель выпускает цилиндрический контейнер радиусом 5 см и высотой 15 см. Стоимость материала для верхней и нижней частей контейнера составляет $0,10/\text{см}^2$, а стоимость материала для боковых сторон - $0,05/\text{см}^2$. Найдите скорость изменения стоимости материала, если радиус увеличивается со скоростью $0,5\text{см/с}$.

Ввод [106]:

```
from sympy import *
import matplotlib.pyplot as plt
import numpy as np
```

Сначала найдем уравнение для стоимости материала для цилиндрического контейнера. Общая площадь поверхности цилиндра составляет: $S = 2\pi r^2 + 2\pi rh$ где r - радиус, а h - высота цилиндра. Стоимость материала для верхней и нижней частей контейнера составляет $0,10/\text{см}^2$, поэтомустоимостьэтихповерхностейсоставляет :C_top/bottom = 0,10(2πr^2) . Стоимостьматериаладлябоковыхсторонконтейнерасоставляет0,05/см^2, поэтому стоимость для этих поверхностей составляет:

$$C_{\text{с}}\text{тороны} = 0,05(2\pi rh).$$

Общая стоимость материала равна сумме стоимости материала для верха/дно и боковых поверхностей:

$$C = C_{\text{в}}\text{ерх/низ} + C_{\text{с}}\text{тороны} = 0,10(2\pi r^2) + 0,05(2\pi rh) = 0,2\pi r^2 + 0,1\pi rh.$$

Ввод [107]:

```
r, h = symbols('r h')
C = 0.2*pi*r**2 + 0.1*pi*r*h
C
```

Out[107]:

$$0.1\pi hr + 0.2\pi r^2$$

Чтобы найти скорость изменения стоимости материала в зависимости от радиуса, нужно продифференцировать C в зависимости от r:

$$dC/dr = 0,4\pi r + 0,1\pi h.$$

Мы знаем, что r увеличивается со скоростью 0,5 см/с, поэтому нам нужно найти соответствующую скорость изменения стоимости материала. Сначала подставим заданные значения r и h:

$r = 5\text{см}\ h = 15\text{см}$

dC/dr оценивается при $(r, h) = (5, 15)$: $dC/dr = 0,4\pi(5) + 0,1\pi(15) \approx 3,93\pi.$

Таким образом, скорость изменения стоимости материала составляет примерно $3,93\pi(\$/\text{с})$, когда радиус увеличивается со скоростью $0,5\text{см/с}$.

Ввод [108]:

```
# Генерируем данные для построения графика
r_vals = np.linspace(0, 10, 100)
C_vals = np.zeros(len(r_vals))
dCdr_vals = np.zeros(len(r_vals))
for i in range(len(r_vals)):
    C_vals[i] = C.subs([(r, r_vals[i]), (h, 15)])
    dCdr_vals[i] = diff(C, r).subs([(r, r_vals[i]), (h, 15)])
C_vals
dCdr_vals
```

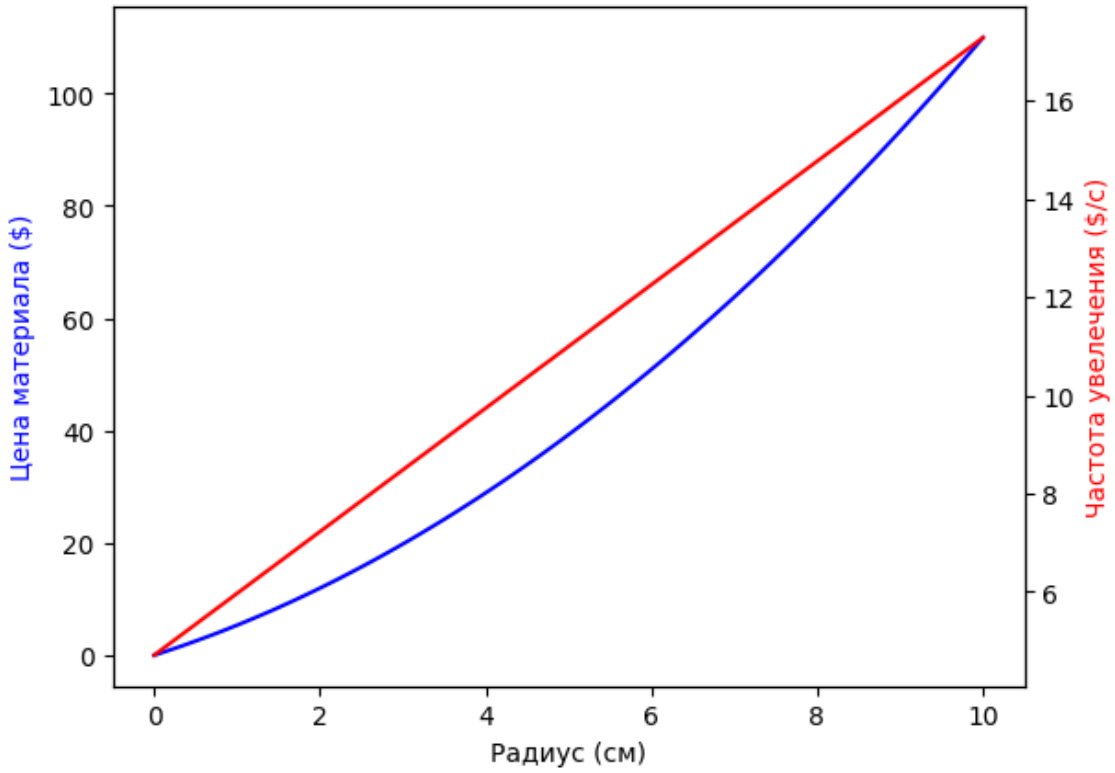
Out[108]:

```
array([ 4.71238898,  4.83932202,  4.96625505,  5.09318809,  5.22012113,
        5.34705416,  5.4739872 ,  5.60092024,  5.72785327,  5.85478631,
        5.98171935,  6.10865238,  6.23558542,  6.36251845,  6.48945149,
        6.61638453,  6.74331756,  6.8702506 ,  6.99718364,  7.12411667,
        7.25104971,  7.37798275,  7.50491578,  7.63184882,  7.75878186,
        7.88571489,  8.01264793,  8.13958097,  8.266514  ,  8.39344704,
        8.52038008,  8.64731311,  8.77424615,  8.90117919,  9.02811222,
        9.15504526,  9.28197829,  9.40891133,  9.53584437,  9.6627774 ,
        9.78971044,  9.91664348, 10.04357651, 10.17050955, 10.29744259,
       10.42437562, 10.55130866, 10.6782417 , 10.80517473, 10.93210777,
       11.05904081, 11.18597384, 11.31290688, 11.43983992, 11.56677295,
       11.69370599, 11.82063902, 11.94757206, 12.0745051 , 12.20143813,
       12.32837117, 12.45530421, 12.58223724, 12.70917028, 12.83610332,
       12.96303635, 13.08996939, 13.21690243, 13.34383546, 13.4707685 ,
       13.59770154, 13.72463457, 13.85156761, 13.97850065, 14.10543368,
       14.23236672, 14.35929976, 14.48623279, 14.61316583, 14.74009886,
       14.8670319 , 14.99396494, 15.12089797, 15.24783101, 15.37476405,
       15.50169708, 15.62863012, 15.75556316, 15.88249619, 16.00942923,
       16.13636227, 16.2632953 , 16.39022834, 16.51716138, 16.64409441,
       16.77102745, 16.89796049, 17.02489352, 17.15182656, 17.27875959])
```

Построение графика стоимости материала в зависимости от радиуса:

Ввод [109]:

```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

# Постройте график стоимости материала в зависимости от радиуса
ax1.plot(r_vals, C_vals, color='blue')
ax1.set_xlabel('Радиус (см)')
ax1.set_ylabel('Цена материала ($)', color='blue')

# Постройте график скорости увеличения в зависимости от радиуса
ax2.plot(r_vals, dCdr_vals, color='red')
ax2.set_ylabel('Частота увелечения ($/c)', color='red')

plt.show()
```



Из графика видно, что скорость увеличения стоимости максимальна, когда радиус мал, и уменьшается по мере увеличения радиуса. Это имеет смысл, поскольку с увеличением радиуса увеличение площади поверхности на каждую единицу увеличения радиуса становится меньше, поэтому стоимость материала растет медленнее.

Ввод [ ]:

```
```