HACKTIV8

//01

FTDS // Calculus // Derivative

Hacktiv8 DS Curriculum Team

Phase 0 Learning Materials

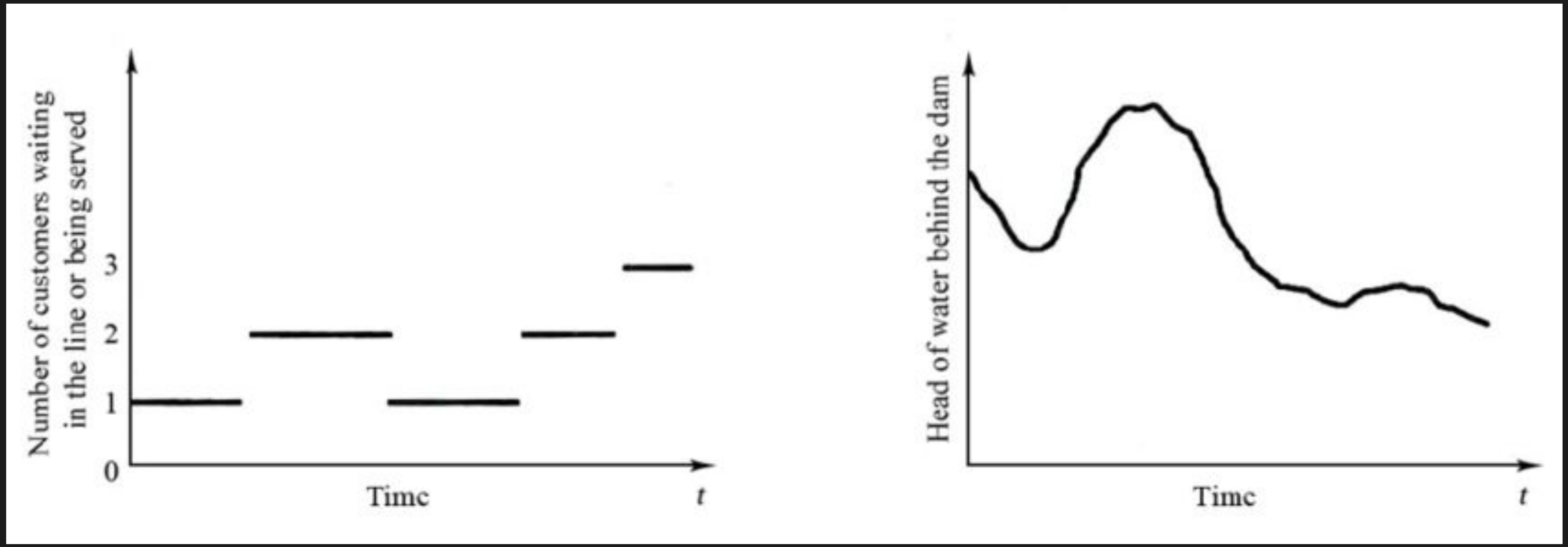
Hacktiv8 DS Curriculum Team

Contents

- Basic understanding of derivative
- Able to calculate the derivative of a function
- Basic understanding of Partial Derivative
- Basic understanding of Gradient, Gradient Descent, Jacobian, and Hessian
- Able to calculate the Partial Derivative of a function
- Able to implement derivative calculation on Python

Calculus studied continuous changes

Diskrit **Kontinu**

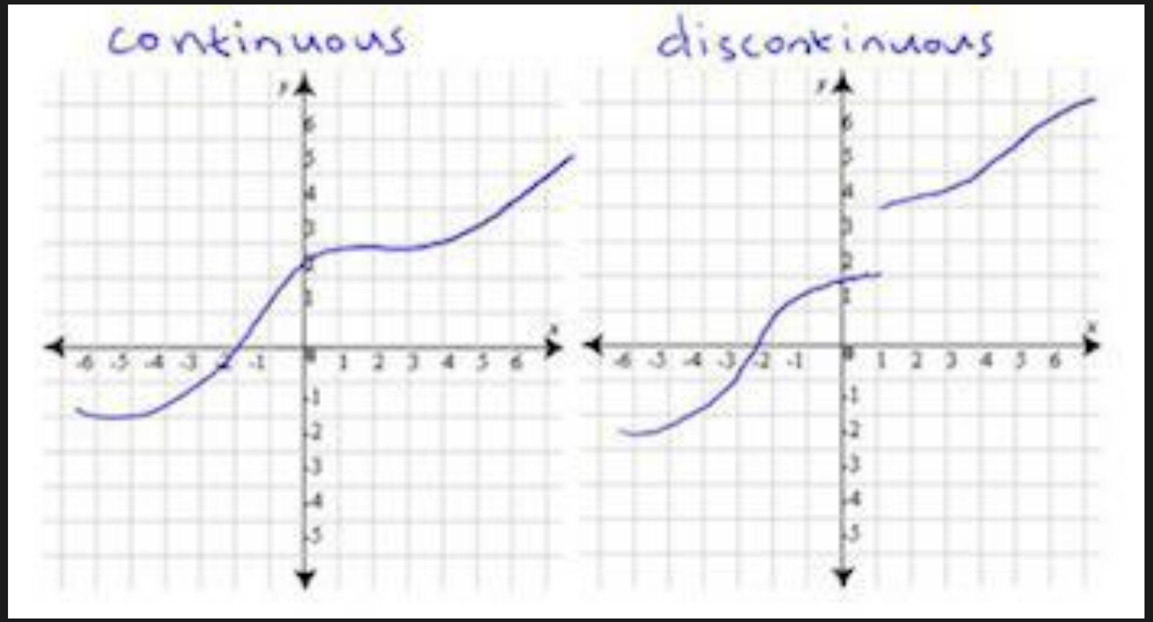


Since calculus studies continuous things, so that concept of limit is revealed.

Let consider a function $f(x) = \frac{(x^2-1)}{x-1}$ **We want to find value of** $f(x)$ **for** $x=1$, $f(1)$

$$f(1) = \frac{(1^2 - 1)}{1 - 1} = \frac{0}{0}$$

Sometimes we can't directly calculate the value of f(x) for certain value of x.



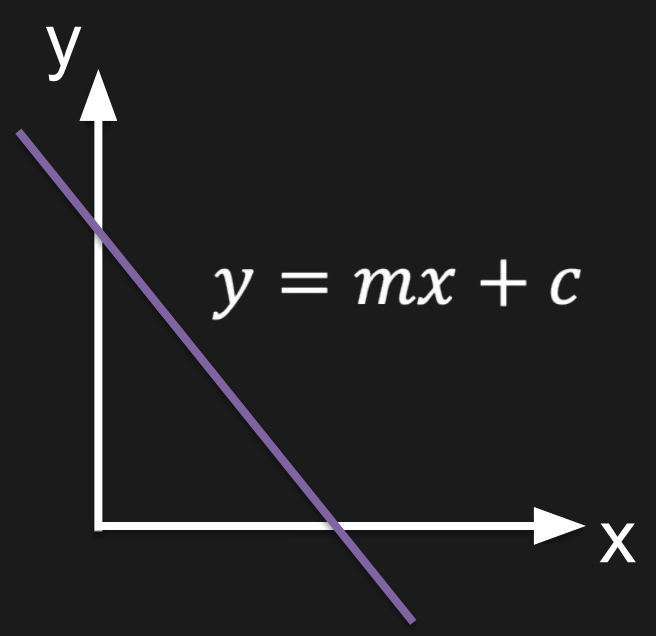
We can choose a value approach to 1

x	$(x^2 - 1)(x - 1)$
0.5	1.50000
0.9	1.90000
0.99	1.99000
0.999	1.99900
0.9999	1.99990
0.99999	1.99999

x close to 1, f(x) close to, it can be written mathematically:

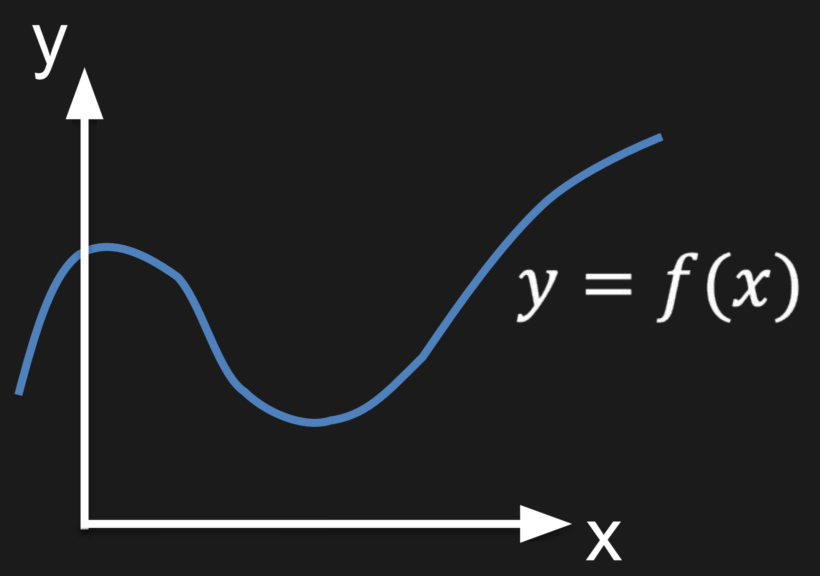
$$\lim_{x \to 1} \frac{(x^2 - 1)}{x - 1} = 2$$

Concept of derivative comes from the concept of line



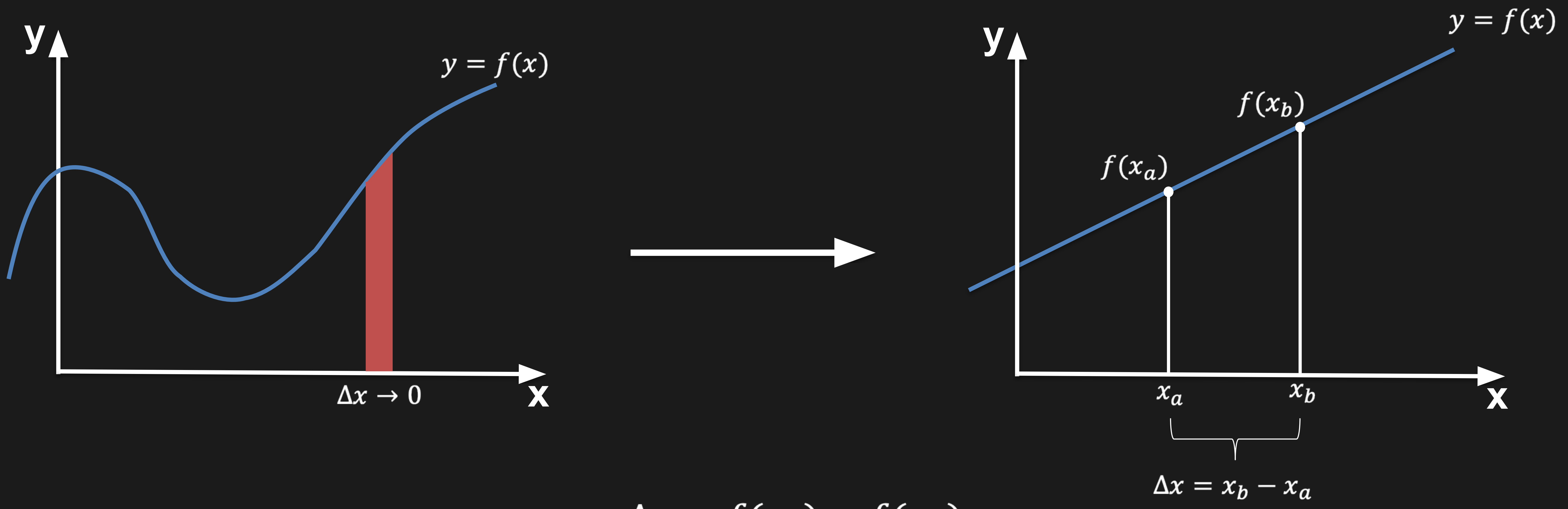
$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

Slope represents an increasing/decreasing change of y as x increases. Other words, the slope indicates how much the change is.



How does the value of y change as x increases?

To make it easier to see a change/measure the slope of a line, we consider a small area of a curve



$$m = \frac{\Delta y}{\Delta x} = \frac{f(x_b) - f(x_a)}{x_b - x_a}$$

$$\boxed{\lim_{\Delta x \to 0} \frac{f(x_b + \Delta x) - f(x_a)}{\Delta x} = \frac{\mathrm{d}y}{\mathrm{d}x} = f'(x)}$$

General formula to calculate derivative of a function f(x):

$$f'(x) = nx^{n-1}$$

Example:

- $f(x) = x$
- $f(x) = 2x^2 + 1$
- $f(x) = 3x^2 + 2x - 4$

The following table is the rules of derivative

	Function f	Derivative f'
Constant	$f(x) = c$	$f'(x) = 0$
Sum	$f(x) = g(x) + h(x)$	$f'(x) = g'(x) + h'(x)$
Product	$f(x) = g(x)h(x)$	$f'(x) = g(x)h'(x) + g'(x)h(x)$
Quotient	$f(x) = \frac{g(x)}{h(x)}$	$f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{h^2(x)}$
Power	$f(x) = x^r$ with $r \neq 0$	$f'(x) = rx^{r-1}$
Exponential	$f(x) = \exp(x)$	$f'(x) = \exp(x)$
Logarithm	$f(x) = \ln(x)$	$f'(x) = \frac{1}{x}$
Sin	$f(x) = \sin(x)$	$f'(x) = \cos(x)$
Cos	$f(x) = \cos(x)$	$f'(x) = -\sin(x)$
Tan	$f(x) = \tan(x)$	$f'(x) = \frac{1}{\cos^2(x)}$
Chain Rule	$f(x) = g(h(x))$	$f'(x) = g'(h(x))\, h'(x)$

In the previous lesson, we have reconized the derivative concept which is:

$$f'(x) = \frac{df(x)}{dx}$$

Is it possible that we derive the derivative of a function? IT'S POSSIBLE!

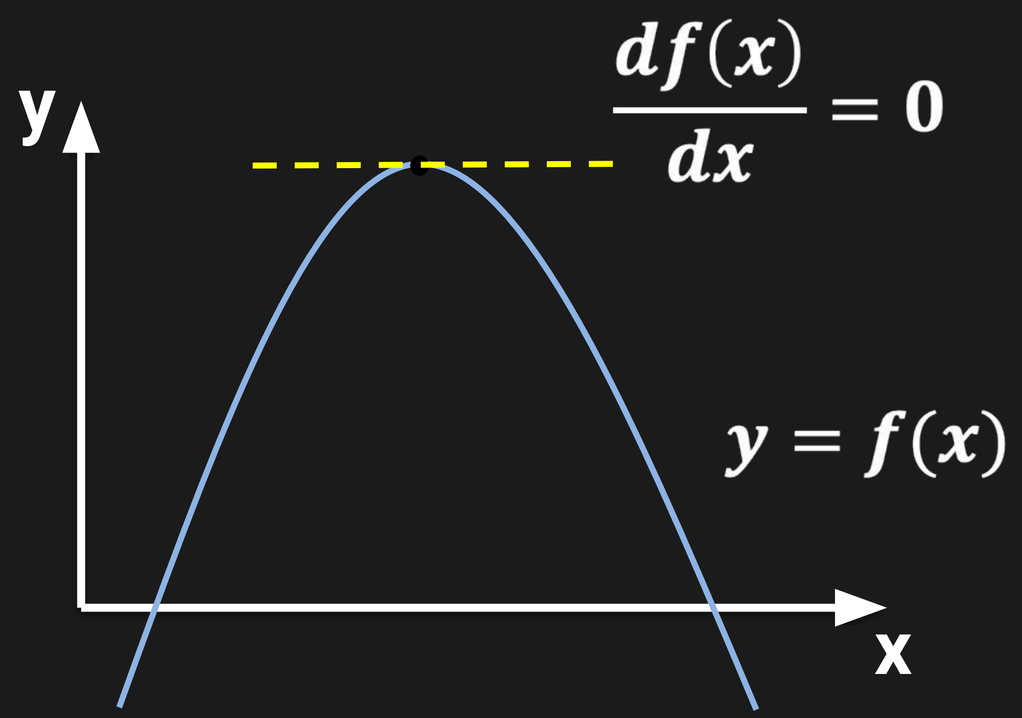
$$f''(x) = \frac{df'(x)}{dx} = \frac{d}{dx}\left(\frac{df(x)}{dx}\right) = \frac{d^2f(x)}{dx^2}$$

$$\boxed{f^n(x) = \frac{d^nf(x)}{dx^n}}$$

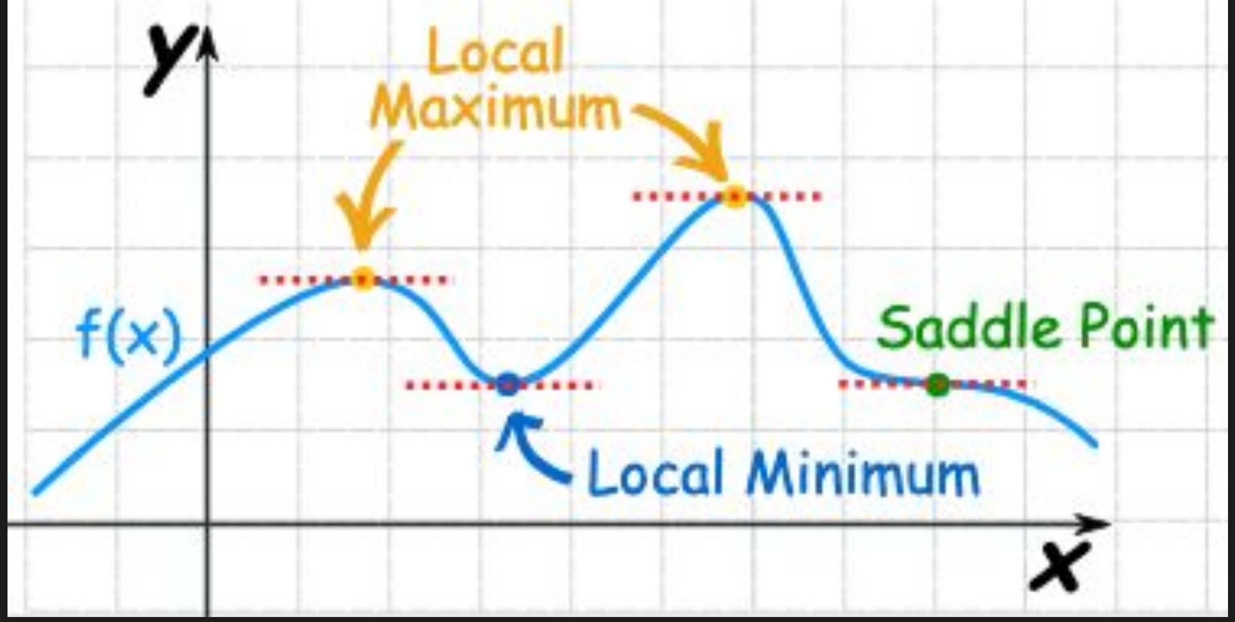
Example:

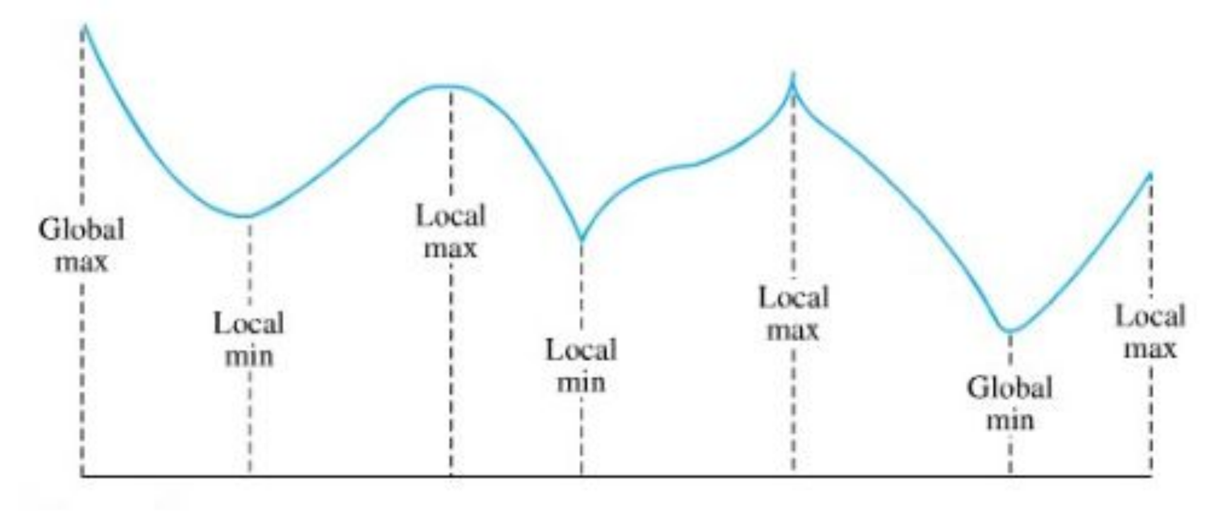
- $f(x) = 3x^2 + 2x - 4$
- $f(x) = 4x^3 - x^2 + 7x + 1$

One of the application of derivative is optimization. The aim of optimization is to find the minimum/maximum value (extreme point) of a function.

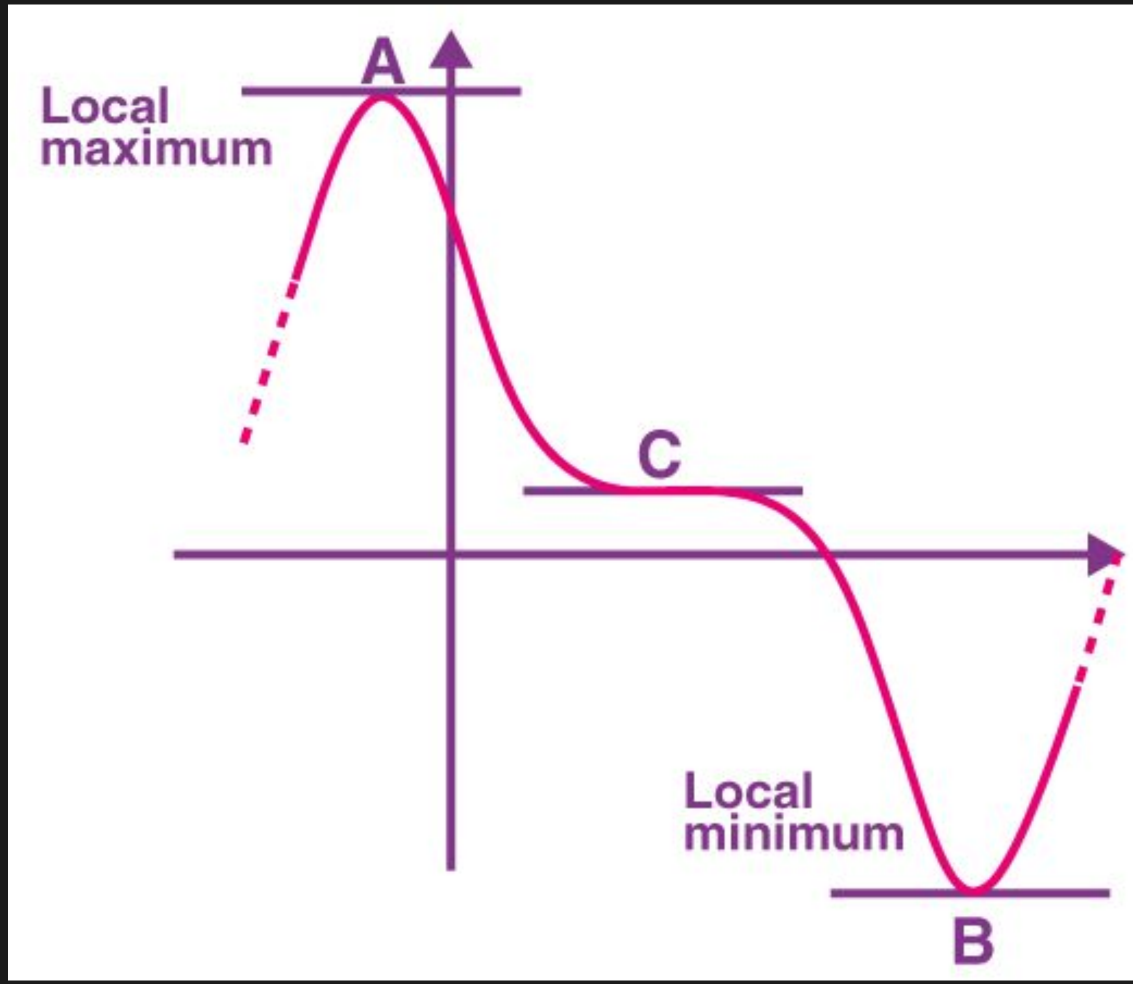


How to find the extreme point that is a local/global min/max?





To determine an extreme point whether it is local minimum or maximum, we can use the second derivative.



Finding extreme points: $f'(x) = 0$

The signatures of minimum/maximum/stationary extreme points:

A. Maximum : $f''(x) < 0$ *De-accelerate

B. Minimum : $f''(x) > 0$ *Accelerate

C. Stasionary : $f''(x) = 0$ *Stagnant/Constant

Derivative on Code // Symbolic

$$f(x) = 2x^2 + 4x - 1$$
$$f'(x) = 4x + 4$$

This method the input or the output as symbols even also the function.

```
import sympy as sy

x = sy.Symbol('x',real=True)
f = 2*x**2+4*x-1

f.diff(x)
```

Output:

$$4x + 4$$

Derivative on Code // Numerical

$$f(x) = 2x^2 + 4x - 1$$
$$f'(x) = 4x + 4$$

In numeric way, we do not consider the input or the output as symbols, yet sample of data/array.

```python
import numpy as np

x = np.linspace(0,20)
y = 2*x**2+4*x-1

df=np.diff(y)/np.diff(x)
```

OR

```python
import numpy as np

x = np.linspace(0,20)
y = 2*x**2+4*x-1

df=np.gradient(y,x)
```

Optimization on Code// Find Minimum

$$f(x) = 2x^2 + 4x - 1$$

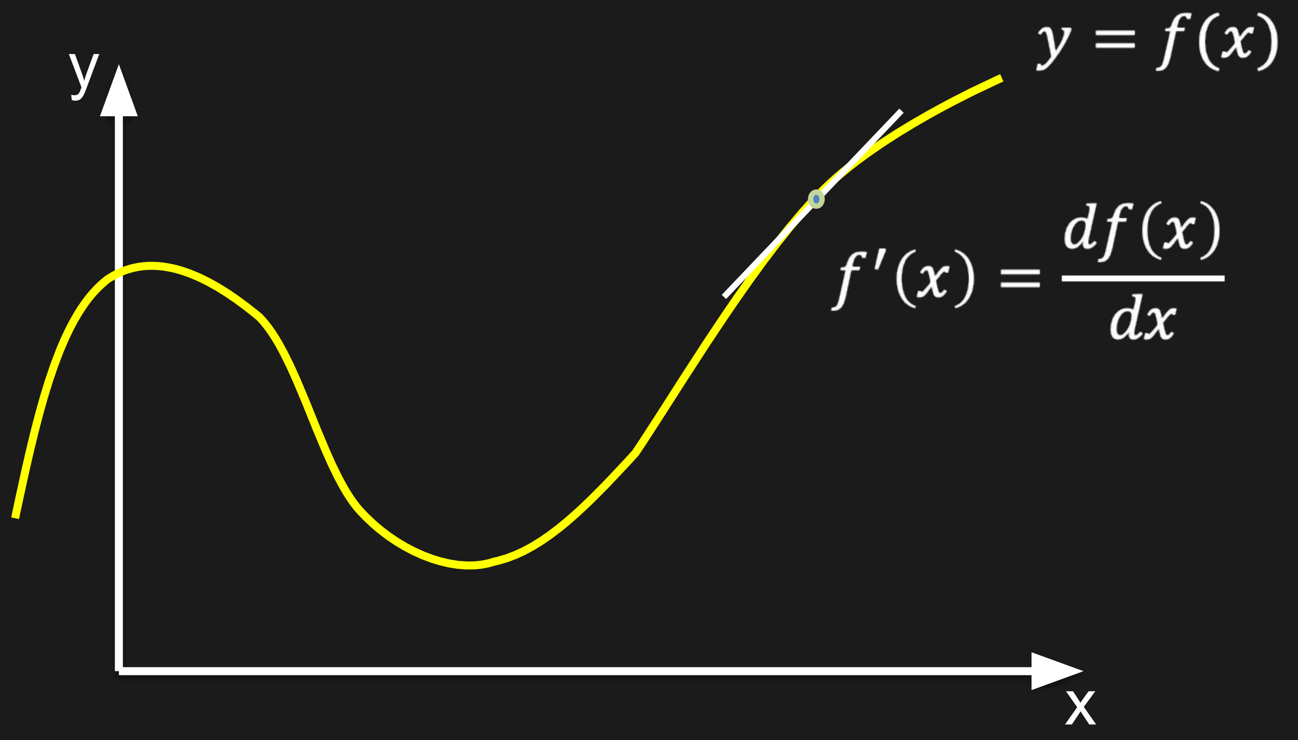
In numeric way, we do not consider the input or the output as symbols, yet sample of data/array.

```
from scipy.optimize import minimize_scalar

def f(x):
    return 2*x**2+4*x-1

opt=minimize_scalar(f)
```

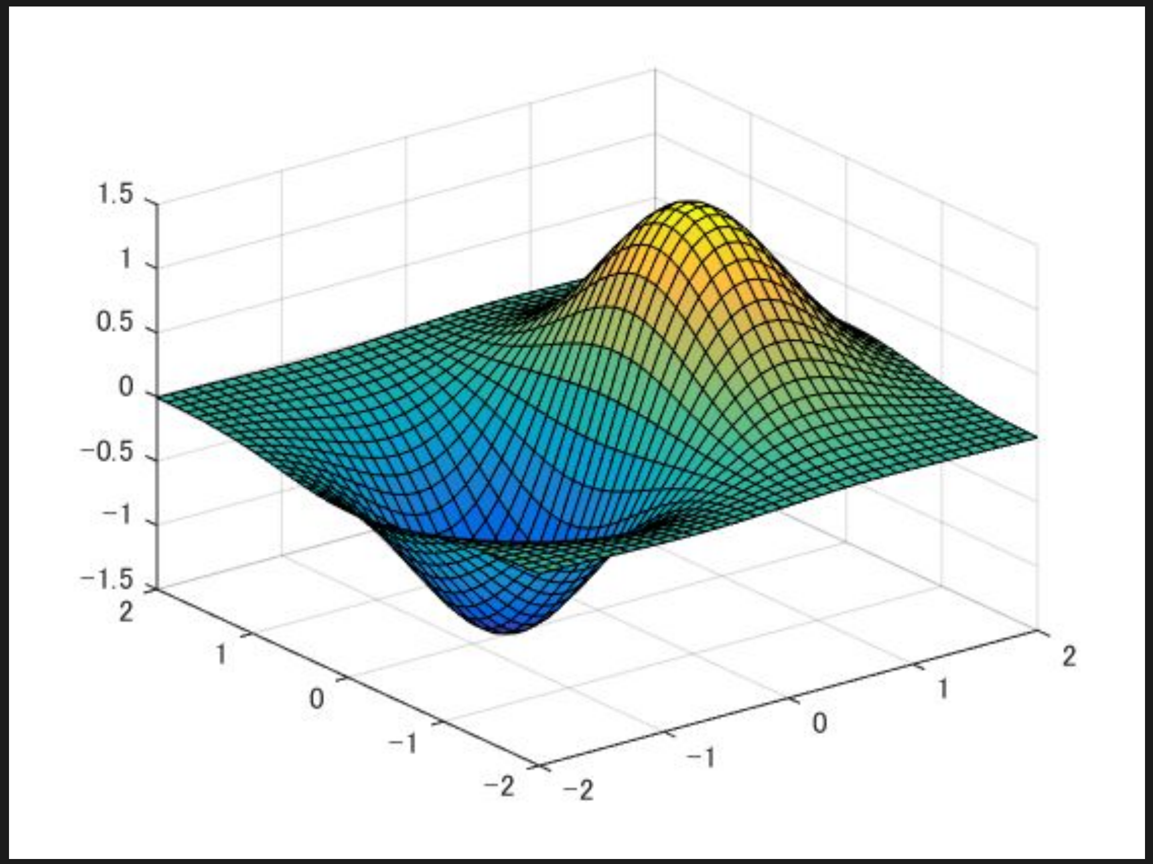
***Output*:**

```
fun: -3.0
nfev: 9
nit: 4
success: True
x: -1.0000000000000002
```

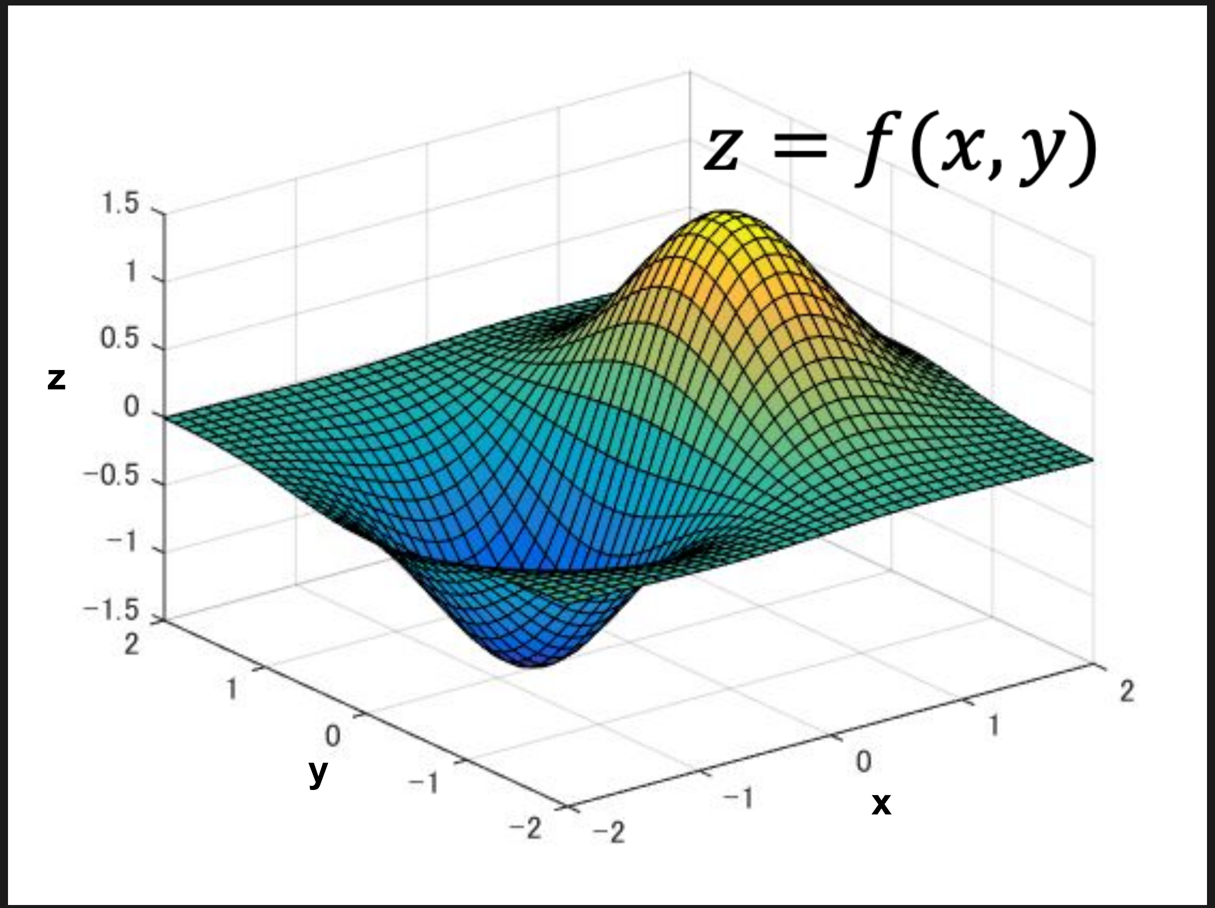
Partial Derivative works with multivariable function



In the prior lesson, we learn how to calculate the rate of change of a curve.



But how do we calculate the rate of change of a surface?



To measure a rate of change of a surface, we can calculate it based on each the axis direction. Let we have a surface z=f(x,y), the partial derivatives are:

$$\frac{\partial f(x,y)}{\partial x} \qquad \frac{\partial f(x,y)}{\partial y}$$

Example:

$$z = f(x, y) = xy^2 + x^3$$

$$\partial_x f = \frac{\partial f}{\partial x} = y^2 + 2x^2$$

$$\partial_y f = \frac{\partial f}{\partial y} = 2xy$$

Second Order

$$z = f(x,y)$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial f(x,y)}{\partial y}\right)$$

$$\frac{\partial f(x,y)}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial y}\right)$$

$$\frac{\partial f(x,y)}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial f(x,y)}{\partial x}\right)$$

Example:

$$z = f(x, y) = xy^2 + x^3$$

$$\partial_x f = \frac{\partial f}{\partial x} = y^2 + 2x^2$$

$$\partial_y f = \frac{\partial f}{\partial y} = 2xy$$

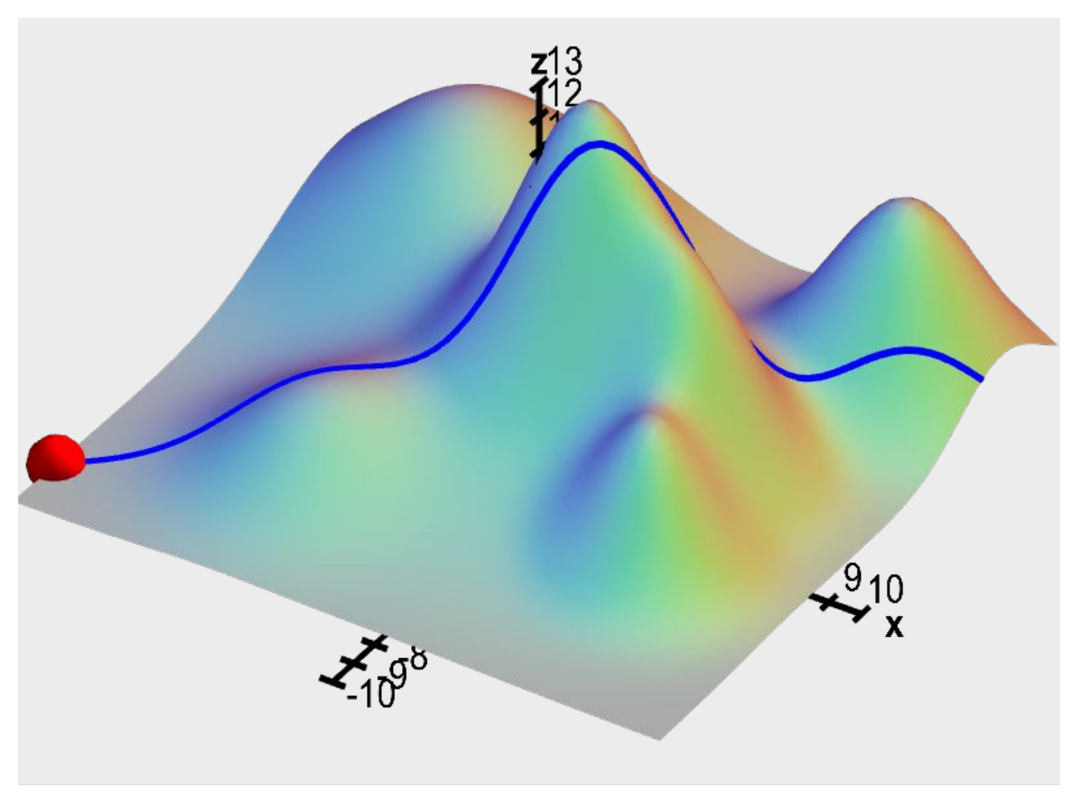
$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(\partial_x f) = 4x$$

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}(\partial_x f) = 2y$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(\partial_y f) = 2x$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(\partial_y f) = 2y$$

When you calculate the partial derivative of a multivariate function, you will get more than one results. Gradient simply store your results into a vector. Yet mathematicaly, gradient is a change rate on the surface.



$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$\nabla f(x,y,z) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix}$$

Example:

$$z = f(x,y) = xy^2 + x^3$$

$$\partial_x f = \frac{\partial f}{\partial x} = y^2 + 2x^2$$

$$\partial_y f = \frac{\partial f}{\partial y} = 2xy$$

$$\nabla f(x,y) = \begin{pmatrix} y^2 + 2x^2 \\ 2xy \end{pmatrix}$$

Sometimes you will have a multi-dimensional function, also the input and the output are multi-dimensional. So, you will work with Jacobian. In brief, Jacobian is a matrix that store the partial derivative of the functions.

$$J_f = J_{i,j} = \frac{\partial}{\partial x_j} f(x)_i = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_i \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_j} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_j} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_i}{\partial x_1} & \frac{\partial f_i}{\partial x_2} & \cdots & \frac{\partial f_i}{\partial x_j} \end{bmatrix}$$

Example:

$$f(x,y) = \begin{bmatrix} x^2y + y \\ 2xy - 2 \end{bmatrix}$$

$$J_f = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy & x^2 + 1 \\ 2y & 2x \end{bmatrix}$$

Like gradient for first order partial derivative, we can also store the second partial derivative results in a matrix. The matrix is called Hessian. Hessian also represents second order gradient.

$$H_f = \nabla^2 f = H_{i,j} = \frac{\partial^2}{\partial x_i x_j} f(x)_i = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 \partial x_j} \\ \frac{\partial^2 f_2}{\partial x_2 \partial x_1} & \frac{\partial^2 f_2}{\partial x_2^2} & \cdots & \frac{\partial f_2}{\partial x_j} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_i}{\partial x_i \partial x_1} & \frac{\partial^2 f_i}{\partial x_i \partial x_2} & \cdots & \frac{\partial^2 f_i}{\partial x_j^2} \end{bmatrix}$$

Example:

$$z = f(x, y) = xy^2 + x^3$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(\partial_x f) = 4x$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(\partial_y f) = 2y$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(\partial_y f) = 2x$$

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}(\partial_x f) = 2y$$

$$H_f = \nabla^2 f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x}(\partial_x f) & \frac{\partial}{\partial x}(\partial_y f) \\ \frac{\partial}{\partial y}(\partial_x f) & \frac{\partial}{\partial y}(\partial_y f) \end{bmatrix} = \begin{bmatrix} 4x & 2y \\ 2y & 2x \end{bmatrix}$$

Partial Derivative on Code // Symbolic// First Order

$$f(x) = 4xy + x \sin z + x^3 + z^8 y$$

This method the input or the output as symbols even also the function.

```
import sympy as sy

x,y,z = sy.symbols('x y z')
f = 4*x*y + x*sy.sin(z) + x**3 + z**8*y
```

sy.diff(f,x) ***Output:*** $3x^2 + 4y + \sin z$

sy.diff(f,y) ***Output:*** $4x + z^8$

sy.diff(f,z) ***Output:*** $x \cos z + 8yz^7$

Partial Derivative on Code // Symbolic// Gradient

$$f(x) = 4xy + x\sin z + x^3 + z^8 y$$

This method the input or the output as symbols even also the function.

```
import sympy as sy
from sympy.tensor.array import derive_by_array

x,y,z = sy.symbols('x y z')
f = 4*x*y + x*sy.sin(z) + x**3 + z**8*y

derive_by_array(f, (x,y,z))
```

Output:

$$\begin{pmatrix} 3x^2 + 4y + \sin z \\ 4x + z^8 \\ x\cos z + 8yz^7 \end{pmatrix}$$

Partial Derivative on Code // Symbolic// Jacobian

$$f(x,y) = \begin{bmatrix} x^2y + y \\ 2xy - 2 \end{bmatrix}$$

This method the input or the output as symbols even also the function.

```
import sympy as sy

x,y,z = sy.symbols('x y z')

f =sy.Matrix([x**y+y, 2*x*y-2])

sy.hessian(f, (x,y))
```

Output:

$$\begin{bmatrix} 2xy & x^2 + 1 \\ 2y & 2x \end{bmatrix}$$

Partial Derivative on Code // Symbolic// Hessian

$$z = f(x,y) = xy^2 + x^3$$

This method the input or the output as symbols even also the function.

```
import sympy as sy

x,y,z = sy.symbols('x y')

f =sy.Matrix([x**y+y, 2*x*y-2])
X=sy.Matrix([x,y])

f.jacobian(X)
```

Output:

$$\begin{bmatrix} 4x & 2y \\ 2y & 2x \end{bmatrix}$$

Partial Derivative on Code // Numeric//

$$z = f(x,y) = xy^2 + x^3$$

Same as the derivative, we can use numpy gradient to compute partial derivative numerically. But, we need to define a matrix or tensor to store the f(x,y) values.

```
import numpy as np

def f(x,y):
    return x**2*y+2*x**3*y+y**4

x=np.linspace(1,10)
y=np.linspace(1,10)

z=np.array([[f(i,j) for i in x] for j in y])

dx,dy=np.gradient(z)
```