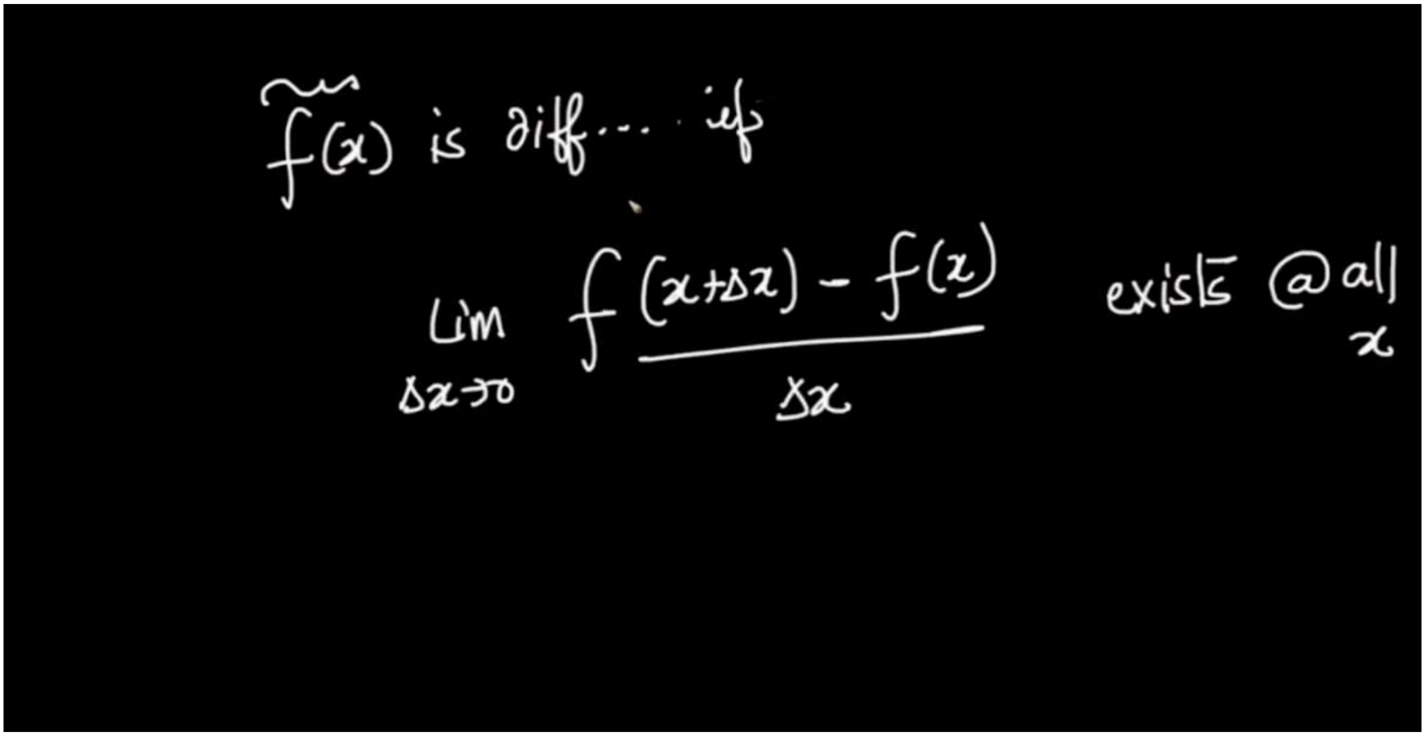
Lets revise certain concepts from Calulus classes which will be helpful in today's class

1. Differentialbility
2. Maxima and Minima
3. Partial Derivates
4. Gradient Descent
5. Implemetation from Gradient Descent from for simple function, $y = (x-5)^2$

If you want to check these topics in detail, you can check back Calulus lectures

Differentiability

- We say f(x) is differentiable at a point x= a,
- if df(x)/dx = lim Δx→0 f(x + Δx) - f(x)/ Δx exists at a

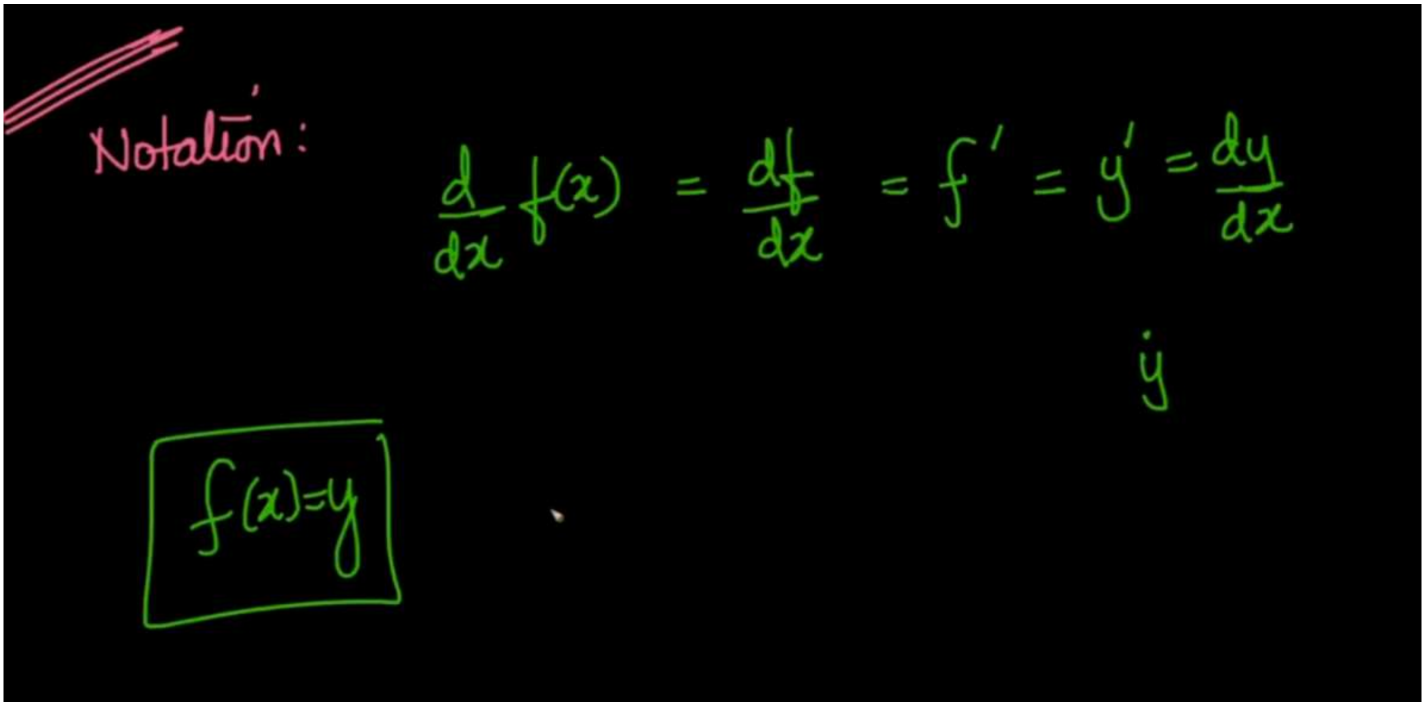


$\tilde{f}(x)$ is diff... if

$\lim_{\Delta x \to 0} f \frac{(x+\Delta x) - f(x)}{\Delta x}$ exists @ all x

Notation

- f(x) = y
- df(x)/dx = df/dx = f' = y' = dy/dx



Notation:

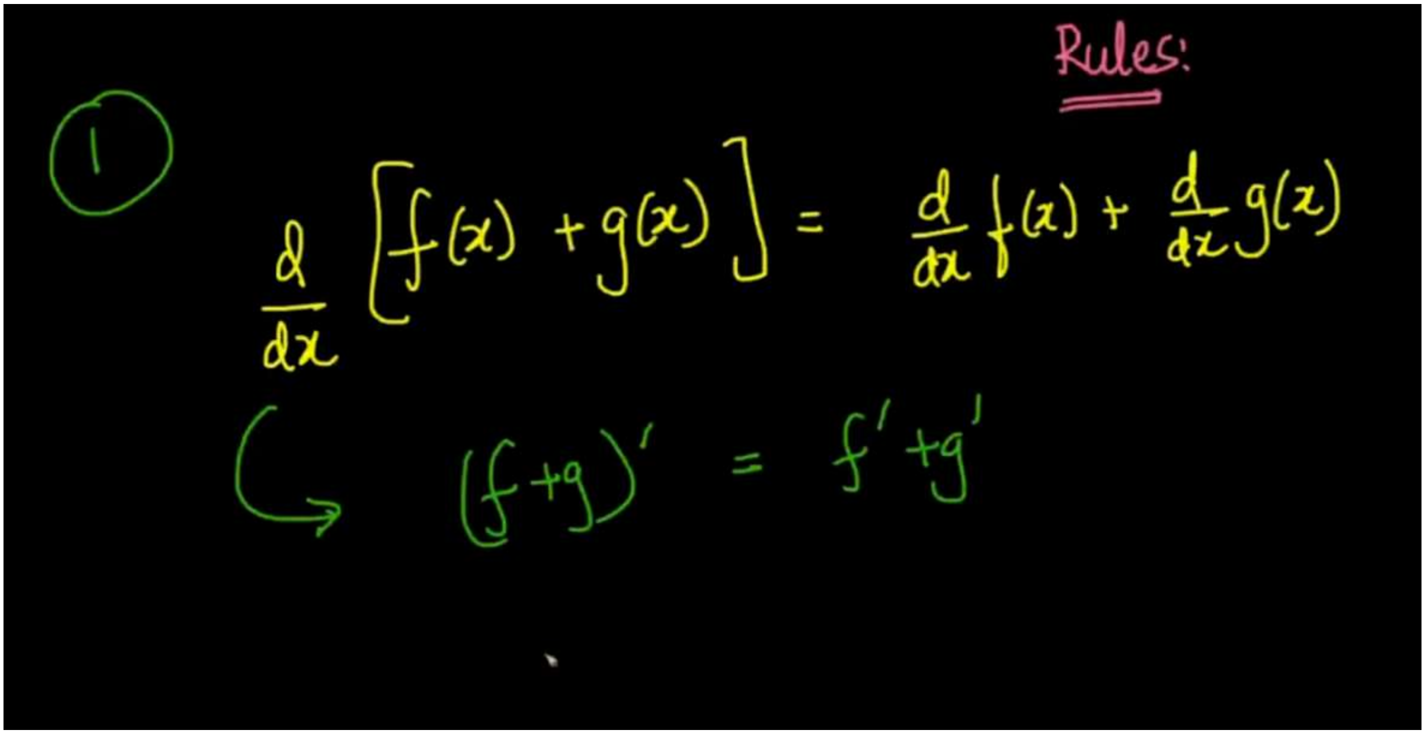
$$\frac{d}{dx}f(x) = \frac{df}{dx} = f' = y' = \frac{dy}{dx}$$

$$\dot{y}$$

$$\boxed{f(x)=y}$$

Basic Rules Of Differentiability

Addition Rule



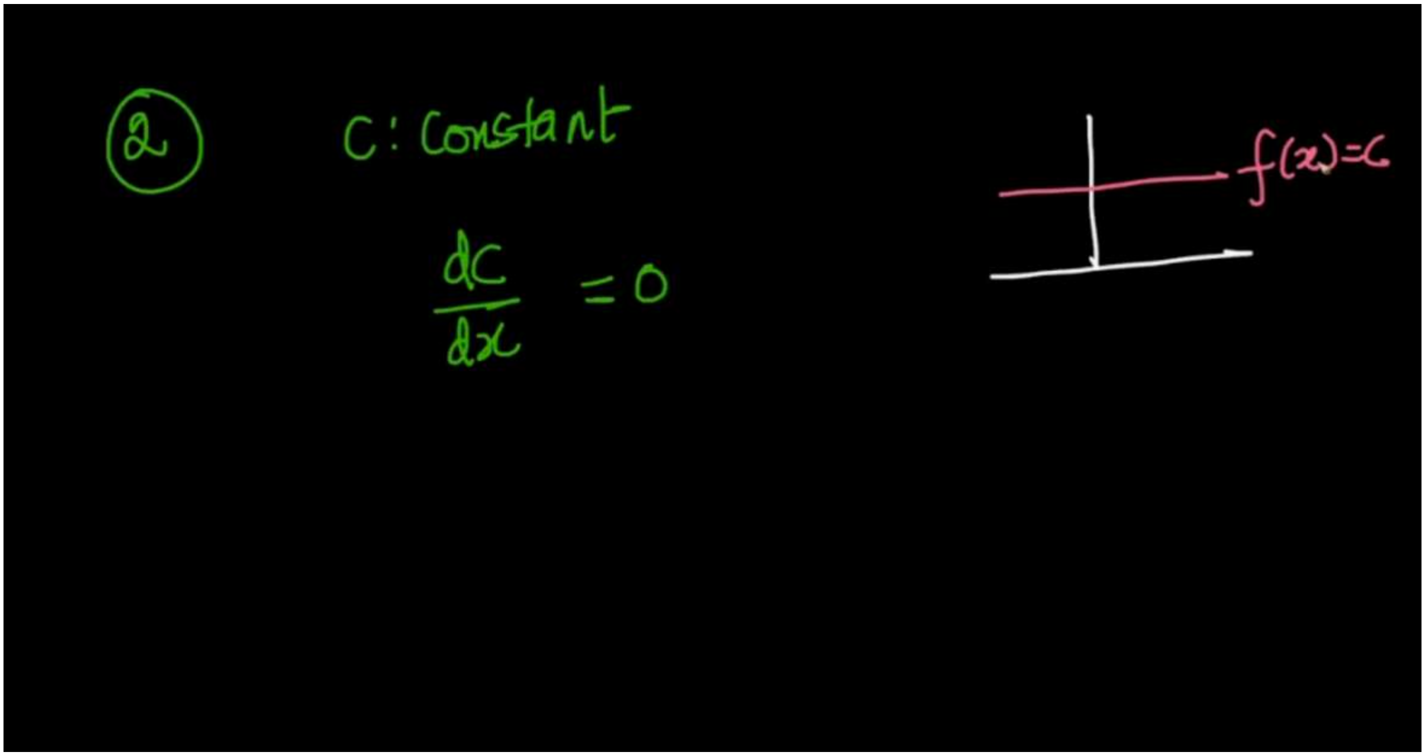
Rules:

1

$$\frac{d}{dx}\left[f(x) + g(x)\right] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$$

$$\hookrightarrow \quad (f+g)' = f'+g'$$

Multiplication Rule

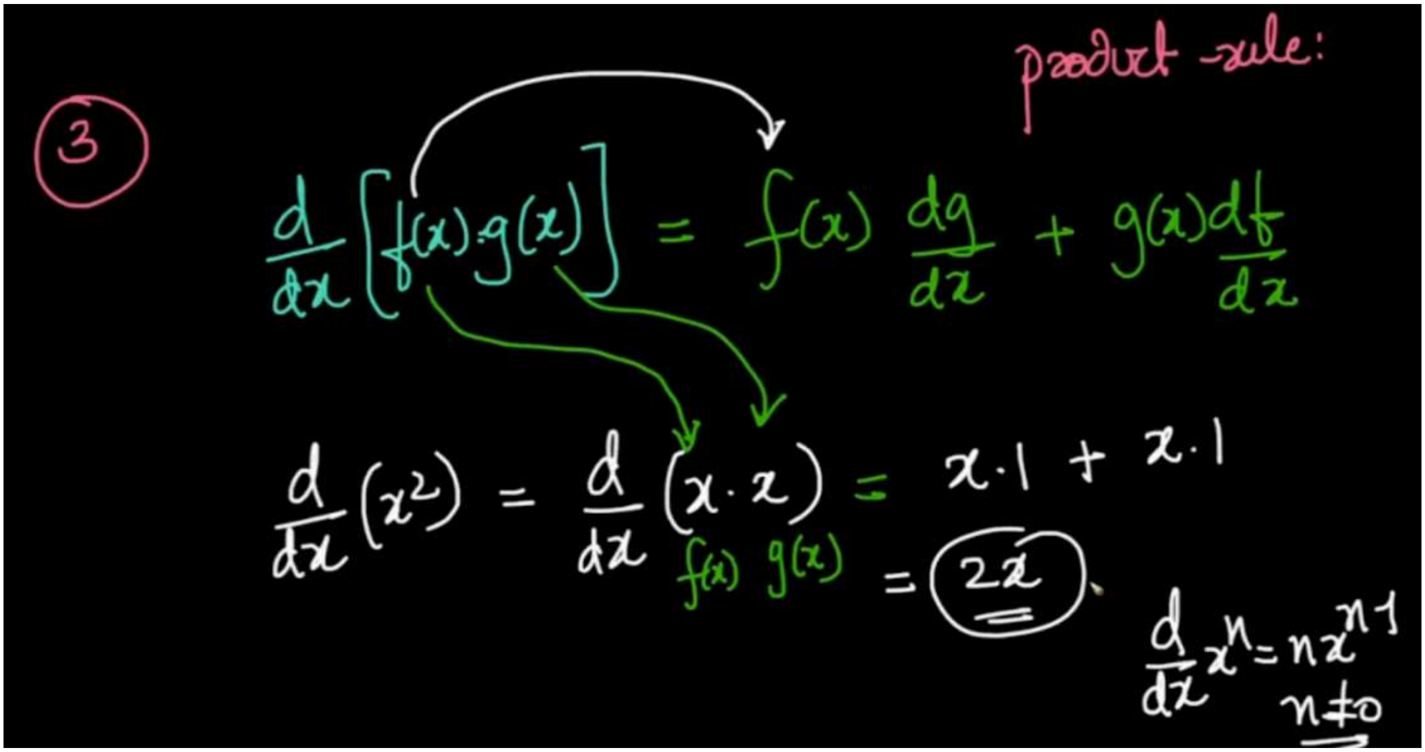


2

c: Constant

$$\frac{dc}{dx} = 0$$

Product Rule - V V Important



3

product rule:

$$\frac{d}{dx}\left[f(x)\cdot g(x)\right] = f(x)\,\frac{dg}{dx} + g(x)\frac{df}{dx}$$

$$\frac{d}{dx}(x^2) = \frac{d}{dx}(\underbrace{x}_{f(x)}\cdot\underbrace{x}_{g(x)}) = x\cdot 1 + x\cdot 1 = \boxed{2x}$$

$$\frac{d}{dx}x^n = nx^{n-1}$$

$n \neq 0$

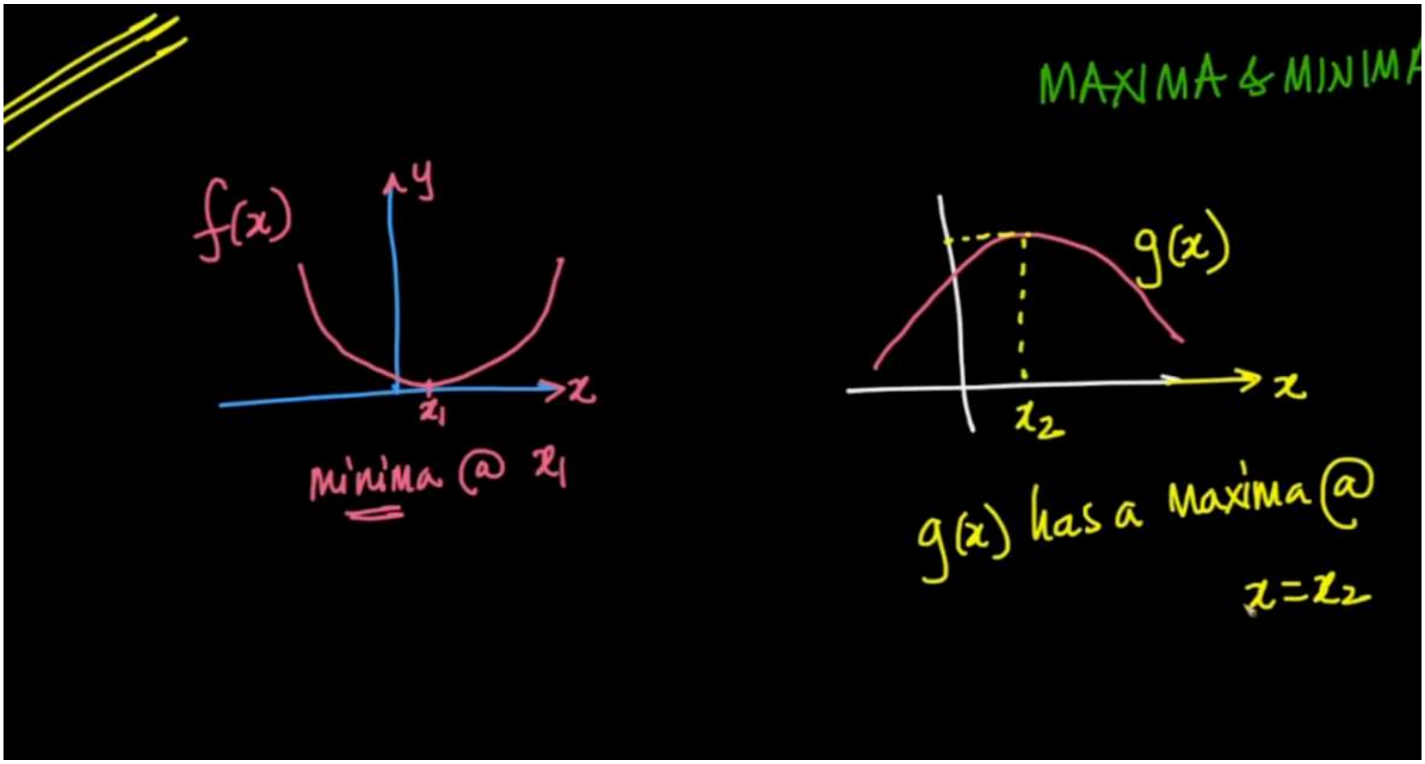
Maxima and Minima

Say we have a function y=f(x)

- We can see that the function has the minimum value at x = x1
- This point is essentially the minima.

Similarly for the function g(x)

- We can see that the function attains the maximum value at x = x2.
- This point is essentially the maxima.



There are all sorts of function - consider this higher degree polynomial

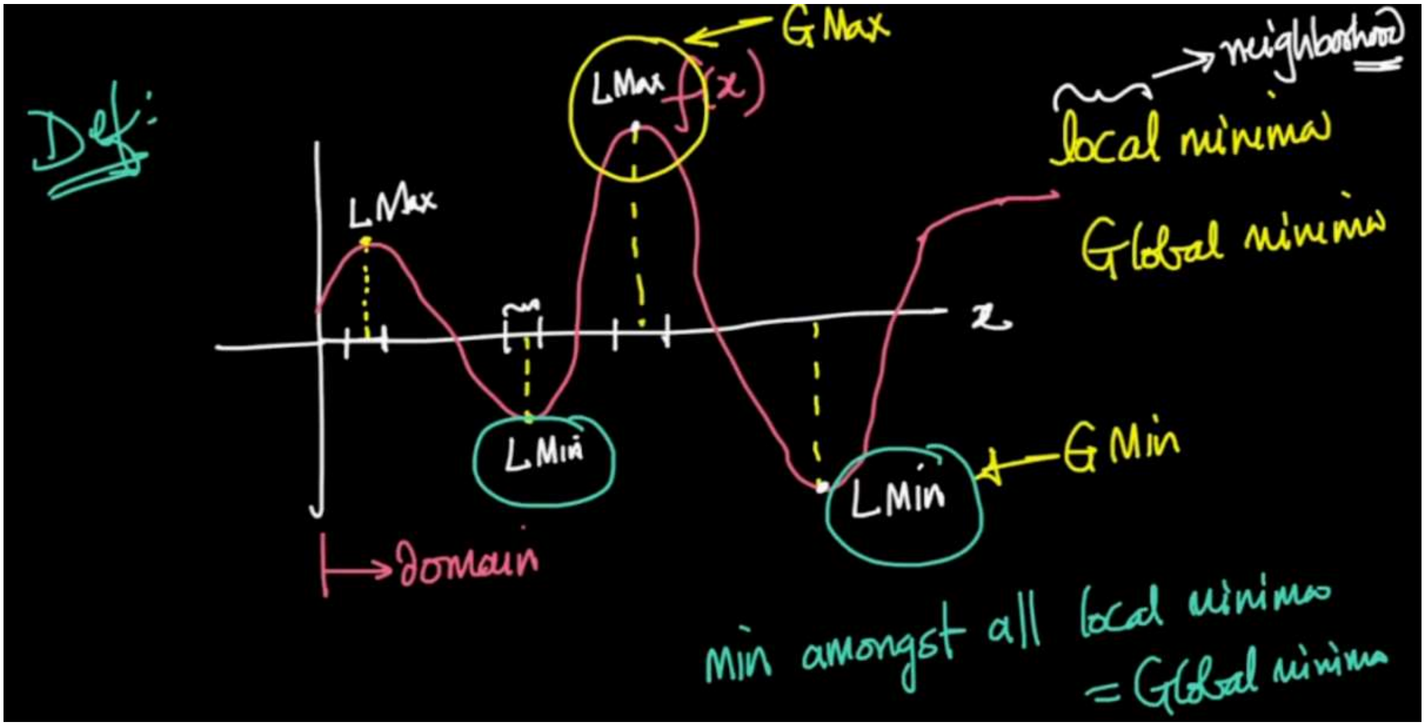
- We can see that there are multiple placese where the function has some minimum and maximum values.

Local minima/maxima

- point where the function has minimum/maximum value with respect to its vicinity/surrounding.
- We have marked it as Lmin and Lmax in the image above.

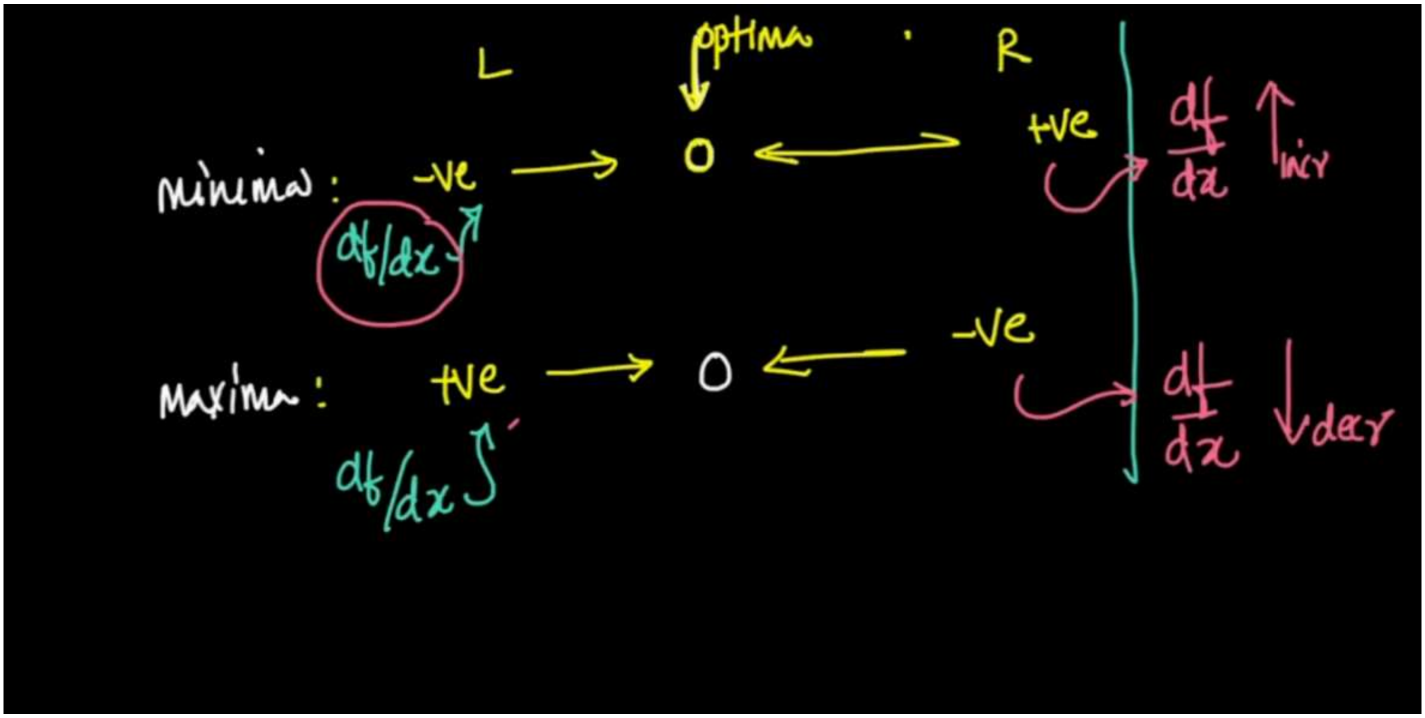
Global minima/maxima

- Minimum/maximum value out of local minima/maxima respectively.
- It is the minimum/maximum value across the whole domain.
- We have marked global minima/maxima as Gmin/Gmax respectively.



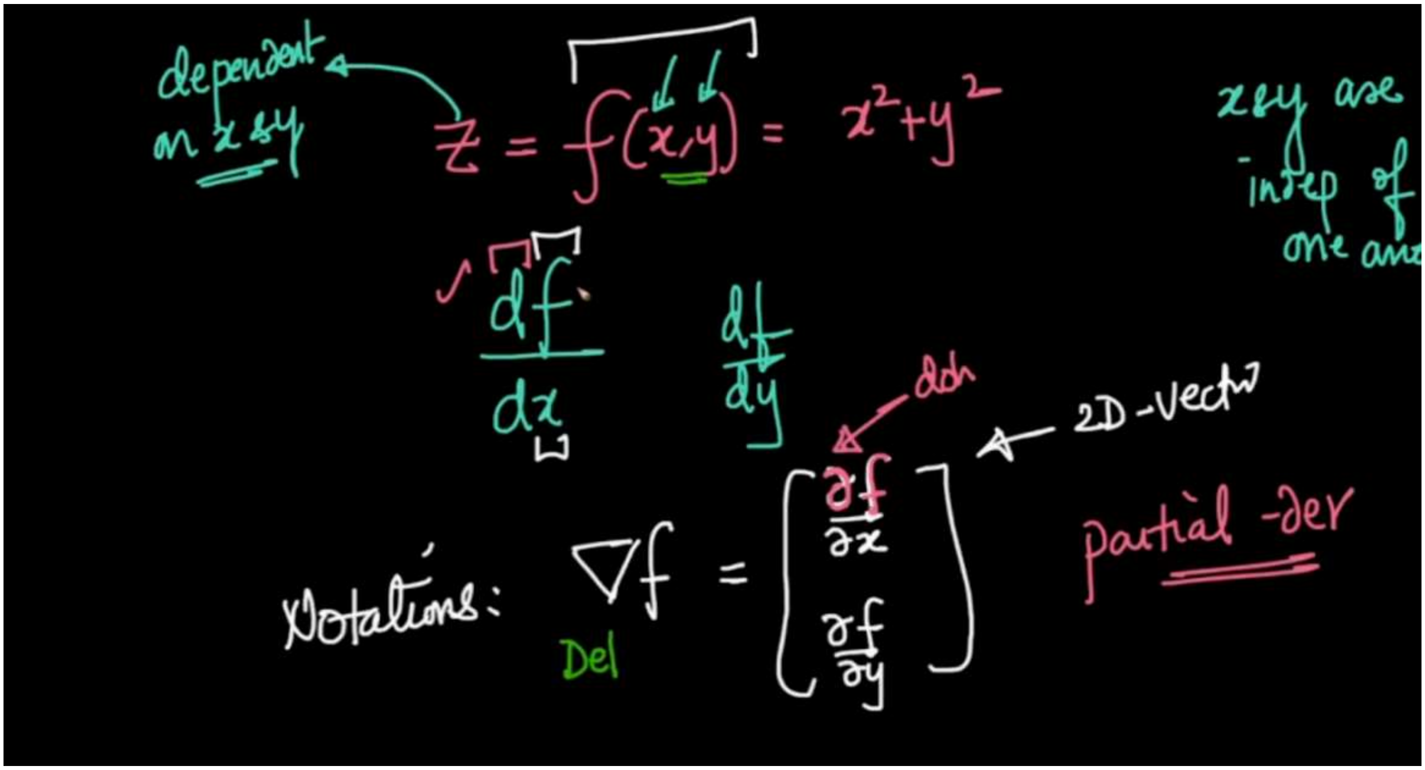
Summary - Maxmima and Minima

- We can say that for minima we have negative slopes in the left hand side and positive slopes in the right hand side.
- Both converge to 0. For the maxima we have positive slopes in the left hand side and negative values on the right side.



Partial Derivative

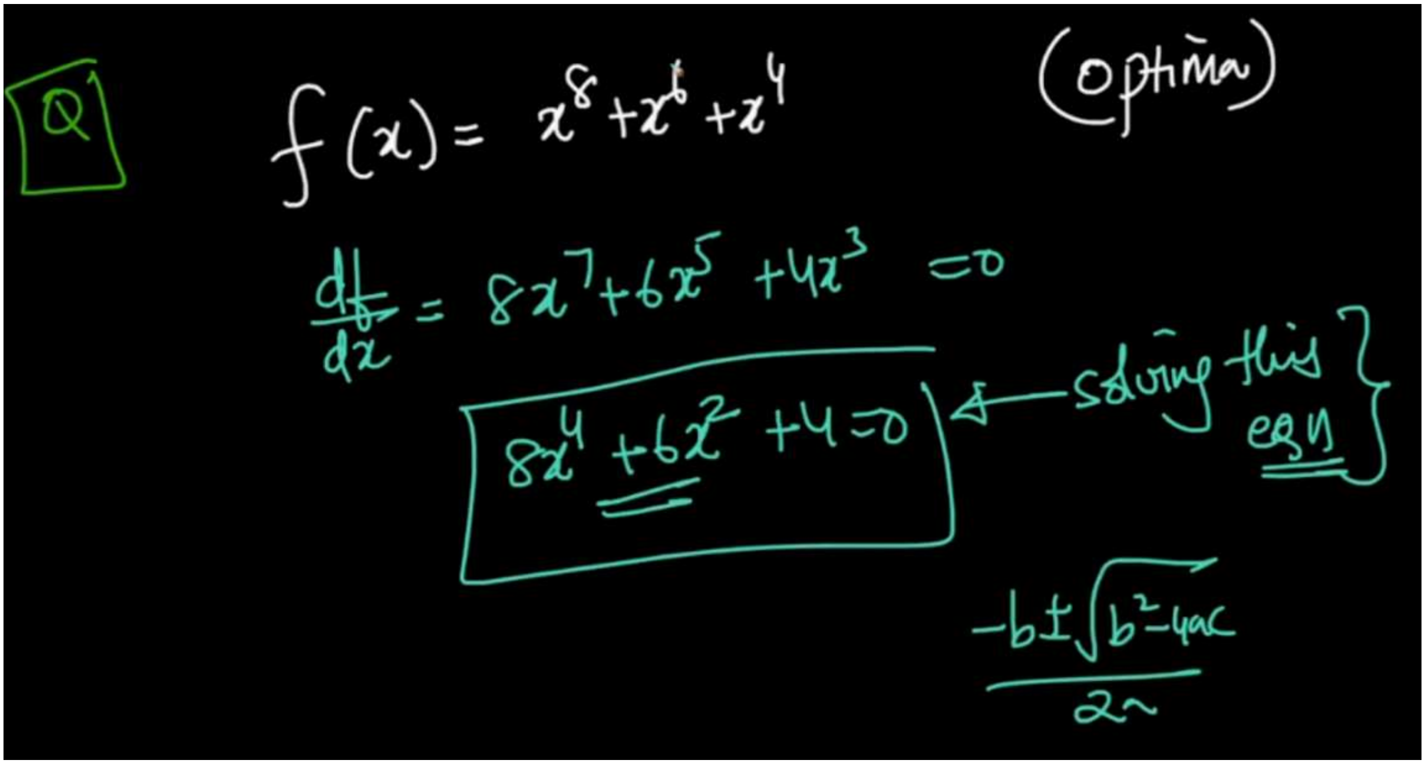
- z = f(x,y) = x2 + y2
- Here x and y are independent variables whereas z is a dependent variable.
- It is dependent on x and y. If we want to take a derivative of this function we can take it either w.r.t x or y.
- So essentially we will be taking the derivative of a function of two variables w.r.t one variable at a time.
- In order to denote this we write, ∇f ∇ is a delta operator.
- It is a 2D vector which consists of derivatives w.r.t single variables also called partial derivatives.
- Partial derivative for a function f w.r.t x is written as δf/δx. So, the 2D vector consists of partial derivatives.


dependent on x & y
$z = f(x,y) = x^2+y^2$
x & y are indep of one ano
$\frac{df}{dx}$
$\frac{df}{dy}$
doh
2D-vector
partial-der
Notations:
$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$
Del

Gradient Descent

- Let’s take an example f(x) = x8 + x6 + x4, we will be finding optima points to it.
- df/dx = 8x7 + 6x5 + 4x3 = 0 or, 8x4 + 6x2 + 4 = 0

- Solving this equation is not trivial, We don’t have any direct formula for all these polynomial equations.
- So that is the reason we need a computational algorithm.



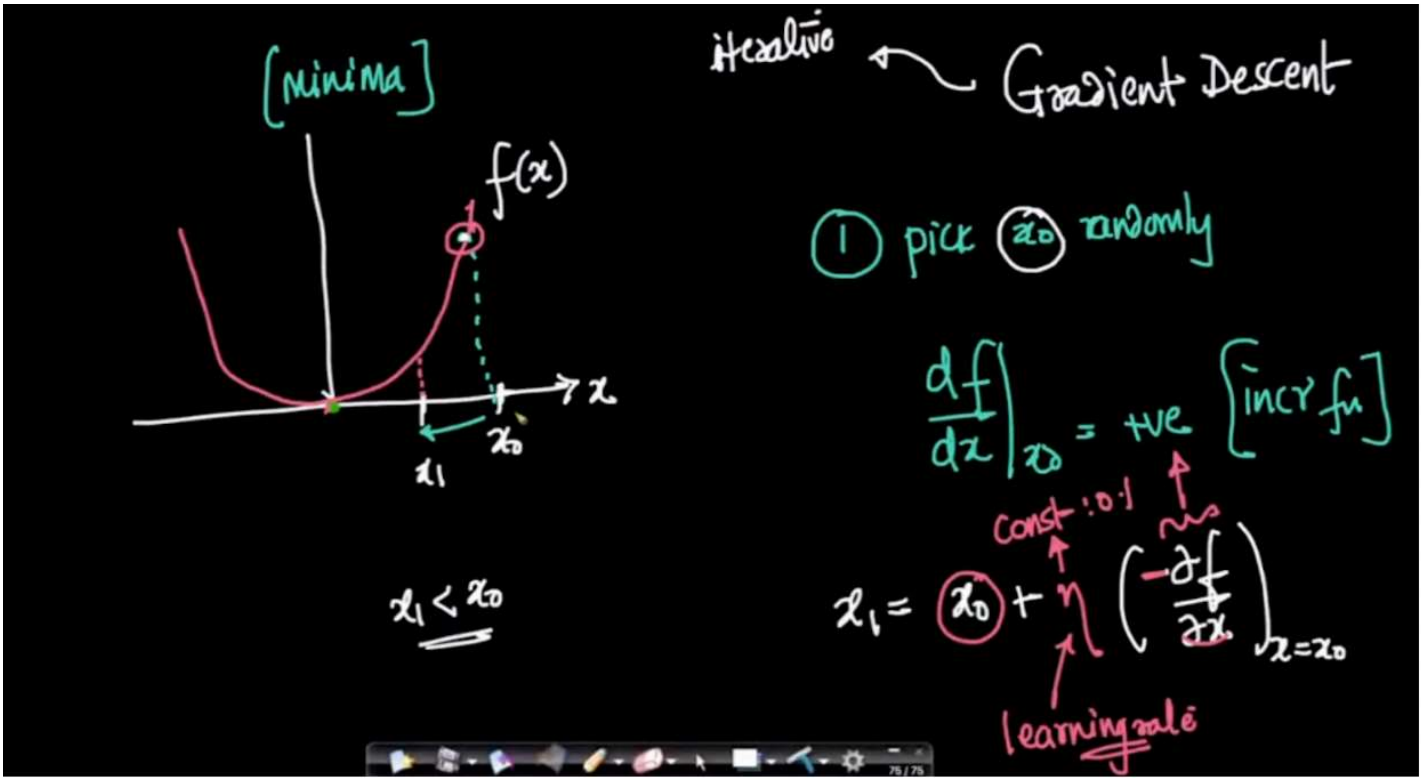
$$f(x) = x^8 + x^6 + x^4$$

$$\frac{df}{dx} = 8x^7 + 6x^5 + 4x^3 = 0$$

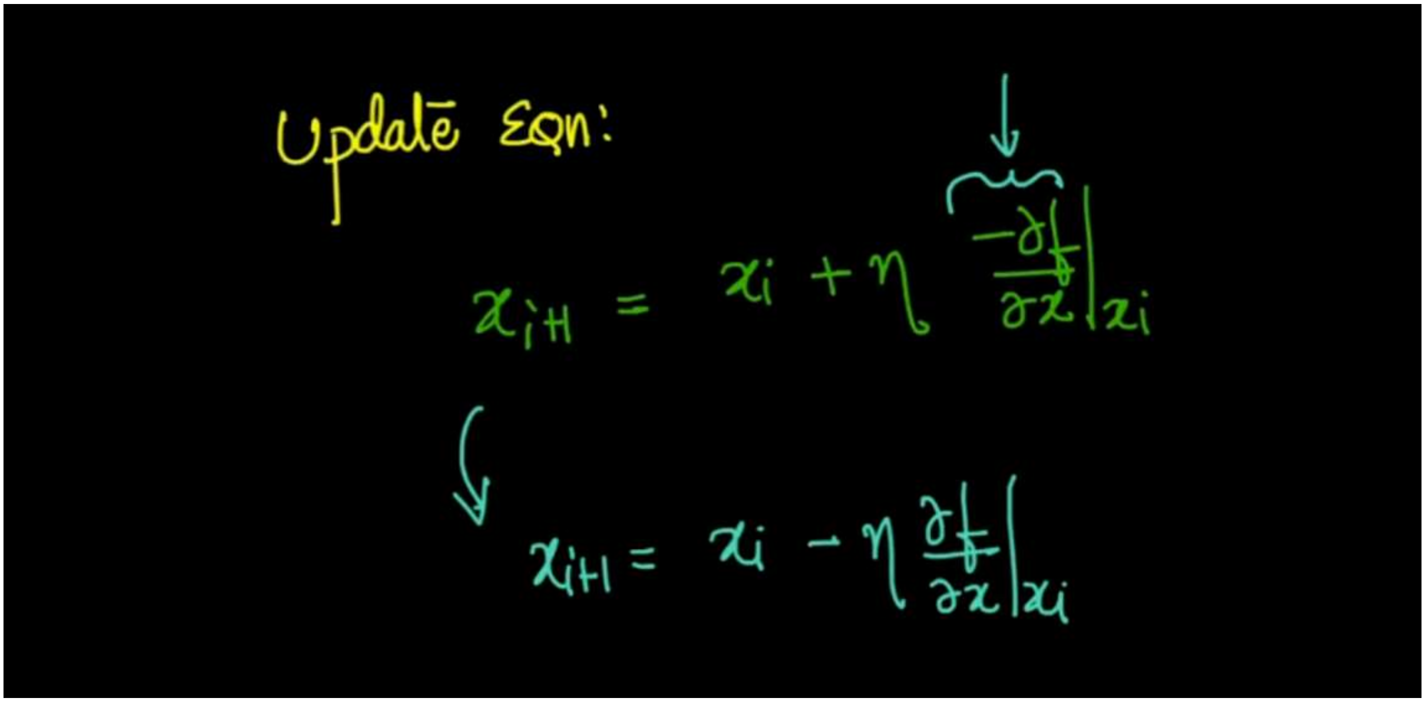
$$\boxed{8x^4 + 6x^2 + 4 = 0}$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- We will now look at the Gradient Descent algorithm. It is an iterative algorithm.
- Imagine we have a function f(x) = x2. We are trying to find its minimum.
  - Pick x0 randomly(refer the image above)
  - Compute df/dx at x = x0, at x0(refer the image) here derivative will be positive. Since the function is increasing here.
  - Our motive is to reach the minimum, so we should move in the opposite direction of the increasing function. So our new point should be x1 = x0 + (-δf/δx)
- Here is a learning rate. It controls the amount of change that you want to bring with one update. It is usually a constant and positive number.


[Minima]
iterative
Gradient Descent
f(x)
x
x_0
x_1
$x_1 < x_0$
① pick x_0 randomly
$\left.\frac{df}{dx}\right|_{x_0}$ = +ve [incr fn]
const : 0.1
$x_1 = x_0 + \eta \left(-\frac{\partial f}{\partial x}\right)_{x=x_0}$
learning rate

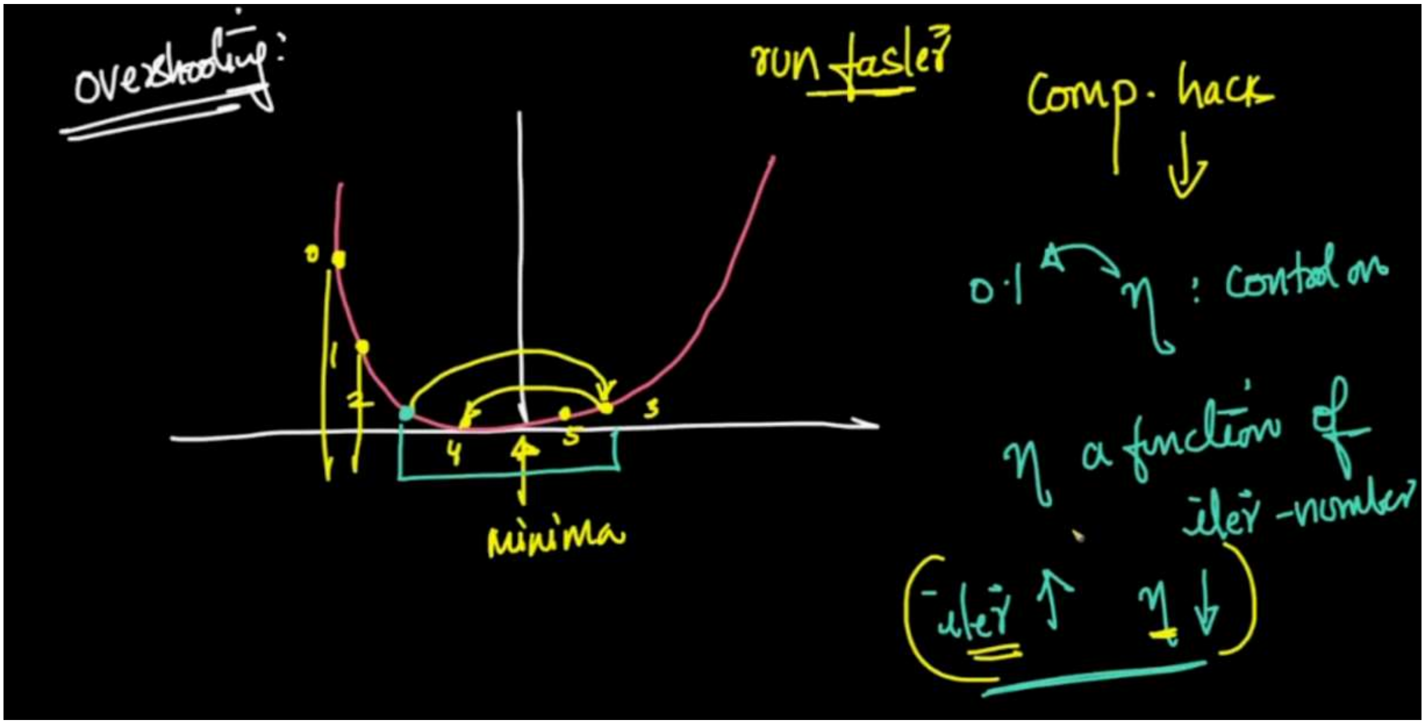
- At times what happens is we set some learning rate η(eta)/ α (alpha).
- Learning rate defines the size of a jump or step size.
    - If we set it to too small a value, then the updates will happen very slowly.
    - If we set it to a large value then it may overshoot the minima.
- Then we have to iterate back again to reach the minimum. At times it results in oscillations back and forth.
- One hack that you can do is we should decrease the value of η as iteration increases.
- This helps to reach minima more easily and not overshoot.
- Note by reducing the η(eta) we are only reducing the chances of overshooting not completely removing it.



Updatē Eqn:

$$x_{i+1} = x_i + \eta \overbrace{\frac{-\partial f}{\partial x}}\Big|_{x_i}$$

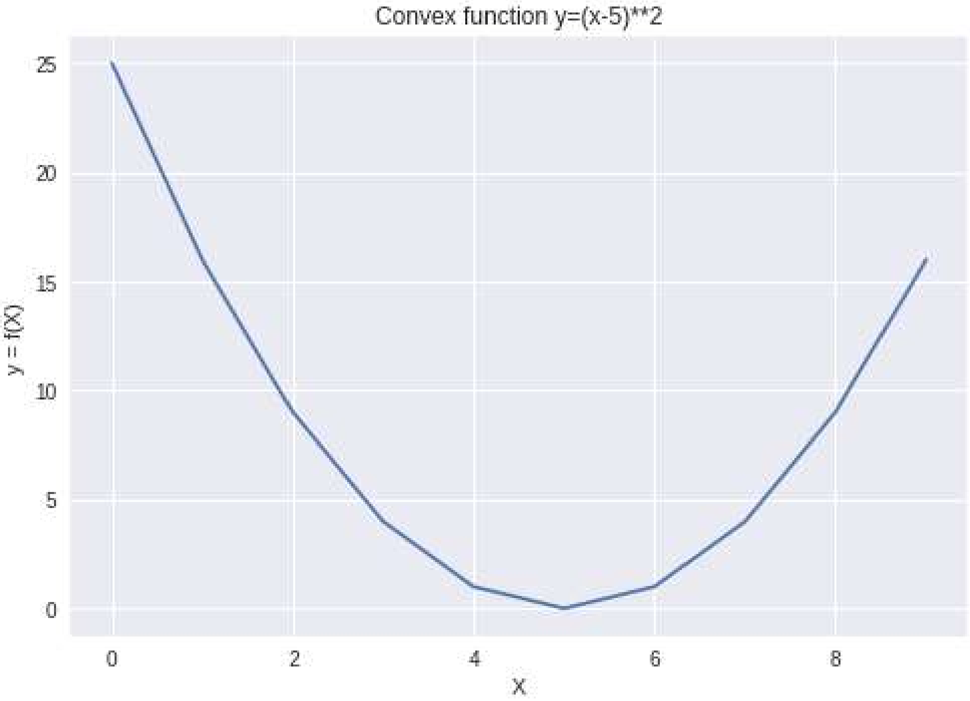
$$x_{i+1} = x_i - \eta \frac{\partial f}{\partial x}\Big|_{x_i}$$



Implementation of Gradient Descent for Univariate function (Single Variable)

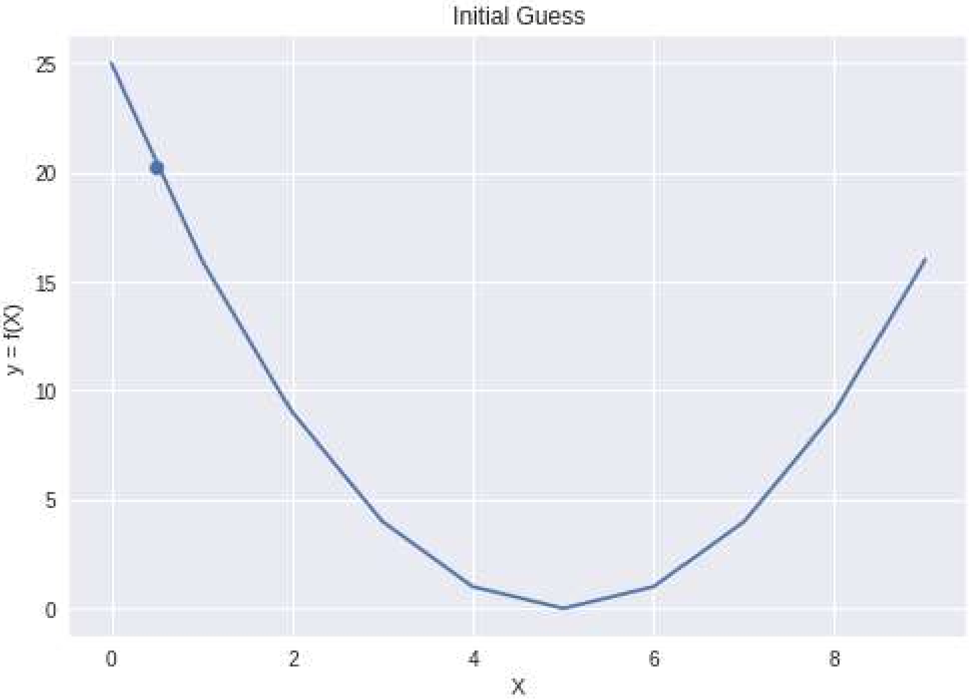
```
import numpy as np
import matplotlib.pyplot as plt
```

```
X = np.arange(10)
#Univariate function which is a Convex Function
Y = (X-5)**2
plt.style.use("seaborn")
plt.plot(X,Y)
plt.ylabel("y = f(X)")
plt.xlabel("X")
plt.title("Convex function y=(x-5)**2") # assume ground truth value is 5
plt.show()
```



Where should we start?

```
x = 0.5
y = (x-5)**2 # assume ground truth value is 5
plt.plot(X,Y)
plt.scatter(x,y)
plt.ylabel("y = f(X)")
plt.xlabel("X")
plt.title("Initial Guess")
plt.show()
```



Imagine that you are standing somewhere on this surface, you look around 360 degrees

Where would you take a step to go downhill and reach the minima as quickly as possible?

- You will take the step in the steepest downward direction.
- This can be mathematically calculated by calculating the gradient at that point.
- The gradient at a point tells you the direction of steepest ascent.
- Thus, to move in direction where the value of the function decreases the most, i.e. in the opposite direction of gradient - steepest descent.

What will be the gradient for y = (x-5)**2?

2*(x-5)

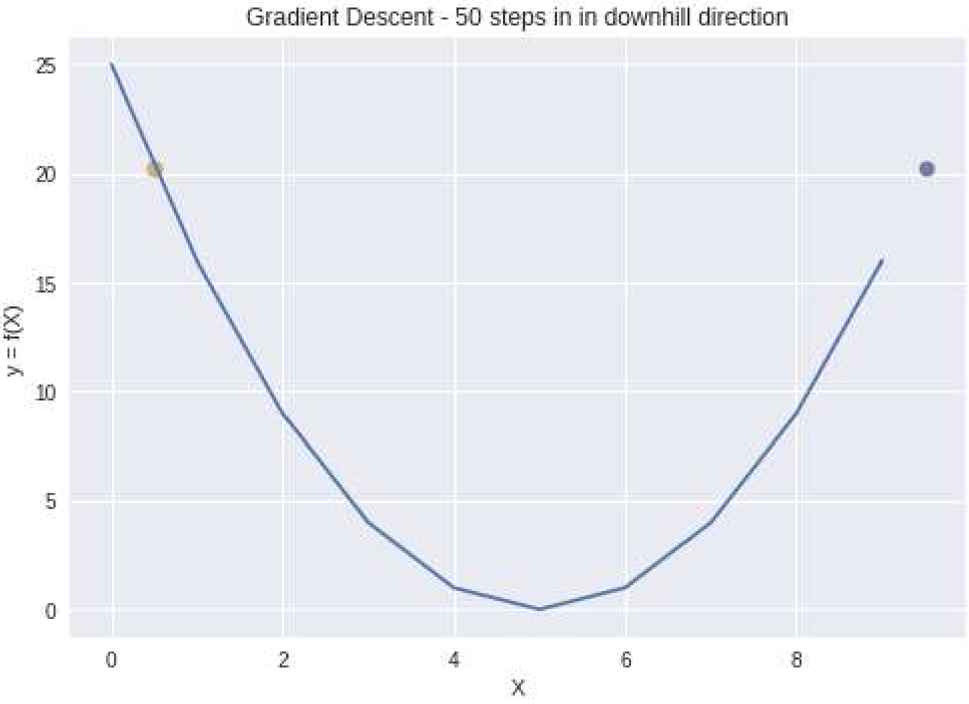
Now, Let's take small steps in the direction steepest descent iteratively and see if we can reach minima

```
import time


fig = plt.figure()
ax = fig.add_subplot(111)
plt.ion()

x = 0.5
y = (x-5)**2 # assume ground truth value is 5
plt.plot(X,Y)
plt.scatter(x,y)
plt.ylabel("y = f(X)")
plt.xlabel("X")
plt.title("Gradient Descent - 50 steps in in downhill direction")

lr = 1.0
errors = []
# 10 steps in the downhill direction
for i in range(10):
    grad = 2*(x-5)
    x = x - grad
    y = (x-5)**2
    error = y - 0
    errors.append(error)
    plt.scatter(x,y)
    fig.canvas.draw()
    time.sleep(0.5)
plt.show()
```



- Notice that we are taking very big steps due to which the process is jumping across the

Lets try smaller steps by multipying the gradient with small value

What is the importance of α/η?

- η is considered as a step size or learning rate.
- The more you keep alpha, the longer the step you take, the smaller the lesser.

Lets take lr = 0.1 and see if the process can get to the minima

```
fig = plt.figure()
ax = fig.add_subplot(111)
plt.ion()
fig.show()
fig.canvas.draw()

x = 0.5
y = (x-5)**2 # assume ground truth value is 5
plt.plot(X,Y)
plt.scatter(x,y)
plt.ylabel("y = f(X)")
plt.xlabel("X")
plt.title("Gradient Descent - 50 steps in in downhill direction")

lr = 0.1
errors = []
# 50 steps in the downhill direction
for i in range(50):
    grad = 2*(x-5)
    x = x - lr*grad
    y = (x-5)**2
    error = y - 0
    errors.append(error)
    plt.scatter(x,y)
    fig.canvas.draw()
    time.sleep(0.5)
plt.show()
```