dialzero FACEBOOK AUTHORIZED SALES PARTNER

* Mathematic for Datascience (Assignment)

Abdul Samad
23399.

Q1
(a)

Sol: Span of $X = R^n$

(b)
Sol: Yes, since X is a square matrix and has full rank. All the columns are linearly independent

(c)
Sol: Since
(i) all the columns of X are independent then y can be written in linear combination of X.
(ii) Reginalt finds the correct α minimizing $\|X\alpha - y\|_2^2$ and calls $\hat{a}$.
(iii) $\hat{a} = \alpha_0, \ldots \alpha_n$
hence
$\boxed{\|X\hat{\alpha} - y\|_2^2 = 0}$ Ans.

(d)
Yes Reginald is correct in his conclusion than independent variable in X are very useful in predicting dependent variable because y can be written as linear combination of x. and the difference between predicted y and actual y is minimized.

$$\| X\hat{\alpha} - y \|_2^2 = 0$$

$$X\hat{\alpha} = y.$$

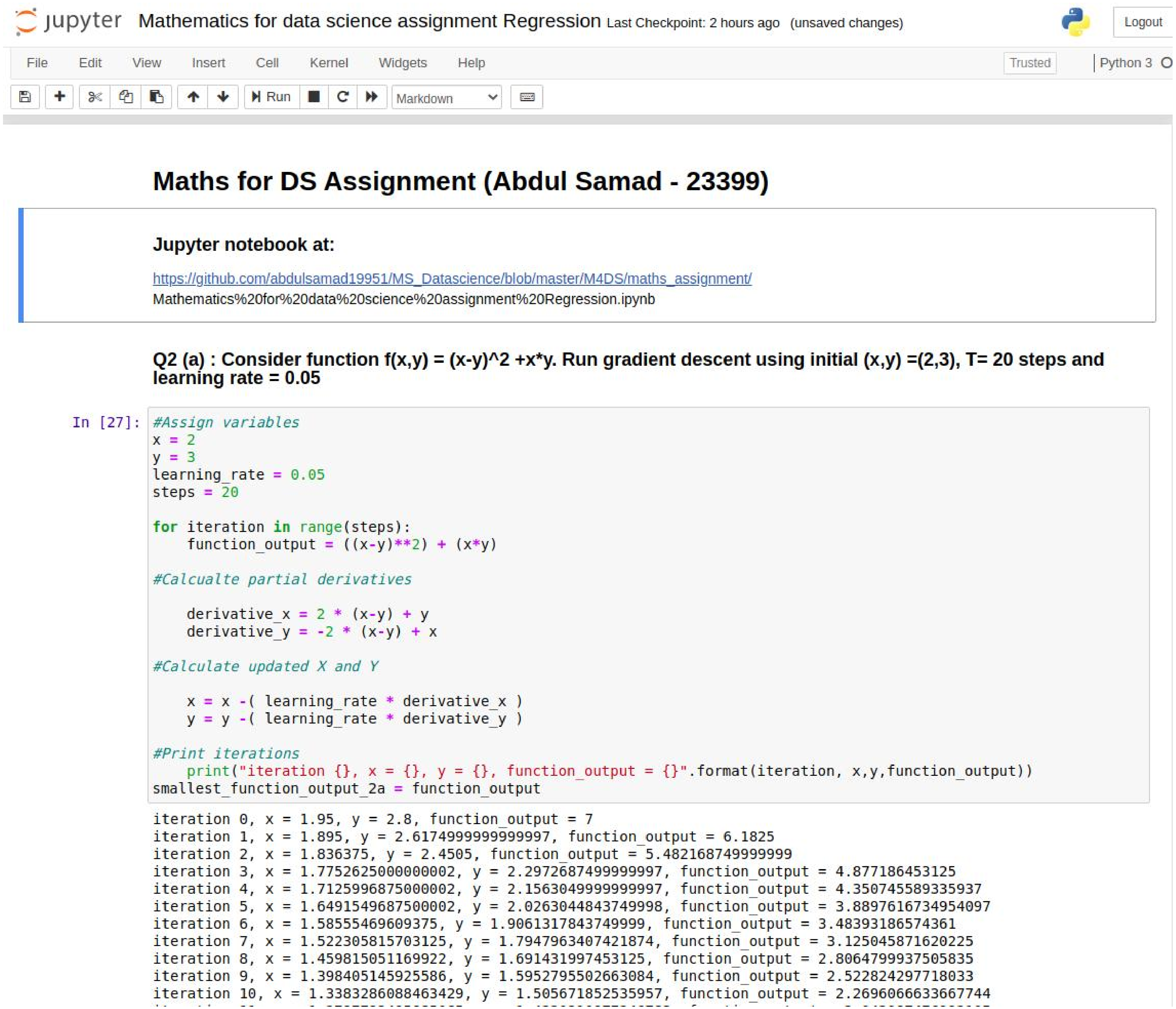
hence $\hat{\alpha}$ hold importance in prediction.

~~f~~Q2:
<u>Derivative</u>

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

$$\cancel{f}\frac{d}{dx} = 2(x-y) + y$$

$$\frac{d}{dy} = -2(x-y) + x$$


jupyter
Mathematics for data science assignment Regression Last Checkpoint: 2 hours ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Markdown

Maths for DS Assignment (Abdul Samad - 23399)

Jupyter notebook at:

https://github.com/abdulsamad19951/MS_Datascience/blob/master/M4DS/maths_assignment/
Mathematics%20for%20data%20science%20assignment%20Regression.ipynb

Q2 (a) : Consider function f(x,y) = (x-y)^2 +x*y. Run gradient descent using initial (x,y) =(2,3), T= 20 steps and learning rate = 0.05

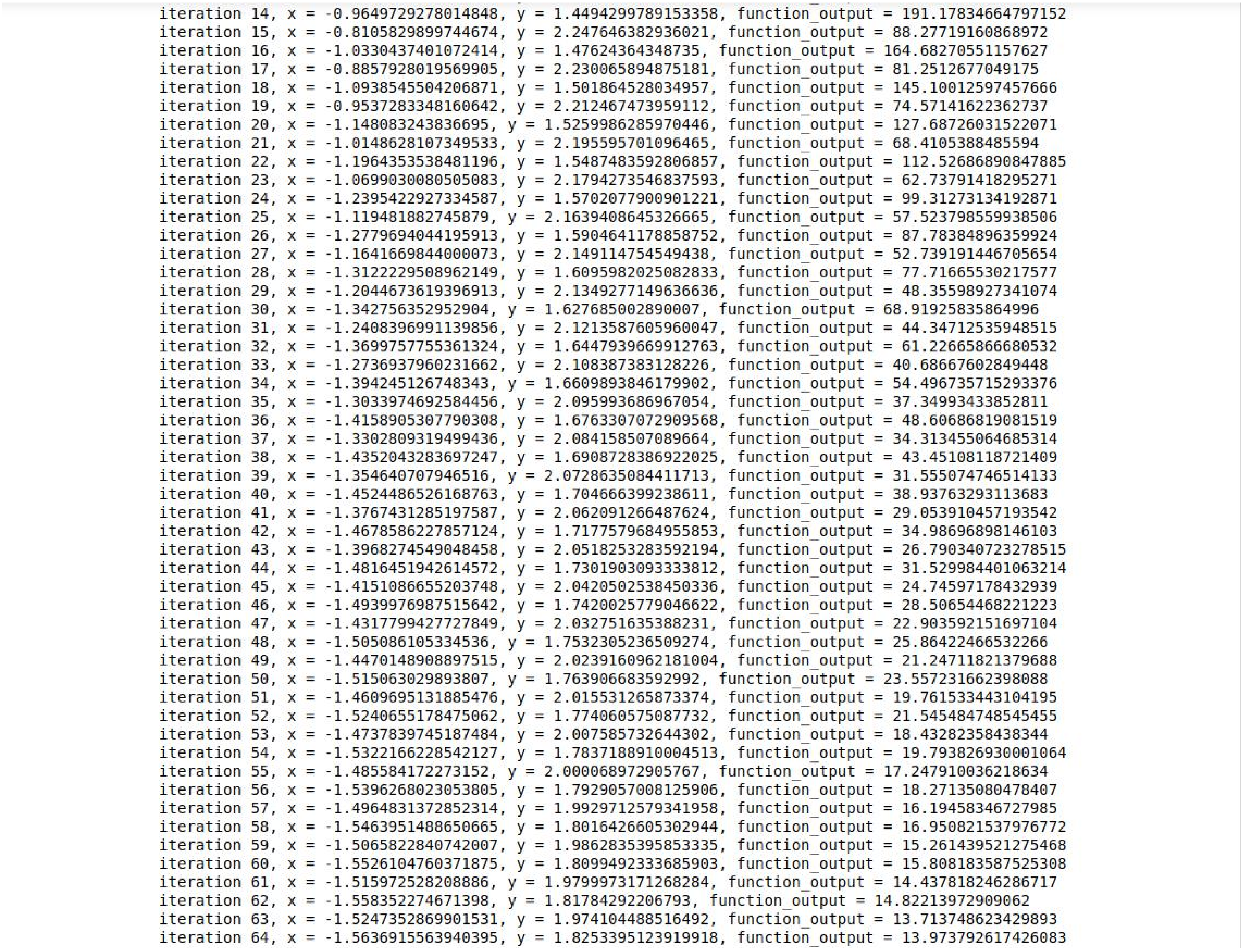
In [27]:
```
#Assign variables
x = 2
y = 3
learning_rate = 0.05
steps = 20

for iteration in range(steps):
    function_output = ((x-y)**2) + (x*y)

#Calcualte partial derivatives

    derivative_x = 2 * (x-y) + y
    derivative_y = -2 * (x-y) + x

#Calculate updated X and Y

    x = x -( learning_rate * derivative_x )
    y = y -( learning_rate * derivative_y )

#Print iterations
    print("iteration {}, x = {}, y = {}, function_output = {}".format(iteration, x,y,function_output))
smallest_function_output_2a = function_output
```

```
iteration 0, x = 1.95, y = 2.8, function_output = 7
iteration 1, x = 1.895, y = 2.6174999999999997, function_output = 6.1825
iteration 2, x = 1.836375, y = 2.4505, function_output = 5.482168749999999
iteration 3, x = 1.7752625000000002, y = 2.2972687499999997, function_output = 4.877186453125
iteration 4, x = 1.7125996875000002, y = 2.1563049999999997, function_output = 4.350745589335937
iteration 5, x = 1.6491549687500002, y = 2.0263044843749998, function_output = 3.8897616734954097
iteration 6, x = 1.58555469609375, y = 1.9061317843749999, function_output = 3.48393186574361
iteration 7, x = 1.522305815703125, y = 1.7947963407421874, function_output = 3.125045871620225
iteration 8, x = 1.459815051169922, y = 1.691431997453125, function_output = 2.8064799937505835
iteration 9, x = 1.398405145925586, y = 1.5952795502663084, function_output = 2.52282429771803
iteration 10, x = 1.3383286088463429, y = 1.505671852535957, function_output = 2.2696066633667744
iteration 11, x = 1.2797793405885065, y = 1.4220210977246783, function_output = 2.043087476992105
iteration 12, x = 1.2229024614158899, y = 1.3438079549816357, function_output = 1.8401059402222188
iteration 13, x = 1.1678026130233827, y = 1.2705722825542667, function_output = 1.6579641941916723
iteration 14, x = 1.1145509658487578, y = 1.201905184950009, function_output = 1.4943392365774435
iteration 15, x = 1.0631911285113824, y = 1.1374422147474461, function_output = 1.347215344339452
iteration 16, x = 1.0137441263976166, y = 1.0768575496982706, function_output = 1.2148316957210588
iteration 17, x = 0.9662125912427685, y = 1.0198590010483244, function_output = 1.095641320174279
iteration 18, x = 0.9205842821709078, y = 0.9661837305056303, function_output = 0.9882785453901884
iteration 19, x = 0.8768350404790985, y = 0.9155945715636127, function_output = 0.8915328656811666
```

Q2 (b) : Consider function f(x,y) = (1-(y-4)) ^2 +35((x+6)-(y-4)^2)^2. Run gradient descent using initial (x,y) =(0,2), T= 100 steps and learning rate = 0.0015

In [28]:
```python
#Assign variables
x = 0
y = 2
learning_rate = 0.0015
steps = 100

for iteration in range(steps):
    function_output = (1 - (y-4))**2  +35 *((x+6)-(y-4)**2)**2
    ((-y**2) + 35 * ((x+6)-(y-4)**2))
#Calcualte partial derivatives

    derivative_x = 70 * ( -y ** 2 + 8 * y + x -10)
    derivative_y = 140 * (y)**3 -1680 *y**2 +5882 * y -140*x*y + 560*x -5610

#Calculate updated X and Y

    x = x -( learning_rate * derivative_x )
    y = y -( learning_rate * derivative_y )

#Print iterations
    print("iteration {}, x = {}, y = {}, function_output = {}".format(iteration, x,y,function_output))
smallest_function_output_2b = function_output
```

```
iteration 0, x = -0.21, y = 1.169, function_output = 149
iteration 1, x = 0.023578904999999956, y = 2.5030167601100013, function_output = 187.88006849523507
iteration 2, x = -0.37359620387128656, y = 1.321378747266069, function_output = 507.02243992542884
iteration 3, x = -0.2109923618270225, y = 2.2035228251761843, function_output = 97.46879224981498
iteration 4, x = -0.4799684886705745, y = 1.2454933118067433, function_output = 237.49697288060761
iteration 5, x = -0.2629045523535189, y = 2.452564960573091, function_output = 163.67331237471882
iteration 6, x = -0.6138712782255527, y = 1.3740108471167247, function_output = 397.5295607383072
iteration 7, x = -0.45535379575050433, y = 2.2174191936190564, function_output = 92.91856638516774
iteration 8, x = -0.7038942424125392, y = 1.3396800763800183, function_output = 203.84546587090884
iteration 9, x = -0.5168686268782328, y = 2.345756978617289, function_output = 124.44105580352289
iteration 10, x = -0.8052628238077113, y = 1.3995715323257156, function_output = 271.08060227877746
iteration 11, x = -0.6406762646813648, y = 2.266363965226154, function_output = 98.95907069068159
iteration 12, x = -0.8878283972778513, y = 1.4176211870495314, function_output = 201.39078707217328
```



```
iteration 14, x = -0.9649729278014848, y = 1.4494299789153358, function_output = 191.17834664797152
iteration 15, x = -0.8105829899744674, y = 2.247646382936021, function_output = 88.2771916086897 2
iteration 16, x = -1.0330437401072414, y = 1.47624364348735, function_output = 164.68270551157627
iteration 17, x = -0.8857928019569905, y = 2.230065894875181, function_output = 81.2512677049175
iteration 18, x = -1.0938545504206871, y = 1.501864528034957, function_output = 145.10012597457666
iteration 19, x = -0.9537283348160642, y = 2.212467473959112, function_output = 74.5714162236273 7
iteration 20, x = -1.148083243836695, y = 1.5259986285970446, function_output = 127.68726031522071
iteration 21, x = -1.0148628107349533, y = 2.195595701096465, function_output = 68.4105388485594
iteration 22, x = -1.1964353538481196, y = 1.5487483592806857, function_output = 112.52686890847885
iteration 23, x = -1.0699030080505083, y = 2.1794273546837593, function_output = 62.73791418295271
iteration 24, x = -1.2395422927334587, y = 1.5702077900901221, function_output = 99.31273134192871
iteration 25, x = -1.119481882745879, y = 2.1639408645326665, function_output = 57.523798559938506
iteration 26, x = -1.2779694044195913, y = 1.5904641178858752, function_output = 87.78384896359924
iteration 27, x = -1.1641669844000073, y = 2.149114754549438, function_output = 52.739191446705654
iteration 28, x = -1.3122229508962149, y = 1.6095982025082833, function_output = 77.71665530217577
iteration 29, x = -1.2044673619396913, y = 2.1349277149636636, function_output = 48.35598927341074
iteration 30, x = -1.342756352952904, y = 1.627685002890007, function_output = 68.91925835864996
iteration 31, x = -1.2408396991139856, y = 2.1213587605960047, function_output = 44.34712535948515
iteration 32, x = -1.3699757755361324, y = 1.6447939669912763, function_output = 61.22665866680532
iteration 33, x = -1.2736937960231662, y = 2.108387383128226, function_output = 40.68667602849448
iteration 34, x = -1.394245126748343, y = 1.6609893846179902, function_output = 54.496735715293376
iteration 35, x = -1.3033974692584456, y = 2.095993686967054, function_output = 37.3499343385281 1
iteration 36, x = -1.4158905307790308, y = 1.6763307072909568, function_output = 48.60686819081519
iteration 37, x = -1.3302809319499436, y = 2.084158507089664, function_output = 34.313455064685314
iteration 38, x = -1.4352043283697247, y = 1.6908728386922025, function_output = 43.45108118721409
iteration 39, x = -1.354640707946516, y = 2.0728635084411713, function_output = 31.555074746514133
iteration 40, x = -1.4524486526168763, y = 1.704666399238611, function_output = 38.93763293113683
iteration 41, x = -1.3767431285197587, y = 2.062091266487624, function_output = 29.053910457193542
iteration 42, x = -1.4678586227857124, y = 1.7177579684955853, function_output = 34.98696898146103
iteration 43, x = -1.3968274549048458, y = 2.0518253283592194, function_output = 26.790340723278515
iteration 44, x = -1.4816451942614572, y = 1.7301903093333812, function_output = 31.529984401063214
iteration 45, x = -1.4151086655203748, y = 2.0420502538450336, function_output = 24.74597178432939
iteration 46, x = -1.4939976987515642, y = 1.7420025779046622, function_output = 28.50654468221223
iteration 47, x = -1.4317799427727849, y = 2.032751635388231, function_output = 22.903592151697104
iteration 48, x = -1.505086105334536, y = 1.7532305236509274, function_output = 25.86422466532266
iteration 49, x = -1.4470148908897515, y = 2.0239160962181004, function_output = 21.24711821379688
iteration 50, x = -1.515063029893807, y = 1.763906683592992, function_output = 23.557231662398088
iteration 51, x = -1.4609695131885476, y = 2.015531265873374, function_output = 19.761533443104195
iteration 52, x = -1.5240655178475062, y = 1.774060575087732, function_output = 21.545484748545455
iteration 53, x = -1.4737839745187484, y = 2.007585732644302, function_output = 18.43282358438344
iteration 54, x = -1.5322166228542127, y = 1.7837188910004513, function_output = 19.793826930001064
iteration 55, x = -1.485584172273152, y = 2.000068972905767, function_output = 17.247910036218634
iteration 56, x = -1.5396268023053805, y = 1.7929057008125906, function_output = 18.27135080478407
iteration 57, x = -1.4964831372852314, y = 1.9929712579341958, function_output = 16.19458346727985
iteration 58, x = -1.5463951488650665, y = 1.8016426605302944, function_output = 16.950821537976772
iteration 59, x = -1.5065822840742007, y = 1.9862835395853335, function_output = 15.261439521275468
iteration 60, x = -1.5526104760371875, y = 1.8099492333685903, function_output = 15.808183587525308
iteration 61, x = -1.515972528208886, y = 1.9799973171268284, function_output = 14.437818246286717
iteration 62, x = -1.558352274671398, y = 1.81784292206793, function_output = 14.82213972909062
iteration 63, x = -1.5247352869901531, y = 1.974104488516492, function_output = 13.713748623429893
iteration 64, x = -1.5636915563940395, y = 1.8253395123919918, function_output = 13.973792617426083
iteration 88, x = -1.6133644934389981, y = 1.8888154741340952, function_output = 9.90832693396303
iteration 89, x = -1.6059657108910605, y = 1.9293894181625986, function_output = 9.85325322761179
iteration 90, x = -1.617159352177712, y = 1.8922459057132461, function_output = 9.826419520702055
iteration 91, x = -1.610881751390882, y = 1.9280324455175852, function_output = 9.783241134806268
iteration 92, x = -1.6209689650779164, y = 1.8954475892316984, function_output = 9.76000649696059
iteration 93, x = -1.615707434529315, y = 1.9269075800647846, function_output = 9.726130444902092
iteration 94, x = -1.6247983748364888, y = 1.8984341384428234, function_output = 9.70626272199846
iteration 95, x = -1.6204537430800883, y = 1.9259998955880677, function_output = 9.679634047858555
iteration 96, x = -1.628651074581105, y = 1.901219363123289, function_output = 9.662798050615235
iteration 97, x = -1.6251302947685844, y = 1.9252943940283676, function_output = 9.641793468778072
iteration 98, x = -1.6325292619156204, y = 1.9038171536095687, function_output = 9.627607308926464
iteration 99, x = -1.6297455242368237, y = 1.9247761484910293, function_output = 9.610948838804106
```

Q2 (c) : Consider function f(x,y) = (x-y)^2 +x*y. Run gradient descent using initial (x,y) =(2,3), T= 20 steps and learning rate = 0.05

For C and D used Batch gradient descent

In [31]:
```python
#Assign variables
x = 2
y = 3
learning_rate = 0.005
steps = 200

for iteration in range(steps):
    function_output = ((x-y)**2) + (x*y)

#Calcualte partial derivatives

    derivative_x = 2 * (x-y) + y
    derivative_y = -2 * (x-y) + x

#Calculate updated X and Y

    x = x -( learning_rate * derivative_x )
    y = y -( learning_rate * derivative_y )

#Print iterations
    if( function_output <= smallest_function_output_2a):
        print("iteration {}, x = {}, y = {}, function_output = {}".format(iteration, x,y,function_output))
```

```
iteration 195, x = 0.910123474767553, y = 0.9618247407453551, function_output = 0.8869404394831466
iteration 196, x = 0.9058313637236042, y = 0.9567571107117393, function_output = 0.8780522960682708
iteration 197, x = 0.9015568356399268, y = 0.9517186964232399, function_output = 0.8692540300545697
iteration 198, x = 0.8972998607656437, y = 0.9467092936372071, function_output = 0.8605447086439368
iteration 199, x = 0.8930604086261733, y = 0.9417287000046632, function_output = 0.8519234094228963
```

Q2 (d) : Consider function f(x,y) = (1-(y-4)) ^2 +35((x+6)-(y-4)^2)^2. Run gradient descent of any variant, try to get smallest function value after T = 100 steps

In [22]:
```
#Assign variables
x = 0
y = 2
learning_rate = 0.0015
steps = 110 # 10 further steps allowed

for iteration in range(steps):
    function_output = (1 - (y-4))**2  +35 *((x+6)-(y-4)**2)**2
    ((-y**2) + 35 * ((x+6)-(y-4)**2))
#Calcualte partial derivatives

    derivative_x = 70 * ( -y ** 2 + 8 * y + x -10)
    derivative_y = 140 * (y)**3 -1680 *y**2 +5882 * y -140*x*y + 560*x -5610

#Calculate updated X and Y

    x = x -( learning_rate * derivative_x )
    y = y -( learning_rate * derivative_y )

#Print iterations
    if( function_output <= smallest_function_output_2b):
        print("iteration {}, x = {}, y = {}, function_output = {}".format(iteration, x,y,function_output))
```

```
iteration 99, x = -1.6297455242368237, y = 1.9247761484910293, function_output = 9.610948838804106
iteration 100, x = -1.636434070635426, y = 1.9062413580089488, function_output = 9.59902285729048
iteration 101, x = -1.634306841872924, y = 1.9244304411448816, function_output = 9.585707938729282
iteration 102, x = -1.6403657791434338, y = 1.90850565830608, function_output = 9.57567008230221
iteration 103, x = -1.6388207712929166, y = 1.9242428916856187, function_output = 9.564915198878415
iteration 104, x = -1.6443239951718036, y = 1.910623450840071, function_output = 9.556426148270743
iteration 105, x = -1.6432930674399215, y = 1.924199572941296, function_output = 9.547621468811574
iteration 106, x = -1.6483078169961345, y = 1.9126077356817126, function_output = 9.540382276972403
iteration 107, x = -1.6477288173722786, y = 1.9242871111464281, function_output = 9.533055249532485
iteration 108, x = -1.652315971868141, y = 1.9144710183925429, function_output = 9.526809731386685
iteration 109, x = -1.6521325258438992, y = 1.924492769337599, function_output = 9.520595929494505
```

In [ ]: