In [54]:

```
from sklearn.datasets import load_boston
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
```

In [55]:

```
DF_Train = pd.read_csv('Train.csv')
DF_Test = pd.read_csv('Test.csv')

print(DF_Train.shape)
print(DF_Test.shape)
print(DF_Train.columns)
print(DF_Test.columns)
DF_Train.head(n=5)
```

```
(1600, 6)
(400, 5)
Index(['feature_1', 'feature_2', 'feature_3', 'feature_4', 'feature_5',
       'target'],
      dtype='object')
Index(['feature_1', 'feature_2', 'feature_3', 'feature_4', 'feature_5'], dtype='object')
```

	feature_1	feature_2	feature_3	feature_4	feature_5	target
0	0.293416	-0.945599	-0.421105	0.406816	0.525662	-82.154667
1	-0.836084	-0.189228	-0.776403	-1.053831	0.597997	-48.897960
2	0.236425	0.132836	-0.147723	0.699854	-0.187364	77.270371
3	0.175312	0.143194	-0.581111	-0.122107	-1.292168	-2.988581
4	-1.693011	0.542712	-2.798729	-0.686723	1.244077	-37.596722

In [56]:

```
DF_Train = DF_Train.values
DF_Test = DF_Test.values

X_Train = DF_Train[:,:-1]
Y_Train = DF_Train[:,-1]
X_Test = DF_Test[:,:]

print(X_Train.shape)
print(Y_Train.shape)
print(X_Test.shape)
print(Y_Train)
```

```
(1600, 5)
(1600,)
(400, 5)
[ -82.15466656  -48.89796018   77.2703707  ... -107.51050797  -47.34155781
 -115.93900296]
```

In [57]:

```
# Normalising Data
u = np.mean(X_Train, axis = 0)
std = np.std(X_Train, axis = 0)
print(std)

X_Train = (X_Train - u)/std
print(X_Train)
```

```
[0.99702582 1.02145636 1.01145129 1.01687155 0.97834768]
[[ 0.29016495 -0.89871183 -0.37238147  0.44177059  0.52502448]
 [-0.84270473 -0.15822922 -0.72365639 -0.99464217  0.59896038]
 [ 0.23300381  0.15706968 -0.10209444  0.72994655 -0.20378187]
 ...
 [ 1.0431652  -0.8532941   1.75476416 -1.79830858  0.44004223]
 [-1.27708547  0.02207793  1.88059294 -1.0207355   0.74035908]
 [-1.89374689 -0.80456069 -1.39187219  0.52221049  1.47960738]]
```

In [58]:

```
### Linear Regression
# theta - (13,)
# X = (506,13)
# m - 506, n-13
# Hypothesis Fn - x is a vector, o/p- value
ones = np.ones((X_Train.shape[0],1))
X_Train = np.hstack((ones,X_Train))

ones = np.ones((X_Test.shape[0],1))
X_Test = np.hstack((ones,X_Test))
print(X_Train)
def hypothesis(X, theta):
    return np.dot(X, theta)
```

```
[[ 1.          0.29016495 -0.89871183 -0.37238147  0.44177059  0.52502448]
 [ 1.         -0.84270473 -0.15822922 -0.72365639 -0.99464217  0.59896038]
 [ 1.          0.23300381  0.15706968 -0.10209444  0.72994655 -0.20378187]
 ...
 [ 1.          1.0431652  -0.8532941   1.75476416 -1.79830858  0.44004223]
 [ 1.         -1.27708547  0.02207793  1.88059294 -1.0207355   0.74035908]
 [ 1.         -1.89374689 -0.80456069 -1.39187219  0.52221049  1.47960738]]
```

In [59]:

```
# Error Fn- o/p = value
def error(X,y,theta):
    e = 0.0
    m = X.shape[0]
    y_ = hypothesis(X,theta)
    e = np.sum((y-y_)**2)
    return e/m
```

In [60]:

```
# Gradient Fn- o/p = (n,)
def gradient(X,y,theta):
    y_ = hypothesis(X,theta)
    grad = np.dot(X.T,(y_-y))
    m = X.shape[0]
    return grad/m
```

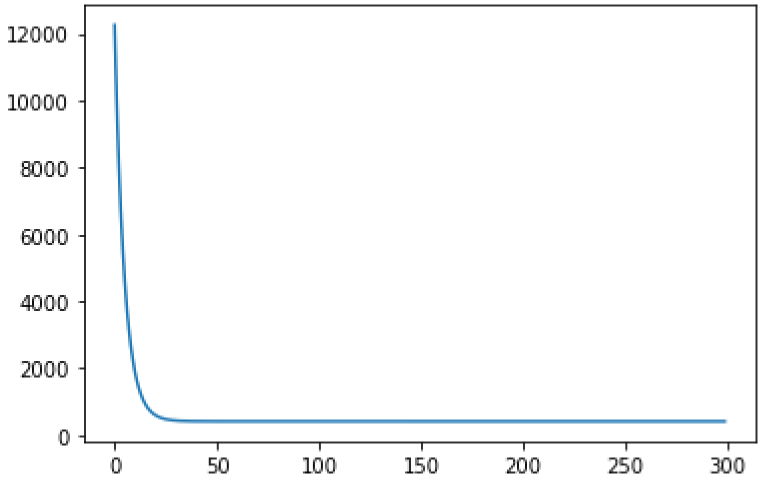
In [77]:

```
# Gradient Descent- o/p = (n,)
def gradient_descent(X,y,learning_rate=0.1, max_epochs=300):

    n = X.shape[1]
    theta = np.zeros((n,))
    error_list = []

    for i in range(max_epochs):
        e = error(X,y,theta)
        error_list.append(e)
        #print(i)
        grad = gradient(X,y,theta)
        theta = theta - learning_rate * grad
    return theta, error_list
```

In [78]:

```
start = time.time()
theta, error_list  = gradient_descent(X_Train,Y_Train)
end = time.time()
print("Time taken: ",end-start)
print(theta)
print(error_list)

plt.plot(error_list)
plt.show()
```

```
Time taken:  0.05884408950805664
[ 0.31883538 29.59359198 94.65067706  8.37544469 45.52303635  2.46461552]
[12256.130414032896, 10026.033684143073, 8217.055252817618, 6749.466627013954, 5558.673187089287, 4592.3340674
171.3849608951386, 2654.5237799458373, 2234.85183834815, 1894.04653113265, 1617.2484830009762, 1392.4052530232
1.3194764678283, 940.7077762355079, 842.6808858642628, 762.9988586786474, 698.2197865343779, 645.5492057333448
59297998, 539.5485154072414, 516.497578530278, 497.74268728982713, 482.48114259194176, 470.06060992121985, 459
96338, 445.0204021068504, 439.5643175010901, 435.12098512481214, 431.50194759201077, 428.5539050830669, 426.15
029, 422.60042706729115, 421.3006895139444, 420.24125320690837, 419.37758077512547, 418.67340992065016, 418.09
12, 417.2490021205568, 416.9374455012189, 416.68326851820393, 416.47587806958705, 416.30664146076737, 416.1685
5, 415.96375709353555, 415.8886222169001, 415.8272731038045, 415.7771744099245, 415.736258177231, 415.7028374
15.65323071970977, 415.6350052789468, 415.62011171830306, 415.60793953738414, 415.5979903566829, 415.58985727
5.5777713515323, 415.5733255770358, 415.56968971829224, 415.56671590459604, 415.5642833232386, 415.5622932570
55933272135326, 415.5582424265418, 415.55735009351656, 415.5566197034253, 415.5560218051132, 415.555532314924,
0335898517, 415.55453460443124, 415.5543144913185, 415.55413419847406, 415.55398650775345, 415.55386551211285,
68514503823, 415.55361857688376, 415.55356402036006, 415.55351930395364, 415.55348264949816, 415.5534526007786
40776560973, 415.553391201965, 415.5533776185551, 415.5533664781947, 415.55335734070616, 415.5533498453675, 41
5191494, 415.5533345128543, 415.5533311165242, 415.55332832942133, 415.55332604208127, 415.5533241647418, 415.
88549, 415.5533203204151, 415.5533194678509, 415.5533187678391, 415.55331819304047, 415.55331772102295, 415.55
677, 415.55331675350635, 415.5533165387044, 415.5533163622492, 415.55331621728493, 415.55331609818387, 415.553
444, 415.55331585383675, 415.55331579953366, 415.55331575490425, 415.55331571822285, 415.55331568807213, 415.5
137, 415.55331562616374, 415.5533156123926, 415.5533156010699, 415.55331559175977, 415.553315584104, 415.55331
5, 415.5533155683725, 415.55331556487016, 415.5533155619894, 415.5533155596197, 415.55331555767043, 415.553315
415.5533155536621, 415.553315552769, 415.5533155520342, 415.5533155514294, 415.55331555093187, 415.5533155505
5.55331554990806, 415.55331554967984, 415.55331554949197, 415.5533155493373, 415.55331554920997, 415.55331554
15.55331554894786, 415.5533155488894, 415.5533155488412, 415.5533155488016, 415.55331554876886, 415.5533155487
5533155487017, 415.55331554868667, 415.5533155486744, 415.5533155486643, 415.5533155486558, 415.5533155486488,
1554863847, 415.55331554863454, 415.55331554863153, 415.5533155486289, 415.5533155486267, 415.55331554862494, 
155486222, 415.55331554862124, 415.55331554862045, 415.5533155486197, 415.5533155486192, 415.5533155486187, 41
4861806, 415.55331554861783, 415.55331554861755, 415.5533155486175, 415.55331554861726, 415.55331554861715, 41
48617, 415.5533155486168, 415.5533155486168, 415.5533155486168, 415.5533155486168, 415.55331554861664, 415.553
675, 415.55331554861664, 415.55331554861664, 415.5533155486166, 415.55331554861664, 415.5533155486166, 415.553
664, 415.55331554861664, 415.55331554861664, 415.5533155486166, 415.55331554861664, 415.5533155486166, 415.553
64, 415.5533155486165, 415.55331554861664, 415.5533155486166, 415.55331554861664, 415.5533155486166, 415.55331
5, 415.5533155486166, 415.5533155486166, 415.5533155486165, 415.5533155486166, 415.5533155486165, 415.5533155
5.5533155486165, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166
331554861664, 415.5533155486165, 415.5533155486165, 415.5533155486166, 415.5533155486166, 415.5533155486166, 4
5486166, 415.5533155486166, 415.5533155486165, 415.5533155486165, 415.55331554861664, 415.55331554861664, 415.
61664, 415.5533155486166, 415.5533155486165, 415.55331554861664, 415.55331554861664, 415.5533155486166, 415.55
66, 415.5533155486166, 415.5533155486165, 415.5533155486165, 415.5533155486166, 415.55331554861664, 415.553315
415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486165, 415.5533155486
533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 4
5486165, 415.5533155486165, 415.5533155486165, 415.5533155486166, 415.5533155486166, 415.55331554861664, 415.5
166, 415.55331554861664, 415.5533155486166, 415.55331554861664, 415.5533155486166, 415.5533155486166, 415.5533
6, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155486166, 415.5533155
5.5533155486166]
```



In [80]:

```
n = X_Test.shape[0]
Y_Pred = []

for i in range(n):
    p = int(hypothesis(X_Test[i],theta))
    Y_Pred.append(p)

print(len(Y_Pred))
print(Y_Pred)
```

```
400
[112, 115, -25, -47, -102, -50, -81, 20, 172, 170, -111, -25, -8, 120, 35, 41, -199, 17, 8, 133, 61, -68, -114
-31, 107, -51, 215, -20, -238, 152, -8, 9, -318, 73, -88, -214, -248, 132, -80, 101, -9, 14, -104, -33, 7, -18
105, -79, 38, 36, -56, 162, 38, 52, 19, 78, -7, -8, -3, -27, 160, -46, 18, -72, -76, 33, -168, -6, 149, -43, -
6, -96, 74, -96, -54, 122, -171, -123, 46, 94, -224, -128, -181, -57, 125, -90, -7, -5, -4, 77, 40, 89, -34, -
-149, 2, 17, -27, -25, -265, 266, 154, 10, 81, -16, -160, 109, -37, -224, 118, -34, -92, 168, 34, -23, 56, 41,
4, -160, -19, 131, 32, 53, -9, -48, 14, -74, 24, 140, 347, 215, 23, -43, -179, 8, 299, 71, -27, 154, -163, 140
172, -157, -99, -176, -23, -113, -18, -64, -81, 63, -75, 86, -62, -115, -83, 185, -46, 218, -62, -89, 44, -5,
22, 21, 50, 148, 42, 89, 170, -280, -181, -111, -88, 25, -56, 74, -25, -57, -34, 61, 88, -89, -155, 44, 238, -
14, -113, 43, 96, -96, -181, 166, 106, 57, -13, -30, -52, -178, -70, -33, -179, -32, 30, -127, 13, -6, -264, -
90, 17, -317, 186, -136, -48, -92, -94, 135, 161, -45, -149, 110, -23, 85, 103, 20, -56, -32, 25, 72, 23, -135
-139, 205, 45, 89, 27, 205, -45, -35, -45, 48, -68, 91, -90, -101, -30, -95, -27, 55, -38, -81, -187, -172, 37
6, 180, -63, 125, 150, -38, -209, -105, 0, -49, -42, 157, 131, 20, -94, 185, -71, 61, 76, 96, -111, 111, -40,
123, 27, -183, -38, -52, -123, 8, -96, -44, -171, -226, 0, -214, 184, -14, 12, 29, 240, -33, 130, 53, 6, -92,
00, 3, -19, -67, -22, -181, -130, 41, -43, -165, 85, -5, 130, -52, -2, 172, 168, 42, -32, 56, 108, -42, -75, -
```

In [82]:

```
# Saving File
df = pd.DataFrame(data=Y_Pred,columns=["target"])
df.to_csv("Pred.csv")
```

In [ ]:

In [ ]: