Implement Gradient Descent Algorithm to find the local minima of a function. For example, find the local minima of the function $y=(x+3)^2$ starting from the point x=2

In [33]:
```
x=2  #start
lr=0.01 #learning rate
precision = 0.000001
previous_step_size = 1
max_iter = 10000
iters =0
gf = lambda x: (x + 3) ** 2
```

In [34]:
```
import matplotlib.pyplot as plt
```

In [37]:
```
gd=[]
```

In [38]:
```
while precision < previous_step_size and iters < max_iter:
    prev = x
    x = x- lr * gf(prev)
    previous_step_size = abs(x - prev)
    iters += 1
    print('Iteration',iters,'Value',x)
    gd.append(x)
```

```
Iteration 265 Value -2.6524039815480225
Iteration 266 Value -2.653612211468459
Iteration 267 Value -2.654812056468897
Iteration 268 Value -2.656003603632489
Iteration 269 Value -2.657186938839627
Iteration 270 Value -2.6583621467886487
Iteration 271 Value -2.6595293110161173
Iteration 272 Value -2.660688513916689
Iteration 273 Value -2.66183983676257
Iteration 274 Value -2.6629833597225776
Iteration 275 Value -2.664119161880816
Iteration 276 Value -2.6652473212549728
Iteration 277 Value -2.6663679148142423
Iteration 278 Value -2.6674810184968964
Iteration 279 Value -2.668586707227495
Iteration 280 Value -2.6696850549337583
Iteration 281 Value -2.6707761345630994
Iteration 282 Value -2.6718600180988314
Iteration 283 Value -2.6729367765760523
Iteration 284 Value -2.674006480097217
```

In [39]:
```
print('Local Minima:',x)
```

```
Local Minima: -2.990001240409911
```

In [44]:
```
plt.plot(range(len(gd)), gd)
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x1dd9c056cd0>]
```