In [1]:
```
import time
import random
import math
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# Customization section:
initial_temperature = 100
cooling = 0.1  # cooling coefficient
number_variables = 3
upper_bounds = [10,10,10]
lower_bounds = [-10, -10,-10]
computing_time = 1 # second(s)
pi = 3.14
```

In [3]:
```
def objective_function(X):
    x=X[0]
    y=X[1]
    z = X[2]
    value = (x)**4 + 4*(x)**3 + 4*(x)**2 + z**2 - 2*z - 20*math.exp(-0.2*y) -
math.exp(math.cos(pi*y)) + 21
    return value
```

In [4]:
```
# Simulated Annealing Algorithm:
initial_solution=np.zeros((number_variables))
for v in range(number_variables):
    initial_solution[v] = random.uniform(lower_bounds[v],upper_bounds[v])
```

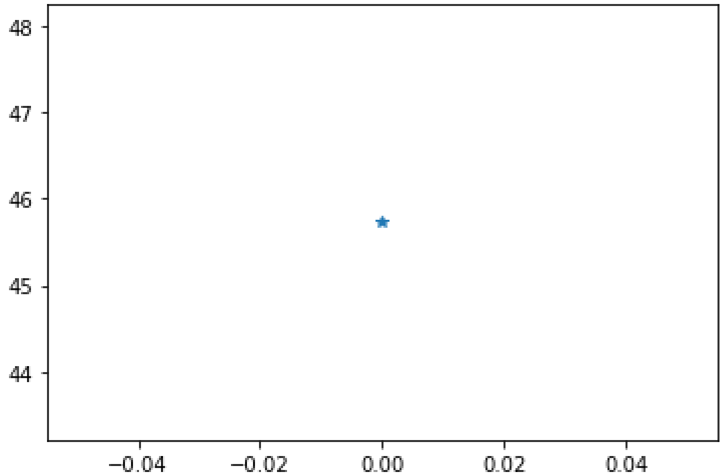
In [5]:
```
current_solution = initial_solution
best_solution = initial_solution
n = 1  # no of solutions accepted
best_fitness = objective_function(best_solution)
current_temperature = initial_temperature # current temperature
start = time.time()
no_attempts = 100 # number of attempts in each level of temperature
record_best_fitness =[]
```

In [6]:
```
for i in range(9999999):
    for j in range(no_attempts):

        for k in range(number_variables):
            current_solution[k] = best_solution[k] + 0.1*(random.uniform(lower
_bounds[k],upper_bounds[k]))
            current_solution[k] = max(min(current_solution[k],upper_bounds[k
]),lower_bounds[k])  # repair the solution respecting the bounds

        current_fitness = objective_function(current_solution)
        E = abs(current_fitness - best_fitness)
        if i == 0 and j == 0:
            EA = E

        if current_fitness < best_fitness:
            p = math.exp(-E/(EA*current_temperature))
            # make a decision to accept the worse solution or not
            if random.random()<p:
                accept = True # this worse solution is accepted
            else:
                accept = False # this worse solution is not accepted
        else:
            accept = True # accept better solution
        if accept==True:
            best_solution = current_solution # update the best solution
            best_fitness = objective_function(best_solution)
            n = n + 1 # count the solutions accepted
            EA = (EA *(n-1) + E)/n # update EA

    print('interation: {}, best_solution: {}, best_fitness: {}'.format(i, best
_solution, best_fitness))
    record_best_fitness.append(best_fitness)
    # Cooling the temperature
    current_temperature = current_temperature*cooling
    # Stop by computing time
    end = time.time()
    if end-start >= computing_time:
        break
```

```
interation: 0, best_solution: [-0.07576368 10.         -4.57972042], best_fit
ness: 45.72989105378121
```

In [7]:
```
plt.plot(record_best_fitness ,'*')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x27d2db061f0>]
```



In [ ]: