Team: Sereth Macwan and Sagar Pathak

```
import random


def f(x): # function for maximizing the value
    return 2-x**2


def hill_climbing(step_size, bound):
    x = random.uniform(bound[0], 0)
    #chooses a random point between (-5,0) so that it definitely
    #generates a closer maxima
    maxima = f(x) #maxima of a random point
    while True:
        x_new = x + step_size #new point
        if x_new > bound[1]:
          #if the new point is outside the bound, then we stop
            return maxima
        if x_new < bound[0]:
            return maxima
        if f(x_new) > maxima:
          #if the new point is a better maxima,
          #then we update the maxima and the point
            maxima = f(x_new) #update maxima
            x = x_new #update point
        else:
            return maxima


def main():

    step_size = 0.5 #definign a step size
    bound = [-5, 5]
    print('Step size: ', step_size)
    print('Maxima: ', hill_climbing(step_size, bound)) #printing the maxima

    print()
    step_size = 0.01
    print('Step size: ', step_size)
    print('Maxima: ', hill_climbing(step_size, bound))


if __name__ == '__main__':
    main()
```

```
Step size:  0.5
Maxima:  1.9992111055766975

Step size:  0.01
```

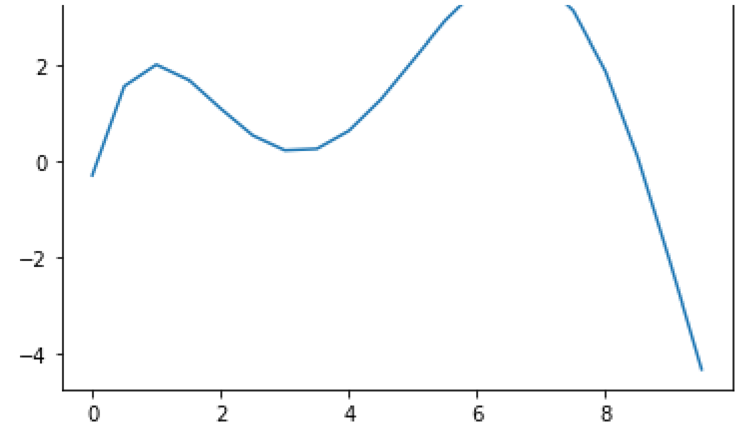
```
Step size:  0.01
Maxima:  1.9999968832170194
```

```
import random  # for using Random function
import numpy as np  # for using numpy
import matplotlib.pyplot as plt  # for plotting


def g(x):
    """
    g(x) = (0.0051x^5 ) - (0.1367x^4) + (1.24x^3) - (4.456x^2) + (5.66x) - 0.287
    :return: g(x)
    """
    # Define the function and return
    return (0.0051 * x ** 5) - (0.1367 * x ** 4) + (1.24 * x ** 3) - (4.456 * x **


def g_prime(x):
    """
    g'(x) = (0.0051x^4 ) - (0.1367x^3) + (1.24x^2) - (4.456x) + (5.66)
    """
    # Define the g(x) derivative and return
    return (0.0051 * x ** 4) - (0.1367 * x ** 3) + (1.24 * x ** 2) - (4.456 * x) +


def random_restart_hill_climbing(x_init, max_iterations, step_size):
    """
    Random restart hill-climbing algorithm
    """
    # Initialize the x_best
    x_current = x_init
    # Initialize the x_best
    x_best = x_init
    for i in range(max_iterations):  # Loop till maximum iterations
        x_current = x_current + step_size * g_prime(x_current)
        # Update x_current
        if g(x_current) > g(x_best):  # Update x_best
            x_best = x_current  # Update x_best
    return x_best  # Return x_best


def main():
    """
    Main function
    """
    x_init = random.uniform(0, 10)
    # Initialize x_init to a random number between 0 and 10
    max_iterations = 20  # Maximum iterations
    step_size = 0.5  # Step size
    # Find x_best
    x_best = random_restart_hill_climbing(x_init, max_iterations, step_size)
    # Print x_best
```

```
    print("x_best when x_init = {}: {}".format(x_init, x_best))

    # plot
    x = np.arange(0, 10, 0.5)  # Create an array of x values
    y = g(x)  # Create an array of y values
    plt.plot(x, y)  # Plot the function
    plt.title("Random Restart Hill Climbing")  # Plot title
    plt.show()  # Show the plot

    # Initialize x_init to a random number between 0 and 10
    x_init = random.uniform(0, 10)
    max_iterations = 100  # Maximum iterations
    step_size = 0.5  # Step size
    # Find x_best
    x_best = random_restart_hill_climbing(x_init, max_iterations, step_size)
    # Print x_best
    print("x_best when max_iterations = {} = {}".format(max_iterations, x_best))

    print("Comparison:")  # Print comparison
    print("x_best when x_init = {}: {}".format(x_init, x_best))  # Print x_best
    print("x_best when max_iterations = {} = {}".format(max_iterations, x_best))
    print("Therefore, the random-restart hill-climbing algorithm is better than the
    print("algorithm because it is more likely to find the global maximum value si


if __name__ == "__main__":  # Execute main function
    main()  # Execute main function
```

```
x_best when x_init = 1.8249607990306704: 1.8249607990306704
```

https://colab.research.google.com/drive/1FVjp1dsaNasJvpu0BY3NAidgbRvhyEkB?usp=sharing




```
x_best when max_iterations = 100 = 6.5359315270498
Comparison:
x_best when x_init = 2.365376550225653: 6.5359315270498
x_best when max_iterations = 100 = 6.5359315270498
Therefore, the random-restart hill-climbing algorithm is better than the random-:
algorithm because it is more likely to find the global maximum value since it ha:
```

✓ 0s completed at 7:46 PM