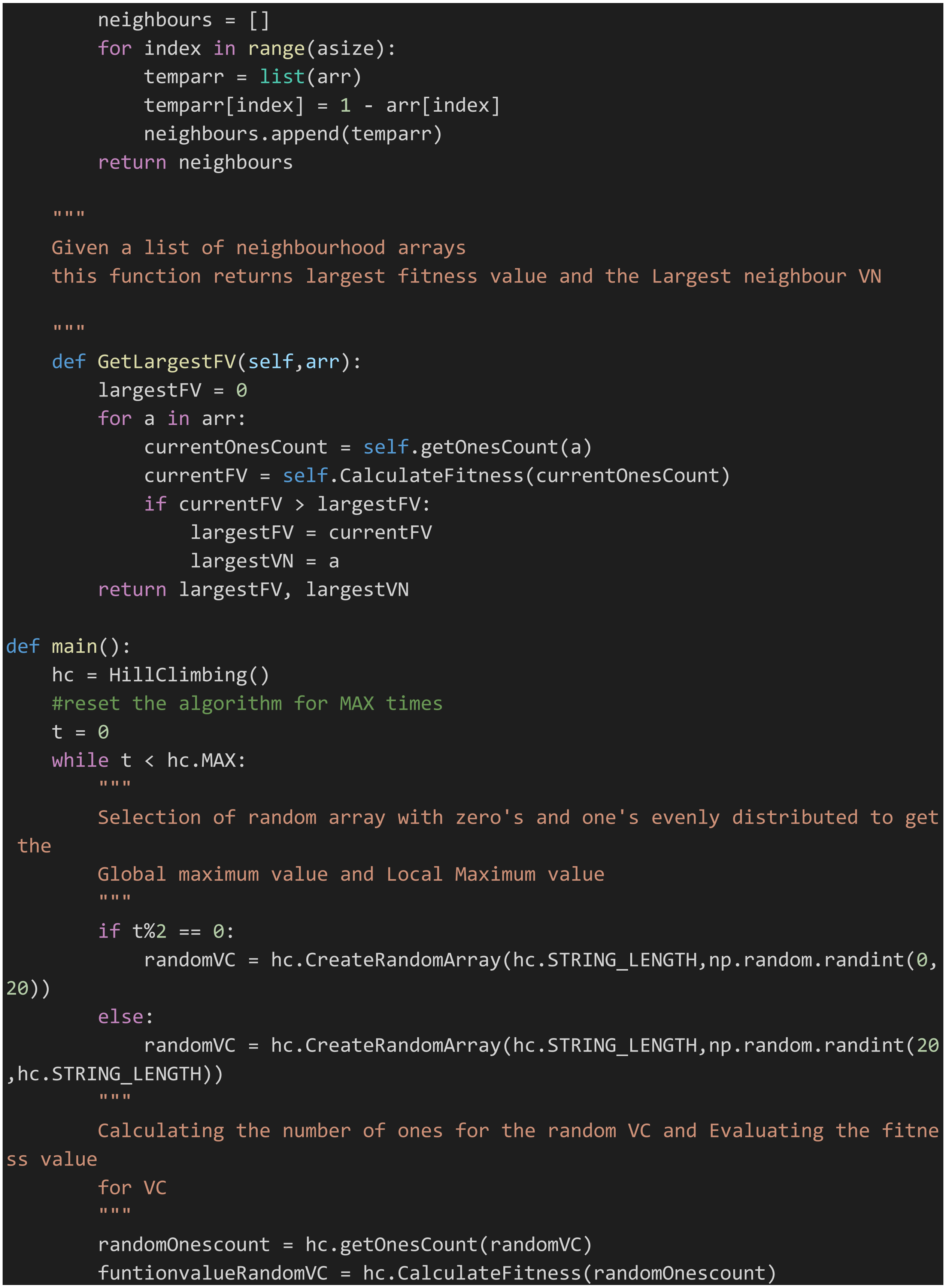
Fall 2020: CSCI 4588/5588 Programming Assignment #1”.

“Name: Tharani Maaneeivaannan
ID:2575198

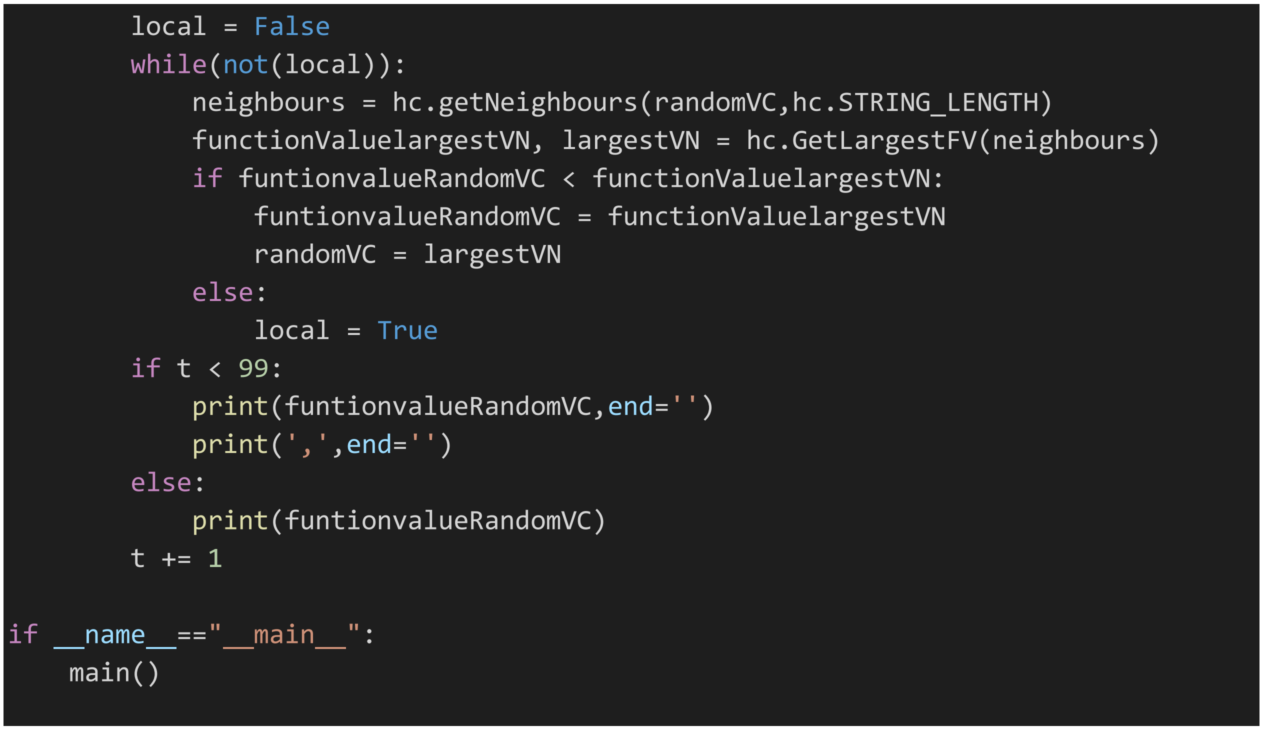
Solution to Part 1 – Hill Climbing Algorithm

Program:

```
import numpy as np
import os
import collections


class HillClimbing:
    """
    Initializing the constants
    STRING_LENGTH and NUMBER_OF_ITERATIONS
    """
    def __init__(self):
        self.STRING_LENGTH = 40
        self.MAX = 100

    """
    The below function creates a random array of 0's and 1's
    given the array size and how many ones you want in that array
    """
    def CreateRandomArray(self,arsize,ones):
        onesArray = np.ones(ones, dtype=np.int)
        zerosArray = np.zeros(arsize-ones, dtype=np.int)
        wholeArray = np.concatenate((onesArray, zerosArray), axis = 0)
        np.random.shuffle(wholeArray)
        return wholeArray

    """
    The below function calculates the fitness value for the given random array
    It counts the number of 1's in the given array and applies that in the functi
on and returns
    the fitness value
    """
    def CalculateFitness(self,onescount):
        return abs(13*onescount-170)

    def getOnesCount(self,arr):
        return collections.Counter(arr)[1]

    """
    Given an array of length asize this function returns a array list of
    one bit changed neighbours of length asize
    """
    def getNeighbours(self,arr,asize):
```

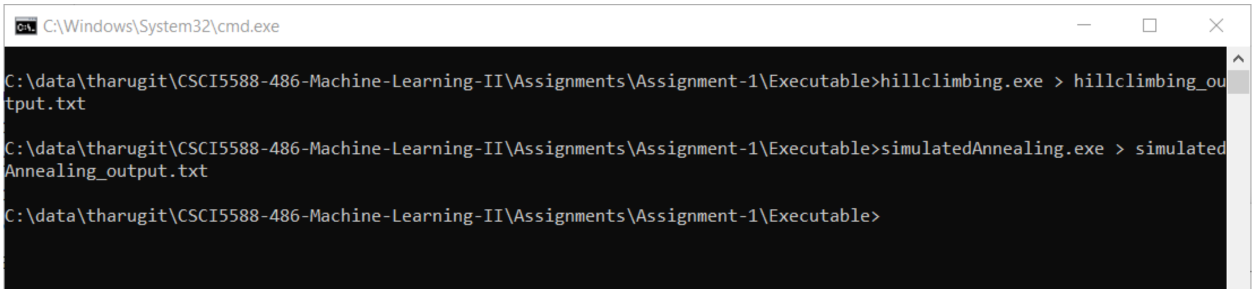


```
        neighbours = []
        for index in range(asize):
            temparr = list(arr)
            temparr[index] = 1 - arr[index]
            neighbours.append(temparr)
        return neighbours

    """
    Given a list of neighbourhood arrays
    this function returns largest fitness value and the Largest neighbour VN

    """
    def GetLargestFV(self,arr):
        largestFV = 0
        for a in arr:
            currentOnesCount = self.getOnesCount(a)
            currentFV = self.CalculateFitness(currentOnesCount)
            if currentFV > largestFV:
                largestFV = currentFV
                largestVN = a
        return largestFV, largestVN

def main():
    hc = HillClimbing()
    #reset the algorithm for MAX times
    t = 0
    while t < hc.MAX:
        """
        Selection of random array with zero's and one's evenly distributed to get
 the
        Global maximum value and Local Maximum value
        """
        if t%2 == 0:
            randomVC = hc.CreateRandomArray(hc.STRING_LENGTH,np.random.randint(0,
20))
        else:
            randomVC = hc.CreateRandomArray(hc.STRING_LENGTH,np.random.randint(20
,hc.STRING_LENGTH))
        """
        Calculating the number of ones for the random VC and Evaluating the fitne
ss value
        for VC
        """
        randomOnescount = hc.getOnesCount(randomVC)
        funtionvalueRandomVC = hc.CalculateFitness(randomOnescount)
```



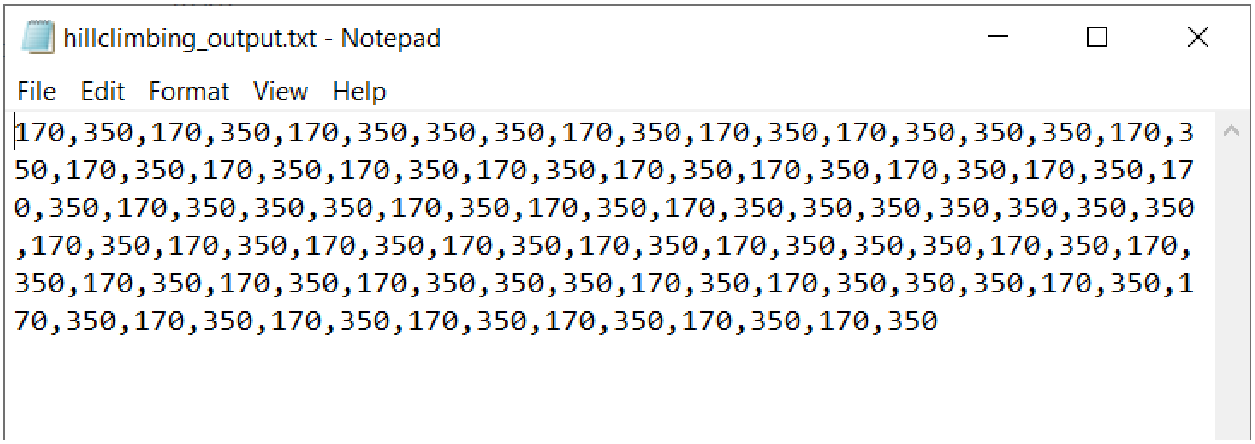
```
        local = False
        while(not(local)):
            neighbours = hc.getNeighbours(randomVC,hc.STRING_LENGTH)
            functionValuelargestVN, largestVN = hc.GetLargestFV(neighbours)
            if funtionvalueRandomVC < functionValuelargestVN:
                funtionvalueRandomVC = functionValuelargestVN
                randomVC = largestVN
            else:
                local = True
        if t < 99:
            print(funtionvalueRandomVC,end='')
            print(',',end='')
        else:
            print(funtionvalueRandomVC)
        t += 1

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

Output:



```
C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>hillclimbing.exe > hillclimbing_ou
tput.txt

C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>simulatedAnnealing.exe > simulated
Annealing_output.txt

C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>
```

Output file is stored inside Executable/Hillclimbing_output.txt

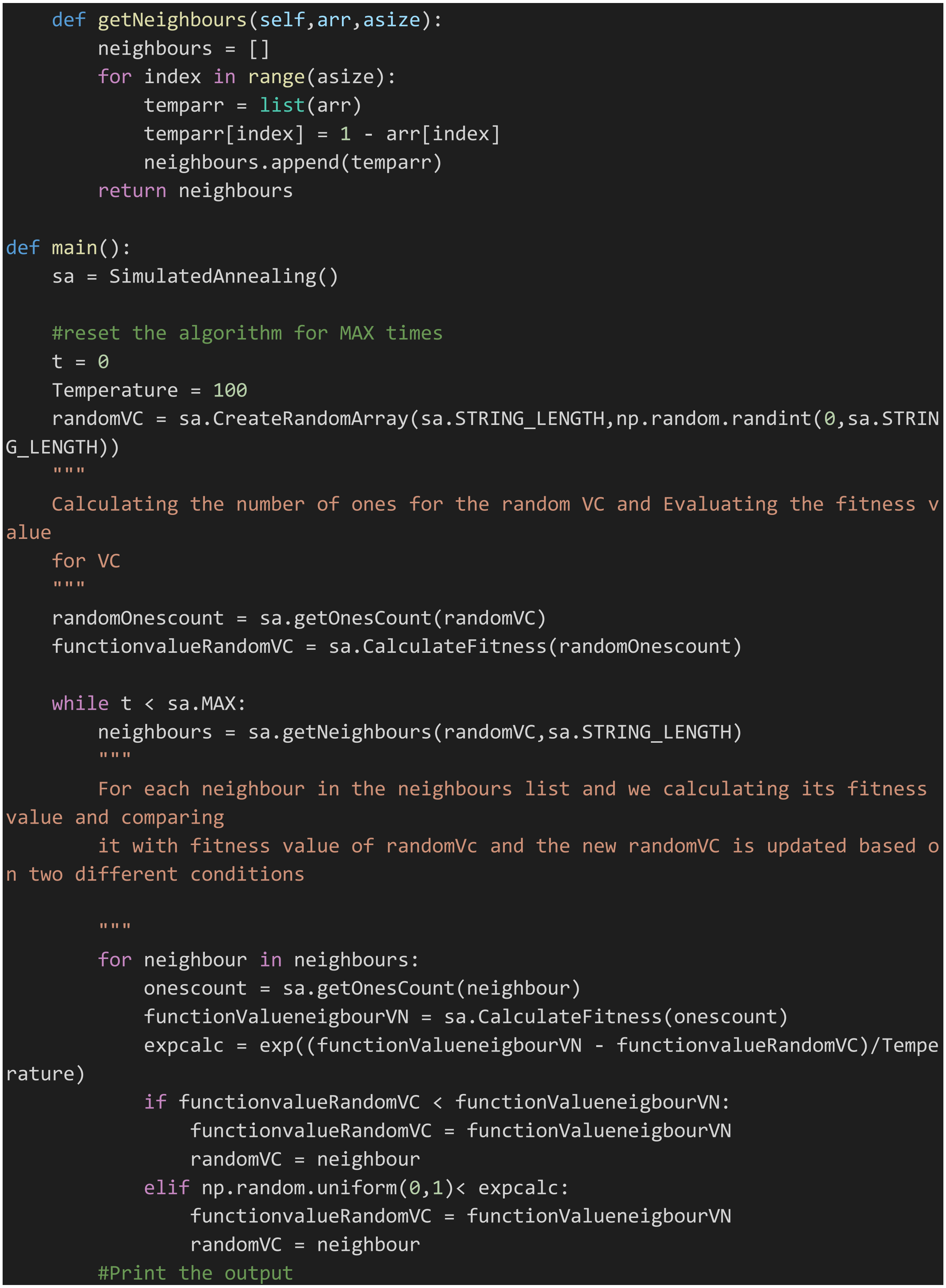


```
170,350,170,350,170,350,350,350,170,350,170,350,170,350,350,350,170,3
50,170,350,170,350,170,350,170,350,170,350,170,350,170,350,170,350,17
0,350,170,350,350,350,170,350,170,350,170,350,350,350,350,350,350,350
,170,350,170,350,170,350,170,350,170,350,170,350,350,350,170,350,170,
350,170,350,170,350,170,350,350,350,170,350,170,350,350,350,170,350,1
70,350,170,350,170,350,170,350,170,350,170,350,170,350
```

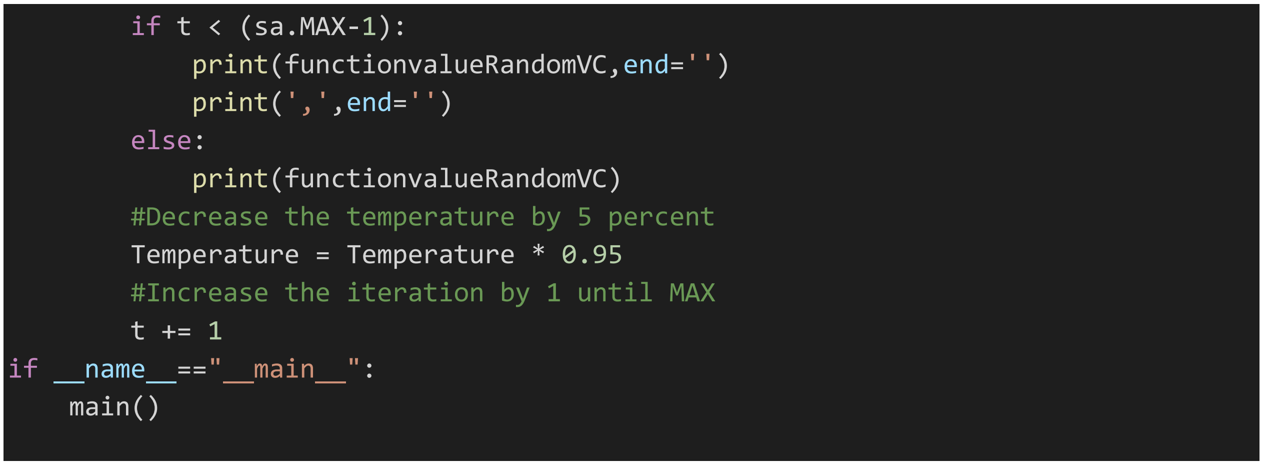
Solution for Part 2: Simulated Annealing

Program:

```
import numpy as np
import os
import collections
from math import exp


class SimulatedAnnealing:
    """
    Initializing the constants
    STRING_LENGTH and NUMBER_OF_ITERATIONS
    """
    def __init__(self):
        self.STRING_LENGTH = 50
        self.MAX = 200

    """
    The below function creates a random array of 0's and 1's
    given the array size and how many ones you want in that array
    """
    def CreateRandomArray(self,arsize,ones):
        onesArray = np.ones(ones, dtype=np.int)
        zerosArray = np.zeros(arsize-ones, dtype=np.int)
        wholeArray = np.concatenate((onesArray, zerosArray), axis = 0)
        np.random.shuffle(wholeArray)
        return wholeArray

    """
    The below function calculates the fitness value for the given random array
    It counts the number of 1's in the given array and applies that in the functi
on and returns
    the fitness value
    """
    def CalculateFitness(self,onescount):
        return abs(14*onescount-190)

    def getOnesCount(self,arr):
        return collections.Counter(arr)[1]

    """
    Given an array of length asize this function returns a array list of
    one bit changed neighbours of length asize
    """
```

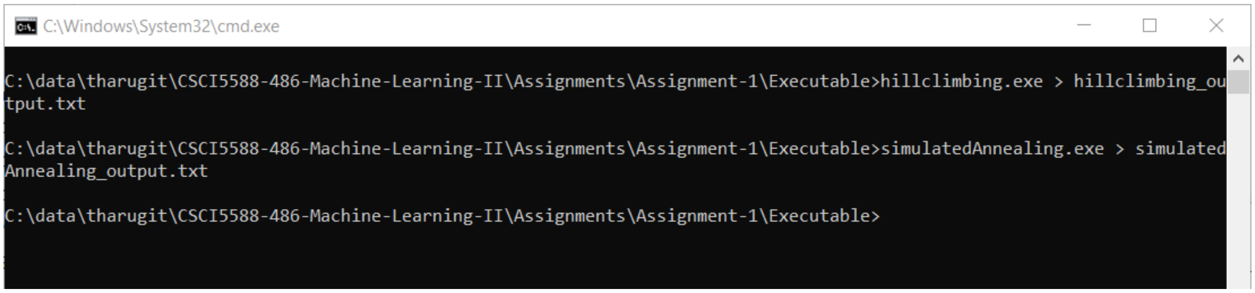


```
    def getNeighbours(self,arr,asize):
        neighbours = []
        for index in range(asize):
            temparr = list(arr)
            temparr[index] = 1 - arr[index]
            neighbours.append(temparr)
        return neighbours

def main():
    sa = SimulatedAnnealing()

    #reset the algorithm for MAX times
    t = 0
    Temperature = 100
    randomVC = sa.CreateRandomArray(sa.STRING_LENGTH,np.random.randint(0,sa.STRING_LENGTH))
    """
    Calculating the number of ones for the random VC and Evaluating the fitness value
    for VC
    """
    randomOnescount = sa.getOnesCount(randomVC)
    functionvalueRandomVC = sa.CalculateFitness(randomOnescount)

    while t < sa.MAX:
        neighbours = sa.getNeighbours(randomVC,sa.STRING_LENGTH)
        """
        For each neighbour in the neighbours list and we calculating its fitness value and comparing
        it with fitness value of randomVc and the new randomVC is updated based on two different conditions

        """
        for neighbour in neighbours:
            onescount = sa.getOnesCount(neighbour)
            functionValueneigbourVN = sa.CalculateFitness(onescount)
            expcalc = exp((functionValueneigbourVN - functionvalueRandomVC)/Temperature)
            if functionvalueRandomVC < functionValueneigbourVN:
                functionvalueRandomVC = functionValueneigbourVN
                randomVC = neighbour
            elif np.random.uniform(0,1)< expcalc:
                functionvalueRandomVC = functionValueneigbourVN
                randomVC = neighbour
        #Print the output
```



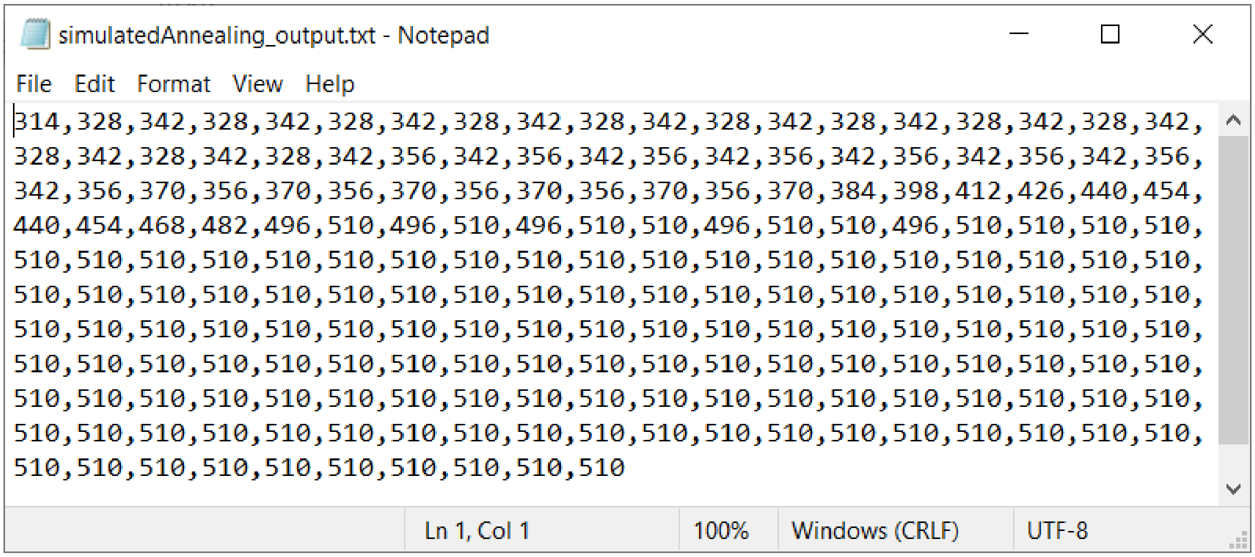
```
        if t < (sa.MAX-1):
            print(functionvalueRandomVC,end='')
            print(',',end='')
        else:
            print(functionvalueRandomVC)
        #Decrease the temperature by 5 percent
        Temperature = Temperature * 0.95
        #Increase the iteration by 1 until MAX
        t += 1
if __name__=="__main__":
    main()
```

Output:



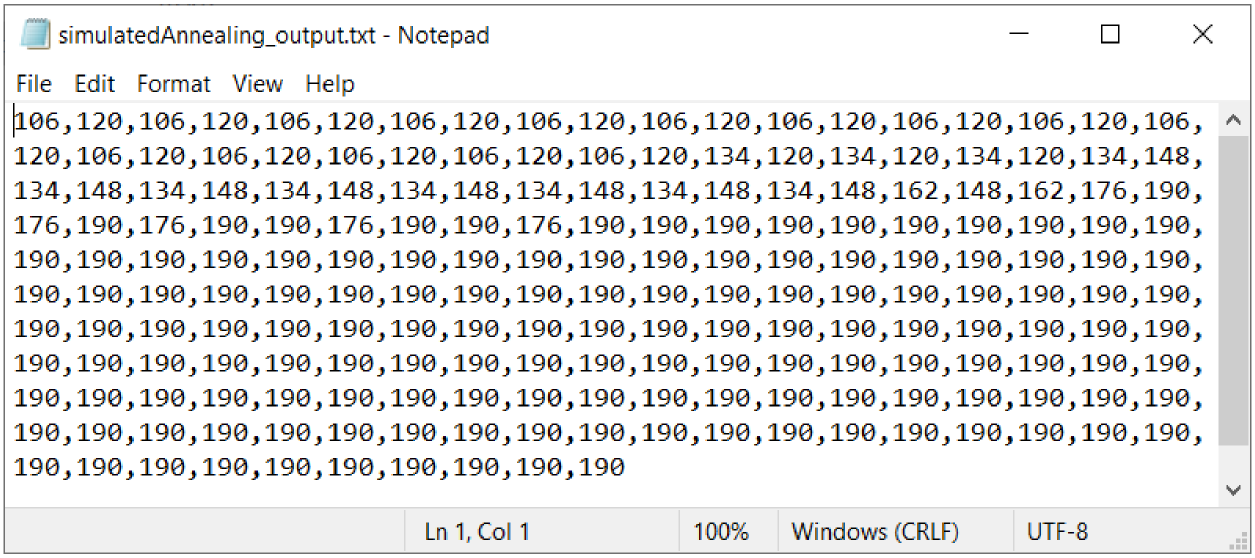
```
C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>hillclimbing.exe > hillclimbing_ou
tput.txt

C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>simulatedAnnealing.exe > simulated
Annealing_output.txt

C:\data\tharugit\CSCI5588-486-Machine-Learning-II\Assignments\Assignment-1\Executable>
```

Output file is stored inside Executable/simulatedAnnealing_output.txt. Based on the random value selection. The output value either evolves to Local Maximum or Global Maximum



simulatedAnnealing_output.txt - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

```
314,328,342,328,342,328,342,328,342,328,342,328,342,328,342,328,342,328,342,
328,342,328,342,328,342,356,342,356,342,356,342,356,342,356,342,356,342,356,
342,356,370,356,370,356,370,356,370,356,370,356,370,384,398,412,426,440,454,
440,454,468,482,496,510,496,510,496,510,510,496,510,510,496,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,
510,510,510,510,510,510,510,510,510,510
```



```
106,120,106,120,106,120,106,120,106,120,106,120,106,120,106,120,106,120,106,
120,106,120,106,120,106,120,106,120,106,120,134,120,134,120,134,120,134,148,
134,148,134,148,134,148,134,148,134,148,134,148,134,148,162,148,162,176,190,
176,190,176,190,190,176,190,190,176,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,190,
190,190,190,190,190,190,190,190,190,190
```