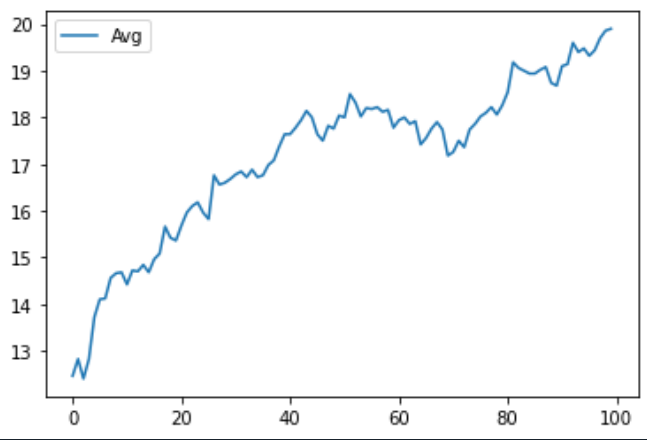
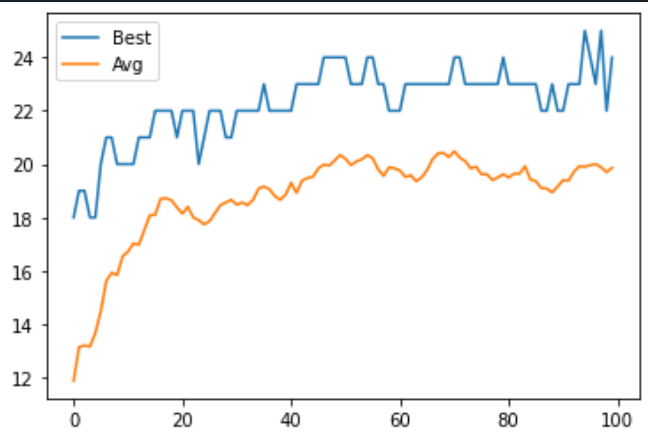
**Evolutionary Algorithm 1 – Assignment 1**

**Simple Genetic Algorithm**

1. Implement the simple genetic algorithm in a programming language of your choice.
2. Use the implemented genetic algorithm to find an individual with all 1s.

**Output**:

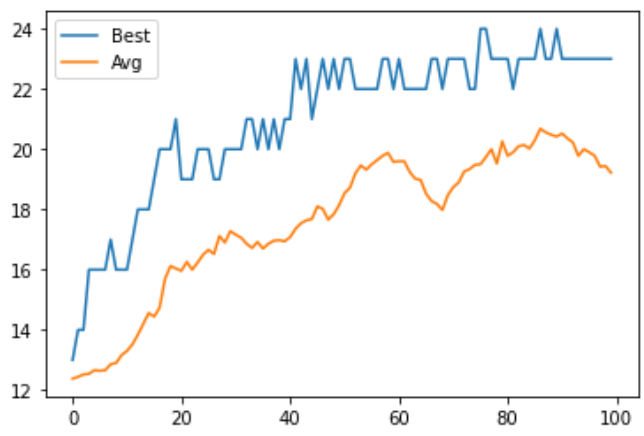




**What I did**: I used the program which was used in recorded practical video.

1. Change the fitness of the algorithm to find an individual with alternating 1s and 0s (either 010101... or 101010...).

**Output**:



**What I did**: I changed the create\_ind() with itertools

**Code**:

def create\_ind():

x = []

x = random.randint(0,1)

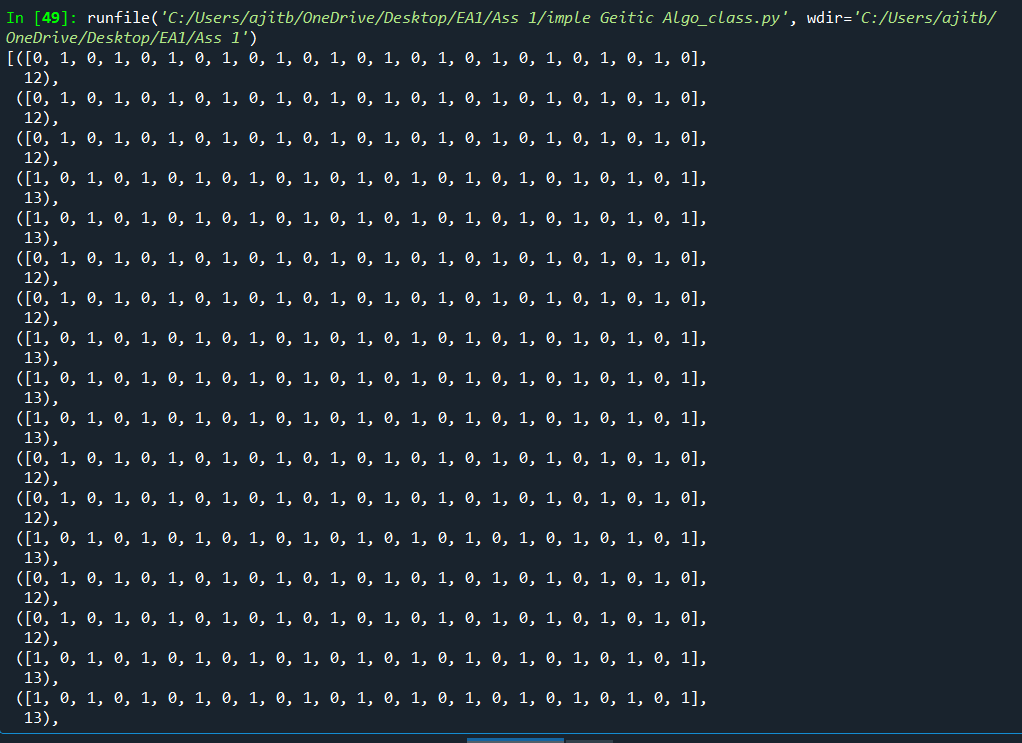
if x:

pattern = [1] \* 1 + [0] \* 1

else: pattern = [0] \* 1 + [1] \* 1

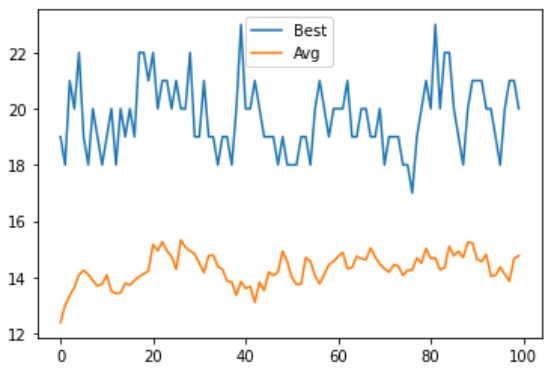
z = list(itertools.islice(itertools.cycle(pattern), DIMENSION))

return z

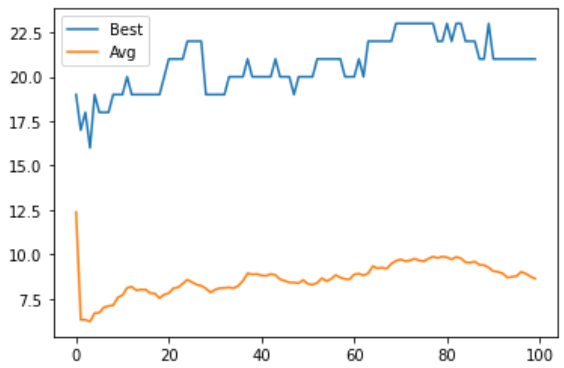


1. Try to change the parameters (probability of mutation/crossover) and see what happens.

**Output**: Changing only probability of mutation from 0.01(from graph 1&2) to 0.09



**Output**: Changing only probability of crossover.



**What I did**:

**Code**: Crossover changes..

def crossover(pop):

offspring = []

for p1, p2 in zip(**pop[::4], pop[2::4**]):

o1, o2 = cross(p1, p2)

offspring.append(o1)

offspring.append(o2)

return offspring

**Code**: Mutation changes..

def mutate(ind):

o=[]

for bit in ind:

if **random.random() < 0.09**:

o.append(1-bit)

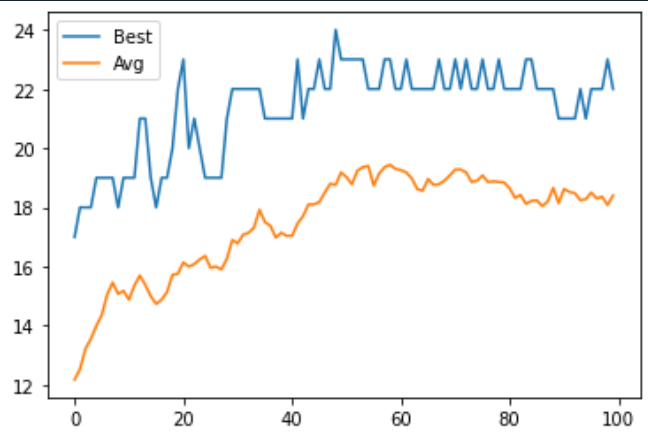
else:

o.append(bit)

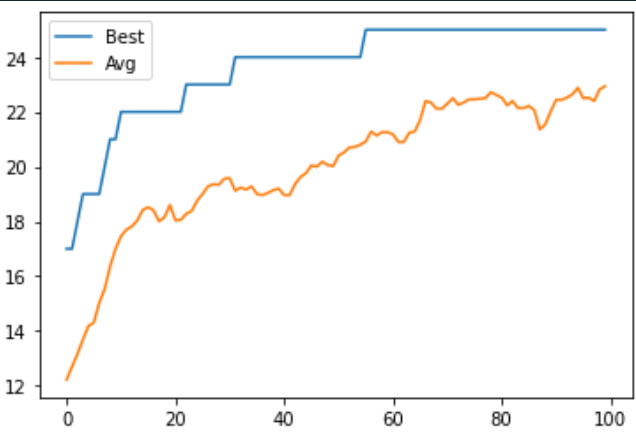
return o

1. Submit a plot comparing the convergence of the algorithm for two different settings of the algorithm.

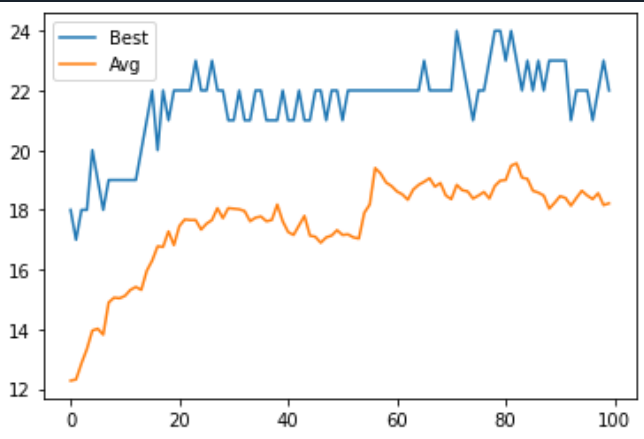
**Original**



**Maximum**

****

**Minimum**

****

**What I did:**

**Code**: For Maximum

pop = offspring[1:] + [max(pop, key=fitness)]

**Code:** For Minimum

pop = offspring[1:] + [min(pop, key=fitness)]

**Platform Used: Spyder IDE**

***-----------------------------------------THE END---------------------------------------------------***