**Approach 1: Brute force**

Generate all possible triplets, return the largest product that a triplet can reach

Time -> O(n^3)

Space -> O(1)

**Approach 2: Sorting**

Sort the array

Now largest triplet product is product of last 3 numbers provided only positive values

But if negative [ -200, -100, 1 , 2, 3] here 1\*2\*3 = 6 < -200\*-300\*1 = 600

Therefore if 2 smaller negative numbers multiplied with largest number gives a product larger then any other triplet then return that

MAX(nums[0]\*nums[1]\*nums[-1] , nums[-1]\*nums[-2]\*nums[-3])

Time -> O(nlogn)

Space -> O(nlogn)

**Approach 3 : Single Pass**

Instead of sorting just find 2 minimums and 3 maximums/largest values in array, return max of these products just like previous approach

**CODE**:

first\_smallest = sys.maxsize

second\_smallest = sys.maxsize

first\_largest = sys.maxsize\*-1

second\_largest = sys.maxsize\*-1

third\_largest = sys.maxsize\*-1

for i in range(len(nums)):

if nums[i] <= first\_smallest:

second\_smallest =first\_smallest

first\_smallest = nums[i]

elif nums[i] <= second\_smallest:

second\_smallest = nums[i]

if nums[i] >= first\_largest:

third\_largest = second\_largest

second\_largest = first\_largest

first\_largest = nums[i]

elif nums[i] >= second\_largest:

third\_largest = second\_largest

second\_largest = nums[i]

elif nums[i] >= third\_largest:

third\_largest = nums[i]

return max(first\_largest \* second\_largest \* third\_largest, first\_smallest \* second\_smallest \* first\_largest)