|  |  |
| --- | --- |
| 01\_practice | 01\_answer |
| def solution(nums):   a = nums[0]  for i in range(1,len(nums)):  if a > nums[i]:  a = nums[i]  answer = i  else:  continue  return answer   print(solution([7,10,5,3,2,15,19])) print(solution([-12,12,30,-15,-5,3,9,-11,14])) print(solution([17,11,5,8,23,29,19,12,25,16,2])) print(solution([7,5,12,-9,-12,22,-30,-35,-21])) | def solution(nums):   answer = 0  minN = 1000000000  n = len(nums)   for i in range(n):  if nums[i] < minN:  minN = nums[i]  answer = i   return answer  print(solution([7,10,5,3,2,15,19])) print(solution([-12,12,30,-15,-5,3,9,-11,14])) print(solution([17,11,5,8,23,29,19,12,25,16,2])) print(solution([7,5,12,-9,-12,22,-30,-35,-21])) |
| 02\_practice = 02\_answer |
| def solution(score, k):   answer = 0   for i in score:  if i>=k:  answer+=1   return answer  print(solution([60,50,80,90,55,70,65,45],60)) print(solution([10,20,30,40,50],60)) print(solution([50,65,75,87,90,55,78,93,100],70)) print(solution([99,30,50,55,68,70,90,100],80)) |
| 03\_practice | 03\_answer |
| def solution(nums):   answer = 0  answer\_list=[]  long\_answer = 0   for i in range(len(nums)):   if nums[i] == 1:  answer += 1   else:  answer\_list.append(answer)  answer = 0   answer\_list.append(answer)   for i in answer\_list:  if long\_answer < i:  long\_answer = i   return long\_answer | def solution(nums):  answer = 0  cnt = 0   for x in nums:  if x == 1:  cnt+=1   else:  answer = max(answer,cnt)  cnt = 0   answer = max(answer, cnt)  return answer  print(solution([1,1,0,1,1,1,0,1,1,1,1,1])) print(solution([0,0,1,0,1,0,0])) print(solution([1,1,1,1,1])) print(solution([1,0,1,1,0,1,1,1,0,1,1,1,0,1])) |
| 04\_practice | 04\_answer |
| def solution(nums, k):   answer = nums[k:len(nums)]+nums[0:k]   return answer   print(solution([3,7,1,5,9,2,8],3)) print(solution([2,12,2,1,3,3,9],2)) print(solution([1,2,5,4,6,7,9],6)) print(solution([1,3,6,8,14,2,1,7],5)) | #deque() 자료구조 이용  from collections import deque   def solution(nums, k):   answer = deque(nums) #nums라는 리스트가 deque구조로 변경됨  for i in range(k):  answer.append(answer.popleft())   return list(answer)   print(solution([3,7,1,5,9,2,8],3)) print(solution([2,12,2,1,3,3,9],2)) print(solution([1,2,5,4,6,7,9],6)) print(solution([1,3,6,8,14,2,1,7],5)) |
| 05\_practice | 05\_answer |
| def solution(nums):   unique\_list = []   for i in nums:  if i not in unique\_list:  unique\_list.append(i)   for i in range(len(unique\_list)):   for j in range(i+1, len(unique\_list)):   if unique\_list[i] < unique\_list[j]:  small\_num = unique\_list[i]  big\_num = unique\_list[j]   unique\_list[i] = big\_num  unique\_list[j] = small\_num   return unique\_list | from collections import deque   def solution(nums):   answer = deque()  answer.appendleft(nums[0])   for i in range(1, len(nums)):  if nums[i-1] != nums[i]:  answer.appendleft(nums[i])   return list(answer)   print(solution([0,1,1,1,2,2,2,3])) print(solution([1,1,2,2,2,3,3,3,3,3,4,4,4,5])) print(solution([0,0,0,3,3,3,5,7,7,7])) print(solution([1,2,3,4,5,6,7,7,7,8,9])) |
| 06\_practice | 06\_answer |
| def solution(nums, target):   for i in nums:  another\_num = target-i   if another\_num in nums and i != another\_num:  if i < another\_num:  small\_num = i  big\_num = another\_num  else:  small\_num = another\_num  big\_num = i   return [small\_num, big\_num]   return [0,0] print(solution([7,3,2,13,9,15,8,11],12)) print(solution([21,12,30,15,6,2,9,19,14],24)) print(solution([12,18,5,8,21,27,22,25,16,2],28)) print(solution([11,17,6,8,21,9,19,12,25,16,2],26)) print(solution([7,5,12,-9,-12,22,-30,-35,-21],-14)) print(solution([7,5,12,20],15)) | def solution(nums, target):  answer = [0] \* 2 #[0,0]  n = len(nums)  for i in range(n-1):  for j in range(i+1, n):  if nums[i]+nums[j] == target:  return sorted([nums[i], nums[j]])   return answer  print(solution([7,3,2,13,9,15,8,11],12)) print(solution([21,12,30,15,6,2,9,19,14],24)) print(solution([12,18,5,8,21,27,22,25,16,2],28)) print(solution([11,17,6,8,21,9,19,12,25,16,2],26)) print(solution([7,5,12,-9,-12,22,-30,-35,-21],-14)) print(solution([7,5,12,20],15)) |