price, money, count = 3,20,4  
  
def solution(price, money, count):  
 lPrice,cnt = 0,1 # 최종 가격과 이용횟수 초기화, 이용횟수는 첫시작이 1  
 while cnt<=count: # 이용횟수 동안의 총 가격 계산  
 lPrice += price\*cnt # 최종 가격 = 기존가격\* N번째  
 cnt+=1  
 if money>=lPrice: answer=0 # 금액이 모자라지 않다면 0을 반환  
 else: answer=lPrice-money # 금액이 모자라다면 모자란 금액 반환  
 return answer  
  
print(solution(price,money,count))

리뷰: 우선 문제를 제대로 이해하지 못하여 의외로 풀이에 드는 시간이 많았습니다. 직관적으로 최종가격 lPrice 를 설정하여 count에 도달할때 까지 기존 가격의 count 를 곱하여 더해주므로 최종 가격을 얻었고 그후에 현재 소지중인 돈과 비교하여 금액이 모자르지 않다면 0을, 모자라다면 놀이기구를 타기위해 필요한 돈에서 현재 소지중인 돈을 빼주었습니다. 문제에서 lPrice 와 cnt 라는 변수를 따로 선언하게 되었는데 따로 변수를 선언하지 않고 더 간결하게 표현할 수 있는 법은 없을지 고민을 해봐야 할것같습니다. 그리고 문제를 독해하는 법을 조금더 익혀야 할 필요성을 느꼈습니다.

prices = [1,2,3,2,3]  
  
def solution(prices):  
 answer = [0]\*len(prices) # 각 시점당 0으로 초기화해준다  
 for i in range(len(prices)-1): # 마지막 인덱스 값은 항상 0이기에 반복에 넣지 않는다  
 if prices[i]>prices[i+1]:# 만약 앞의 값이 바로 뒤에 값보다 크다면  
 answer[i]+=1 # 1초동안은 가격이 떨어지지 않은 것으로 보고 +1 해주고 다음 반복으로  
 continue  
 for j in range(i+1,len(prices)):  
 if prices[i] <= prices[j]:  
 answer[i]+=1 # 만약 가격이 떨어지지 않으면 +1  
 else: break # 만약 가격이 떨어졌다면 반복 탈출  
 return answer  
  
print(solution(prices))

리뷰: 예전 프로그래머스에서 풀었던 문제와 매우 유사하여 푸는 시간은 얼마 걸리지 않았던것 같습니다. 우선 핵심이라고 하자면 앞의 값이 바로 뒤에 값보다 크다면 1초동안은 유지 되었다고 보고 1초를 유지하였다고 해야하는 부분인데 이는 마지막 인덱스값이 항상 0임을 이용하여 첫번째 반복 구간에서 바로뒤의 값과 비교하여 더 크다면 +1 을 하고 바로 다음 반복으로 갈수 있도록 continue 해주었습니다. 아무래도 for문을 두번 돌리다 보니 O(n^2)을 갖게 되는데 2중 반복이아닌 방법이 없을지 조금더 고민하고 개선해봐야 겠습니다.