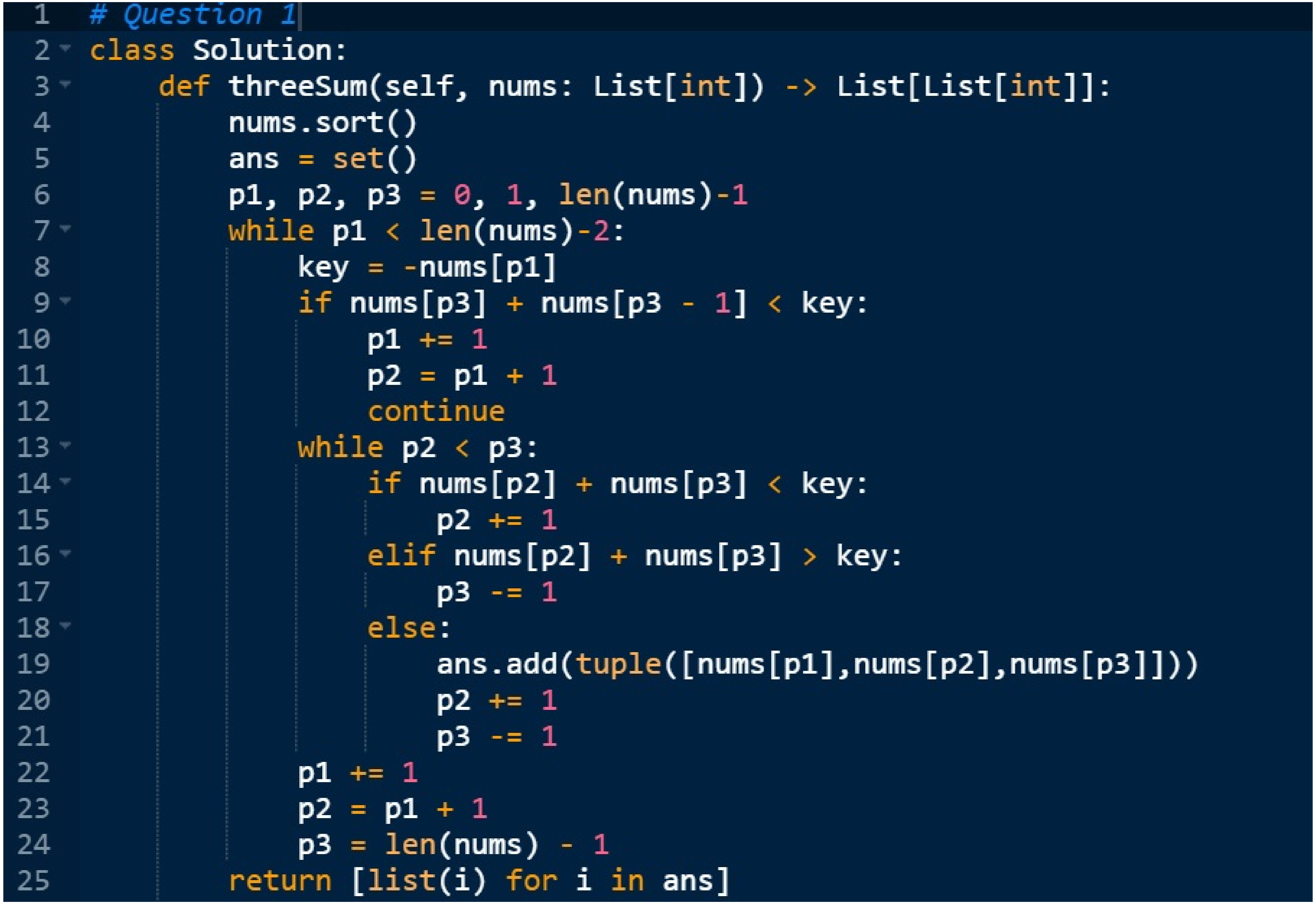
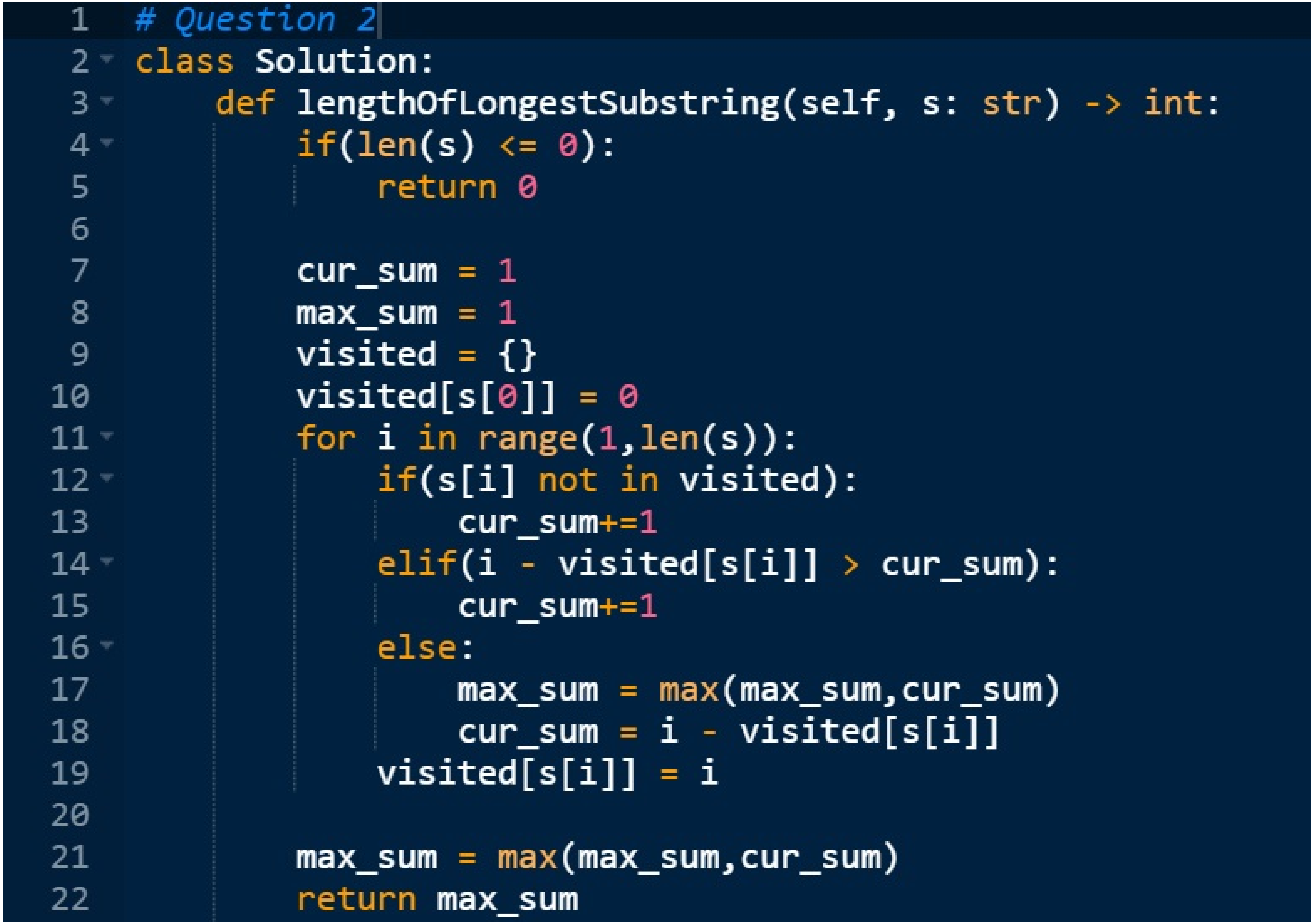
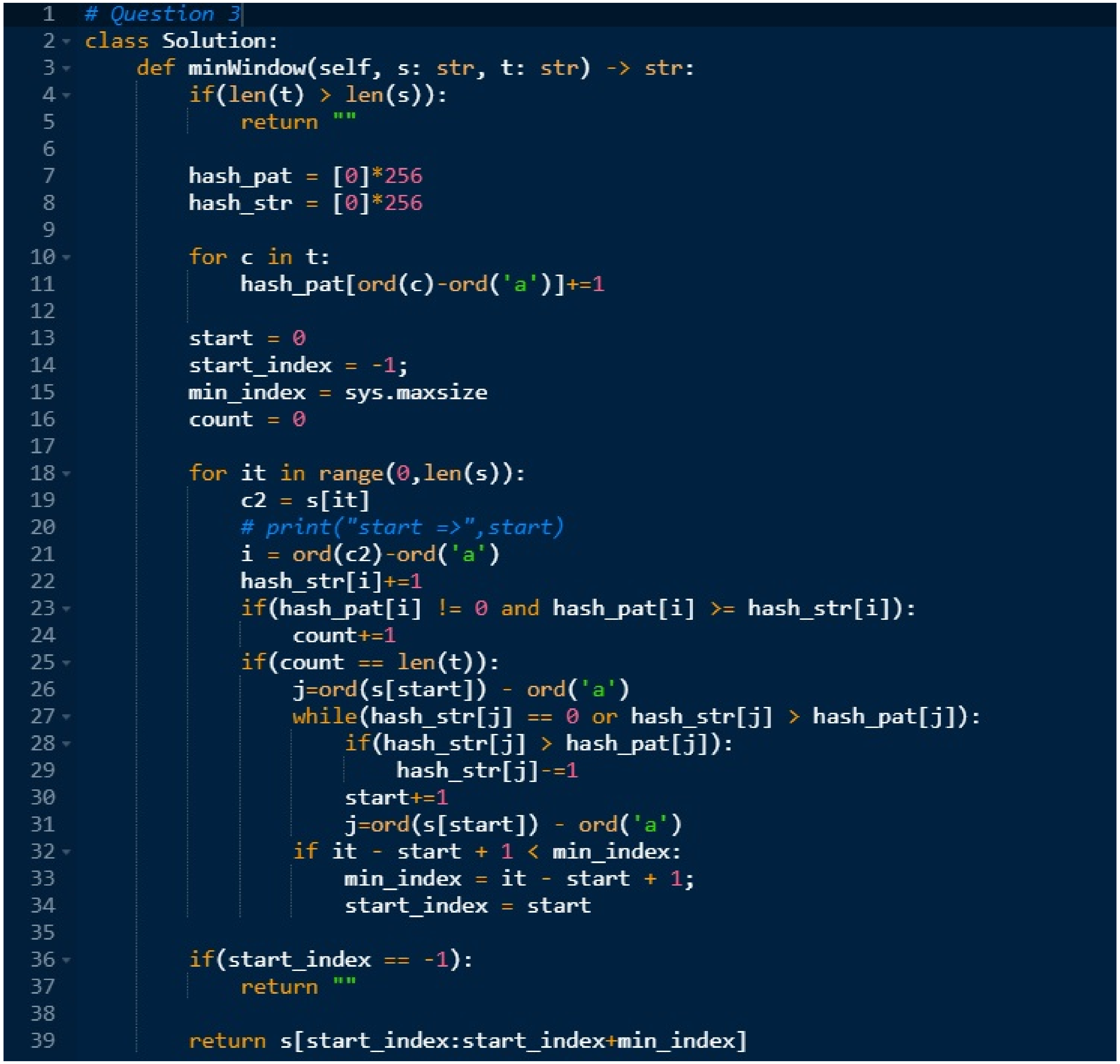


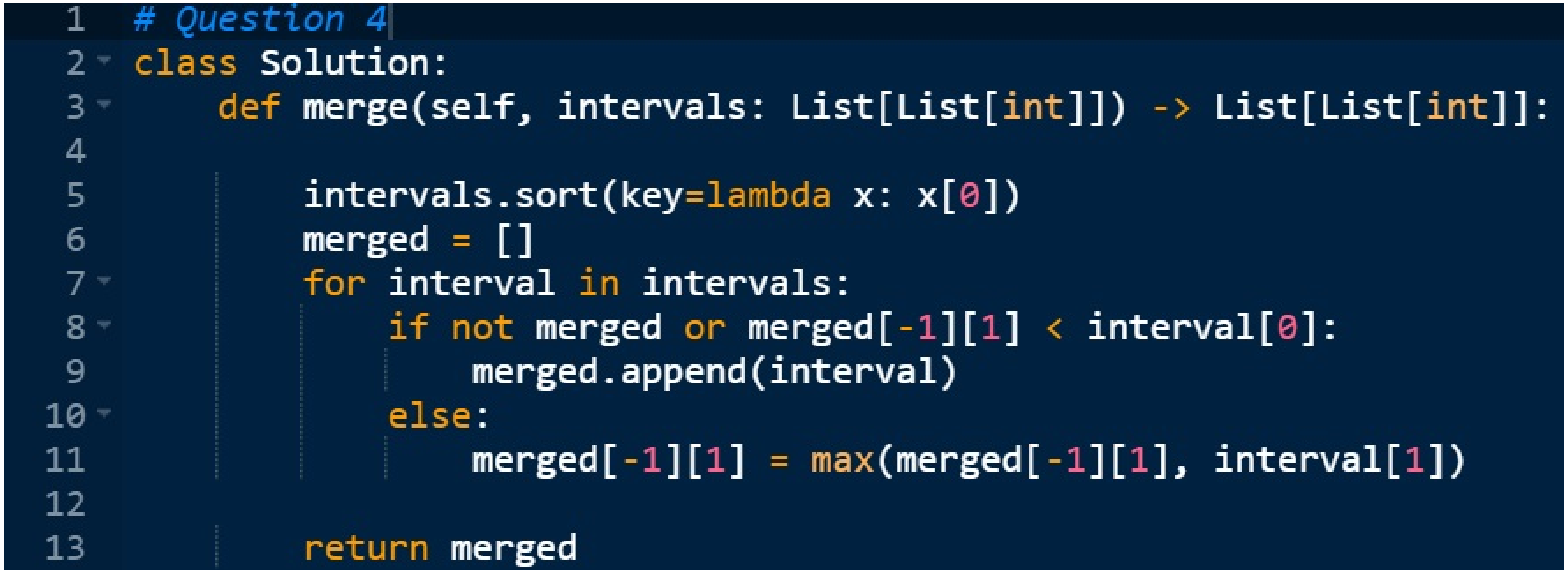
```python
# Question 1
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        nums.sort()
        ans = set()
        p1, p2, p3 = 0, 1, len(nums)-1
        while p1 < len(nums)-2:
            key = -nums[p1]
            if nums[p3] + nums[p3 - 1] < key:
                p1 += 1
                p2 = p1 + 1
                continue
            while p2 < p3:
                if nums[p2] + nums[p3] < key:
                    p2 += 1
                elif nums[p2] + nums[p3] > key:
                    p3 -= 1
                else:
                    ans.add(tuple([nums[p1],nums[p2],nums[p3]]))
                    p2 += 1
                    p3 -= 1
            p1 += 1
            p2 = p1 + 1
            p3 = len(nums) - 1
        return [list(i) for i in ans]
```



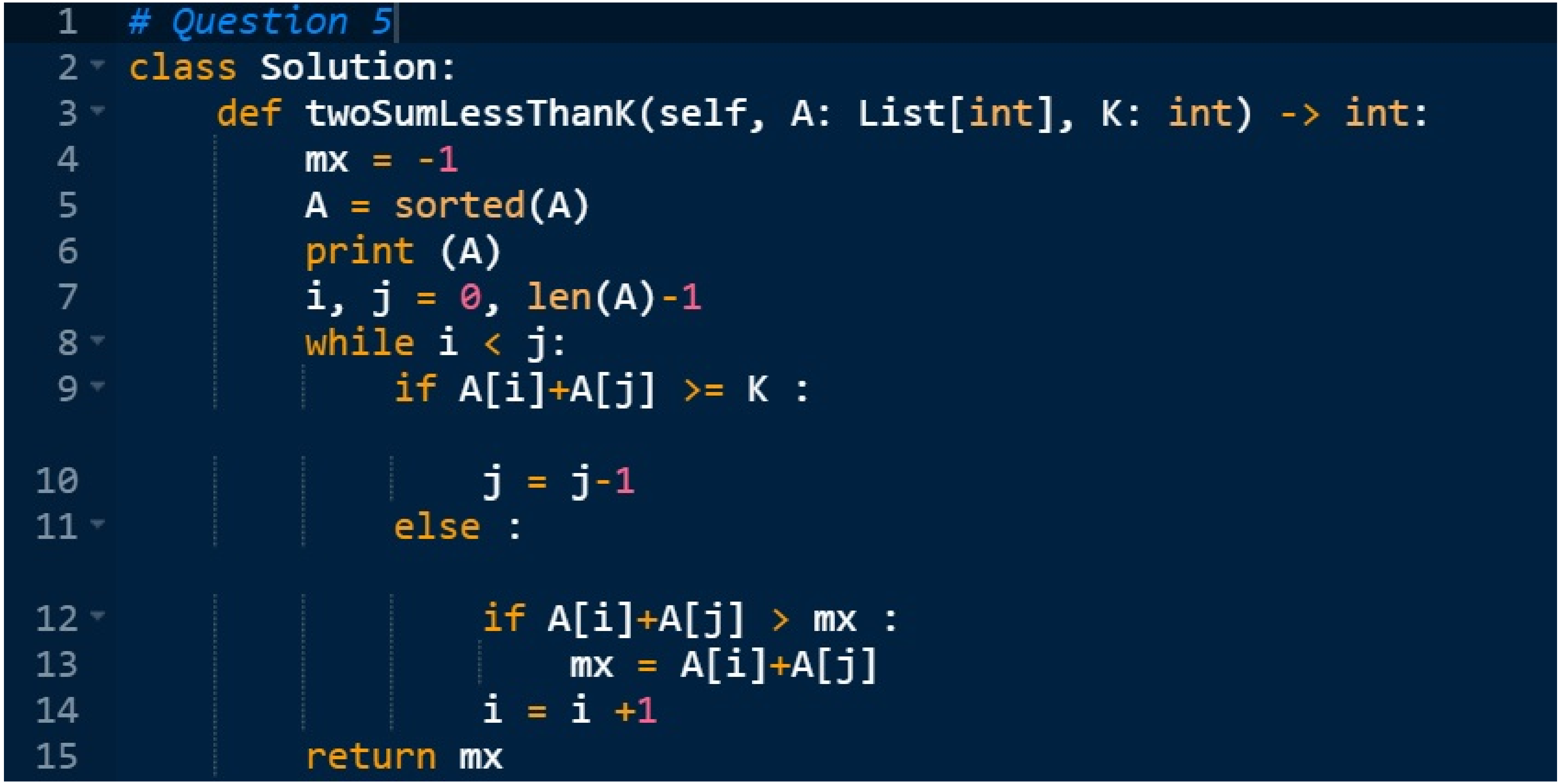
```python
# Question 2
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        if(len(s) <= 0):
            return 0

        cur_sum = 1
        max_sum = 1
        visited = {}
        visited[s[0]] = 0
        for i in range(1,len(s)):
            if(s[i] not in visited):
                cur_sum+=1
            elif(i - visited[s[i]] > cur_sum):
                cur_sum+=1
            else:
                max_sum = max(max_sum,cur_sum)
                cur_sum = i - visited[s[i]]
            visited[s[i]] = i

        max_sum = max(max_sum,cur_sum)
        return max_sum
```



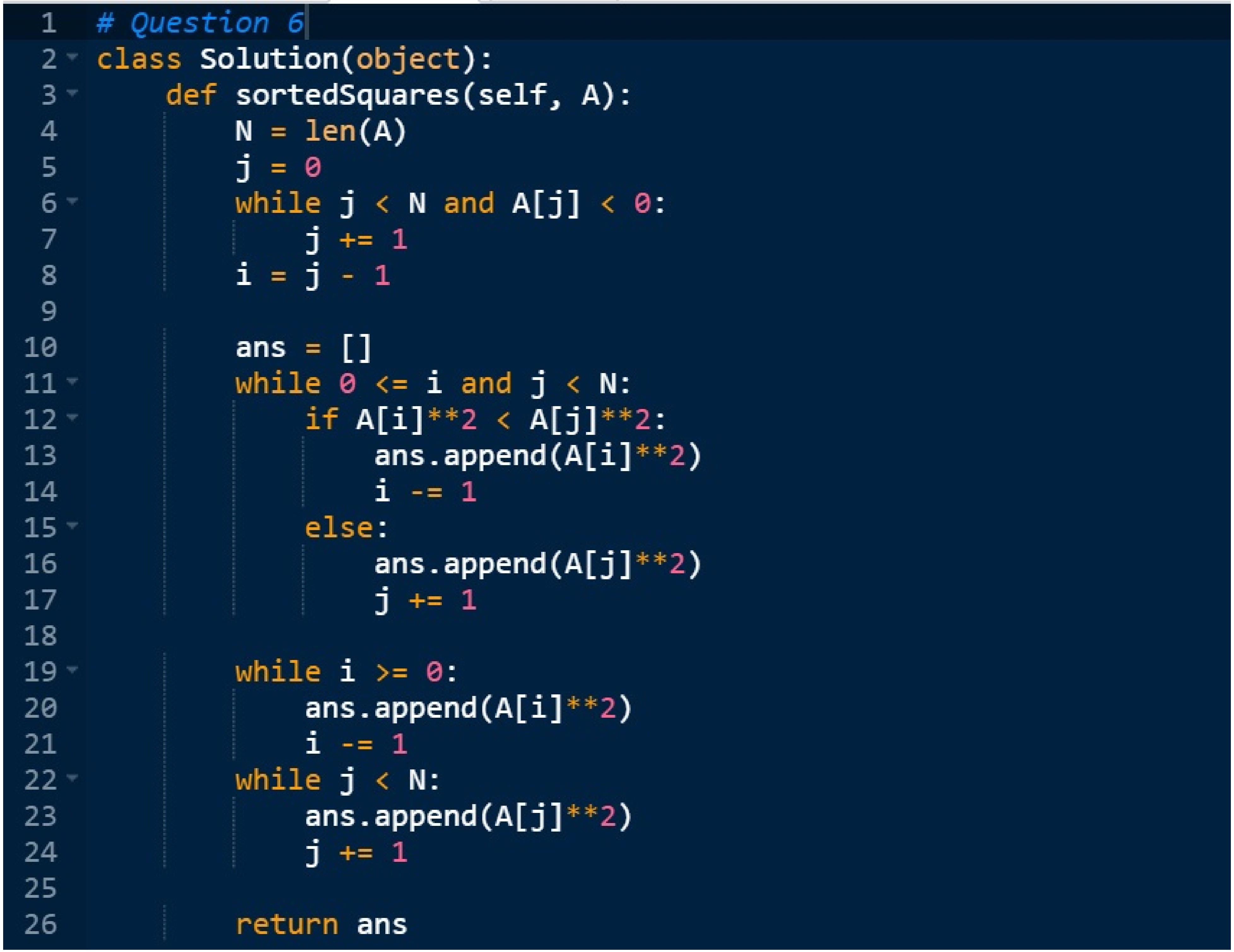
```python
# Question 3
class Solution:
    def minWindow(self, s: str, t: str) -> str:
        if(len(t) > len(s)):
            return ""

        hash_pat = [0]*256
        hash_str = [0]*256

        for c in t:
            hash_pat[ord(c)-ord('a')]+=1

        start = 0
        start_index = -1;
        min_index = sys.maxsize
        count = 0

        for it in range(0,len(s)):
            c2 = s[it]
            # print("start =>",start)
            i = ord(c2)-ord('a')
            hash_str[i]+=1
            if(hash_pat[i] != 0 and hash_pat[i] >= hash_str[i]):
                count+=1
            if(count == len(t)):
                j=ord(s[start]) - ord('a')
                while(hash_str[j] == 0 or hash_str[j] > hash_pat[j]):
                    if(hash_str[j] > hash_pat[j]):
                        hash_str[j]-=1
                    start+=1
                    j=ord(s[start]) - ord('a')
                if it - start + 1 < min_index:
                    min_index = it - start + 1;
                    start_index = start

        if(start_index == -1):
            return ""

        return s[start_index:start_index+min_index]
```



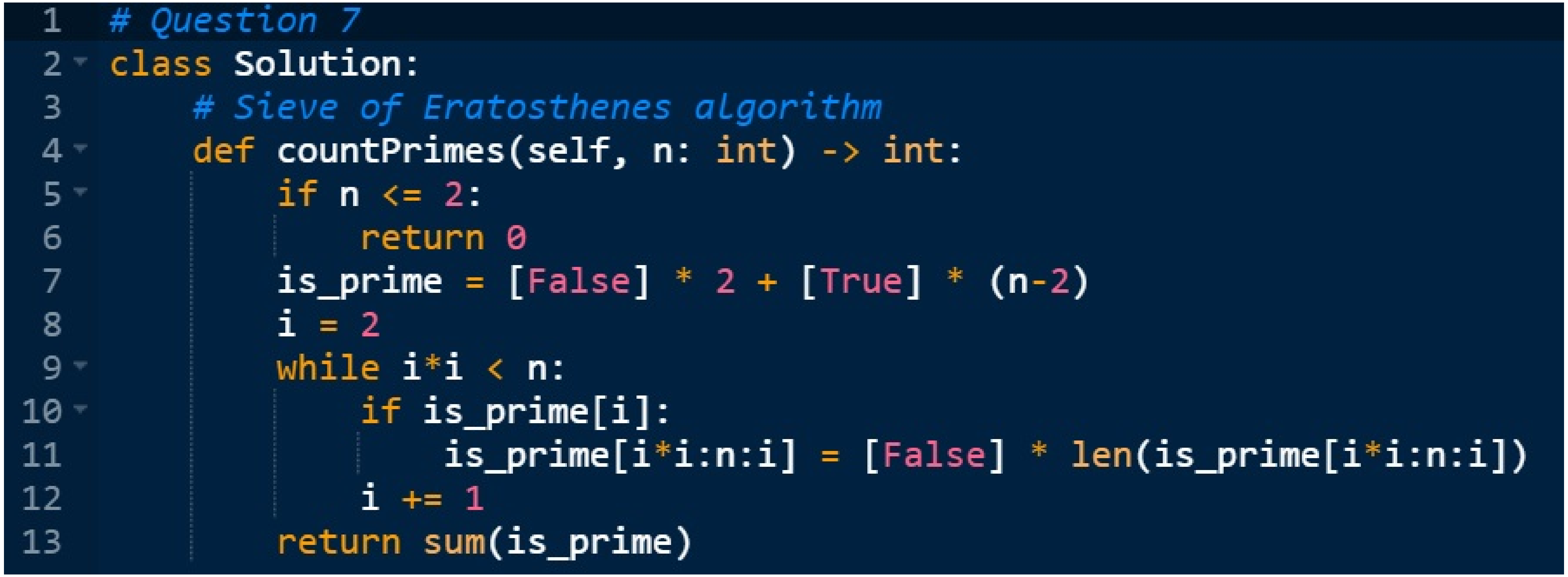
```python
# Question 4
class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:

        intervals.sort(key=lambda x: x[0])
        merged = []
        for interval in intervals:
            if not merged or merged[-1][1] < interval[0]:
                merged.append(interval)
            else:
                merged[-1][1] = max(merged[-1][1], interval[1])

        return merged
```



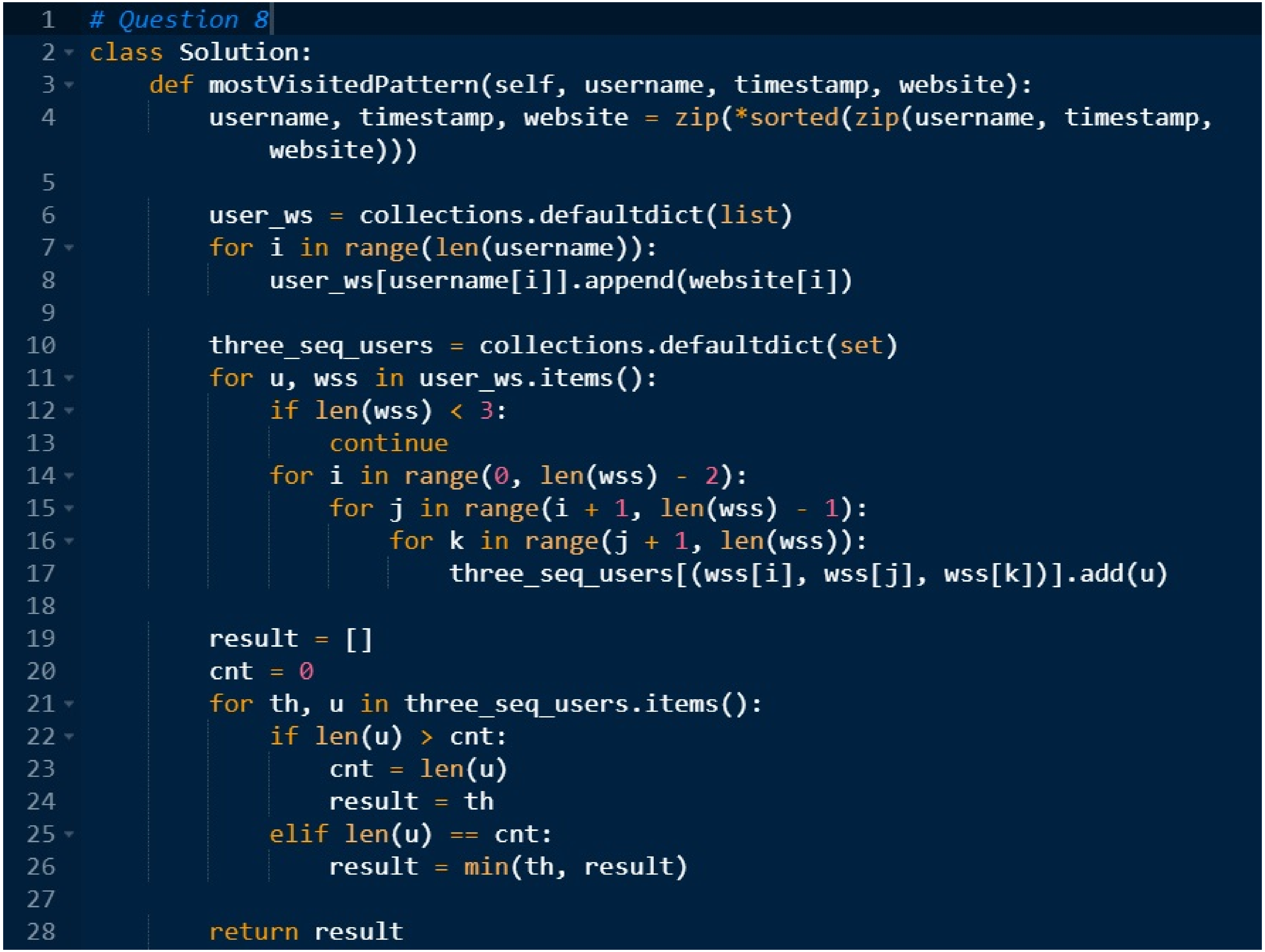
```python
# Question 5
class Solution:
    def twoSumLessThanK(self, A: List[int], K: int) -> int:
        mx = -1
        A = sorted(A)
        print (A)
        i, j = 0, len(A)-1
        while i < j:
            if A[i]+A[j] >= K :

                j = j-1
            else :

                if A[i]+A[j] > mx :
                    mx = A[i]+A[j]
                i = i +1
        return mx
```



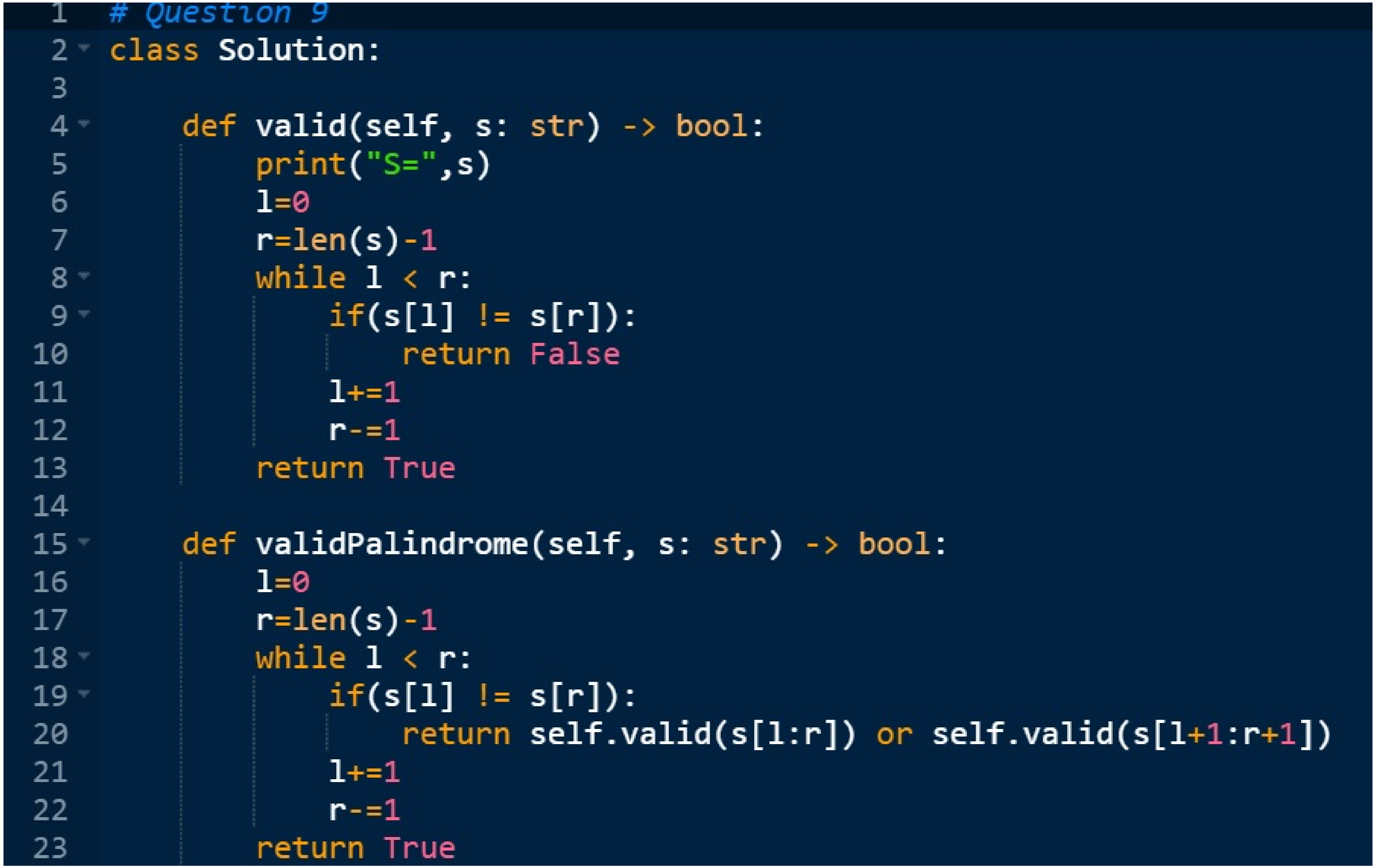
```python
# Question 6
class Solution(object):
    def sortedSquares(self, A):
        N = len(A)
        j = 0
        while j < N and A[j] < 0:
            j += 1
        i = j - 1

        ans = []
        while 0 <= i and j < N:
            if A[i]**2 < A[j]**2:
                ans.append(A[i]**2)
                i -= 1
            else:
                ans.append(A[j]**2)
                j += 1

        while i >= 0:
            ans.append(A[i]**2)
            i -= 1
        while j < N:
            ans.append(A[j]**2)
            j += 1

        return ans
```



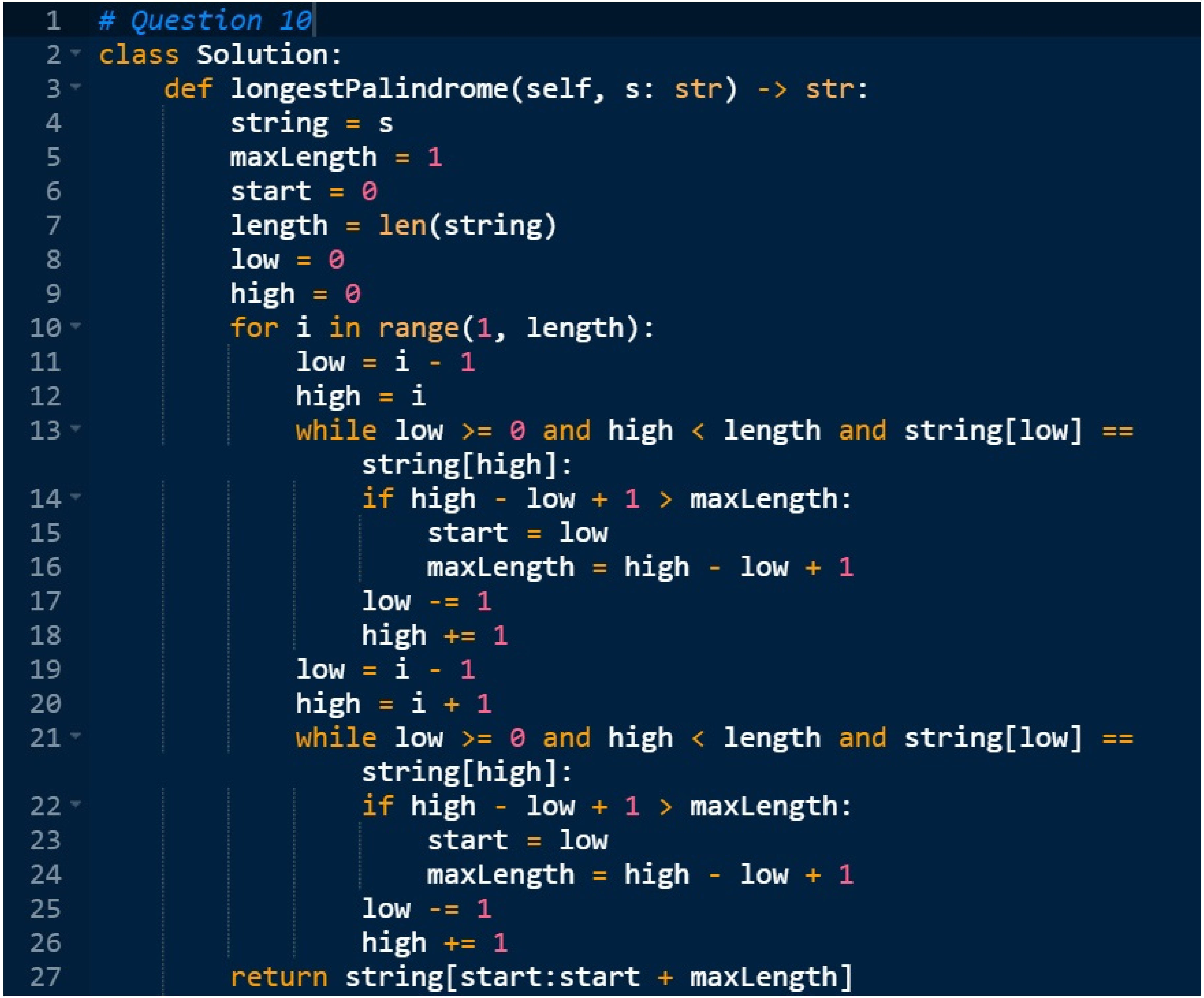
```python
# Question 7
class Solution:
    # Sieve of Eratosthenes algorithm
    def countPrimes(self, n: int) -> int:
        if n <= 2:
            return 0
        is_prime = [False] * 2 + [True] * (n-2)
        i = 2
        while i*i < n:
            if is_prime[i]:
                is_prime[i*i:n:i] = [False] * len(is_prime[i*i:n:i])
            i += 1
        return sum(is_prime)
```



```python
# Question 8
class Solution:
    def mostVisitedPattern(self, username, timestamp, website):
        username, timestamp, website = zip(*sorted(zip(username, timestamp,
            website)))

        user_ws = collections.defaultdict(list)
        for i in range(len(username)):
            user_ws[username[i]].append(website[i])

        three_seq_users = collections.defaultdict(set)
        for u, wss in user_ws.items():
            if len(wss) < 3:
                continue
            for i in range(0, len(wss) - 2):
                for j in range(i + 1, len(wss) - 1):
                    for k in range(j + 1, len(wss)):
                        three_seq_users[(wss[i], wss[j], wss[k])].add(u)

        result = []
        cnt = 0
        for th, u in three_seq_users.items():
            if len(u) > cnt:
                cnt = len(u)
                result = th
            elif len(u) == cnt:
                result = min(th, result)

        return result
```



```python
# Question 9
class Solution:

    def valid(self, s: str) -> bool:
        print("S=",s)
        l=0
        r=len(s)-1
        while l < r:
            if(s[l] != s[r]):
                return False
            l+=1
            r-=1
        return True

    def validPalindrome(self, s: str) -> bool:
        l=0
        r=len(s)-1
        while l < r:
            if(s[l] != s[r]):
                return self.valid(s[l:r]) or self.valid(s[l+1:r+1])
            l+=1
            r-=1
        return True
```



```python
# Question 10
class Solution:
    def longestPalindrome(self, s: str) -> str:
        string = s
        maxLength = 1
        start = 0
        length = len(string)
        low = 0
        high = 0
        for i in range(1, length):
            low = i - 1
            high = i
            while low >= 0 and high < length and string[low] ==
                string[high]:
                if high - low + 1 > maxLength:
                    start = low
                    maxLength = high - low + 1
                low -= 1
                high += 1
            low = i - 1
            high = i + 1
            while low >= 0 and high < length and string[low] ==
                string[high]:
                if high - low + 1 > maxLength:
                    start = low
                    maxLength = high - low + 1
                low -= 1
                high += 1
        return string[start:start + maxLength]
```