



Patricia Roque
Other

PDF generated at: 20 Sep 2024 04:58:31 UTC
[View this report on HackerRank](#)

Score

93.8% • 75 / 80
scored in TIP102: Unit 1 Version A (Standard) - Fall 2024 in 50 min 1 sec on 19 Sep 2024 21:06:50 PDT

Candidate Information

Email	mhyles@bu.edu
Test	TIP102: Unit 1 Version A (Standard) - Fall 2024
Candidate Packet	View
Taken on	19 Sep 2024 21:06:50 PDT
Time taken	50 min 1 sec/ 90 min
Univ/College Name	Boston University
Personal Member ID	111572
Email Address with CodePath	mhyles@bu.edu
Github username with CodePath	itsmhyles
Invited by	CodePath

Skill Distribution



There is no associated skills data that can be shown for this assessment

Tags Distribution







There is no associated tags data that can be shown for this assessment

Questions

Coding Questions • 60.0 / 60.0

Status	No.	Question	Time Taken	Skill	Score
	1	Unique Coding	1 min 54 sec	-	20/20
	2	Needle in a Haystack Coding	2 min 1 sec	-	20/20
	3	Flowerbed Coding	29 min 3 sec	-	20/20

Multiple Choice + Debugging • 15.0 / 20.0

Status	No.	Question	Time Taken	Skill	Score
	4	Find the bug! Coding	11 min 12 sec	-	5/5
	5	What is the output of the following code snippet? Multiple Choice	13 sec	-	0/5
	6	What is the output of the following code snippet? Multiple Choice	1 min 42 sec	-	5/5
	7	What is the output of the following code snippet? Multiple Choice	3 min 48 sec	-	5/5

1. Unique

 Correct

Coding

Question description

Given a string `s`, return `True` if every character in the string is unique. Return `False` if any characters in `s` are repeated.

Example 1

Example Input: `s = "abcdef"`Expected Output: `True`

Example 2

Example Input: s = "aabbcc"

Expected Output: False

Example 3

Example Input: s = ""

Expected Output: True

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11
12  #
13  # Complete the 'has_all_unique_characters' function below.
14  #
15  # The function is expected to return a BOOLEAN.
16  # The function accepts STRING s as parameter.
17  #
18
19  def has_all_unique_characters(s):
20      # Write your code here
21      checker = []
22      for i in range(len(s)):
23          if s[i] not in checker:
24              checker.append(s[i])
25      return len(checker) == len(s)
26
27
28
29  if __name__ == '__main__':
30      input_data = sys.stdin.read().strip().splitlines()
31
32      for line in input_data:
33          result = has_all_unique_characters(line)
34          print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Empty String	Easy	Hidden	Success	0	0.0419 sec	11.1 KB
Single Character String	Easy	Hidden	Success	0	0.0462 sec	11 KB
All Unique Characters	Easy	Hidden	Success	0	0.0435 sec	10.9 KB
Duplicate Characters	Easy	Hidden	Success	0	0.0557 sec	10.9 KB
Mixed Characters	Easy	Hidden	Success	0	0.0526 sec	11 KB
Case Sensitivity	Easy	Hidden	Success	0	0.06 sec	10.9 KB
Special Characters	Easy	Hidden	Success	0	0.0458 sec	10.9 KB
Long String with Unique Characters	Easy	Hidden	Success	0	0.0419 sec	11 KB
String with Spaces	Easy	Hidden	Success	0	0.0414 sec	11.1 KB
String with Repeated Spaces	Easy	Hidden	Success	0	0.0431 sec	11.1 KB

Pass/Fail Case

Easy

Hidden

Success

20

0.037
sec

11 KB

 No comments.

2. Needle in a Haystack

 Correct

Coding

Question description

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

Input: `haystack = "sadbutsad"`, `needle = "sad"`

Output: `0`

Explanation: "sad" occurs twice, starting at indices `0` and `6`.
The first occurrence is at index `0`, so we return `0`.

Example 2:

Input: `haystack = "leetcode"`, `needle = "leeto"`

Output: `-1`

Explanation: "leeto" did not occur in "leetcode", so we return `-1`.

Example 3:

Input: `haystack = "sad"` `needle = "sadbutsad"`

Needle is longer than haystack, so we return `-1`.

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8
9
10
11 #
12 # Complete the 'find_the_needle' function below.
13 #
14 # The function is expected to return an INTEGER.
15 # The function accepts following parameters:
16 # 1. STRING haystack
17 # 2. STRING needle
18 #
19
20 def find_the_needle(haystack, needle):
21     # Write your code here
22     for i in range(len(haystack)):
23         if needle in haystack:
24             return haystack.index(needle)
25         else:
26             return -1
27
28 if __name__ == '__main__':
29     input_data = sys.stdin.read().strip().splitlines()
30
31     for line in input_data:
32         haystack, needle = [part.strip() for part in line.split(',')]
33
34         result = find_the_needle(haystack, needle)
35
36         print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Needle is found in the haystack	Easy	Hidden	Success	0	0.0455 sec	10.5 KB

Needle is not found in the haystack	Easy	Hidden	Success	0	0.0337 sec	10.3 KB
Haystack is an empty string	Easy	Hidden	Success	0	0.0454 sec	10.5 KB
Needle is longer than the haystack	Easy	Hidden	Success	0	0.0358 sec	10.3 KB
Needle is the same as the haystack	Easy	Hidden	Success	0	0.0304 sec	10.4 KB
Needle at the beginning of the haystack	Easy	Hidden	Success	0	0.0328 sec	10.5 KB
Needle at the end of the haystack	Easy	Hidden	Success	0	0.0366 sec	10.5 KB
Needle in the middle of the haystack	Easy	Hidden	Success	0	0.0337 sec	10.4 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0369 sec	10.5 KB

🚫 No comments.

3. Flowerbed

 Correct

Coding

Question description

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in **adjacent** plots.

Given an integer array `flowerbed` containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer `n`, return `True` if `n` new flowers can be planted in the `flowerbed` without violating the no-adjacent-flowers rule and `False` otherwise.

Hint: When deciding where to plant a new flower, focus on each plot in the `flowerbed` and check its neighboring plots. You only need to consider the plot directly before and directly after the current plot to determine if a flower can be planted there. Remember that the flowerbed is linear, so you don't need to worry about wrapping around.

Example 1:

Input: `flowerbed = [1,0,0,0,1]`, `n = 1`

Output: `True`

Example 2:

Input: `flowerbed = [1,0,0,0,1]`, `n = 2`

Output: `False`

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11 #
12 # Complete the 'can_place_flowers' function below.
13 #
```

```
14 # The function is expected to return a BOOLEAN.
15 # The function accepts following parameters:
16 # 1. INTEGER_ARRAY flowerbed
17 # 2. INTEGER n
18 #
19
20 def can_place_flowers(flowerbed, n):
21     # Write your code here
22     comparison = 0
23     last = 0
24     for i in range(len(flowerbed)-1):
25         current, nexts = flowerbed[i], flowerbed[i+1]
26         if i>0 and flowerbed[i-1]>0:
27             last = 1
28         else:
29             last = 0
30
31         if current + nexts + last == 0:
32             comparison +=1
33     if flowerbed[len(flowerbed)-1] + flowerbed[len(flowerbed)-2] == 0:
34         comparison +=1
35
36     return comparison >= n
37
38
39 if __name__ == '__main__':
40     input_data = sys.stdin.read().strip().splitlines()
41
42     for line in input_data:
43         match = re.match(r"(\[.*\]),\s*(\d+)", line)
44         if match:
45             flowerbed_str = match.group(1)
46             n_str = match.group(2)
47
48             flowerbed = ast.literal_eval(flowerbed_str)
49             n = int(n_str)
50             result = can_place_flowers(flowerbed, n)
51             print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Enough empty plots to plant the	Easy	Hidden	Success	0	0.047 sec	11.2 KB

required number of flowers						
Not enough empty plots to plant the required number of flowers	Easy	Hidden	Success	0	0.0694 sec	11.1 KB
Flowerbed is already fully planted	Easy	Hidden	Success	0	0.0387 sec	11.1 KB
Flowerbed is entirely empty	Easy	Hidden	Success	0	0.046 sec	11.1 KB
Edge case with one plot in the flowerbed	Easy	Hidden	Success	0	0.0454 sec	10.9 KB
Edge case with one plot in the flowerbed	Easy	Hidden	Success	0	0.053 sec	10.9 KB
Required number of flowers to plant is zero	Easy	Hidden	Success	0	0.0397 sec	11 KB
Required number of flowers to plant is zero	Easy	Hidden	Success	0	0.0423 sec	11 KB
Flowerbed has alternating empty and planted plots	Easy	Hidden	Success	0	0.0388 sec	11.1 KB

Flowerbed with large number of plots	Easy	Hidden	Success	0	0.0388 sec	11.1 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0407 sec	11.1 KB

! No comments.

4. Find the bug!

✓ Correct

Coding

Question description

The provided code incorrectly implements the function `reverse_lst` which should accept a list `lst` and return the original list with the elements in reverse order.

```
def reverse_lst(lst):
    left = 0
    right = len(lst) - 1

    while left < right:
        lst[left] = lst[right]
        lst[right] = lst[left]
        left -= 1
        right += 1

    return lst

lst = [1, 2, 3, 4, 5]
print(reverse_lst(lst))

lst = [10, 20, 30, 40]
print(reverse_lst(lst))
```

Identify the bug(s) within the given implementation and select the corrected code that will successfully reverse the list.

Candidate's Solution

Language used: Python 3

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import ast
9
10
11
12  #
13  # Complete the 'reverse_lst' function below.
14  #
15  # The function is expected to return an INTEGER_ARRAY.
16  # The function accepts INTEGER_ARRAY lst as parameter.
17  #
18
19  def reverse_lst(lst):
20      left = 0
21      right = len(lst) - 1
22
23
24      while left < right:
25          temp = lst[left]
26          lst[left] = lst[right]
27          lst[right] = temp
28          left += 1
29          right -= 1
30
31      return lst
32
33  if __name__ == '__main__':
34      input_str = sys.stdin.read().strip()
35      # Convert the input string to a list of integers
36      input_list = ast.literal_eval(input_str)
37      # Reverse the list
38      result = reverse_lst(input_list)
39      # Print the reversed list
40      print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0428 sec	11.1 KB

! No comments.

5. What is the output of the following code snippet?

⊗ Incorrect

Multiple Choice

Question description

```
name = "codepath"  
name[0] = "C"  
print(name)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☒ Codepath

☐ Ccodepath

☐ c☐ Throws an error because strings are immutable and characters cannot be changed once the string is created.

 No comments.

6. What is the output of the following code snippet?

 Correct

Multiple Choice

Question description

```
def mystery_function(s):  
    count = 0  
    for i in range(1, len(s)):  
        if s[i] == s[i - 1]:  
            count += 1  
    return count  
  
result = mystery_function("AABBAB")  
print(result)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ 1

☒ 2☐ 3☐ 4 No comments.

7. What is the output of the following code snippet?

 Correct

Multiple Choice

Question description

```
def mystery_function(lst, threshold):  
    total = 0  
    i = 0  
    while i < len(lst) and total + lst[i] <= threshold:  
        total += lst[i]  
        i += 1  
    return total
```

```
result = mystery_function([1, 2, 3, 4, 5], 7)  
print(result)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

☐ 3

☒ 6



☐ 7

☐ 10

 No comments.