93.8

Mary Ashwitha Gopu

Other

PDF generated at: 27 Feb 2025 00:38:16 UTC

View this report on HackerRank

Score

93.8% • 75 / 80

scored in TIP102: Unit 2 Version A (Standard) - Spring 2025 in 21 min 41 sec on 26 Feb 2025 16:15:04 PST

Candidate Information

Email	maryashwithagopu@gmail.com
Test	TIP102: Unit 2 Version A (Standard) - Spring 2025
Candidate Packet	View
Taken on	26 Feb 2025 16:15:04 PST
Time taken	21 min 41 sec/ 90 min
Personal Member ID	110188
Email Address with CodePath	maryashwithagopu@gmail.com
Github username with CodePath	ashwithamary
Invited by	CodePath

Suspicious Activity detected

Code similarity

Code similarity • 1 question

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 • 65 / 65

Status	No.	Question	Time Taken	Skill	Score
✓	1	First Repeating Element Coding	7 min	-	20/20

✓	2	Intersection of Two Arrays Coding	3 min 43 sec	-	20/20
✓	3	Roman To Integers Coding	4 min 12 sec	-	20/20
✓	4	Debugging Coding	2 min 57 sec	-	5/5

Multiple Choice + Debugging • 10 / 15

Status	No.	Question	Time Taken	Skill	Score
✓	5	What will be the output of the following code snippet? Multiple Choice	1 min 36 sec	-	5/5
✗	6	Bart Simpson in Springfield Multiple Choice	1 min 26 sec	-	0/5
✓	7	What does this mystery function do? Multiple Choice	35 sec	-	5/5

1. First Repeating Element

✓ Correct

Coding

Question description

Given an integer array, return the minimum index of a repeating element by doing a single traversal of the array. If there are no repeating elements, return `None`.

Example 1:
Input: [1, 2, 3, 4, 5]
Output: None

Example 2:
Input: [1, 2, 3, 1]
Output: 0

Candidate's Solution Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast


#
# Complete the 'find_min_index_of_repeating' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY arr as parameter.
#

def find_min_index_of_repeating(arr):
    # Write your code here
    seen = set()
    for i,element in enumerate(arr):
        if element in seen:
            return arr.index(element)
        seen.add(element)
    return None

if __name__ == '__main__':
```

```
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_data = sys.stdin.read().strip().splitlines()

    for line in input_data:
        try:
            arr = ast.literal_eval(line.strip())
            result = find_min_index_of_repeating(arr)
            outfile.write(str(result) + '\n')
            # outfile.write(str(result) + '\n')
        except (ValueError, SyntaxError):
            print("Invalid input")
    outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Success	0	0.0303 sec	11.4 KB
Testcase 1	Easy	Hidden	Success	0	0.0282 sec	11.3 KB
Testcase 2	Easy	Hidden	Success	0	0.0289 sec	11.3 KB
Testcase 3	Easy	Hidden	Success	0	0.0318 sec	11.3 KB
Testcase 4	Easy	Hidden	Success	0	0.0293 sec	11.3 KB
Testcase 5	Easy	Hidden	Success	0	0.0282 sec	11.2 KB
Testcase 6	Easy	Hidden	Success	0	0.0306 sec	11.3 KB

Testcase 7	Easy	Hidden	Success	0	0.0332 sec	11.2 KB
Testcase 8	Easy	Hidden	Success	0	0.0298 sec	11.2 KB
Testcase 9	Easy	Hidden	Success	0	0.0295 sec	11.3 KB
Testcase 10	Hard	Hidden	Success	0	0.0281 sec	11.2 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0315 sec	11.3 KB

No comments.

2. Intersection of Two Arrays

Correct

Coding

Question description

Given two integer arrays `nums1` and `nums2`, return an array of their intersection. Each element in the result must be unique. The array must be returned in **ascending** order.

Example 1:
Input: nums1 = [1,2,2,1], nums2 = [2,2]
Output: [2]

Example 2:

Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [4,9]

Candidate's Solution Language used: **Python 3**

```python
#!/bin/python

import math
import os
import random
import re
import sys
import ast
import json

#
# Complete the 'intersection' function below.
#
# The function is expected to return an INTEGER_ARRAY.
# The function accepts following parameters:
#  1. INTEGER_ARRAY nums1
#  2. INTEGER_ARRAY nums2
#

def intersection(nums1, nums2):
    # Write your code here
    ans = set()
    for num1 in nums1:
        if num1 in nums2:
            ans.add(num1)
    ans = sorted(list(ans))
    return ans

if __name__ == '__main__':
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_lines = sys.stdin.read().strip().split('\n')

    results = []
    for line in input_lines:
        input_list = ast.literal_eval(line)
        nums1 = input_list[0]
        nums2 = input_list[1]

```

```
39          result = intersection(nums1, nums2)
40          results.append(result)
41
42      for result in results:
43          outfile.write(str(result) + '\n')
44      outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Success	0	0.0345 sec	11.4 KB
Testcase 1	Easy	Hidden	Success	0	0.0377 sec	11.5 KB
Both arrays empty	Easy	Hidden	Success	0	0.0344 sec	11.4 KB
One array empty	Easy	Hidden	Success	0	0.043 sec	11.5 KB
No intersection	Easy	Hidden	Success	0	0.0324 sec	11.5 KB
Same arrays	Easy	Hidden	Success	0	0.0311 sec	11.4 KB
Arrays with duplicates	Easy	Hidden	Success	0	0.0302 sec	11.4 KB
Single element arrays	Easy	Hidden	Success	0	0.0301 sec	11.4 KB

Intersection with different element orders	Easy	Hidden	Success	0	0.038 sec	11.5 KB
Intersection with different element orders (anotheracceptable answer)	Easy	Hidden	Success	0	0.0373 sec	11.6 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0347 sec	11.6 KB

No comments.

3. Roman To Integers

Correct

Coding

Question description

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two ones added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, there is a special rule: A smaller number may appear before a number to **subtract** from it, in specific circumstances. For example, instead of `IIII`, the number four is written as `IV`, or "five minus one". Here's a full list of possible subtractions:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Example 1:
Input: s = "III"
Output: 3
Explanation: III = 3.

Example 2:
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.

Example 3:
Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

Candidate's Solution Language used: **Python 3**

```
#!/bin/python3

import math
import os
import random
import re
import sys


#
# Complete the 'roman_to_integer' function below.
```

```
#
# The function is expected to return an INTEGER.
# The function accepts STRING s as parameter.
#

def roman_to_integer(s):
    # Write your code here
    roman = {'I':1, 'V':5, 'X':10, 'L': 50, 'C': 100, 'D':500, 'M':1000}
    total =0
    prev=0

    for char in reversed(s):
        curr = roman[char]
        if curr< prev:
            total -= curr
        else:
            total += curr
        prev=curr

    return total

if __name__ == "__main__":
    # Read all input
    input_data = sys.stdin.read().strip().split("\n")
    results = []

    for line in input_data:
        if not line.strip():  # If the input line is empty
            results.append(0)  # Return 0 for empty strings
            continue

        try:
            # Process Roman numeral string
            roman_string = line.strip()

            # Redirect debugging output to stderr
            original_stdout = sys.stdout
            try:
                sys.stdout = sys.stderr  # Redirect stdout to stderr for debugging prints
                result = roman_to_integer(roman_string)
            finally:
                sys.stdout = original_stdout  # Restore stdout

            # Append the result
            results.append(result)
```

```
        except KeyError:
            # Handle invalid Roman numeral input
            results.append("Invalid Roman numeral")

    # Print all results (one per line)
    for res in results:
        print(res)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Success	0	0.0234 sec	10.8 KB
Testcase 1	Easy	Hidden	Success	0	0.0258 sec	10.8 KB
Testcase 3	Easy	Hidden	Success	0	0.0242 sec	10.8 KB
Testcase 4	Easy	Hidden	Success	0	0.0313 sec	10.8 KB
Pass/Fail Testcases	Easy	Hidden	Success	20	0.0235 sec	10.8 KB

No comments.

4. Debugging

Correct

Coding

Question description

The provided incorrectly implements the function `get_top_player`. When correctly implemented, `get_top_player` should accept a dictionary which maps player names to their score and return the name of the highest scoring player. You are guaranteed all scores will be unique.

Identify any bug(s) within the given implementation and correct the code so that it successfully passes the provided test cases.

Candidate's Solution Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast



#
# Complete the 'get_top_player' function below.
#
# The function is expected to return a STRING.
#

def get_top_player(dictionary):
  high_score = 0
  top_player = ""

  for name, score in dictionary.items():
    if score >= high_score:
      high_score = score
    #   print(high_score)
      top_player = name
    #   print(top_player)

  return top_player
if __name__ == '__main__':
    input_data = sys.stdin.read().strip()
    dictionary = ast.literal_eval(input_data)

    result = get_top_player(dictionary)
    print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0341 sec	11.3 KB

No comments.

5. What will be the output of the following code snippet?

Correct

Multiple Choice

Question description

```
word = "encourage"
char_count = {}
for char in word:
    if char not in char_count:
        char_count[char] = 1
    else:
        char_count[char] += 1
char_count['e'] += 2

print(char_count['e'])
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ 2

- 3
- 4 ✓
- Causes a KeyError because the code tries to assign a value to a key in the dictionary that does not exist

No comments.

6. Bart Simpson in Springfield

Incorrect

Multiple Choice

Question description

Given the dictionary `bart` below, which of the following options would print `"Springfield"` to the console?

```
bart = {"first name": "Bart", "last name": "Simpson", "age": 10, "hometown": "Springfield"}
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [x] print(bart["hometown"])<!-- notionvc: 79a56a09-b46c-46da-87f1-07cf58ff6c45 -->
- [ ] print(bart.get(”hometown”))<!-- notionvc: b28cd1c8-ecc1-43a1-9269-175efacbb590 -->
- [ ] print(bart.pop(”Springfield”))<!-- notionvc: 3165c580-f5da-48e3-adea-0e495d239be7 -->
- [ ] Both A and B

No comments.

7. What does this mystery function do?

Correct

Multiple Choice

Question description

```
def mystery_function(old_dictionary):
    new_dictionary = {}

    for key, value in old_dictionary.items():
        new_dictionary[value] = key
```

```
    return new_dictionary

# Example usage
old_dictionary = {'a': 1, 'b': 2, 'c': 3}
new_dictionary = mystery_function(old_dictionary)
print(new_dictionary)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [x] ** It swaps the keys and values from the original dictionary. Example: <code>{'a': 1, 'b': 2, 'c': 3}</code> becomes <code>{1: 'a', 2: 'b', 3: 'c'}</code>. <!-- notionvc: cbf80623-1bd0-4a68-8896-cc2927697ef3 --> **

- [ ] It doubles the values in the dictionary and keeps the keys the same. Example: <code>{'a': 1, 'b': 2, 'c': 3}</code> becomes <code>{'a': 2, 'b': 4, 'c': 6}</code>. <!-- notionvc: 793b7521-2eda-46f7-afb8-9adab447b127 -->

- [ ] It concatenates the keys and values into a single string for each key-value pair. Example: <code>{'a': 1, 'b': 2, 'c': 3}</code> becomes <code>{'a1': 'a1', 'b2': 'b2', 'c3': 'c3'}</code>. <!-- notionvc: 588aa935-0cdb-4d53-a15d-21889e094e22 -->

It sorts the dictionary by keys in ascending order. Example: <code> {'b': 2, 'a': 1, 'c': 3}</code> becomes <code> {'a': 1, 'b': 2, 'c': 3}</code>. <!-- notionvc: f8416ee2-de4a-403e-be2a-bdd5280d70aa -->

No comments.