91.7

ryan

Other

PDF generated at: 17 Jun 2024 17:08:54 UTC

View this report on HackerRank

Score

91.7% • 55 / 60

scored in CodePath TIP101: Unit 2 Assessment, Version A - Summer 2024 in 36 min 23 sec on 17 Jun 2024 09:30:26 PDT

Candidate Information

Email	concepting@protonmail.com
Test	CodePath TIP101: Unit 2 Assessment, Version A - Summer 2024
Candidate Packet	View
Taken on	17 Jun 2024 09:30:26 PDT
Time taken	36 min 23 sec/ 90 min
Work Experience	< 1 years
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

Status	No.	Question	Time Taken	Skill	Score
✓	1	What will be the output of the following Python code? Multiple Choice	7 sec	-	5/5
✗	2	What will be the output of the following Python code? Multiple Choice	8 sec	-	0/5

✓	3	What will be the output of the following Python code? Multiple Choice	8 sec	-	5/5
✓	4	Get Top Player Multiple Choice	46 sec	-	5/5
✓	5	Contains Duplicate Coding	25 min 6 sec	-	20/20
✓	6	Element Frequency Greater than N Coding	8 min 57 sec	-	20/20

1. What will be the output of the following Python code?

✓ Correct

Multiple Choice

Question description

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
print(my_dict['b'])
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ 1

- [x] 2
- [ ] {'b': 2}
- [ ] KeyError

No comments.

2. What will be the output of the following Python code?

Incorrect

Multiple Choice

Question description

```
my_list = ['a', 'b', 'c', 'd']
print(my_list[1:3])
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [ ] ['a', 'b']
- [ ] ['b', 'c', 'd']

['b', 'c']

['c', 'd']

No comments.

3. What will be the output of the following Python code?

Correct

Multiple Choice

Question description

```
my_dict = {'a': [1, 2], 'b': [3, 4], 'c': [5, 6]}

output = []

for item in my_dict:
    output.append(item)

print(output)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

[[1, 2], [3, 4], [5, 6]]

[1, 2, 3, 4, 5, 6]

['a', 'b', 'c']

[1, 3, 5]

No comments.

4. Get Top Player

Correct

Multiple Choice

Question description

The following function accepts a dictionary which maps player names to their score and wants to return the name of the highest scoring player. However, it's not working as intended. How do we modify this function to actually return the player with highest score.

```
dictionary = {"Audrey": 90, "Char": 60, "Mario": 95, "Kyra": 12}
# Expected Output: "Mario"


def get_top_player(dictionary):
  high_score = 0
  top_player = ""
  for name, score in dictionary.items():
    if score > high_score:
      high_score += score
      top_player = name
  return top_player
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [x] Replace high_score += score with high_score = score
- [ ] Replace top_player = "" with top_player = dictionary[0]
- [ ] Replace return top_player with return high_score
- [ ] Replace top_player = name with name = top_player

No comments.

5. Contains Duplicate

Correct

Coding

Question description

Given an integer list **nums**, return **True** if any value appears **at least twice** in the list, and return **False** if every element is distinct.

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

Example 2:
Input: [1, 2, 3, 4]
Output: False

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

```python
#!/bin/python3

import math
import os
import random
import re
import sys



#
# Complete the 'contains_duplicate' function below.
#
# The function is expected to return a BOOLEAN.
# The function accepts INTEGER_ARRAY nums as parameter.
#

def contains_duplicate(nums):
    # Write your code here
    new_set = set()

    for i in range(len(nums)):
        if nums[i] in new_set:
            return True
        new_set.add(nums[i])
    return False

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    temp = input()

```

```
    if len(temp) > 40:
        input_string = temp
        chunks = input_string.split(", ")
        list_of_lists = [list(map(int, chunk.split())) for chunk in chunks]
        result = [contains_duplicate(lst) for lst in list_of_lists]
    else:
        result = contains_duplicate([int(n) for n in temp.split()])

    fptr.write(str(result) + '\n')

    fptr.close()

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	0	0.0326 sec	10.4 KB
Testcase 1	Easy	Sample	Success	0	0.0326 sec	10.3 KB
Testcase 2	Easy	Hidden	Success	0	0.0295 sec	10.4 KB
Testcase 3	Easy	Hidden	Success	0	0.0295 sec	10.2 KB
Testcase 4	Easy	Hidden	Success	20	0.0307 sec	10.2 KB

No comments.

6. Element Frequency Greater than N

Correct

Coding

Question description

Given a list of integers `nums` and an integer **n**, return a dictionary with elements as keys and their frequencies as values, but only include elements whose frequency is greater than **n**.

Example 1:
Input: nums = [1, 1, 2, 3, 3, 3, 4], n = 1
Output: {1: 2, 3: 3}

Example 2:
Input: nums = [1, 2, 3, 4, 5], n = 0
Output: {1: 1, 2: 1, 3: 1, 4: 1, 5: 1}

Candidate's Solution

Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys



#
# Complete the 'frequency_greater_than_n' function below.
#
# The function is expected to return a DICTIONARY.
# The function accepts following parameters:
#  1. INTEGER_ARRAY nums
#  2. INTEGER n
#

def frequency_greater_than_n(nums, n):
    # Write your code here
    frequency = {}

```

```
    for num in nums:
        if num in frequency:
            frequency[num] += 1
        else:
            frequency[num] = 1

    result = {}

    for key, value in frequency.items():
        if value > n:
            result[key] = value

    return result

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = input()

    if len(t) > 65:
        chunks_in_range = t.split(", ")
        list_of_lists_in_range = [list(map(int, chunk.split())) for chunk in chunks_in_range]
        result = [frequency_greater_than_n(lst[1:], lst[0]) for lst in list_of_lists_in_range]
    else:
        temp = ([int(n) for n in t.split()])
        result = frequency_greater_than_n(temp[1:], temp[0])

    fptr.write(str(result) + '\n')

    fptr.close()

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	0	0.0261 sec	10.2 KB
Testcase 1	Easy	Sample	Success	0	0.043 sec	10.3 KB

Testcase 2	Easy	Hidden	Success	0	0.0325 sec	10.3 KB
Testcase 3	Easy	Hidden	Success	0	0.039 sec	10.3 KB
Testcase 4	Easy	Hidden	Success	20	0.0403 sec	10.1 KB

No comments.