



You can view this report online at : <https://www.hackerrank.com/x/tests/1742860/candidates/57752105/report>

Full Name: Instructor
Email: aisha.batool@sse.habib.edu.pk
Test Name: CS101 - PW12 - Fall23
Taken On: 5 Nov 2023 19:22:35 PKT
Time Taken: 10 min 33 sec/ 10080 min
Student Roll Number: 02163
Section: N/A
Invited by: Aisha
Skills Score:

Tags Score:

CS101	20/20
Hard	10/10
Input	10/10
Iteration	10/10
Lists	10/10
NestedLists	10/10
Python 3	10/10
Strings	10/10

100%

70/70

scored in **CS101 - PW12 - Fall23** in 10 min 33 sec on 5 Nov 2023 19:22:35 PKT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Sum > Coding	2 min 54 sec	10/ 10	✓
Q2	Basic Statistics > Coding	33 sec	10/ 10	✓
Q3	Anatomy of a Number > Coding	34 sec	10/ 10	✓
Q4	Distinct > Coding	32 sec	10/ 10	✓
Q5	Prime Factorization Theorem > Coding	1 min 7 sec	10/ 10	✓
Q6	Treasure hunt > Coding	1 min 48 sec	10/ 10	✓
Q7	Edit Distance > Coding	37 sec	10/ 10	✓
Q8	Difficulty Meter > Multiple Choice	4 sec	0/ 0	✓



Correct Answer

Score 10

Sum > Coding

QUESTION DESCRIPTION

Write a Python function `sum_lst(x)` to sum all the items in a list.

```
>> sum_lst([1, 6, 7, 9])
23
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 # Complete the sum_lst function below.
3 def sum_lst(x):
4     sumi = 0
5     for i in x:
6         sumi+= i
7     return sumi
8
9
10
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2.5	0.2221 sec	10.7 KB
Testcase 1	Easy	Sample case	✔ Success	2.5	0.1337 sec	10.7 KB
Testcase 2	Easy	Hidden case	✔ Success	2.5	0.1179 sec	10.8 KB
Testcase 3	Easy	Hidden case	✔ Success	2.5	0.0728 sec	10.8 KB

No Comments

QUESTION 2



Correct Answer

Score 10

Basic Statistics > Coding

QUESTION DESCRIPTION

Problem

Mean, *median*, and *mode* are different measures of center in a numerical data set. They each try to summarize a dataset with a single number to represent a "typical" data point from the dataset.

Mean: The "average" number; found by adding all data points and dividing by the number of data points.

Example: The mean of 4, 1, and 7 is $(4+1+7)/3=12/3=4$

Median: The middle number; found by ordering all data points and picking out the one in the middle (or if there are two middle numbers, taking the mean of those two numbers).

Example: The median of 4, 1, and 7 is 4 because when the numbers are put in order (1, 4, 7), the number 4 is in the middle.

Mode: The most frequent number—that is, the number that occurs the highest number of times.

Example: The mode of {4, 2, 4, 3, 2, 2} is 2 because it occurs three times, which is more than any other number.

-- [Khan Academy](#)

Write the functions `mean` `median` `mode` that return the corresponding quantity for their parameter list.

Sample

```
>>> mean([1,4,7])
4.0
>>> median([1,4,7])
4
>>> mode([4,2,4,3,2,2])
2
```

Input Format

The first line contains a number followed by a space separated list of numbers on the next line. The value of the number on the first line determines the operation to be performed on the list as follows.

- 1 : the mean is to be found
- 2 : the median is to be found
- 3 : the mode is to be found

INTERVIEWER GUIDELINES

Solution

```
def mean(lst):
    return sum(lst)/len(lst)

def median(t):
    t = sorted(t)
    mid = len(t)//2
    if len(t)%2 == 0:
        return (t[mid-1]+t[mid])/2
    return t[mid]

def mode(t):
    elems = []
    for n in t:
        if n not in elems:
            elems.append(n)
    freq = [t.count(n) for n in elems]
    max_frq = freq[0]
    max_idx = 0
    for i in range(1,len(freq)):
        frq = freq[i]
        if frq > max_frq:
            max_frq = frq
            max_idx = i
    return elems[max_idx]

if select == 1:
    print(mean(lst))
elif select == 2:
    print(median(lst))
elif select == 3:
    print(mode(lst))
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 def mean(lst):
3     return sum(lst)/len(lst)
```

```

4
5 def median(t):
6     t = sorted(t)
7     mid = len(t)//2
8     if len(t)%2 == 0:
9         return (t[mid-1]+t[mid])/2
10    return t[mid]
11
12 def mode(t):
13     elems = []
14     for n in t:
15         if n not in elems:
16             elems.append(n)
17     freq = [t.count(n) for n in elems]
18     max_frq = freq[0]
19     max_idx = 0
20     for i in range(1, len(freq)):
21         frq = freq[i]
22         if frq > max_frq:
23             max_frq = frq
24             max_idx = i
25     return elems[max_idx]
26
27 if select == 1:
28     print(mean(lst))
29 elif select == 2:
30     print(median(lst))
31 elif select == 3:
32     print(mode(lst))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	1.1	0.0432 sec	9.69 KB
Testcase 1	Easy	Sample case	✔ Success	1.1	0.0602 sec	9.74 KB
Testcase 2	Easy	Sample case	✔ Success	1.2	0.0994 sec	9.51 KB
Testcase 3	Easy	Hidden case	✔ Success	1.1	0.0938 sec	9.98 KB
Testcase 4	Easy	Hidden case	✔ Success	1.1	0.0471 sec	9.56 KB
Testcase 5	Easy	Hidden case	✔ Success	1.1	0.0409 sec	9.65 KB
Testcase 6	Easy	Hidden case	✔ Success	1.1	0.064 sec	9.72 KB
Testcase 7	Easy	Hidden case	✔ Success	1.1	0.0579 sec	9.81 KB
Testcase 8	Easy	Hidden case	✔ Success	1.1	0.0873 sec	9.59 KB

No Comments

QUESTION 3



Correct Answer

Score 10

Anatomy of a Number > Coding

Lists

NestedLists

CS101

QUESTION DESCRIPTION

Problem

An n -digit number is the *weighted sum* of each of its individual digits, e.g. $42 = 4 \cdot 10^1 + 2 \cdot 10^0$. Write a function named `breakdown` that returns a string containing the breakdown of its argument, `n`.

Similarly, a list of n digits can be put together to form a single n -digit number, e.g. $[4, 2] \rightarrow 42$. Write a function, `buildup` that takes a list of digits and returns a corresponding number composed of the digits.

Sample

```
>>> breakdown(42)
'42 = 4*10^1 + 2*10^0'
>>> buildup([4,2])
'42'
```

Input Format

The first line contains a number whose value determines the format of the next line as follows.

- 1 : the next line contains a single number to be broken down
- 2 : the next line contains a space separated list of digits that need to be composed into a single number

Constraints

All numbers appearing in the input are non-negative.

INTERVIEWER GUIDELINES

Solution

```
def breakdown(num):
    num = str(num)
    n = len(num)
    lst = []
    for i in range(n):
        c = num[i]
        lst += [c + '*10^'+str(n-1-i)]
    return num + ' = ' + ' + '.join(lst)

def buildup(lst):
    return int(''.join([str(n) for n in lst]))
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def breakdown(num):
2     num = str(num)
3     n = len(num)
4     lst = []
5     for i in range(n):
6         c = num[i]
7         lst += [c + '*10^'+str(n-1-i)]
8     return num + ' = ' + ' + '.join(lst)
9
10 def buildup(lst):
11     return int(''.join([str(n) for n in lst]))
12
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	1.25	0.0531 sec	9.39 KB
Testcase 1	Easy	Sample case	✔ Success	1.25	0.0408 sec	9.74 KB
Testcase 2	Easy	Sample case	✔ Success	1.25	0.0585 sec	9.52 KB
Testcase 3	Easy	Hidden case	✔ Success	1.25	0.0491 sec	9.62 KB
Testcase 4	Easy	Hidden case	✔ Success	1.25	0.0501 sec	9.72 KB

Testcase 5	Easy	Hidden case	✔ Success	1.25	0.0403 sec	9.52 KB
Testcase 6	Easy	Hidden case	✔ Success	1.25	0.0467 sec	9.45 KB
Testcase 7	Easy	Hidden case	✔ Success	1.25	0.0772 sec	9.54 KB

No Comments

QUESTION 4



Correct Answer

Score 10

Distinct > Coding

QUESTION DESCRIPTION

Write a Python function `distinct(L)` that returns a list containing the elements in the list, `L`, that do not repeat.

```
>> distinct([1, 6, 7, 6])
[1, 7]

>> distinct([1, 6, 1, 6])
[]
```

INTERVIEWER GUIDELINES

```
def distinct(L):
    lst = []
    for i in L:
        if L.count(i)==1:
            lst.append(i)
    return lst
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 def distinct(L):
3     lst = []
4     for i in L:
5         if L.count(i)==1:
6             lst.append(i)
7     return lst
8
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	5	0.0862 sec	10.7 KB
Testcase 1	Easy	Hidden case	✔ Success	5	0.0772 sec	10.7 KB

No Comments

QUESTION 5

Prime Factorization Theorem > Coding



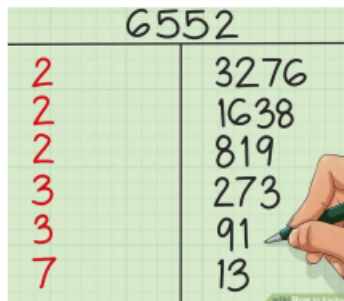
Correct Answer

Score 10

QUESTION DESCRIPTION

A special number is a number that is equal to the sum of its prime factors.

"A prime number can only be divided by 1 or itself, so it cannot be factored any further! Every other whole number can be broken down into prime number factors. It is like the Prime Numbers are the basic building blocks of all numbers." - [Math is fun](#)



Write a function 'prime_factors' that takes as parameter 'num', a positive integer, and returns a list containing its prime factors.

```
>>> prime_factors(6552)
[2, 2, 2, 3, 3, 7, 13]
>>> prime_factors(1567)
[1567]
```

INTERVIEWER GUIDELINES

Solution

```
import math
def prime_factors(n):
    prime_factors = []
    # Extract all factors of n starting from 2. No new factors are to
    # be found beyond sqrt(n).
    for factor in range(2, int(math.sqrt(n))+1):
        # Store factor if it divides n and update n.
        q, r = divmod(n, factor)
        while r == 0:
            prime_factors.append(factor)
            n = q
            q, r = divmod(n, factor)
    # A composite n be updated to 1 when the loop exits. A prime n
    # would remain unaffected. If so, store n as its prime factor.
    if n > 1:
        prime_factors.append(n)
    return prime_factors
```

CANDIDATE ANSWER

Language used: Python 3

```
1
2 import math
3 def prime_factors(n):
4     prime_factors = []
5     # Extract all factors of n starting from 2. No new factors are to
```

```

6      # be found beyond sqrt(n).
7      for factor in range(2, int(math.sqrt(n))+1):
8          # Store factor if it divides n and update n.
9          q, r = divmod(n, factor)
10         while r == 0:
11             prime_factors.append(factor)
12             n = q
13             q, r = divmod(n, factor)
14         # A composite n be updated to 1 when the loop exits. A prime n
15         # would remain unaffected. If so, store n as its prime factor.
16         if n > 1:
17             prime_factors.append(n)
18     return prime_factors
19

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	3.33	0.0656 sec	9.52 KB
Testcase 1	Easy	Sample case	✔ Success	3.33	0.0825 sec	9.48 KB
Testcase 2	Easy	Hidden case	✔ Success	3.34	0.067 sec	9.89 KB

No Comments

QUESTION 6



Correct Answer

Score 10

Treasure hunt > Coding

CS101

Python 3

QUESTION DESCRIPTION

Background

A treasure hunt is a game in which players follow a series of clues given in a particular order to reach a goal or an object. Each clue points to the location of the next clue.

By assigning each clue a number between 0 and $n - 1$, where n is the number of clues, a list can represent the sequence of clues to follow. The next clue can be found in the position of the current clue. See examples below.

Task

Given a list of clues and a starting position, write a program that visits the starting clue in the list, then visits the next clue that it points to, and so on, until the current clue has already been visited. To mark a clue visited, set its value to *None*. Note that not all clues may be visited in this process.

Function Description

To implement the given task, write the following recursive function:

1. **visit** that takes two arguments: **clues**, **pos**. The function should visit the given position **pos**, if it has not already been visited, by setting its value in the list **clues** to *None*, and then visit the next position, and so on. The function should not print or return anything.

Constraints

- You must use recursion—you may not use *for* or *while* loops, or any other form of iteration.
- You may not reference global variables.
- Your function must not return any value.
- You may not use any helper functions.
- Input and output will be handled by HackerRank—you should not read input or display values yourself.
- all clues in the list are valid positions, that is, they are all in *range(len(clues))*

▼ Input Format For Custom Testing

The first line contains *clues*, a list of numbers. Each number represents the position of the next clue to visit.

The second line contains *pos*, the position of the starting clue to visit.

▼ Sample Case 0

Sample Input For Custom Testing

```
[2, 3, 4, 0, 1]
3
```

Sample Output

```
[None, None, None, None, None]
```

Explanation

Begin by visiting position 3.

Next position to visit is 0. Mark it None. -> [2, 3, 4, None, 1]

Next position to visit is 2. Mark it None. -> [None, 3, 4, None, 1]

Next position to visit is 4. Mark it None. -> [None, 3, None, None, 1]

Next position to visit is 1. Mark it None. -> [None, 3, None, None, None]

Next position to visit is 3. Mark it None. -> [None, None, None, None, None]

Since position 3 is already visited, stop.

▼ Sample Case 1

Sample Input For Custom Testing

```
[2, 0, 1, 4, 3]
4
```

Sample Output

```
[2, 0, 1, None, None]
```

Explanation

Begin by visiting position 4.

Next position to visit is 3. Mark it None. -> [2, 0, 1, 4, None]

Next position to visit is 4. Mark it None. -> [2, 0, 1, None, None]

Since position 4 is already visited, stop.

INTERVIEWER GUIDELINES

```
def visit(clues, pos):
    next_pos = clues[pos]
    t[pos] = None
    if next_pos is not None:
        visit(clues, next_pos)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def visit(clues, pos):
2     next_pos = clues[pos]
```

```

3     clues[pos] = None
4     if next_pos is not None:
5         visit(clues, next_pos)
6

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	1	0.1218 sec	11.6 KB
TestCase 1	Easy	Sample case	✔ Success	1	0.0939 sec	11.8 KB
TestCase 2	Easy	Sample case	✔ Success	1	0.1062 sec	11.7 KB
TestCase 3	Easy	Sample case	✔ Success	1	0.1126 sec	11.6 KB
TestCase 4	Easy	Sample case	✔ Success	1	0.1915 sec	11.6 KB
TestCase 5	Easy	Hidden case	✔ Success	1	0.1028 sec	11.7 KB
TestCase 6	Easy	Hidden case	✔ Success	1	0.0906 sec	11.5 KB
TestCase 7	Easy	Hidden case	✔ Success	1	0.0989 sec	11.8 KB
TestCase 8	Easy	Hidden case	✔ Success	1	0.1331 sec	11.5 KB
TestCase 9	Easy	Hidden case	✔ Success	1	0.0988 sec	11.7 KB

No Comments

QUESTION 7



Correct Answer

Score 10

Edit Distance > Coding

Hard

Strings

Iteration

Input

QUESTION DESCRIPTION

The *edit distance* between 2 strings is a measure of their dissimilarity. A pair of strings with a small distance between them are more similar than a pair with a large distance. Edit distance is typically used in spell correction where words within a small distance of an erroneous word are suggested as replacements. It can also be used to measure the similarity of DNA sequences.

There are various ways to compute the edit distance between 2 strings. We will compute it as follows.

Given strings, `s1` and `s2` with lengths `l1` and `l2` respectively, the edit distance between them is the number of characters that need to be changed in one string to obtain the other. The strings may be of different length, leading to 2 cases.

- `s1` and `s2` are of equal length: the edit distance is the number of differing characters *in the same position* and *regardless of case*.
- `s1` and `s2` are of unequal length: the edit distance is the sum of
 - the difference in length, and
 - the minimum edit distance between the shorter string and any of the longer string's substrings of the same length.

Function Description

Write a function `edit_distance()` which takes `s1` and `s2` as parameters and returns their edit distance.

Constraints

- `isinstance(s1, str)` and `isinstance(s2, str)`
- `len(s1) * len(s2) > 0`

▼ Input Format For Custom Testing

The first line contains `s1` and the second contains `s2`.

The output must be the edit distance between them.

Your function need not take input or print any output.

▼ Sample Case 0

Sample Input For Custom Testing

```
kitten  
KittEn
```

Sample Output

```
0
```

Explanation

This is an instance of Case 1: the strings are of equal length. All positions have the same characters ignoring case.

▼ Sample Case 1

Sample Input For Custom Testing

```
kitten  
mitten
```

Sample Output

```
1
```

Explanation

This is an instance of Case 1: the strings are of equal length. All but 1 positions have the same characters ignoring case.

▼ Sample Case 2

Sample Input For Custom Testing

```
take  
RaTE
```

Sample Output

```
2
```

Explanation

This is an instance of Case 1: the strings are of equal length. All but 2 positions have the same characters ignoring case.

▼ Sample Case 3

Sample Input For Custom Testing

```
space  
pace
```

Sample Output

```
1
```

Explanation

This is an instance of Case 2: the strings are of unequal length. The difference in length is 1, and the substring, *pace*, of the longer string has the minimum edit distance, 0, with the shorter string.

▼ Sample Case 4

Sample Input For Custom Testing

```
pace  
spaces
```

Sample Output

2

Explanation

This is an instance of Case 2: the strings are of unequal length. The difference in length is 2, and the minimum edit distance of any substring of the longer string with shorter one is 0, which is the distance between the substring, *pace*, and the shorter string.

▼ Sample Case 5

Sample Input For Custom Testing

yohsin
university

Sample Output

8

Explanation

This is an instance of Case 2: the strings are of unequal length. The difference in length is 4, and the minimum edit distance of any substring of the longer string with shorter one is 4, which is the distance between the substring, *versit*, and the shorter string.

Hint

Write your function for Case 1 first and then add Case 2.

INTERVIEWER GUIDELINES

Solution

```
def edit_distance(s1, s2):
    s1 = s1.lower()
    s2 = s2.lower()
    l1 = len(s1)
    l2 = len(s2)
    if l1 == l2: # equal length
        distance = 0
        for i in range(l1):
            if s1[i] != s2[i]:
                distance += 1
    else: # unequal length
        # make s1 and l1 correspond to the shorter string
        if l1 > l2:
            tmp = s1
            s1 = s2
            s2 = tmp
            tmp = l1
            l1 = l2
            l2 = tmp
        distance = l2 # large value to begin with.
        # compare s1 with all appropriate substrings of s2, record the
        smallest distance.
        for i in range(l2-l1+1):
            distance = min(distance, edit_distance(s1, s2[i:i+l1]))
        # add the difference in length.
        distance += l2 - l1
    return distance
```

CANDIDATE ANSWER

Language used: **Python 3**

```

2 def edit_distance(s1, s2):
3     s1 = s1.lower()
4     s2 = s2.lower()
5     l1 = len(s1)
6     l2 = len(s2)
7     if l1 == l2: # equal length
8         distance = 0
9         for i in range(l1):
10             if s1[i] != s2[i]:
11                 distance += 1
12     else: # unequal length
13         # make s1 and l1 correspond to the shorter string
14         if l1 > l2:
15             tmp = s1
16             s1 = s2
17             s2 = tmp
18             tmp = l1
19             l1 = l2
20             l2 = tmp
21         distance = l2 # large value to begin with.
22         # compare s1 with all appropriate substrings of s2, record the
23         smallest distance.
24         for i in range(l2-l1+1):
25             distance = min(distance, edit_distance(s1, s2[i:i+l1]))
26         # add the difference in length.
27         distance += l2 - l1
28     return distance

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	1	0.0664 sec	9.4 KB
Testcase 1	Easy	Sample case	✔ Success	1	0.0456 sec	9.61 KB
Testcase 2	Easy	Sample case	✔ Success	1	0.0399 sec	9.51 KB
Testcase 3	Easy	Hidden case	✔ Success	1	0.079 sec	9.16 KB
Testcase 4	Easy	Sample case	✔ Success	1	0.0514 sec	9.44 KB
Testcase 5	Easy	Hidden case	✔ Success	1	0.0547 sec	9.39 KB
Testcase 6	Easy	Hidden case	✔ Success	1	0.0481 sec	9.5 KB
Testcase 7	Easy	Hidden case	✔ Success	1	0.0509 sec	9.26 KB
Testcase 8	Easy	Hidden case	✔ Success	1	0.0455 sec	9.4 KB
Testcase 9	Easy	Hidden case	✔ Success	1	0.032 sec	9.39 KB

No Comments

QUESTION 8



Correct Answer

Score 0



Difficulty Meter > Multiple Choice

QUESTION DESCRIPTION

On a scale of 1 to 5, with 1 being very easy and 5 being extremely challenging, how would you rate this worksheet?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

-  ☒ 1
-  ☐ 2
-  ☐ 3
-  ☐ 4
-  ☐ 5

No Comments

PDF generated at: 5 Nov 2023 14:33:37 UTC