# Data Validation - 90 points (Coding)

A third-party provider uses a custom data format to exchange information with us. We need to validate the input before we process it further.

The character encoding is US-ASCII. Valid characters are all the characters between  $0 \times 20$  (space) and  $0 \times 7E$  (~).

All fields are delimited by '|', and '~' is the escape character. There are only three valid escape sequences:

- '~|' stands for '|'
- '~~' stands for '~'
- '~n' stands for new line

A line represents one record, it must start and end with '|'.

The first line contains the field names. The remaining lines contain the records.

Names can't be empty and must be unique, there is no restriction on values.

If a record has more fields than there are names defined, the last name defined will be used and '\_#' will be appended to the field name where # is the number of extra record starting at 1.

Here is a valid example:

|name|address|~n|Patrick|patrick@test.com|pat@test.com|~n|
Annie||annie@test.com|~n

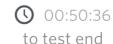
This represents the following data:

name	address	address_1
Patrick	patrick@test.com	pat@test.com
Annie		annie@test.com

Valid input should output statistics about the data: number of records, number



Graduate 2016





message "0:0:0:format\_error"



You have to write a validate function which verifies the input string conforms to the requirements and generates either the expected output or the

1

### **YOUR ANSWER**

3

2

7

5

```
Python 2
                                                            Ö
 Draft saved 01:15 am
    #!/bin/python
 1
 2
 3
    import sys
 4
    import os
 5
 6
 7
    # Complete the function below.
 8 ▼def check segment(input, start, end):
 9
        dic = \{\}
        if input[start] != '|':
10 ▼
             dic[-1] = ''
11
12
            return dic
13
        cleanup = ''
14
        i = start + 1
15 ₩
        while i < end:
16
             if input[i] == '~':
17 ▼
                 if i + 1 < end:
18 ▼
                      if input[i+1] == '~':
                          cleanup += 'a'
19
                          i += 2
20
                      elif input[i+1] == '|':
21 🔻
22
                          cleanup += 'b'
23
                          i += 2
24
                      elif input[i+1] == 'n' and input[i-1]
    == '|':
25
                          dic[i-1] = cleanup[:-1]
                          return dic
26
27 🔻
                      else:
                          dic[-1] = ''
28
29
                          return dic
30 ▼
                 else:
                     dic[-1] = ''
31
                     return dic
32
33 ▼
             elif ord(input[i]) < ord(' ') or ord(input[i])</pre>
    > ord('~'):
34
                 dic[-1] = ''
35
                 return dic
36 ▼
             else:
                 cleanup += input[i]
37
38
                 i += 1
```

```
39
        return dic
40
41
42 ▼def validate( input):
        res = check segment(input, 0, len(input))
43
44
        if -1 in res:
45
            return '0:0:0:format error'
46
        else:
47
            fl end = res.keys()[0]
48
            first line = input[1:fl end]
            fnms = first line.split('|')
49
50
            nmfnms = len(fnms)
51
            last name = fnms[-1]
52
            records = 0
            nmfds = 0
53
54
            nonempty = 0
55
            names = \{\}
            pos = fl end + 3
56
            while pos < len(input):
57 ▼
58
                res = check segment(input, pos, len(input))
59
                if -1 in res:
                     return '0:0:0:format error'
60
61
                pos = res.keys()[0] + 3
62
                line = res.values()[0]
63
                if len(line) == 0:
64
                     return '0:0:0:format error'
65
                print line
66
                fields = line.split('|')
                name = fields[0].strip().lower()
67
68
                if len(name) == 0 or name in names:
69
                     return '0:0:0:format error'
70
                else:
71
                     names[name] = 1
72 ¬
                for f in fields:
73
                     if len(f) > 0:
74
                         nonempty += 1
75
                records += 1
76
                nmfds = max(nmfds, len(fields))
77
            empty = records * nmfds - nonempty
78
            extra = nmfds - nmfnms
79
            if extra > 0:
                last_name += ('_'+str(extra))
80
            return str(records) + ":"+str(nmfds) + ":" +
81
    str(empty) + ":" + last name
82
```

```
83
    f = open(os.environ['OUTPUT PATH'], 'w')
84
85
86
    _input = raw_input()
87
88
    res = validate( input);
89
    f.write(res + "\n")
90
91
    f.close()
92
                                                Line: 63 Col: 31
```

Test against custom input

**Run Code** 

Submit code & Continue

Download sample testcases The input/output files have Unix line endings. Do not use Notepad to edit them on windows.

## Status: Compiled successfully. 3/9 test cases passed.

#### **Testcase 1: Success**

#### **Your Output**

3:3:3:address 1

#### **Expected Output**

3:3:3:address 1

#### **Debug Output**

Patrick|patrick@test.com|pat@test.com
Annie|annie@test.com
Zoe

#### **Testcase 2: Wrong Answer**

#### **Your Output**

Output hidden

#### **Testcase 3: Success**

#### **Your Output**

Graduate 2016:: powered by HackerRank Output hidden **Testcase 4: Wrong Answer Your Output** Output hidden **Testcase 5: Success Your Output** Output hidden **Testcase 6: Wrong Answer Your Output** Output hidden **Testcase 7: Wrong Answer Your Output** Output hidden **Testcase 8: Wrong Answer Your Output** Output hidden **Testcase 9: Wrong Answer** 

**Your Output** 

Output hidden

About Privacy policy Terms of service