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 `0x20` (space) and `0x7E` (~).
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

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

error message.

YOUR ANSWER

Draft saved 01:15 am

Python 2

```
#!/bin/python

import sys
import os


# Complete the function below.
def check_segment(input, start, end):
    dic = {}
    if input[start] != '|':
        dic[-1] = ''
        return dic
    cleanup = ''
    i = start + 1
    while i < end:
        if input[i] == '~':
            if i + 1 < end :
                if input[i+1] == '~':
                    cleanup += 'a'
                    i += 2
                elif input[i+1] == '|':
                    cleanup += 'b'
                    i += 2
                elif input[i+1] == 'n' and input[i-1] == '|':
                    dic[i-1] = cleanup[:-1]
                    return dic
                else:
                    dic[-1] = ''
                    return dic
            else:
                dic[-1] = ''
                return dic
        elif ord(input[i]) < ord(' ') or ord(input[i]) > ord('~'):
            dic[-1] = ''
            return dic
        else:
            cleanup += input[i]
            i += 1
```

```
    return dic


def  validate( input):
    res = check_segment(input, 0, len(input))
    if -1 in res:
        return '0:0:0:format_error'
    else:
        fl_end = res.keys()[0]
        first_line = input[1:fl_end]
        fnms = first_line.split('|')
        nmfnms = len(fnms)
        last_name = fnms[-1]
        records = 0
        nmfds = 0
        nonempty = 0
        names = {}
        pos = fl_end + 3
        while pos < len(input):
            res = check_segment(input, pos, len(input))
            if -1 in res:
                return '0:0:0:format_error'
            pos = res.keys()[0] + 3
            line = res.values()[0]
            if len(line) == 0:
                return '0:0:0:format_error'
            print line
            fields = line.split('|')
            name = fields[0].strip().lower()
            if len(name) == 0 or name in names:
                return '0:0:0:format_error'
            else:
                names[name] = 1
            for f in fields:
                if len(f) > 0:
                    nonempty += 1
            records += 1
            nmfds = max(nmfds, len(fields))
        empty = records * nmfds - nonempty
        extra = nmfds - nmfnms
        if extra > 0:
            last_name += ('_'+str(extra))
        return str(records) + ":"+str(nmfds) + ":" +
str(empty) + ":" + last_name

```

```
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

```
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
```