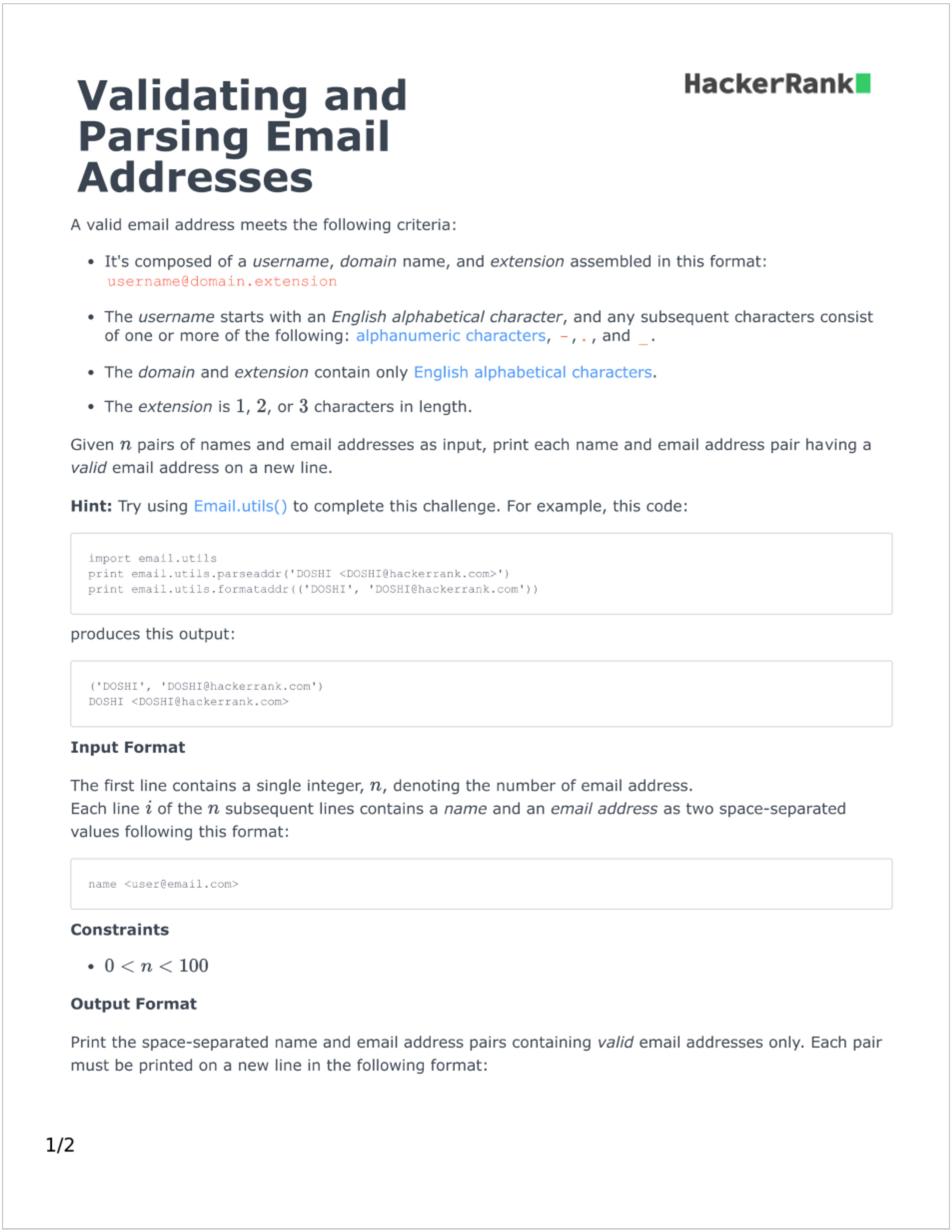
Validating and Parsing Email Addresses

08 June 2025 10:39 AM



Validating and Parsing Email Addresses

HackerRank

A valid email address meets the following criteria:

- It's composed of a *username*, *domain* name, and *extension* assembled in this format: `username@domain.extension`
- The *username* starts with an *English alphabetical character*, and any subsequent characters consist of one or more of the following: alphanumeric characters, `-`, `.`, and `_`.
- The *domain* and *extension* contain only English alphabetical characters.
- The *extension* is 1, 2, or 3 characters in length.

Given n pairs of names and email addresses as input, print each name and email address pair having a *valid* email address on a new line.

Hint: Try using Email.utils() to complete this challenge. For example, this code:

```
import email.utils
print email.utils.parseaddr('DOSHI <DOSHI@hackerrank.com>')
print email.utils.formataddr(('DOSHI', 'DOSHI@hackerrank.com'))
```

produces this output:

```
('DOSHI', 'DOSHI@hackerrank.com')
DOSHI <DOSHI@hackerrank.com>
```

Input Format

The first line contains a single integer, n, denoting the number of email address.
Each line i of the n subsequent lines contains a *name* and an *email address* as two space-separated values following this format:

```
name <user@email.com>
```

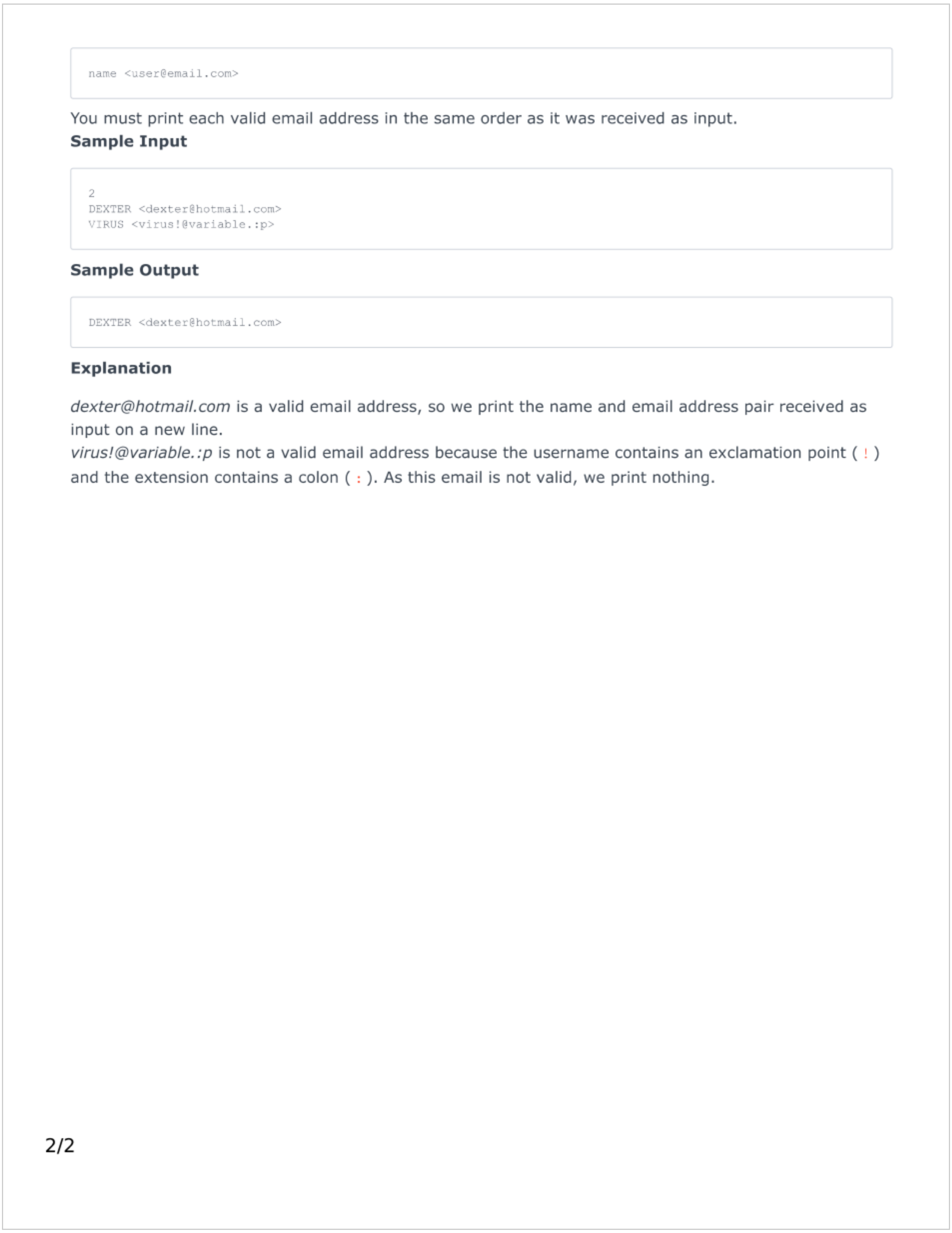
Constraints

- $0 < n < 100$

Output Format

Print the space-separated name and email address pairs containing *valid* email addresses only. Each pair must be printed on a new line in the following format:

1/2



```
name <user@email.com>
```

You must print each valid email address in the same order as it was received as input.

Sample Input

```
2
DEXTER <dexter@hotmail.com>
VIRUS <virus!@variable.:p>
```

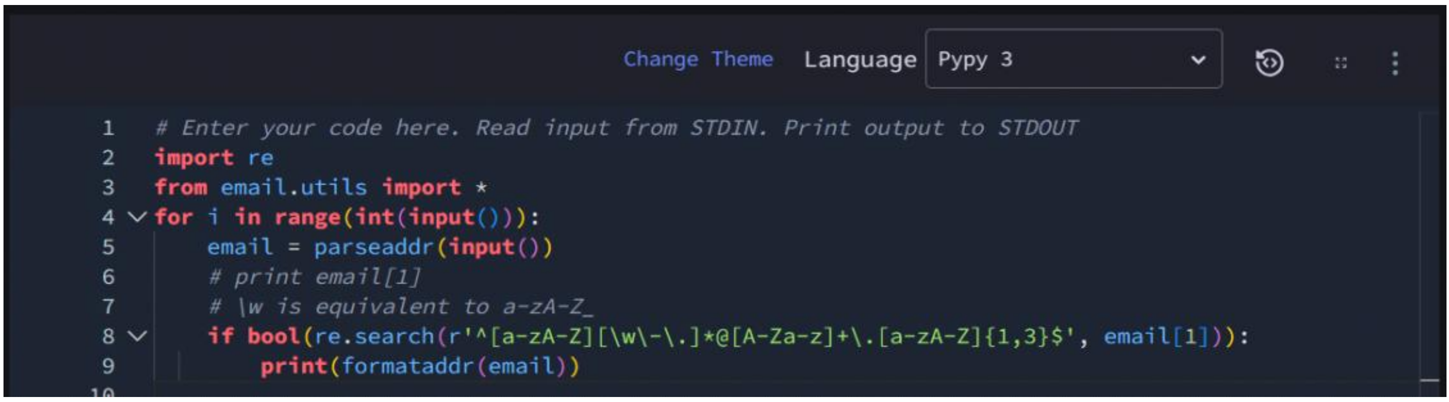
Sample Output

```
DEXTER <dexter@hotmail.com>
```

Explanation

dexter@hotmail.com is a valid email address, so we print the name and email address pair received as input on a new line.
virus!@variable.:p is not a valid email address because the username contains an exclamation point (!) and the extension contains a colon (:). As this email is not valid, we print nothing.

2/2



```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
from email.utils import *
for i in range(int(input())):
    email = parseaddr(input())
    # print email[1]
    # \w is equivalent to a-zA-Z_
    if bool(re.search(r'^[a-zA-Z][\w\-\.]*@[A-Za-z]+\.[a-zA-Z]{1,3}$', email[1])):
        print(formataddr(email))
```