In [1]:

```
# Detect Floating point number
```

In [ ]:

```
import re

class Main():
    def __init__(self):
        self.n = int(input())

        for i in range(self.n):
            self.s = input()
            print(bool(re.match(r'^[-+]?[0-9]*\.[0-9]+$', self.s)))

if __name__ == '__main__':
    obj = Main()
```

In [2]:

```
# Re.split()
```

In [ ]:

```
regex_pattern = r"[.,]"

import re
print("\n".join(re.split(regex_pattern, input())))
```

In [3]:

```
# group(),groups()
```

In [ ]:

```
import re

expression=r"([a-zA-Z0-9])\1+"

m = re.search(expression,input())

if m:
    print(m.group(1))
else:
    print(-1)
```

In [4]:

```
# re.findall()
```

In [ ]:

```
import re

Storage = re.findall(r'(?<=[qwrtypsdfghjklzxcvbnm])([aeiou]{2,})(?=[qwrtypsdfghjklzxcvbnm
if Storage:
    for i in Storage:
        print(i)
else:
    print(-1)
```

In [5]:

```
# re.start(), re.end()
```

In [ ]:

```
import re

S, k = input(), input()
matches = re.finditer(r'(?=(' + k + '))', S)

anymatch = False
for match in matches:
    anymatch = True
    print((match.start(1), match.end(1) - 1))

if anymatch == False:
    print((-1, -1))
```

In [6]:

```
# regex substitution
```

In [ ]:

```
import re

def change(match):
    if match.group(1) == '&&':
        return 'and'
    else:
        return 'or'

for _ in range(int(input())):
    print(re.sub(r"(?<= )(\|\||&&)(?= )", change,input()))
```

In [7]:

```
# Validating Roman numerals
```

In [ ]:

```
regex_pattern = r'M{0,3}(C[MD]|D?C{0,3})(X[CL]|L?X{0,3})(I[VX]|V?I{0,3})$'

import re
print(str(bool(re.match(regex_pattern, input()))))
```

In [8]:

```
# Validating phone numbers
```

In [ ]:

```
import re

N = int(input())

for i in range(N):
    number = input()
    if(len(number)==10 and number.isdigit()):
        output = re.findall(r"^[789]\d{9}$",number)
        if(len(output)==1):
            print("YES")
        else:
            print("NO")
    else:
        print("NO")
```

In [9]:

```
# Validating and parsing email addresses
```

In [ ]:

```
import re
n = int(input())
for _ in range(n):
    x, y = input().split(' ')
    m = re.match(r'<[A-Za-z](\w|-|\.|_)+@[A-Za-z]+\.[A-Za-z]{1,3}>', y)
    if m:
        print(x,y)
```

In [10]:

```
# Hex color code
```

In [ ]:

```
import re

T = int(input())
in_css = False
for _ in range(T):
    s = input()
    if '{' in s:
        in_css = True
    elif '}' in s:
        in_css = False
    elif in_css:
        for color in re.findall('#[0-9a-fA-F]{3,6}', s):
            print(color)
```

In [11]:

```
# html parser
```

In [ ]:

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print ('Start :',tag)
        for ele in attrs:
            print ('->',ele[0],'>',ele[1])

    def handle_endtag(self, tag):
        print ('End   :',tag)

    def handle_startendtag(self, tag, attrs):
        print ('Empty :',tag)
        for ele in attrs:
            print ('->',ele[0],'>',ele[1])

MyParser = MyHTMLParser()
MyParser.feed(''.join([input().strip() for _ in range(int(input()))]))
```

In [12]:

```
# html parser-part2
```

In [ ]:

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):


    def handle_comment(self, data):
        if (len(data.split('\n')) != 1):
            print(">>> Multi-line Comment")
        else:
            print(">>> Single-line Comment")
        print(data.replace("\r", "\n"))
    def handle_data(self, data):
        if data.strip():
            print(">>> Data")
            print(data)


html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

In [13]:

```
# Delete html tags
```

In [ ]:

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        [print('-> {} > {}'.format(*attr)) for attr in attrs]

html = '\n'.join([input() for _ in range(int(input()))])
parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

In [14]:

```
# Validating uid
```

In [ ]:

```
import re

for i in range(int(input())):
    N = input().strip()
    if N.isalnum() and len(N) == 10:
        if bool(re.search(r'(.*[A-Z]){2,}',N)) and bool(re.search(r'(.*[0-9]){3,}',N)):
            if re.search(r'.*(.).*\1+.*',N):
                print('Invalid')
            else:
                print('Valid')
        else:
            print('Invalid')
    else:
        print('Invalid')
```

In [15]:

```
# Validating credit card numbers
```

In [ ]:

```
import re
TESTER = re.compile(
    r"^"
    r"(?!.*(\d)(-?\1){3})"
    r"[456]"
    r"\d{3}"
    r"(?:-?\d{4}){3}"
    r"$")
for _ in range(int(input().strip())):
    print("Valid" if TESTER.search(input().strip()) else "Invalid")
```

In [16]:

```
# Validating postal codes
```

In [ ]:

```
regex_integer_in_range = r"^[1-9][\d]{5}$"
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"

import re
P = input()

print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

In [17]:

```
# Matrix Script
```

In [ ]:

```
import math
import os
import random
import re
import sys


first_multiple_input = input().rstrip().split()

n = int(first_multiple_input[0])

m = int(first_multiple_input[1])

matrix = []

for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)

encoded_string = "".join([matrix[j][i] for i in range(m) for j in range(n)])
pat = r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])'
print(re.sub(pat,' ',encoded_string))
```