Group(), Groups() & Groupdict()

group()

A *group()* expression returns one or more subgroups of the match.

Code

```
>>> import re
>>> m = re.match(r'(\w+)@(\w+)\.(\w+)','username@hackerrank.com')
>>> m.group(0)      # The entire match
'username@hackerrank.com'
>>> m.group(1)      # The first parenthesized subgroup.
'username'
>>> m.group(2)      # The second parenthesized subgroup.
'hackerrank'
>>> m.group(3)      # The third parenthesized subgroup.
'com'
>>> m.group(1,2,3)   # Multiple arguments give us a tuple.
('username', 'hackerrank', 'com')
```

---

groups()

A *groups()* expression returns a tuple containing all the subgroups of the match.

Code

```
>>> import re
>>> m = re.match(r'(\w+)@(\w+)\.(\w+)','username@hackerrank.com')
>>> m.groups()
('username', 'hackerrank', 'com')
```

---

groupdict()

A *groupdict()* expression returns a dictionary containing all the named subgroups of the match, keyed by the subgroup name.

Code

```
>>> m = re.match(r'(?P<user>\w+)@(?P<website>\w+)\.(?P<extension>\w+)','myname@hackerrank.com')
>>> m.groupdict()
{'website': 'hackerrank', 'user': 'myname', 'extension': 'com'}
```

Task

You are given a string S.
Your task is to find the first occurrence of an alphanumeric character in S (read from left to right) that has consecutive repetitions.

Input Format

A single line of input containing the string S.

Constraints

$0 < len(S) < 100$

Output Format

Print the first occurrence of the repeating character. If there are no repeating characters, print `-1`.

Sample Input

```
..12345678910111213141516171820212223
```

Sample Output

```
1
```

Explanation

`..` is the first repeating character, but it is not alphanumeric.
`1` is the first (from left to right) alphanumeric repeating character of the string in the substring `111`.