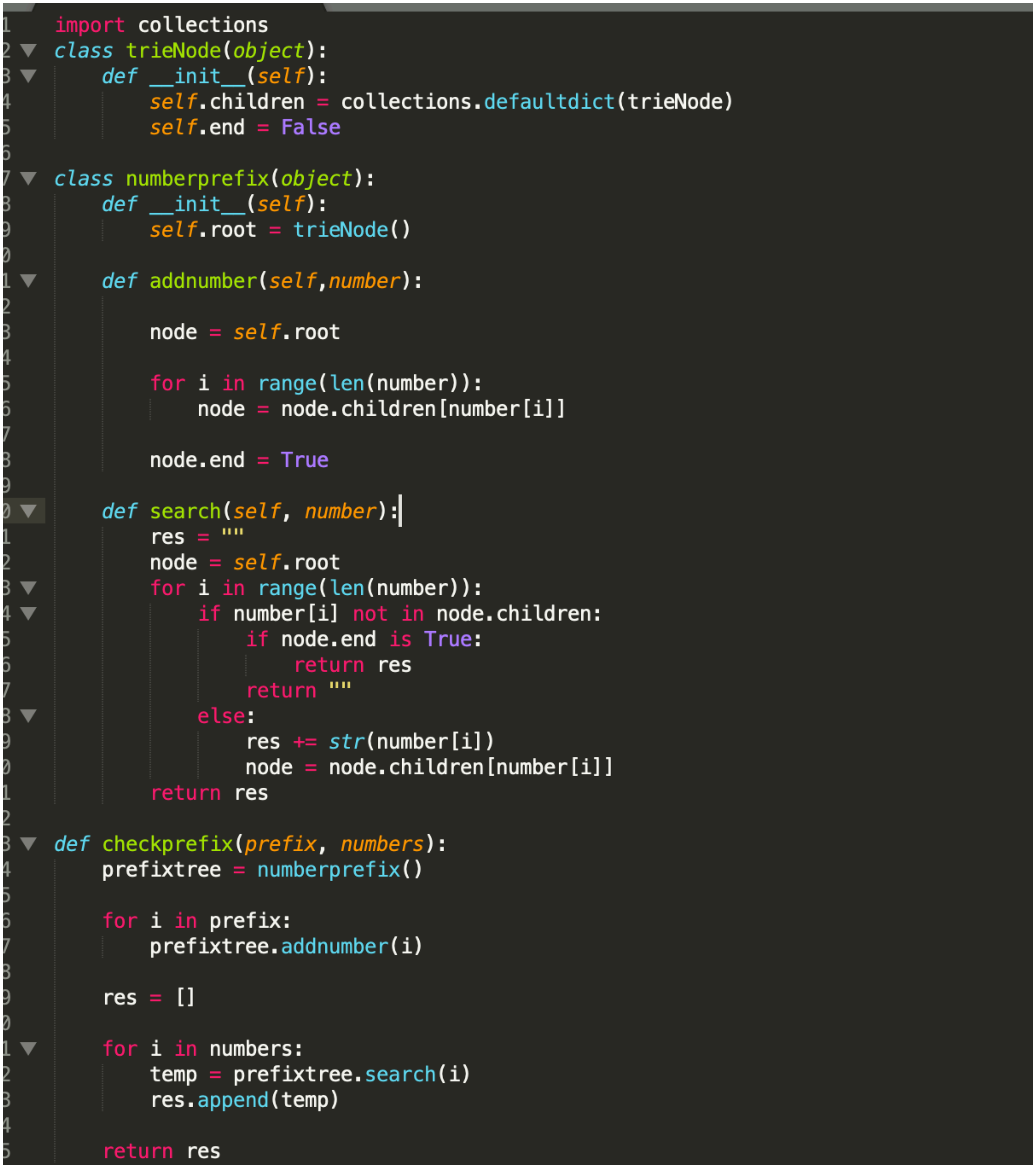
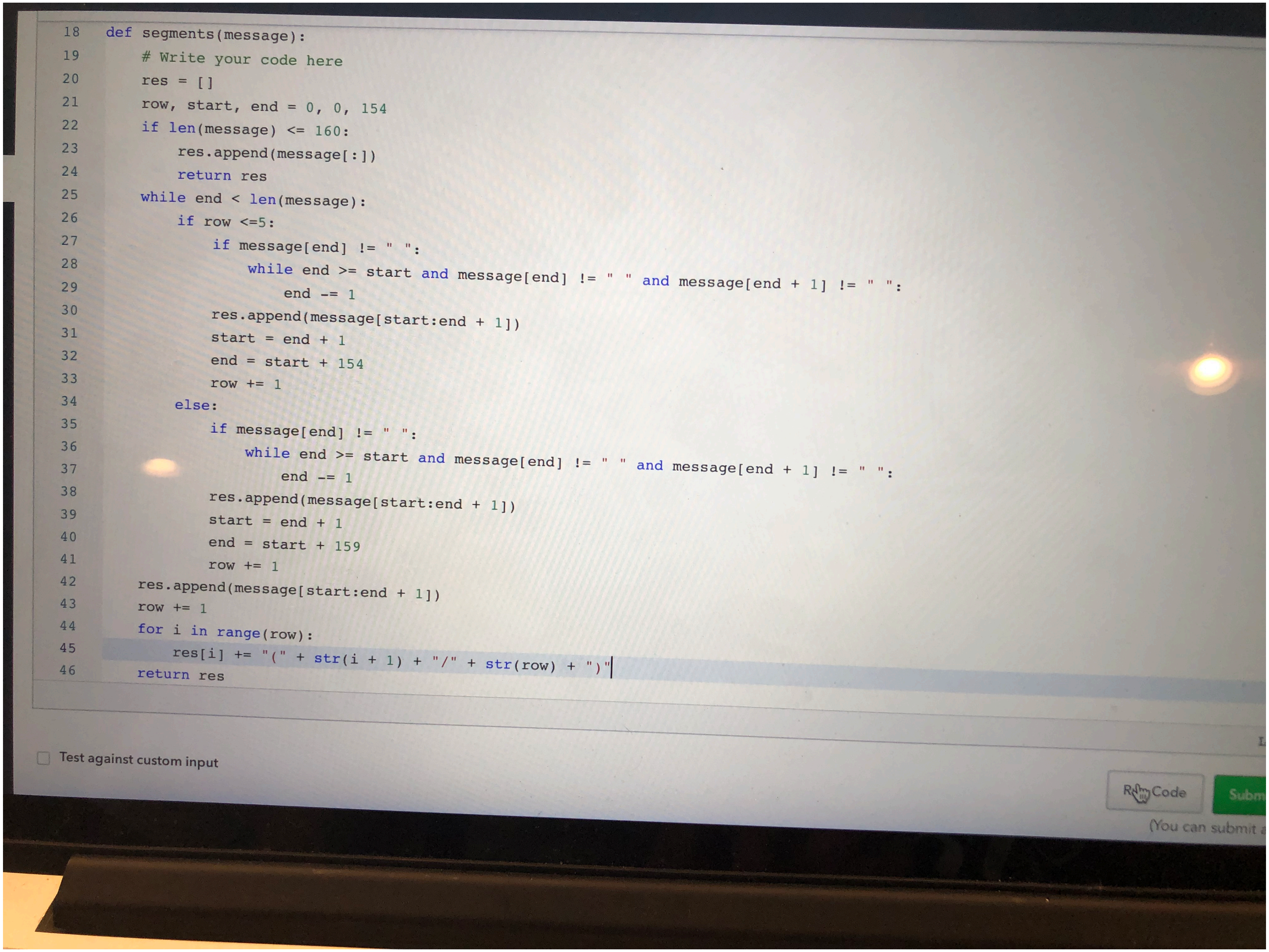


```python
import collections
class trieNode(object):
    def __init__(self):
        self.children = collections.defaultdict(trieNode)
        self.end = False

class numberprefix(object):
    def __init__(self):
        self.root = trieNode()

    def addnumber(self,number):

        node = self.root

        for i in range(len(number)):
            node = node.children[number[i]]

        node.end = True

    def search(self, number):
        res = ""
        node = self.root
        for i in range(len(number)):
            if number[i] not in node.children:
                if node.end is True:
                    return res
                return ""
            else:
                res += str(number[i])
                node = node.children[number[i]]
        return res

def checkprefix(prefix, numbers):
    prefixtree = numberprefix()

    for i in prefix:
        prefixtree.addnumber(i)

    res = []

    for i in numbers:
        temp = prefixtree.search(i)
        res.append(temp)

    return res
```

第三题的意思是：
给定超长一段话
比如：
sdfgh fds fsdfds fsd fuds fjsdf kdfja fklafkl ad;ldkosa af dsf ksd d, fsdf, sdfdsfsdfsd.f s fs dsf fdsaiunvcskdcmsidcewcu, fdsiufjsdifkdjsfjisdfaos. fsdfdgfgsf.......
希望你根据给定的长度，切割这段话
如果每一行的长度限制为15，那第一行会是
sdfgh fds(1/xx)
第二行会是
空格fsdfds (2/xx)
注意第二行开头的空格，因为第一行最后塞不下，但必须preserve这个空格，所以加在第二行头上。（打成空格因为地里和hackerrank一样会trim leading space）
第三行会是
fsd fuds (3/xx)
这样的
corner case比较多，但是比原题简单太多了。



```python
def segments(message):
    # Write your code here
    res = []
    row, start, end = 0, 0, 154
    if len(message) <= 160:
        res.append(message[:])
        return res
    while end < len(message):
        if row <=5:
            if message[end] != " ":
                while end >= start and message[end] != " " and message[end + 1] != " ":
                    end -= 1
            res.append(message[start:end + 1])
            start = end + 1
            end = start + 154
            row += 1
        else:
            if message[end] != " ":
                while end >= start and message[end] != " " and message[end + 1] != " ":
                    end -= 1
            res.append(message[start:end + 1])
            start = end + 1
            end = start + 159
            row += 1
    res.append(message[start:end + 1])
    row += 1
    for i in range(row):
        res[i] += "(" + str(i + 1) + "/" + str(row) + ")"
    return res
```

Test against custom input