```
# A7.py - program with functions to generate a string with three
# random lowercase letters, and returns a list of a specified size
# conforming to a regex string
# uses xeger library from here "https://github.com/crdoconnor/xeger"
# @author Erik Stryshak
# @sid 41069864
# @version 1.0

import random
import string
import re
from xeger import Xeger

class TripleGen(object):

    def __init__(self, regex_string, count):
        self.regex_string = regex_string
        self.count = count

    def genRandomTriple(self):
        # use random and string classes to generate three random
        # lowercase letters and return a concatenated string
        ch_one = random.choice(string.ascii_lowercase)
        ch_two = random.choice(string.ascii_lowercase)
        ch_three = random.choice(string.ascii_lowercase)
        return ch_one + ch_two + ch_three

    def genFilteredTriple(self, regex_string, count):
        # using the Xeger library, generate "count" random strings
        # that adhere to the regex string passed in
        return_list = []
        for x in range(0, count):
            str_gen = Xeger(limit=10)
            return_list.append(str_gen.xeger(regex_string))
        return return_list

    def getList(self):
        # calls the genFilteredTriple method with this object's
        # regex string and count
        return self.genFilteredTriple(self.regex_string, self.count)

class MyIterator(object):

    def __init__(self, iterator_list):
        # constructor sorts the incoming list alphabetically
        self.list = sorted(iterator_list)
        self.next_pointer = 1

    def next(self):
    # use hasNext function to check if there is a next value
    # if it does not exist, raise StopIteration
        if self.hasNext():
            self.next_pointer += 1
            return self.list[self.next_pointer-1]
        raise StopIteration
```

```
    def hasNext(self):
    # use size of the list and next_pointer to determine if
    # there is a next value in the list
        if len(self.list) < self.next_pointer + 1:
            return False
        return True

def main():
    # create a TripleGen object
    triple_gen = TripleGen("[bcdfgmnpr][aeiou][dgnprstwxyz]", 4)

    print ("Regex string used: [bcdfgmnpr][aeiou][dgnprstwxyz]")
    print ("Generating 4 random triples results in:")
    print (triple_gen.getList()[0])
    print (triple_gen.getList()[1])
    print (triple_gen.getList()[2])
    print (triple_gen.getList()[3])

    print ("\nFunction genRandomTriple examples:")
    print (triple_gen.genRandomTriple())
    print (triple_gen.genRandomTriple())
    print (triple_gen.genRandomTriple())
    print (triple_gen.genRandomTriple())

    print ("\nIterater Demonstration")
    my_it = MyIterator(triple_gen.getList())
    # should return 2nd, 3rd, and 4th elements
    print ("Has next = " + str(my_it.hasNext()))
    print ("Next element = " + my_it.next())
    print ("Has next = " + str(my_it.hasNext()))
    print ("Next element = " + my_it.next())
    print ("Has next = " + str(my_it.hasNext()))
    print ("Next element = " + my_it.next())
    # should return false and throw StopIteration
    print ("Has next = " + str(my_it.hasNext()))
    print ("Next element = " + my_it.next())

if __name__ == "__main__":
    main()
```

```
======== RESTART: C:/Users/Rik/Desktop/Programming_Languages/A7/A7.py ========
Regex string used: [bcdfgmnpr][aeiou][dgnprstwxyz]
Generating 4 random triples results in:
pad
rap
car
bex

Function genRandomTriple examples:
yeb
woa
djm
qey

Iterater Demonstration       .
Has next = True
Next element = gun
Has next = True
Next element = nod
Has next = True
Next element = rox
Has next = False
Traceback (most recent call last):
  File "C:/Users/Rik/Desktop/Programming_Languages/A7/A7.py", line 96, in <module>
    main()
  File "C:/Users/Rik/Desktop/Programming_Languages/A7/A7.py", line 93, in main
    print ("Next element = " + my_it.next())
  File "C:/Users/Rik/Desktop/Programming_Languages/A7/A7.py", line 55, in next
    raise StopIteration
StopIteration
>>>
```

```
grammar triplegen;

stmt       : TOKENONE VOWEL TOKENTWO

TOKENONE   : [bcdfgmnpr]
VOWEL      : [aeiou]
TOKENTWO   : [dgnprstwxyz]
```