Practice Quiz: Lists

TOTAL POINTS 3

1. **Given a list of filenames, we want to rename all the files with the extension hpp to the extension h by generating a list of tuples of the form (old_name, new_name).**

   That is, given the following list of filenames

   filenames = ["program.c", "stdio.hpp", "sample.hpp", "a.out", "math.hpp", "hpp.out"]

   complete the starter code to generate the following newfilenames list of tuples

   newfilenames = [('program.c', 'program.c'), ('stdio.hpp', 'stdio.h'), ('sample.hpp', 'sample.h'), ('a.out', 'a.out'), ('math.hpp', 'math.h'), ('hpp.out', 'hpp.out')]

   1 / 1 point

```
1   filenames = ["program.c", "stdio.hpp", "sample.hpp", "a.out", "math.hpp", "hpp
        .out"]
2   newfilenames = []
3
4   for i in filenames:
5       if i.endswith('.hpp'):
6           newfilenames.append(tuple([i]+[i.replace('.hpp','.h')]))
7
8       else:
9           newfilenames.append(tuple([i]+[i]))
10
11  print (newfilenames) # Should be [('program.c', 'program.c'), ('stdio.hpp',
        'stdio.h'), ('sample.hpp', 'sample.h'), ('a.out', 'a.out'), ('math.hpp', 'mat
        .h'), ('hpp.out', 'hpp.out')]
```

Run

Reset

   ✓ **Correct**

   > Great work! You're starting to see the benefits of knowing how to operate with lists and strings.

2. **The permissions of a file in a Linux system are split into three sets of three permissions: read, write, and execute for the owner, group, and others. Each of the three values can be expressed as an octal number summing each permission, with 4 corresponding to read, 2 to write, and 1 to execute. Or it can be written with a string**

   1 / 1 point

using the letters r, w, and x or - when the permission is not granted. For example: 640 is read/write for the owner, read for the group, and no permissions for the others; converted to a string, it would be: "rw-r-----" 755 is read/write/execute for the owner, and read/execute for group and others; converted to a string, it would be: "rwxr-xr-x" Fill in the blanks to make the code convert a permission in octal format into a string format.

```
1   def octal_to_string(octal):
2       result = ""
3       value_letters = [(4,"r"),(2,"w"),(1,"x")]
4       # Iterate over each of the digits in octal
5       for x in [int(n) for n in str(octal)]:
6           # Check for each of the permissions values
7           for value, letter in value_letters:
8               if x >= value:
9                   result += letter
10                  x -= value
11              else:
12                  result += '-'
13      return result
14
15  print(octal_to_string(755)) # Should be rwxr-xr-x
16  print(octal_to_string(644)) # Should be rw-r--r--
17  print(octal_to_string(750)) # Should be rwxr-x---
18  print(octal_to_string(600)) # Should be rw-------
```

Run

Reset

✓ **Correct**

> You nailed it! This is how we work with lists of tuples, how exciting is that!

3. **Let's create a function that turns text into pig latin: a simple text transformation that modifies each word moving the first character to the end and appending "ay" to the end. For example, python ends up as ythonpay.**

```
1   def pig_latin(text):
2     say = ""
3     # Separate the text into words
4     words = text.split()
5     for word in words:
6       # Create the pig latin word and add it to the list
7       texts = word[1:] + word[0] + "ay" + " "
8       say += texts
9       # Turn the list back into a phrase
10    return say
11
12  print(pig_latin("hello how are you")) # Should be "ellohay owhay reaay ouyay"
13  print(pig_latin("programming in python is fun")) # Should be "rogrammingpay niay
      ythonpay siay unfay"
```

Run

Reset

✓ **Correct**

> Nice! You're using some of the best string and list functions to make this work. Great job!