Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE
100%

Practice Quiz: Lists

TOTAL POINTS 3

---

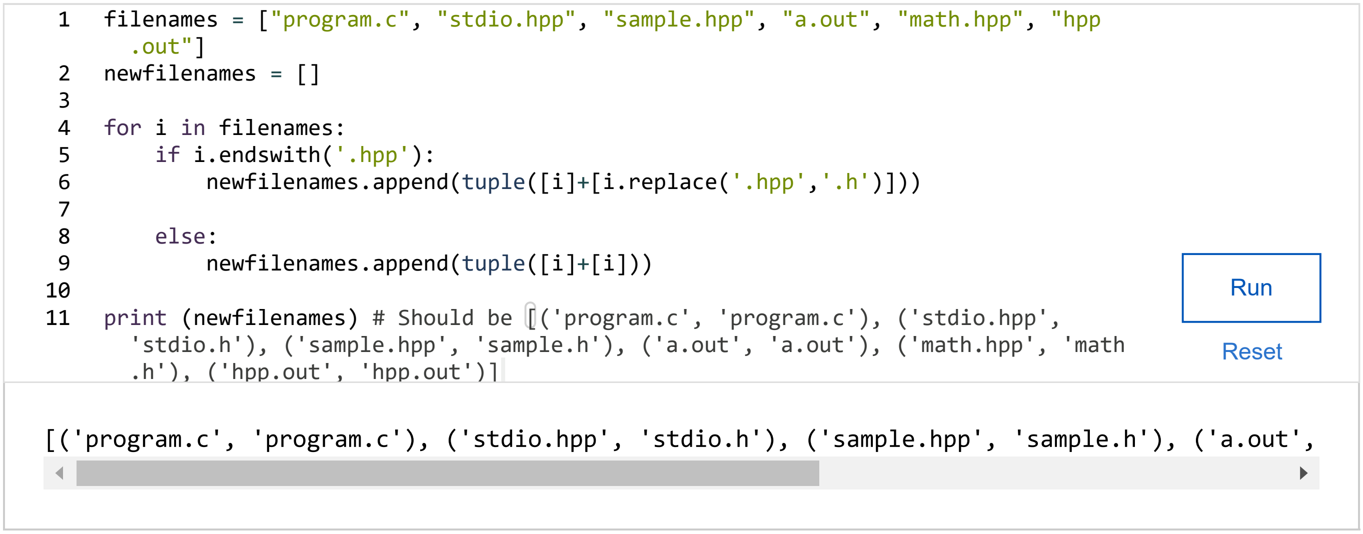
1. **Given a list of filenames, we want to rename all the files with the extension hpp to the extension h by generating a list of tuples of the form (old_name, new_name).** **1 / 1 point**

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')]



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

for i in filenames:
    if i.endswith('.hpp'):
        newfilenames.append(tuple([i]+[i.replace('.hpp','.h')]))

    else:
        newfilenames.append(tuple([i]+[i]))

print (newfilenames) # Should be [('program.c', 'program.c'), ('stdio.hpp',
  'stdio.h'), ('sample.hpp', 'sample.h'), ('a.out', 'a.out'), ('math.hpp', 'math
  .h'), ('hpp.out', 'hpp.out')]
```

```
[('program.c', 'program.c'), ('stdio.hpp', 'stdio.h'), ('sample.hpp', 'sample.h'), ('a.out',
```

✓ **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 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** **1 / 1 point**

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.

```python
def octal_to_string(octal):
    result = ""
    value_letters = [(4,"r"),(2,"w"),(1,"x")]
    # Iterate over each of the digits in octal
    for x in [int(n) for n in str(octal)]:
        # Check for each of the permissions values
        for value, letter in value_letters:
            if x >= value:
                result += letter
                x -= value
            else:
                result += '-'
    return result

print(octal_to_string(755)) # Should be rwxr-xr-x
print(octal_to_string(644)) # Should be rw-r--r--
print(octal_to_string(750)) # Should be rwxr-x---
print(octal_to_string(600)) # Should be rw-------
```

Run

Reset

```
rwxr-xr-x
rw-r--r--
rwxr-x---
rw-------
```

✓ **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.**

```python
def pig_latin(text):
  say = ""
  # Separate the text into words
  words = text.split()
  for word in words:
    # Create the pig latin word and add it to the list
    texts = word[1:] + word[0] + "ay" + " "
    say += texts
    # Turn the list back into a phrase
  return say

print(pig_latin("hello how are you")) # Should be "ellohay owhay reaay ouyay"
print(pig_latin("programming in python is fun")) # Should be "rogrammingpay niay
  ythonpay siay unfay"
```

Run

Reset

```
ellohay owhay reaay ouyay
rogrammingpay niay ythonpay siay unfay
```

✓ **Correct**

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