

Strings

- ✔

Video:

Basic Structures Introduction

1 min
- ✔

Video:

What is a string?

2 min
- ✔

Video:

The Parts of a String

4 min
- ✔

Reading:

String Indexing and Slicing

10 min
- ✔

Video:

Creating New Strings

6 min
- ✔

Reading:

Basic String Methods

10 min
- ✔

Video:

More String Methods

4 min
- ✔

Reading:

Advanced String Methods

10 min
- ✔

Video:

Formatting Strings

5 min
- ✔

Reading:

String Formatting

10 min
- ✔

Reading:

String Reference Cheat Sheet

10 min
- ✔

Reading:

Formatting Strings Cheat Sheet

10 min
- ✔

Practice Quiz:

Practice Quiz: Strings

5 questions
- Lists
- ✔

Video:

What is a list?

4 min

✔

Reading:

Lists Defined

10 min

✔

Video:

Modifying the Contents of a List

5 min

✔

Reading:

Modifying Lists

10 min

✔

Video:

Lists and Tuples

3 min

✔

Reading:

Tuples

10 min

✔

Video:

Iterating over Lists and Tuples

7 min

✔

Reading:

Iterating Over Lists Using Enumerate

10 min

✔

Video:

List Comprehensions

4 min

✔

Reading:

List Comprehensions

10 min

✔

Reading:

Lists and Tuples Operations Cheat Sheet

10 min

✔

Practice Quiz:

Practice Quiz: Lists

6 questions

Dictionaries

✔

Video:

What is a dictionary?

5 min

✔

Reading:

Dictionaries Defined

10 min

✔

Video:

Iterating over the Contents of a Dictionary

4 min

✔

Reading:

Iterating Over Dictionaries

10 min

✔

Video:

Dictionaries vs. Lists

3 min

✔

Reading:

Dictionary Methods Cheat Sheet

10 min

📖

Practice Quiz:

Practice Quiz: Dictionaries

5 questions

Module Review

▶

Video:

Basic Structures Wrap Up

1 min

▶

Video:

In Marga's Words: My Most Challenging Script

1 min

📖

Quiz:

Module 4 Graded Assessment

10 questions

💬

Discussion Prompt:

Discussion Prompt

5 min

PRACTICE QUIZ • 30 MIN

✔

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE

83.33%

Practice Quiz: Lists

Practice Quiz: Lists

TOTAL POINTS 6

✔

Submit your assignment

✔

Receive grade

TO PASS 80% or higher

1.

Given a list of filenames, we want to rename all the files with extension hpp to the extension h. To do this, we would like to generate a new list called newfilenames, consisting of the new filenames. Fill in the blanks in the code using any of the methods you've learned thus far, like a for loop or a list comprehension.

1

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

2

newfilenames = []

3

for file in filenames:

4

if '.hpp' in file:

5

newfilenames.append((file,file[:-2]))

6

else:

7

newfilenames.append((file,file))

8

9

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')]

10

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

Try again

Grade

View Feedback

0 / 1 point

We keep your highest score

Run

Reset

Incorrect

Not quite. Be sure to review the lists and strings techniques.

2.

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

say += word[1:]+word[0]+'ay'

8

if word != words[len(words)-1]:

9

say += ' '

10

Turn the list back into a phrase

11

return say

12

13

14

print(pig_latin("hello how are you")) # Should be "ellohay owhay reabay ouyay"

15

print(pig_latin("programming in python is fun")) # Should be "rogrammingpay niay ythonpay siay unfay"

ellohay owhay reabay ouyay

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!

3.

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 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 digit 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 digit >= value:

9

result += letter

10

digit -= 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-----

rwxr-xr-x

rw-r--r--

rwxr-x---

rw-----

Run

Reset

Correct

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

4.

Tuples and lists are very similar types of sequences. What is the main thing that makes a tuple different from a list?

☐ A tuple is mutable

☐ A tuple contains only numeric characters

☒ A tuple is immutable

☐ A tuple can contain only one type of data at a time

Correct

Awesome! Unlike lists, tuples are immutable, meaning they can't be changed.

5.

The group_list function accepts a group name and a list of members, and returns a string with the format: group_name: member1, member2, ... For example, group_list("g", ["a","b","c"]) returns "g: a, b, c". Fill in the gaps in this function to do that.

1

def group_list(group, users):

2

members = "

3

members += ", ".join(users)

4

return ("{}:{}".format(group, members))

5

6

print(group_list("Marketing", ["Mike", "Karen", "Jake", "Tasha"])) # Should be "Marketing: Mike, Karen, Jake, Tasha"

7

print(group_list("Engineering", ["Kim", "Jay", "Tom"])) # Should be "Engineering: Kim, Jay, Tom"

8

print(group_list("Users", "")) # Should be "Users:"

Marketing: Mike,Karen,Jake,Tasha

Engineering: Kim,Jay,Tom

Users:

Correct

Nice job! You're doing well, working with list elements!