/ Users lexirohrer Downloads Classes HCDE310 hw0-lexirohrer hw0.py

hw0

hw0.py

```
1   ...
27
28  def hello():...
35
36
37  hello()
38  print("... Let's say that again... \n")
39  ...
48
49  hello()
50
51  ...
57
58  length = 2
59  width = 3
60  height = 2
61  #
62  me = "Lexi Rohrer"
63  print("Volume =", width * length * height)
64  print("My name is", me)
65
66  ...
72
```

Run: hw0

```
/Users/lexirohrer/RelativeFreq/bin/python /Users/lexirohrer/Downloads/Classes/HCDE310/hw0-lexirohrer/hw0.py
--------------------------------
Welcome to HCDE 310.

We are glad you are here
and we hope you enjoy the class.
--------------------------------
... Let's say that again...

--------------------------------
Welcome to HCDE 310.

We are glad you are here
and we hope you enjoy the class.
--------------------------------
Volume = 12
My name is Lexi Rohrer

Process finished with exit code 0
```

1: Project

Z: Structure

2: Favorites

Lexi Rohrer
HCDE 310
9/25/2019

HW.0 Written Responses

Description of your program (part 7)

In counting the number of words using a computer program, the fundamental strategy I would use would be to identify 'words' by chunking the characters in the document into 'words' with spaces separating them. In other words, I would begin by having the program identify consecutive segments of characters with a space before and after the segment. These consecutive segments of characters separated by space characters would be identified as words and counted.

After running the program and identifying any miscounts or errors in the output, my first instinct would be to look at punctuation, dashed words, and apostrophes/contractions as sources of possible error. If these character types were causing challenges, I would manually code them in as character types the program should ignore in its search for "word segments."

Reflection on writing your program (part 8)

In my computational linguistics class, we were often asked to articulate the approach behind our programs in plain english, so this felt like a comfortable exercise -- not too difficult. This process is always satisfying for me, as I much prefer the logical problem solving part of computer science to the actual minutiae of coding the solution. I find that much of this logical problem solving can be done with a written description of the approach to the problem before beginning to write in code.

From this course, I'd love to gain proficiency in python as well as just overall confidence with computational projects -- I still really can't self-identify as someone who 'gets' the computational side of HCDE. I think this may stem from lack of knowledge about higher level, more abstract CS concepts, which leaves me confused about what I do and don't know. If you have resources (like youtube videos) that just explain topics like what a server actually is, what key words like "bash" and "git" mean, etc., that would be super helpful!