Lab 17: Anagrams

Soundgarden == Nude Dragons

Anagrams: Summary of what will be covered

- Run through the lab solution
- Inspect the Unit Tests section
- Run through *lab_functions.py* module
- Breakdown of *unit_tester_2()* function
- Questions?

Running Time: <5 mins

Anagrams: Run through the lab solution

Sanitize user input (lines 30-37) - Do this first, get it out of the way

Short-circuit evaluation (lines 39-40) - Binary comparison:
If the length of the input words don't match, you don't need to look any further

Processing

a. Split words into lists (lines 43-45)
b. Sort letters in each list (lines 48-49)
c. Compare the lists (lines 52-54)

Input & Output

a. Prompt user for two words (lines 59-66)
b. Call the function to get the results (lines 69-73)

Anagrams: Run through the lab solution (cont’d)

Compare v1 and v2

a. v1 uses discrete operations to split, sort, and compare the words using the list(string) and list.sort() methods, and comparison operator (lines 43-52)

b. v2 uses the sorted() function and comparison operator to evaluate the words in one line of code (line 43)

See: https://www.diffchecker.com/KW6LD2Ca

Anagrams: Inspect the Unit Tests section

test_data (list of tuples)

a. Each tuple in the list is a test case
b. The positive (True) test cases cover input that includes spaces, capitalization, and punctuation
c. The negative (False) test cases cover length and letter mismatches

Import testing function and output the results

a. Import unit_tester_2() from *lab_functions.py*
b. Execute unit_tester_2() and print test results

Anagrams: Run through *lab_functions.py* module

This module has two functions, unit_tester_1() & unit_tester_2()

Each function takes 2 parameters:

a. input_output, which is the test_data defined in the module
b. function_name, which is the name of the function being tested

unit_tester_1 works with functions that take 1 parameter, like *palindrome checker*, *credit card validator*, *blackjack advice*

unit_tester_2 works with functions that take 2 parameters, like *anagram checker*

Anagrams: Breakdown of *run_tests_2* function

Compare expected_output and actual_output

> If the test fails, increment failed_test_count by 1 and append test result string to the failed_test_msg variable,

Return the result (after all test cases are complete)

- If *failed_test_count > 0*, print failed_test_msg
- If *failed_test_count == 0*, print “All tests passed.”

Anagrams: Questions?

```
import random
import re
def answer(q):
    ynm = ["Yes!", "No!", "Maybe!"]
    qa = {"who": "The identity is unknown.",
          "what": "Cannot predict now.",
          "when": "When the time is right.",
          "where": "The path is unclear.",
          "how": "There are many ways.",
          "why": "Your question is deep.",
          "are": ynm[random.randint(0,2)],
          "will": ynm[random.randint(0,2)],
          "can": ynm[random.randint(0,2)]}
    character_regex = r'^\w+'
    question_lower = q.lower()
    start_word = ''.join(re.findall(character_regex, question_lower)
    return qa.get(start_word,"Concentrate and ask again")
def get_all_answers(questions):
    for question in questions:
        response = answer(question)
        print(f"{question}? {response}")
questions = []
while True:
    q = input("Submit your question, or type (done): ")
    if q == 'done':
        get_all_answers(questions)
        break
    else:
        questions.append(q)
```