EECS 1015 Lab 9

Goal

- Get familiar with the basics of collections

Tasks

1. Guide you through writing a process with collections
2. Learn to debug scripts involving collections
3. Learn to write a script with collections
4. Learn to write a script with collections
5. Learn to write a script with collections
6. Learn to write a script with collections
7. Learn to write a script with collections

Total Credit: 100 pts

It is an academic offense to copy code from other students, provide code to other students, let other people (including the teaching staff) debug your code, or debug other students' code.

We will check code similarities at the end of the semester.

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Questions 3 – 6 may not be arranged by the difficulty level. You are highly recommended to take a look and decide the order of completing the questions. You will experience something similar in the exams and it is important to learn how to triage your tasks.

Note: For question 3 – 6, you must come up with a function description in your own words, and cannot copy the words from handout.

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 1: Follow the Steps (30 pts)

You would like to calculate the average given numbers in a string format (e.g., number of fruites). The numbers are separated by '|', and there could be spaces around the numbers and separators. Your task is to write a Python function to calculate this average. The input string will always contain exactly 5 numbers in the specified format.

Write a function called calculate_average() that takes a string in the given format as input and returns the average of the numbers as a floating-point value.

You are highly recommended to attempt the questions yourself before you look at the solution as a way to learn.

Q1.1 Start by completing signature and design recipe of the function calculate_average().

You are tasked with calculating the average of numbers in a string format. The numbers are separated by '|', and there could be spaces around the numbers and separators. Your task is to write a Python function to calculate this average. The input string will always contain exactly 5 numbers in the specified format.

Solution:

```python
def calculate_average(numbers_str: str) -> float:
    """
    Calculates the average of numbers in the specified string format.

    Parameters:
        numbers_str (str): A string containing 5 numbers separated by '|',
        with possible spaces.

    Returns:
        float: The average of the numbers.

    >>> calculate_average('1|2|3|4|5')
    3.0
    >>> calculate_average('2 | 10|  5|7|11')
    7.0
    """
```

Q1.2 Break Down the Question

So how do we write a script to achieve the functionality? Recall that one of the principles we mentioned in the first lecture is that we use building blocks to build systems. Let's break down the problem. Please note that you can use the same procedure for complicated problems in the future.

Recall that in the previous labs, your code usually has the following the structure:

- Take the input and the check for the preconditions
- Do some calculation with the input

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

- Show the result

Solution:

For do some calculation with the input, let's see what the structure of our script should look like if our example is "2| 7|5| 10 |11":

1. Find the number in "2| 7|5| 10 |11"
    a. Split the list based on '|', we will get ["2", " 7", "5", " 10 ", "11"]
    b. For each element in the list, remove the space before and after
    c. Convert to integer
2. Sum all the numbers
3. Divide by the number of numbers (in this case, 5)

Our goal here is to write a script that can handle any expressions in this format, not just "2| 7|5| 10 |11" So let's generalize our structure:

1. Take the input in the format of "x1 | x2| x3 | x4|x5"
2. Do some calculation with the input
    2.1 Find the number in "x1 | x2| x3 | x4|x5"
    2.2 Split the list based on '|', we will get ["x1 ", " x2", " x3 ", " x4", "x5"]
        i. For each element in the list, remove the space before and after
        ii. Convert to integer
    2.3 Sum all the numbers
    2.4 Divide by the number of numbers (in this case, 5)
3. Show the result

Q1.3 Check the pre-conditions

Solution:

```python
def calculate_average(numbers_str: str) -> float:
    """
    Calculates the average of numbers in the specified string format.

    Parameters:
        numbers_str (str): A string containing 5 numbers separated by '|',
        with possible spaces.

    Returns:
        float: The average of the numbers.

    >>> calculate_average('1|2|3|4|5')
    3.0
    >>> calculate_average('2 | 10|  5|7|11')
    7.0
    """

    assert type(numbers_str) == str, "The input must be a string"

    numbers = numbers_str.split('|')

    assert len(numbers) == ALLOWED_NUMBER, "Input string must contain exactly 5 numbers."
```

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

When checking how many numbers, we need to split the input string into individual numbers. The '|' character is used as the separator, and there may be spaces around it.

After splitting the string, you should ensure that there are exactly 5 elements (numbers) in the resulting list. If there are not exactly 5 elements, it indicates an invalid input. We want to make sure that the input string contains exactly 5 numbers, as this is a requirement. To do this, we use an 'assert' statement to check if the number of elements in the list matches the expected count of 5.

Note: We skipped the precondition checking of whether there are numbers of strings, but not alphabetic characters. Think about how to check it?

Q1.4 Calculate the Total

Next, you need to calculate the sum of the 5 numbers in the list. Remember to strip any leading or trailing spaces from each number.

Solution:

To find the average, we need the total sum of the 5 numbers. We iterate through the list of numbers, remove any leading or trailing spaces, and convert each number to an integer. Then, we calculate the sum of these integers.

```
1   ALLOWED_NUMBER = 5
2
3   def calculate_average(numbers_str: str) -> float:
4       """
5       Calculates the average of numbers in the specified string format.
6
7       Parameters:
8           numbers_str (str): A string containing 5 numbers separated by '|',
9           with possible spaces.
10
11      Returns:
12          float: The average of the numbers.
13
14      >>> calculate_average('1|2|3|4|5')
15      3.0
16      >>> calculate_average('2 | 10|  5|7|11')
17      7.0
18      """
19
20      assert type(numbers_str) == str, "The input must be a string"
21
22      numbers = numbers_str.split('|')
23
24      assert len(numbers) == ALLOWED_NUMBER, "Input string must contain exactly 5 numbers."
25
26      total = 0
27      total = total + int(numbers[0].strip())
28      total = total + int(numbers[1].strip())
29      total = total + int(numbers[2].strip())
30      total = total + int(numbers[3].strip())
31      total = total + int(numbers[4].strip())
```

Note: the current coding style isn't ideal. We encourage you to refactor your code next week to improve your coding style.

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Q1.5 Calculate the Average and Return

Once you have the total sum of the numbers, you should calculate the average by dividing the total by 5 (since there are 5 numbers). And you should return the calculated average as a float.

Solution:

```python
ALLOWED_NUMBER = 5

def calculate_average(numbers_str: str) -> float:
    """
    Calculates the average of numbers in the specified string format.

    Parameters:
        numbers_str (str): A string containing 5 numbers separated by '|',
        with possible spaces.

    Returns:
        float: The average of the numbers.

    >>> calculate_average('1|2|3|4|5')
    3.0
    >>> calculate_average('2 | 10|  5|7|11')
    7.0
    """

    assert type(numbers_str) == str, "The input must be a string"

    numbers = numbers_str.split('|')

    assert len(numbers) == ALLOWED_NUMBER, "Input string must contain exactly 5 numbers."

    total = 0
    total = total + int(numbers[0].strip())
    total = total + int(numbers[1].strip())
    total = total + int(numbers[2].strip())
    total = total + int(numbers[3].strip())
    total = total + int(numbers[4].strip())

    average = total / len(numbers)

    return average
```

Note (again!) the current coding style isn't ideal since we haven't covered looping over collections yet. We'll be learning about it in the next class. We encourage you to refactor your code next week to improve your coding style.

We skipped a few things in the code, you are encouraged to think about those after completing the lab as additional exercises:

- Please note that here we skip the precondition check for each element (e.g., each must be an int). You can think about how to check these to guarantee that all numbers are integers.
- If we need to accept floating numbers, then how would you change the precondition check?
- We used a naïve way of calculating the average, can you think of ways to calculate the result with built-in functions?

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

- Are you able to write a program that takes any arbitrary number of numbers (rather than just 5?). You may leave the question until next week after we learn loops over collections.

Submission

- Go to eClass -> our course -> Week 9 -> Practice -> Lab -> Lab 9 Submission
- You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission (You need spend some time debugging!).

Rubrics:

- You will get full mark for this question if you submit the solution (with no typo) to the required location on Prairielearn before the deadline.

Note:

- Since we provide solutions to this question, it is ok to submit it as is.
- In order for the autograder to work properly:
    - You must NOT change the name of the function
    - You must NOT change the order of the arguments
- The autograder may take some time to grade

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 2: Debugging (30 pts)

Sam tried to convert the fruit quantity represented in a list into a dictionary where the keys are the fruits' name predefined as global variables. For the i-th value of the input_list, the key is the ith element of the elements in the fruit, and the values are the quantities of the fruites. The length of input_list must be 4, and they represent the quantity of the fruits in the order of apple, banana, grape, and orange.

For example when the input list is [10, 10, 30, 40], the output should be {'apple': 10, 'banana': 10, 'grape': 30, 'orange': 40}

However, it seems that there are some issues with the function. Your task is to identify and fix the problems to make the function work correctly.

The code to debug (lab9_task2.py) may have syntax errors, run-time errors, semantic errors, or errors in test cases. To help with debugging, you can use the debugger to execute the code line by line (you should fix the syntax errors before doing this).

After completing the question, you can think about why Sam wants to do this conversion. What are the limitations of using a list?

Note (again!) the current coding style isn't ideal since we haven't covered looping over collections yet. We'll be learning about it in the next class. We encourage you to refactor your code next week to improve your coding style.

Requirement

- You must not change the function definition

Submission

- Copy the Python code to Lab 9 -> Task 2 on Prairielearn.
- You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission
- Note: In order for the autograder to work properly:
    - You must NOT change the name of the function
    - You must NOT change the order of the arguments

Rubrics:

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

- you will receive 5 points if your script correctly provides a solution for the general case, e.g., list_to_dict([10, 15, 20, 25]), ANS: {'apple': 10, 'banana': 15, 'grape': 20, 'orange': 25}
- you will receive 2.5 points if your script correctly provides a solution for edge case – Zero case list_to_dict([]), ANS: Assertion Error
- you will receive 2.5 points if your script correctly provides a solution for condition: all quantities are the same, list_to_dict([10, 10, 10, 10]), ANS: {'apple': 10, 'banana': 10, 'grape': 10, 'orange': 10}
- you will receive 2.5 points if your script correctly provides a solution for violation of contract (number of elements), e.g., list_to_dict([5, 10, 15]), ANS: Assertion Error
- you will receive 2.5 points if your script correctly provides a solution for violation of contract (type of the elements), e.g., list_to_dict([0.5, 5, 10, 15]), ANS: Assertion Error
- you will receive 15 points for providing a solution for three other test cases (5 pts each)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 3: Coding (10 pts)

Imagine you are tasked with developing an inventory management system for a small store.

Given a dictionary called inventory to represent the store's inventory, where keys are product names (strings) and values are the quantities (integers) of each product in stock. For example:

```
inventory = {
    "Laptop": 20,
    "Smartphone": 15,
    "Tablet": 30,
    "Headphones": 10
}
```

You task is to write the add_to_inventory function:

add_to_inventory (Inventory, item, quantity)

This function returns the current quantity of the item. Below are two examples of how add_to_inventory should work. As shown in the example, add_to_inventory returns the new product quantity.

```
>>> add_to_inventory({"Laptop": 20, "Smartphone": 15,  "Tablet": 30,
"Headphones": 10}, "Keyboard", 5)
5
>>> add_to_inventory({"Laptop": 20, "Smartphone": 15,  "Tablet": 30,
"Headphones": 10}, " Smartphone ", 10)
25
```

In the first example, we add 5 Keyboards. Since Keyboard does not exist in the original inventory, we add this new item and the quantity is 5. Thus, the quantity to return is the quantity of this inventory, which is 5.

In the second example, we add 10 Smartphones. Smartphone already exist in the original inventory with a quality of 15. Thus the quantity to return is 15 + 10 = 25.

When using the function, the quantity must be a non-negative integer.

Submission

- Copy the Python code to Lab 9 -> Task 3 on Prairielearn.
- You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission.
- Note: In order for the autograder to work properly:
    - You must NOT change the name of the function
    - You must NOT change the order of the arguments

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Rubrics:

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise
    - You will receive 0.25 pt for function signature
    - You will receive 0.25 pt for function description
    - You will receive 0.5 pt for doctest (At least 4, and they cannot be the cases below)
        - To accommodate the autograder, please make sure each test case is self-contained in a line and the expected output is in the next line (i.e., do not define the variable inventory in one line and call the function in the next line and refer to the variable).
    - You will receive 1 point if your script correctly provides the correct solution for a general test case, e.g.,
        - new_quantity = add_to_inventory({"Laptop": 20, "Smartphone": 15, "Tablet": 30, "Headphones": 10}, "Keyboard", 5)
        - Ans: 5
    - You will receive 1 point if your script correctly provides the correct solution for edge case – empty inventory, e.g.,
        - new_quantity = add_to_inventory({}, "Apple", 1)
        - Ans: 1
    - You will receive 1 points if your script correctly provides the correct solution for edge case – singleton inventory, e.g.,
        - new_quantity = add_to_inventory({"Plates": 3}, "Plates", 4)
        - Ans: 7
    - You will receive 1 point if your script correctly provides a solution for condition – the item to be added was not in inventory, e.g.,
        - add_to_inventory({"Orange": 45, "Apple": 60}, "Pear", 30)
        - Ans: 30
    - You will receive 1 point if your script correctly provides a solution for condition – the item to be added was already in inventory, e.g.,
        - add_to_inventory({"Orange": 45, "Apple": 60}, "Apple", 30)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

        - Ans: 60
    - You will receive 1 point if your script correctly provides a solution for violation of contract, e.g.,
        - add_to_inventory({"Chocolate Cake": 3, "Cheese Cake": 5}, "Strawberry Cake", -4)
        - Ans: AssertionError
    - you will receive 3 points for providing a solution for three other test cases (1 pt each)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 4: Implementation (10 pts)

For this task you will write a function called "intersecting_chars" that will find the characters appearing across multiple strings. This function will take as input a single string that contains two underscore characters ("_") that separate the three things to be compared.

For example, one input might be "aBce_aabe_bCC", in this case the three strings to consider would be "aBce", "aabe" and "bCC". The output of the function will be a list containing the intersecting characters in ascending order. The output of the above example would be ['b', 'c'] as 'b' and 'c' are the only characters appearing in all three strings. The intersecting characters are case-insensitive and your answer should be in all lowercase.

Let's give one more example:

input: a string

"_CCda_acz"

output: a list of intersecting characters

[]

reasoning:

'a' and 'c' are in the second and third string, but there are no characters in the first string, therefore there are no characters appearing in all three strings.

Keep in mind the following requirements when writing your function

- input may be invalid (more than, or less than 2 "_" characters. In this case you should assert an error)
- "_a_b" is a valid input, it means the first string is empty
- there will only be letters in the string, no numbers or special characters
- you can match characters across strings in a case-insensitive manner but your answer should be in all lowercase

Hint: This task can be done with loops but it will be easier if you use collections. You can convert a list to a set by wrapping it with set(). For example, if you had the following list

```
>>> x = [1, 1, 5, 1, 2, 3, 1]
```

then you can convert it to a set as follows

```
>>> set(x)
{1, 5, 2, 3}
```

Requirement

- You cannot use any other libraries or packages to get the answer for you

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Submission

- Copy the Python code to Lab 9 -> Task 4 on Prairielearn.
- You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission
- Note: In order for the autograder to work properly:
    - You must NOT change the name of the function
    - You must NOT change the order of the arguments

Rubric

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise
    - You will receive 0.25 pt for function signature
    - You will receive 0.25 pt for function description
    - You will receive 0.5 pt for doctest (At least 4, and they cannot be the cases below)
    - You will receive 1 pt if your script correctly provides the solution to: General case, e.g., "ab_aaab_bCC", ans: ['b']
    - You will receive 1 pt if your script correctly provides the solution to: Condition – have common characters but with both upper and lower case, e.g., "aBcd_aaab_bCC", ans: ['b']
    - You will receive 1 pt if your script correctly provides the solution to: Condition – have common characters but with common upper cases only, e.g., "Bdk_YB_MWB", ans: ['b']
    - You will receive 1 pt if your script correctly provides the solution to: Condition – no common character, e.g., "m_CCda_acz", ans: []
    - You will receive 1 pt if your script correctly provides the solution to: Condition – Multiple common characters, e.g., "CAB_cab_cxaxb", ans: ['a', 'b', 'c']
    - You will receive 1 pt if your script correctly provides the solution to: Violation of contract, e.g., "za3g34_0a_a", ans: AssertionError
    - You will receive 3 pts if your script correctly provides the solution to other examples (3 cases and 1 pt each)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 5: Implementation (5 pts)

In this task, you will implement a function and no starter code is provided. The function should be called remove_item and accept two arguments. Given a **list** of the name of products and the **name** of the product to remove from the list, the function should return a list with the product removed. For example:

```
>>> remove_item(["cellphone", "laptop", "microphone"], "laptop")
['cellphone', 'microphone']
>>> remove_item(["cellphone", "laptop", "microphone"], "webcam")
["cellphone", "laptop", "microphone"]
```

The items in the list can be any string and there is no other specifications of the format of the string.

Submission

Copy the Python code to Lab 9 -> Task 5 on Prairielearn.

You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission (You need to spend some time debugging!).

Rubrics:

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise
    - You will receive 0.25 pts if you provide a description of the function.
    - You will receive 0.25 pts if you provide the argument annotations correctly
    - You will receive 0.5 pt if you include at least 2 test cases (and they cannot be the same as test cases provided) and your code pass those test cases
    - You will receive 1 pts if your function passes general test (e.g., remove_item(["cellphone", "laptop", "microphone"], "laptop"), expected output: ['cellphone', 'microphone'])
    - You will receive 1 pts if your function passes the corner test - zero data (e.g., remove_item([], "laptop"), expected output: [])
    - You will receive 1 pts if your function passes the corner test - singleton data (e.g., remove_item(["laptop"], "laptop"), expected output: [])
    - You will receive 1 pts if your function passes the condition test – the item to be removed not in the list (e.g., remove_item(["cellphone", "laptop", "microphone"], "webcam"), expected output: ["cellphone", "laptop", "microphone"])

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 6: Implementation (5 pts)

In this task, you will implement a function and no starter code is provided. The function should be called remove_item and accept two arguments. Given a **list** of the name of products and the **index** of the product to remove from the list, the function should return a list with the product removed. For example:

```
>>> remove_item(["cellphone", "laptop", "microphone"], 1)
['cellphone', 'microphone']
>>> remove_item(["cellphone", "laptop", "microphone"], 4)
AssertionError
```

The items in the list can be any string and there is no other specifications of the format of the string.

Submission

Copy the Python code to Lab 9 -> Task 6 on Prairielearn.

You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission (You need to spend some time debugging!).

Rubrics:

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise
    - You will receive 0.25 pts if you provide a description of the function.
    - You will receive 0.25 pts if you provide the argument annotations correctly
    - You will receive 0.5 pt if you include at least 2 test cases (and they cannot be the same as test cases provided) and your code pass those test cases
    - You will receive 1 pts if your function passes general test (e.g., remove_item(["cellphone", "laptop", "microphone"], 1), expected output: ['cellphone', 'microphone'])
    - You will receive 1 pts if your function passes the corner test - zero data (e.g., remove_item([], 0), expected output: AssertionError)
    - You will receive 1 pts if your function passes the corner test - singleton data (e.g., remove_item(["laptop"], 0), expected output: [])
    - You will receive 1 pts if your function passes the condition test – the item to be removed not in the list (e.g., remove_item(["cellphone", "laptop", "microphone"], 5), expected output: AssertionError)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin

Task 7: Implementation (10 pts)

In this task, you will implement a function and no starter code is provided. The function should be called remove_item and accept two arguments. Given a **tuple** of the name of products and the **index** of the product to remove from the list, the function should return a tuple with the product removed. For example:

```
>>> remove_item(("cellphone", "laptop", "microphone"), 1)
('cellphone', 'microphone')
>>> remove_item(["cellphone", "laptop", "microphone"], 4)
AssertionError
```

The items in the tuple can be any string and there is no other specifications of the format of the string.

Submission

Copy the Python code to Lab 9 -> Task 6 on Prairielearn.

You can resubmit your code as many times as you need, but you need to wait for 5 minutes before submission (You need to spend some time debugging!).

Rubrics:

- You will not receive any mark if you do not submit it to the designated location on Prairielearn before the deadline
- You will not receive any mark if your code violates the above requirements
- Otherwise
    - You will receive 0.5 pts if you provide a description of the function.
    - You will receive 0.5 pts if you provide the argument annotations correctly
    - You will receive 1 pt if you include at least 2 test cases (and they cannot be the same as test cases provided) and your code pass those test cases
    - You will receive 2 pts if your function passes general test (e.g., remove_item(("cellphone", "laptop", "microphone"), 2), expected output: ['cellphone', ' laptop'])
    - You will receive 2 pts if your function passes the corner test - zero data (e.g., remove_item((), 0), expected output: AssertionError)
    - You will receive 2 pts if your function passes the corner test - singleton data (e.g., remove_item(("laptop",), 0), expected output: ())
    - You will receive 2 pts if your function passes the condition test – the item to be removed not in the list (e.g., remove_item(("cellphone", "laptop", "microphone"), 5), expected output: AssertionError)

Copyright © Xiao Xuan Wang, Alexander Bianchi, Meiying Qin