ESI-4357 Web DSS Exam 1 Summer 2020

Cover Sheet

SUBMISSION INSTRUCTIONS

Your answers will be submitted to Canvas. The following is required in order to receive full credit for the assignment:

1. You will type your answers to a jupyter file called **'Exam1_Skeleton'**
2. Type your name in the top cell.
3. To submit, attach 'Exam1_Skeleton' as a jupyter notebook file (.ipynb). **DO NOT ZIP.**
4. If you forget to submit, or incorrectly submit, you will receive a zero. There will be no exceptions, so **verify** the file you attached after hitting submit. Multiple submissions are allowed, provided that they are completed before the deadline.
5. By attending this exam, you accept **Honor Pledge** below:

"We, the members of the University of Florida community, pledge to hold ourselves and our peers to the highest standards of honor and integrity by abiding by the Honor Code.

On my honor, I have neither given nor received unauthorized aid in doing this assignment."

1. **(Easy – 6 pts)**
   Given a list L, where L=$[x_1, x_2, \dots, x_n]$, create a **recursive** function that prints sums of every two adjacent numbers in the list format such as $[x_1 + x_2,\ x_2 + x_3, \dots,\ x_{n-1} + x_n]$. Stop recursion when there is one item left.
   See the example below.

```
fun([1,2,3,4])  ➞ [1,2,3,4]
                  [3,5,7]
                   [8,12]
                   [20]
```

2. (Normal – 7 pts)

Your task is to write a program which allows teachers to create multiple-choice tests with 5 questions in different subjects. You are given two classes named **Test** and **Student** (*see Exam1_Skeleton.ipynb*). There are 2 different subjects: mathematics and physics. Pass marks and answer keys are already entered in the skeleton for Mathematics and Physics course.

a. Fill **student_list** given to you in the skeleton code with 10 Student class objects.

b. Generate 2 different scantrons (one for mathematics, one for physics) for each student by picking a random letter from **choice_list** for each question. *(Hint: random.choice(A) gives you a random element in list A.)*

c. In Exam1_Skeleton.ipynb, fill the function, called **take_test()**, which takes **students' answers** and **a test class object (MATH or PHY)**. Compare students' randomly generated scantron to **the answer key** and update the (initially empty) dictionary assigned to tests_taken as follows:

- The **tests_taken dictionary** should be capable of storing the subject of the test, the student's grade on the test, and whether they passed or failed based on their grade. See the format as an example below. (This is just an example to show how an output looks like for a student in student_list. Since you create random scantrons your answers may not be the same.)

```
student_list[0].tests_taken → {}
student_list[0].take_test(MATH, pass student's math test scantron here)
student_list[0].tests_taken → {"Mathematics": {"Score": 80, "Result": "Pass"}}
student_list[0].take_test(PHY, pass student's physics test scantron here)
student_list[0].tests_taken → {"Mathematics": {"Score": 80, "Result": "Pass"},
                               "Physics": {"Score":40, "Result": "Fail"}}

student_list[0].tests_taken["Mathematics"]["Score"] → 80
student_list[0].tests_taken["Mathematics"]["Result"] →"Pass"
student_list[0].tests_taken["Physics"]["Score"] → 40
student_list[0].tests_taken["Physics"]["Result"] →"Fail"
```

d. Calculate and print the average score in math test.

e. Calculate and print the pass rate in physics test

3. (Very Easy – 6 pts)

There is a special relationship between the Golden Ratio (1.61803398874989484820…) and Fibonacci Numbers (0, 1, 1, 2, 3, 5, 8, 13, 21, … etc., each number is the sum of the two numbers before it).

When we take any two successive (one after the other) Fibonacci Numbers, their ratio converges to the Golden Ratio as you can see in the table.

A	B	B/A
2	3	1.5
3	5	1.666666666…
5	8	1.6
8	13	1.625
…	…	….
144	233	1.618055556…
233	377	1.618025751…
…	…	…

a. Start a Fibonacci series with **2 and 3 (no recursion needed)** and keep building the series in a list until the absolute error gets smaller than 10^{-20}.
b. Print the **last two numbers** of the Fibonacci series you built and **their ratio**.

4. (Normal – 6 pts)

A **subsequence** is a sequence that can be derived from another sequence by deleting some or no elements **without changing the order** of the remaining elements. For example, *(ABD)* is a subsequence of *(ABCDEF)*. However, *(CBA)* is not a subsequence of *(ABCDEF)* since the order among the elements does not hold.

- You are given a list of word (**wordlist**) and a string (**longstring**). Your aim is to detect how many words in wordlist is a subsequence of longstring.
- You are given an empty list called **subs** which stores the words that are a subsequence.
- There are duplicates in the word list, if you have already added a word to list sub, skip searching the same word again.
- You might want to use **continue** and **break** commands just to make your code faster.

```
print(sub) ➞ ['in', 'spite', 'of', 'war', 'hate', 'lilacs', 'are',
'at', 'my', 'gate', 'the', 'glory', 'just', 'to', 'live', 'is',
'so', 'divine', 'when', 'lift', 'their', 'eyes' 'mine']
```