Lists: Operate on Lists

In these activities, you will learn various ways to operate on lists.

Objectives:

- AP Computer Science Principles Standards:
  - AAP-2.N 2.B Write expressions that use list indexing and list procedures
  - AAP-2.N 4.B Evaluate expressions that use list indexing and list procedures
  - AAP-2.O 2.B Write iteration statements to traverse a list
  - AAP-2.O 4.B Determine the result of an algorithm that includes list traversals
- Programming Objectives:
  - Perform operations on lists and their elements
  - Use operations to analyze lists and their elements

Activity Summary:

Activity 1: TRY – Students will take in a list of values and display their average.
Activity 2: TRY – Students will take in a list of numbers and display the sum of all even integers.
Activity 3: DO – Students will write a program that will take in a variety of types of grades and append them to the correct list. The program will then display the class average based on weighted averages.

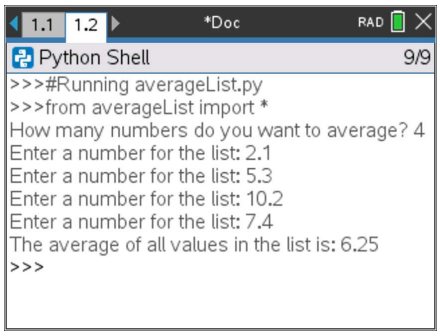
TRY activities have less structure to allow students to try applying their knowledge from activity 1. DO activities have significantly less structure to the activity and should show whether or not students have grasped the concept of WHILE loops. They can be seen as a culminating activity or performance task.

Activity 1: TRY

In this activity, students will take in a list of values and display their average.

Write a program that will take in a list of values from the user. It will then display the average of all values stored in the **list**.

1. Begin a new Python file and name it **averageList.py**.
2. Write an input statement that asks the user for how many values they want to average. This user input will be used to control how many values to add to the list. Store the user input in a variable.
3. Create an empty list to store the values in. Let's name it **nums**.



4. Create a **for** loop that runs the correct amount of times, using the user input as the stopping value. Inside of the loop, ask the user for values to add to the list. Use the **.append()** command to add values to the list as the loop iterates.

Tech tip:
What kind of data will the user be entering? Remember to cast their input appropriately so we can do math with the values. Cast using **int()** for integer input or **float()** for decimal input

There are multiple ways to find the average (mean) of elements in a **list**. Two ways will be described below, try them both! One makes use of *built-in functions* in Python, the other makes use of **for** loops to *traverse* the list. Both are beneficial to be aware how to use.

5. First, let's try using the *built-in functions* in Python. Think through the process of finding the average (mean) of a set of numbers. What does it entail? Finding the sum of all numbers and then dividing by how many numbers are in the set, right?
   a. Python has a **sum()** list function that you can select from **menu > Built-ins > Lists.**
   b. Python also had a **len()** function that will return the length of the list, or how many elements are stored in it.
   c. Combine the use of these two functions to find the average:
      mean = sum(nums)/len(nums)
   d. Print **mean** for the user to view.

6. For the second version, we will use a second **for** loop so we can practice traversing a list. We still need to find the sum of all elements and then divide by the total amount of elements. Which parts of that process belong inside of the loop block? Outside of the loop block? Is there anything that should be done before the loop ever begins? Write your thoughts here, then turn them into code. Print the results and see if both methods give you the same mean. Try it before looking ahead to the next box!

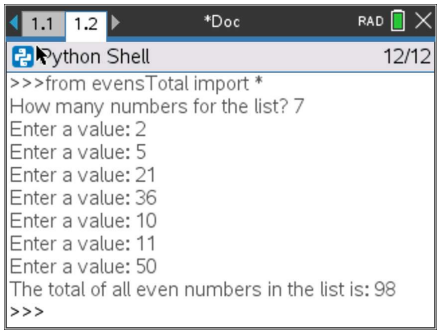
7. For the second option above, you would want to create a **total** variable initialized to 0 before creating a second **for** loop. Or to be honest, you could make this part of your original **for** loop! Either way, initialize **total** prior to the loop where you will be modifying the value of **total**.
8. The **for** loop should start at 0 and stop at **len(nums).** Inside of the loop, add to **total** through each iteration.
9. After adding all of the elements from the list, the loop is complete. Divide **total** by the length of the list, and print!

Tech tip:
Reminder: Access an element in the list by using the name of the list followed by the location: **nums[2]**

Activity 2: TRY

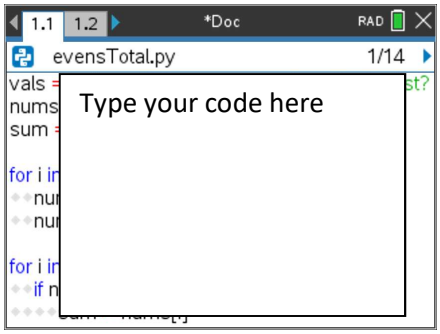
In this activity, students will take in a list of numbers and display the sum of all even integers.

Write a program that asks the user to enter a list of numbers. The program should then calculate and display the total of all even numbers added to the list.



1. Begin a new Python file and name it **evensTotal.py**.
2. Write an input statement that asks the user for how many values they want to add to the list and store the input in a variable.
3. Create an empty list to store the values in. Let's name it **nums**.
4. Create a **for** loop that runs the correct amount of times, using the user input as the stopping value. Inside of the loop, ask the user for values to add to the list. Use the **.append()** command to add values to the list as the loop iterates.
5. Now it is time to find the sum of all even numbers in the list. Create a **for** loop that will access every element in the list. Since we are finding the sum of various elements in the list, where do you think we should create and initialize the **sum** variable?
6. As the loop iterates, check each element to see if it is even or not. How do we identify even number with code?

   If the element is even, add it to **sum**.



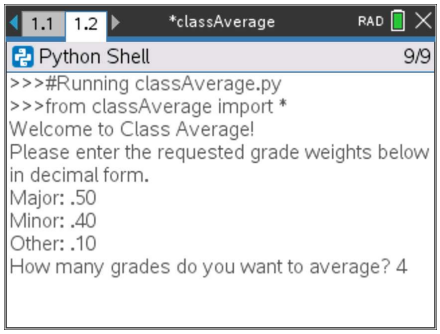
7. Last, display the results to the user.

Possible extensions: Can you modify the code to display the sum of all odd numbers? Multiples of 5? What else can you think of?

Activity 3: DO

In this activity, students will write a program that will take in a variety of types of grades and append them to the correct list. The program will then display the class average based on weighted averages.

Write a program that will ask the user to enter grade weights for different categories of grades: Major, Minor, and Other. Ask the user how many grades they would like to enter, and then take in those grades, adding them to three different lists based on grade type. Finally, the program should calculate the user's grade average based on the grades entered and types.



1. Begin a new Python file and name it **classAverage.py**.
2. What variables do to think you will need? What **lists**? Think about what input you will be taking in and what you will be doing with that input. Plan your lists, variables, and initialization here:

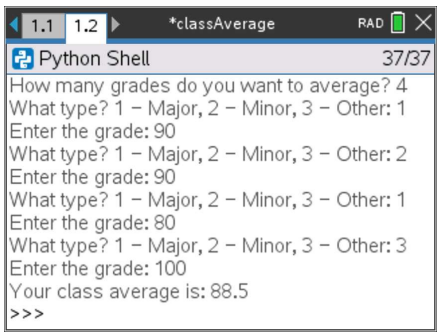
3. Ask the user what percent weight each grade category should have based on the type of course they are taking (ex: Major – 50, Minor – 40, Other – 10). Store the user's answers in variables.

4. Ask the user how many grades they would like to average. This will drive your **for** loop

5. Inside of the **for** loop, ask the user to input the type of grade, and then what the grade is. Your program should append the grade to the correct list depending on what type of grade it is. What code structure would benefit you in this task?

Tech Tip:
If your text wraps around to the next line while running the program and you are asking the user for input, you may encounter an *error*. Keep prompts to a single line of text.

6. Now that the grades have been entered into the lists, it is time to find the user's class average. Think about how this would be accomplished without code!
   HINT: class average = (major percent*major grades averaged) + (minor percent*minor grades averaged) + (other percent*other grades averaged)



   Now it's your turn, turn this process into code! When you are done, display the average to the user.