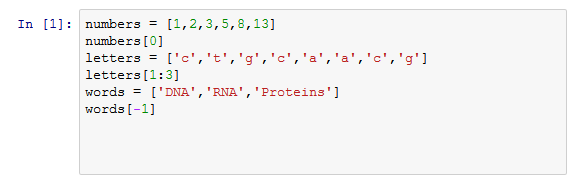
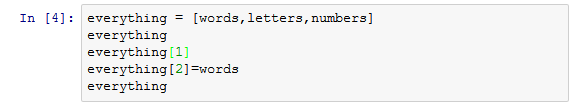
**Objectives**

1. Work with List data structures
2. Apply For loops to your desired application

**Lists**

Lists are similar to strings in that they are a data structure with a sequence of element. Instead of characters, you can put any data type into a list structure (numbers, strings, other lists, etc…). Each element is stored under a certain index and can be retrieved using bracket notation. To retrieve a range of indices you can use slice notation (*i.e.* [start:stop] ) like we did with strings. Enter the following and inspect the outputs individually.



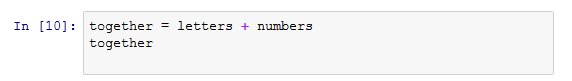


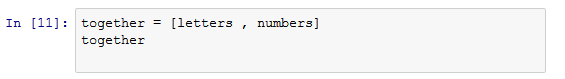
**Thinking Critically about Lists**

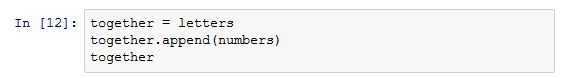
1. When making lists of lists, do the dimensions need to match (i.e. do all sublists need to have the same length?)
2. Do lists require that all elements be of the same data type (i.e. all strings, all numbers, etc…)?
3. Write a list of how strings and lists are similar and different.

**Building a List**

There are multiple ways to add on to a list in python. Which method you choose is dependent on what you want to add to your list and what you want your list to look like. Type the code below in individual cells in your notebook and inspect each output individually:







**Thinking Critically about Adding to Lists**

1. How do these methods of adding to a list differ from each other?
2. Add the String ‘Lipids’ to the words list so that Lipids is the 4th element (index = 3)
3. Create a separate list called ‘complementary’ that contains the complementary nucleotide for each element in the letters list. (complementary = [‘g’,’a’,’c’,’t’,’t’,’g’,’c’]). Create a new list of lists that stores the ‘letters’ and ‘complementary’ lists as two individual elements.
4. Add the next two numbers in the fibonnaci sequence to the numbers list ( add 21 and 34) so that length of the numbers list is 8.

**Built in List Methods**

Lists have many different built in methods to aid in data manipulation. If you recall from lesson 3, strings are immutable objects and could not be changed. In contrast, lists are mutable objects and can be changed by the built in methods. This means any built in list function you run will change the original list.









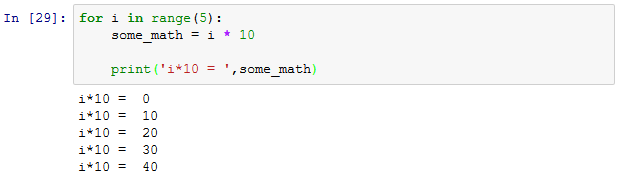


You can view all of the built in list functions by running ‘help(list)’ or by using the tab drop down menu ( type: numbers followed by a period then press tab).

**For Loops**

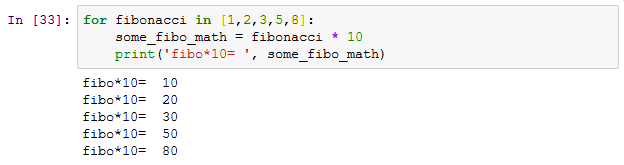
Loops are coding structures that repeat the same portion of code multiple times. For Loops define how many times to repeat the code in the beginning. While loops repeats the code until a conditional statement has been met. Since we haven’t gone over conditional statements yet, today’s lesson will only focus on for loops.

In python the basic loop structure is seen below:

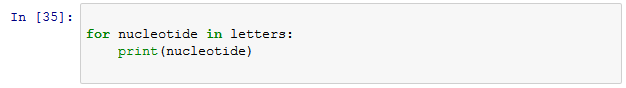


‘i' is a new variable we declare, this will keep track of our iterations. When indexing a range, ‘i' is a integer starting at the beginning of the range (typically 0) and going till the end (for the example above, i stops at 4). Every for statement ends with a ‘:’. Underneath the for statement is code, indented over by one tab, that will repeat for every iteration. In this example we want to do some math on our variable ‘i’ every iteration. Because i is increasing with every iteration, out output variable ‘some\_math’ also changes.

For loops are especially useful for iterating through the items in a list. The iterating variable cycles through each element of the list.



For loops can cycle through a list of any type. Type and run the following in a cell:



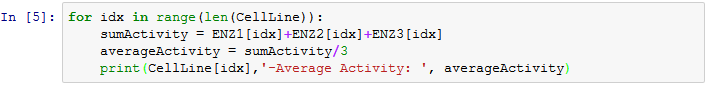
Notice that nucleotide became each of the characters in the list letters as it iterated through loop.

**Practicing For Loops**

Download and Open the ipynb titled ‘Lesson4-variables’. Run all of the cells (**Cell>>Run All**). This notebook contains normalized enzyme activity data for 3 enzymes. Each enzyme list contains 9 values that correspond to the activity of a cell line with matching index in the CellLine list. For example ENZ1[0] corresponds to Enzyme 1 activity level for CellLine[0].

1. Use a for loop to calculate the average of enzyme activity for ENZ1, ENZ2, and ENZ3.

We can iterate through multiple lists simultaneously. This is useful when you need the same index across multiple lists for a calculation. For example, lets say we want to calculate the average activity between the three enzymes for every cell line. To do this test the following code in a cell:



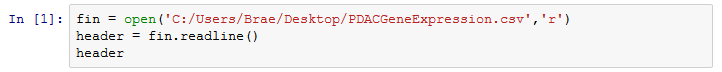
In this example we want to iterate through a range that is equal to the length of our CellLine list. We use the variable ‘idx’ as an index to get the corresponding elements from each of the three enzyme activity lists.

**Thinking Critically about Iterating Lists with For Loops**:

1. How would you change the above code to save the ‘averageActivity’ to a new list titled ‘AVG’?
2. There are multiple ways that for loops can iterate through multiple lists. Google the following search to find a new way to iterate through multiple lists. Adjust the code above to use this new way:
   1. ‘How to iterate through multiple lists in a for loop python’

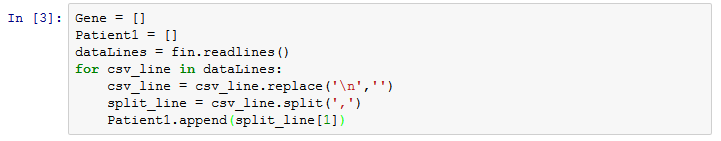
**Iterating Files using For Loops**

For loops can be used to iterate through all the lines of a file. Download the PDACGeneExpression.csv file in the lesson 4 folder. This is an abbreviated file downloaded from the TCGA database. It contains normalized gene expression values for patients with pancreatic cancer. Open the file and read in the first line.



You should see from the line output that this file is in the comma separated value format. Split the ‘head’ variable using comma’s as the separator and save it as a variable.

 Investigate the output: notice each element of the split list is a header title. The first element is the HugoSymbol which is used to identify the genes. The remaining elements in the list are the sample id’s. First we are going to pull all the gene names and the first patients gene expression values from our file. To do so we are going to iterate through every remaining line of the file and append the first and second elements to two lists titled Gene and Patient1.



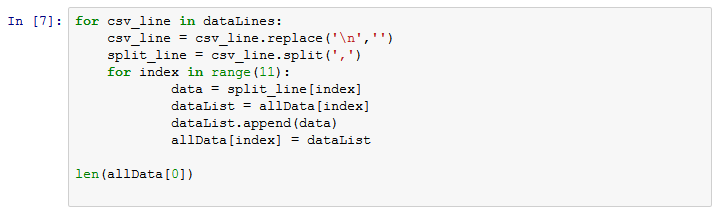
**Thinking Critically with Files and For Loops**

1. What is the length of the Gene and Patient1 Lists? Would you expect them to be the same?

**Nested Loops**

Often we need to work with 2 or more dimensional arrays. For example the gene expression data we used above is a two dimensional array with multiple columns and rows. We can create individual lists for each column but this is time consuming and error prone. Instead we will take advantage of nested loops to quickly create a two dimensional list (a list of lists). Create a new list titled ‘allData’. Using the ‘head’ list from earlier, append a list to the ‘allData’ variable for every column of information. 

The output of allData is a list with 11 empty lists inside. Modify the cell with the for loop we used earlier to get data from a single patient. In the for loop, add a second for loop that will cycle through all 11 columns data. We will index our ‘allData’ list to pull out each individual list. To every individual list (column) we will append the corresponding data value from the ‘split\_line’ variable (row). Validate that the length of the first sublist in ‘allData’ is the same as the length of the ‘Patient1’ list.



Homework:

In the Lesson 5 folder are two CSV files that represent a fluorescence image of cells. A fluorescence stain was used to determine the activity of two different enzymes. Each value represents a pixel of the image. The magnitude of the pixel reflects the activity of the enzyme on that sample.

1. **Import the Channel 1 and Channel2 image files and use a nested for loop to create a list of lists. Each sublist will hold the pixel value for a given row and colun. When you append data to each list use the function ‘eval()’ to convert your string to a float.**
2. **Calculate the sum of the list of lists using your nested function. This summed intensity value roughly correlates to the enzyme activity of the two enzymes stained for in the image. Record the channel that had the greatest summed intensity**
3. **Find the most intense (maximum value) in both channels and save their values in your Jupyter Notebook.**