CS21 Lab – Lists and Tuples
Write your answers in comments in your code as you work through these problems.

Part One

1. Create `years`, a **list** of the values 2016,1620,1984,1776,1860,1492. Then write the `print` statements in order to display each of the following:

   a. The data type of `years` (should be `list`)
   b. The entire list, using only one print statement and no loops
   c. The value 1620, by indexing into the list with one positive integer.
   d. The values [1984, 1776, 1860], using list slicing to index into `years`.
   e. The minimum and maximum values in the list. (Use `min` and `max`.)

   f. A vertical listing of `years`, using a `for` loop as shown on page 346 of your text.

   g. A vertical listing (as shown), using the following strategy:

      i. Use the `len` function to define n, the number of elements in years.
      ii. Use a `for` loop with n as its `range` argument.
      iii. Use sep = '' in your `print` statement.

```
Yr0: 2016
Yr1: 1620
Yr2: 1984
Yr3: 1776
Yr4: 1860
Yr5: 1492
```

2. Repeat the above steps in #1 to create `years2`, a **tuple** containing the same sequence of values. Besides the variable name, what else must you change?

3. Write an assignment statement to replace the `years` (this is the list, not the tuple) value 1860 with 1865. In your statement, the target (left side of equal sign) should use years with one positive index.

4. Add this assignment statement: `years2[4] = 1865`. What happens? Why?
   Add a Try-Except statement to catch the type of error that was thrown.

5. Use the `remove` method to remove 1984 from `years`. Does `remove` work for `years2`? Why or why not?

6. Use the `sort` method to rearrange `years` in ascending order. Then use `reverse` to arrange `years` in descending order. Does this work with `years2`? Why or why not?

   Notice that you do NOT need an assignment statement when using sort, remove, etc. Simply call the method: `years.sort()`

Part Two

1. Create a new list, `dorms`. The list should be empty. Verify this by using the `len` function to print the length of the list.

2. Use the `append` method to add "Converse" to `dorms`.

3. Use the `insert` method to place "Mason" at the front (index = 0) of the list. Then use the `index` method to locate the position of "Converse"', using a print statement to display result.

4. Add values to the end of your list using two different approaches. After each update, inspect your result using a `print` statement.

   Step 1: Using the append method:

   ```
   dorms.append(['UHeights','Redstone','Trinity'])
   ```

   Question: What is the value of `dorms[2]`? What *type* is the value in `dorms[2]`?

   Step 2: Using list concatenation (using the + operator), add the list `['Christie','Wright','Patterson']` to the end of `dorms`.

5. Given the following code:

```
central = ["WE","Converse"]
athletic = ["UHeights","Harris/Millis","MAT", "L/L"]
redstone = ["CWP", "SMH", "WDW"]
housing = [central, athletic, redstone]
```

What is the type of `housing`?
What is the type of `housing[0]`?
What is the type of `housing[0][0]`?
Using only the housing variable, how would you print out "MAT"?

Part Three

1. Create a new list, `odds`. Use the list and range functions for this purpose. The list should contain the odd numbers between 51 and 67, inclusive. Verify your work by displaying `odds`. In the same manner, create a second list, `evens`, containing the even numbers between 1 and 10 (inclusive). Finally, concatenate `evens` and `odds` to create `nums`.

2. Create a list called nums2 using the assignment statement: `nums2 = nums`.

3. Use the del statement to remove the 6th element in **`nums`**. Print `nums` and `nums2`. What do you notice?

4. Now make `nums2` a real copy of `nums` as shown at the bottom of page 363. Repeat Step 3. What is different?