CS417 Lab 11: List Comprehensions

Many python tasks involve operations on lists. Often one list (the source) will be used to make a new list (the target). These operations often have three sub-operations:

- Visit every element in the source.
- Apply a filter to each element. This is an `if` that determines whether the element should be skipped (Filters are optional, and may be omitted).
- Apply a transformation to each element, yielding a new value.

The new values will make up a new list.

- Example 1: build a target list, containing the squares of the values in the source:

  ```
  target = []
  for x in source:           # visit every element
      target.append(x * x)  # transform (square) each element
  ```

  This example can be re-written as a single statement, using a list comprehension:

  ```
  target = [x * x    for x in source]
  ```

- Example 2: build a target list, containing the squares of the even values in the source:

  ```
  target = []
  for x in source:               # visit every element
      if x % 2 == 0:             # filter to get only even elements
          target.append(x * x)  # transform (square) each element
  ```

  This example can be re-written as a single statement, using a list comprehension:

  ```
  target = [x * x    for x in source    if x % 2 == 0]
  ```

  Notice that a list comprehension has three parts: [(transformation) (visit elements) (filter)]

- Example 3: what is the sum of the first 10 squares, $1 + 2^2 + 3^2 + ... + 10^2$? It's a one-liner using a list comprehension:

  ```
  sum_10_squares = sum([x*x for x in range(11)])
  ```

Origins in Set Notation

List comprehensions imitate set-builder notation in mathematics. You may have seen the first 10 squares expressed thus:

$\{x^2 \mid x \in \{1,2,\ldots,10\}\}$

or the set of all even numbers in set S:

{x ∈S | x mod 2 = 0}

Exercises

First, download the file `comprehensions.py`. Open that file, and notice the list `upto_ten` which has the value [1,2, ... ,10].

Remember that a list comprehension has three parts:

```
    [(transformation)  (visit elements)  (filter)]
```

For each of the following values, write a one-line python statement that uses a list comprehension.

Your program may NOT contain any while or for loops that are not part of a list comprehension.

1. `evens_to_20`, a list of all the even numbers

   ```
         [2, 4, ..., 20]
   ```

2. `upto_ten_odd`, a list of all the odd numbers from 1 to 10

   ```
         [1, 3, ..., 9]
   ```

3. `upto_ten_squared`, a list of all the squares

   ```
         [1, 4, 9, ..., 100]
   ```

4. `upto_10_and_square`, a list of pairs of numbers and their squares:

   ```
         [[1,1], [2,4], [3,9], ..., [10,100]]
   ```

5. `upto_5_pairs`, a list of all the possible pairs of numbers from 1 to 5:

   ```
       [[1,1],[1,2], ..., [1,5],[2,1],[2,2], ..., [2,15], ... [5,5]]
   ```

   For this question, your comprehension will need two for-loops inside it: `for x in range(1,6) for y in range(1,6)`. The transformation is just `[x,y]`.

6. `upto_5_products`, a list of all the possible products of numbers from 1 to 5. This is just like the previous question, but the transformation is `x*y`. As in the previous question, you will need two `for`-loops in your comprehension.

7. Given a 2-dimensional list matrix, such as

   ```
   [[1,2,3], [4,5], [6,7]]
   ```

   flatten it into a one dimensional list, such as

   ```
   [1,2,3,4,5,6,7]
   ```

   To do this, you will need two loops in your comprehension: one loop goes through all the entries in the list matrix, and the second loop goes through each entry.

8. Given a list of names, get a list of their first letters (their initials of the names).

9. Given a list of pairs, such as

   ```
   [['Malinda',1.5], ['Indrajit', 1.6],...]
   ```

   where the first is a name, and second is a height, get the names of people whose height exceeds 1.7.

10. Given a list of employee records, such as

    ```
        [
          [1,'Bagley','Malinda R',12],
          [2,'Wray','Indrajit H',15],
        ...
        ]
    ```

    where each record has an `id`, a last name, a first name, and pay rate, get a list with just the `ids`.

11. Using the same list of employees, get a list with pairs: [lastname, payrate], of the employees who are paid 15 or more.

12. Given two lists of strings, one with colors, one with things, return a list of strings, with all possible combinations of colored things. For example, if

    ```
        colors = ['red', 'blue']
        things = ['ball', 'box']
    ```

    you should get this list:

    ```
        ['red ball', 'red box', 'blue ball', 'blue box']
    ```

Turn in your work

To turn you work in, go to `mycourses.unh.edu`, find CS417 and the lab. Click the “Submit” button, and upload `comprehensions.py` . At the end of the lab session, submit any work you have completed. You can submit again until midnight, with no lateness penalty.