Cardinal Sin

No camelCase
Multiple Lists

You can use multiple existing lists to create the new list:

\[
pairs = [(x, y) \text{ for } x \text{ in } \text{xcoords} \text{ and } y \text{ in } \text{ycoords}]
\]

What are the elements of \( pairs \)?
- each is an \((x, y)\) pair in the cross-product of \( \text{xcoords} \) and \( \text{ycoords} \)
- this works even if the two input lists have different lengths

What if you want elements paired by position?

\[
pairs = [(\text{xcoords}[i], \text{ycoords}[i]) \text{ for } i \text{ in } \text{range(len(xcoords))}]
\]

Can use \( \text{min} \) if the input lists have different lengths.

More filtering

Let \( \text{data} \) be a list containing lists of instrument readings:

\[
[[d0_0, d0_1, ... d0_n], [d1_0, d1_1, ... d1_m] ...]
\]

Create a list of the \( \text{min} \) readings for each sublist.

\( \text{min} \) is a built-in function

\[
\text{mins} = [\text{min}(L) \text{ for } L \text{ in } \text{data}]
\]

Above gives error if a sublist is empty – can’t \( \text{min}([]) \)

\[
\text{mins} = [\text{min}(L) \text{ for } L \text{ in } \text{data if len(L) > 0}]
\]
“Double iteration”

Let data be a list containing lists of instrument readings:

\[[d_{0,0}, d_{0,1}, \ldots d_{0,n}], [d_{1,0}, d_{1,1}, \ldots d_{1,m}] \ldots\]

Combine all of the elements into a single list.

\[
\text{all\_data} = [x \text{ for } L \text{ in data for } x \text{ in } L]
\]

If we only want non-negative values:

\[
\text{all\_data} = [x \text{ for } L \text{ in data for } x \text{ in } L \text{ if } x \geq 0]
\]

Exercise 1

Let data be a list containing lists of instrument readings:

\[[d_{0,0}, d_{0,1}, \ldots d_{0,n}], [d_{1,0}, d_{1,1}, \ldots d_{1,m}] \ldots\]

Write a list comprehension that creates a list of the mean values of each of the sublists in data:

\[
\text{means} = [\text{sum}(L)/\text{len}(L) \text{ for } L \text{ in data if } \text{len}(L) > 0]
\]

Why is the filter necessary?
Exercise 2

Let chars be a list of unique characters. Write a list comprehension that uses chars to create a list of 3-letter “words” such that no two letters in a word are the same.

Without the restriction on no repeated characters:

\[
\text{words} = [x+y+z \text{ for } x \text{ in } \text{chars} \text{ for } y \text{ in } \text{chars} \text{ for } z \text{ in } \text{chars}]
\]

This version eliminates repeated characters:

\[
\text{words} = [x+y+z \text{ for } x \text{ in } \text{chars} \text{ for } y \text{ in } \text{chars} \text{ for } z \text{ in } \text{chars} \text{ if } x!=y \text{ and } y!=z \text{ and } x!=z]
\]

Exercise 3

Write a list comprehension that uses a list of random integers in the range 1..100 to create a list of those integers in the list that contain a 7. You are not give the list of random integers – you must create it in the comprehension.

\[
\text{sevens} = [x \text{ for } x \text{ in } [\text{randint}(1,100) \text{ for } _\text{ in } \text{range}(20)] \text{ if } ‘7’ \text{ in } \text{str}(x)]
\]