CSC110 Lecture 3: Comprehensions and Introduction to Functions

Print this handout

Exercise 1: Practice with comprehension expressions

Here is a summary of the three types of comprehensions, for your reference:

Comprehension type	Syntax
set comprehension	`{ <expression> for <variable> in <collection> }`
list comprehension	`[ <expression> for <variable> in <collection> ]`
dict comprehension	`{ <key_expr>: <value_expr> for <variable> in <collection> }`

1. Suppose we assign the variable `numbers = [1, 2, 3]`.
    a. Fill in the table below.

Expression	Value
`numbers[0]`	
`numbers[1]`	
`numbers[2]`	
`numbers[0] ** 3`	
`numbers[1] ** 3`	

Expression	Value
`numbers[2] ** 3`	

b. Write a comprehension that evaluates to the *list* of every integer in `numbers` cubed (i.e., raised to the power of 3).

```
>>>


[1, 8, 27]   # Evaluating your expression should produce this
```

c. Write a comprehension that evaluates to a *dictionary* mapping every integer in `numbers` to three times that integer.

```
>>>


{1: 3, 2: 6, 3: 9}   # Evaluating your expression should produce
        this
```

Hint: the *identity dictionary comprehension* has the following form:

```
>>> {x : x for x in numbers}
```

d. Write a comprehension that evaluates to the given output dictionary shown.

```
>>>


{3: 1, 6: 2, 9: 3}   # Evaluating your expression should produce
        this
```

e. Write a comprehension that is a translation of the set builder expression $\{\frac{x}{x+1} \mid x \in \texttt{numbers}\}$.

Exercise 2: Comprehensions and `range`

1. Write down the integers that are contained in each of the following Python `range` expressions

   a. `range(0, 5)`

   b. `range(5, 10)`

2. For each of the following descriptions, write a comprehension that evaluates to the described collection.

   a. The set of integers between 30 and 50, inclusive.

   b. The list of integers between -30 and 30, inclusive (in ascending order).

   c. The set of the squares of the natural numbers less than 2000.

   d. A mapping from a number to its square, for the natural numbers less than 2000.

3. You are given a variable `s` that refers to a (very very long) string:

```
>>> s = 'nonsensenonsensenonsensenonsensenonsensenonsensenonsense'
```

Write a list comprehension expression that evaluates to a list containing the first 20 characters in the string, in the order they appear.

Hint: `s[19]` is the last character in `s` that should be included in the list.

Exercise 3: Practice with built-in functions

Suppose we have assigned the following variables in the Python console:

```
>>> n = -5
>>> numbers_list = [1, 10, n]
>>> numbers_set = {100, n, 200}
```

1. Complete the following table showing the value of each variable.

Variable	Value
`n`	
`numbers_list`	
`numbers_set`	

2. Write down what each of the following expressions evaluate to. *Do this by hand first! (Then check your work in the Python console.)*

   You may find it helpful to consult Appendix A.1 Python Built-In Function Reference (https://www.teach.cs.toronto.edu/~csc110y/fall/notes/A-python-builtins/01-builtins.html).

```
>>> abs(n)



>>> sorted(numbers_list)



>>> sorted(numbers_set) + sorted(numbers_list)



>>> len(numbers_set)



>>> len(numbers_list) == n



>>> sum(numbers_set) - n
```

3. The variable `numbers` refers to a list that contains a mix of positive and negative integers (e.g. `[-1, 2, 3]`). Write a comprehension that evaluates to the set of the absolute values of every integer in `numbers`.

   (Hint: the structure is the same as earlier problems on this worksheet. Use the `abs` function.)

Additional exercises

1. *Comprehension practice*. For each of the following mappings, write a Python dictionary expression that evaluates to the mapping.

a. A mapping from a number to its square, for the first 50 natural numbers.

b. A mapping from input to output of the function $f(x) = \frac{x}{x-1}$, for integer inputs greater than 1 and less than 2000.

c. A mapping from a number to a list that contains the same number of items, where every item is the string `'Hello'`, for the first 50 natural numbers. (e.g., `3` maps to the list `['Hello', 'Hello', 'Hello']`.)

d. A mapping from an integer to the set of integers between 0 and that integer inclusive, for integers 1 to 20, inclusive.

2. *Comprehensions with multiple variables*. Suppose you have the lists: `nums1 = [1, 2, 3]` and `nums2 = [4, 5, 6]`.

   a. Using both `nums1` and `nums2`, write a comprehension that evaluates to: `[[1, 4], [1, 5], [1, 6], [2, 4], [2, 5], [2, 6], [3, 4], [3, 5], [3, 6]]`.

   b. Using both `nums1` and `nums2`, write a comprehension that evaluates to: `[[4, 1], [5, 1], [6, 1], [4, 2], [5, 2], [6, 2], [4, 3], [5, 3], [6, 3]]`.

   c. Using both `nums1` and `nums2`, write a comprehension that evaluates to: `{5, 6, 7, 8, 9}`.

3. *Function practice*. Using the same variables defined in Exercise 3, determine the value of each of the following Python expressions.

```
>>> type(n)



>>> type(abs(n))



>>> type(numbers_list == n)



>>> type(numbers_list) == type(n)



>>> max(numbers_list + [5])



>>> max(numbers_list) + 5



>>> max(sorted(numbers_list)) == max(numbers_list)
```

4. *Interpreting errors*. Your friend is practicing in the Python console again, and is trying to add two numbers. They type in the following, and get an error:

```
>>> sum(3, 4)
Traceback (most recent call last):
  ... [some output omitted] ...
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

Once again, explain this error to your friend, and how they can correctly add two numbers in Python. (*Hint*: treat "iterable" as another word for "collection".)