CPSC 5061—Programming I for Business

Worksheet 15

Comprehensions

We're going to take a step back from classes for a minute and reinforce our understanding of the collection datatypes, **`tuple`**, **`list`**, and **`dict`**. These core built-in datatypes are ubiquitous in Python including as components and tools within our own classes.

Motivation

We use some patterns so frequently that it makes sense to make them as easy as possible to program. Python has added some built-in syntactical support to common for-loop patterns used to build tuples, lists, and dictionaries.

Here's the canonical list-building pattern. First the simple version:

```
list_built = []
for datum in data:
    list_built.append(some_func(datum))
# list_built is a list of all the (tranformed) data
```

And then the more general version with a condition:

```
list_built = []
for datum in data:
    if condition(datum):
        list_built.append(some_func(datum))
# list_built is a list of all the (tranformed) data that we want
```

Concept

A syntactic shortcut for the list-building pattern is a *list comprehension*. Python likely had inspiration from the Haskell language which had a way of putting set-building notation from math into their language. Python does similarly with their short-cutting list comprehension.

The above loop patterns can each be written in a single statement.

```
list_built = [some_func(datum) for datum in data]

list_built = [some_func(datum) for datum in data if condition(datum)]
```

Exercises

Rewrite each of the following loops using a list comprehension.

1.
```
list1 = []
for i in range(9, -3, -1):
    list1.append(i*i)
```

2.
```
list2 = []
for c in '1234987723':
    list2.append(int(c))
```

3.
```
list3 = []
for i in range(100_000):
    if i%3 == 0:
        list3.append(i)
```

Rewrite each of the following list comprehensions as a for loop.

4.
```
x = [i**3.2 for i in range(300)]
```

5.
```
y = [int(c) for c in s if c.isdigit()]
```

6.
```
z = [d[k] for k in d if type(k) == int and k%2 == 0]
```

7.
```
w = [(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]
```

Dictionary Comprehensions

Similar to lists, dictionary-building loops can also be done using comprehensions. The syntax is similar, borrowing from list comprehensions and dictionary literals as you would expect.

```
squares = {}
for i in range(100):
    squares[i] = i*i
```

becomes:

```
squares = {i: i*i for i in range(100)}
```

Exercises

Rewrite each of the following loops using a dictionary comprehension.

8.
```
dict1 = {}
for weekday_num in iso_weekdays:
    if weekday_num < 6:
        dict1[iso_weekdays[weekday_num]] = 0
```

9.
```
swap = {}
for key in d:
    swap[d[key]] = key
```

Rewrite each of the following dictionary comprehensions as a for loop.

10. `rand_map = {randint(lo, hi): i for i in range(10_000)}`

11. `x_y_points = {x: f(x) for x in x_vals if abs(x) < 1.56e14}`

Tuple Comprehensions are Not Tuples

Unlike list and dictionary comprehensions, comprehensions with parentheses produce something like a tuple, but it is really a generator object. You cannot index it, but you can use it in a for loop. And only when you use it in a for loop is it evaluated and then only one item at a time. This is useful to save time and space by not physically producing the whole sequence all at once.

```
a_list = (x for x in seq if we_like(x))
```

acts like a tuple in a for loop:

```
for star in a_list:
    print(star, 'is a star!')
```

An example is a gigantically-long sequence for which we don't need to all the contents to be around at once; we just care about getting one at a time.

Here's a contrived example:

```
samples = (uniform(lo,hi) for i in range(1_000_000_000))
for sample in samples: # does NOT call uniform a billion times at once
    weight += weigh(sample)
    if weight >= target:
        break
```

Exercises

Rewrite the following loop using a generator comprehension.

12.
```
values = ()
  for i in range(1, n+1):
      next_val = input('value ' + str(i) + ': ')
      values += (next_val,)
```

Rewrite the following generator comprehension as a similar for loop.

13.
```
big_vals = (x for x in x_vals if f(x) > big_lim)
```