SOLNS

In-Class Activity: List and Dictionary Comprehensions in Python

Work with a partner. You will likely be asked to turn in the completed activity with your next homework assignment.

Comprehensions are compact ways of defining **lists** or **dictionaries** in one line of code.

List Comprehensions.

We will start with **list comprehensions**. For example, suppose we want to construct a list of the squares of integers from 0 to 9

Using a loop we would type:	Here is the same result obtained using a **list comprehension**:

`my_squares = []`
`for x in range(10):`
`  my_squares.append(x**2)`
`print(my_squares)`
	`my_squares2 = [x*x for x in range(10)]`
	`print(my_squares2)`

Both result in the same output of: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

The general concept of a **list comprehension** is: *values = [expression for item in collection]* where the collection can be a list, range, etc.

1. Go through the commands in the first column of the table below, filling out the other columns as you proceed, testing the code in the console or Jupyter:

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`x_values = [i for i in range(10)]`	[0, 1, 2, ..., 9]	✓	
`y_values = [x**2 for x in x_values]`	[0, 1, 4, ..., 81]	✓	

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?

`import random`
`rand_nums = [random.random() for i in range(5)]`
(you can make up random #s with 1 decimal place)
	[0.2, 0.3, 0.3, 0.7, 0.1]	✓	

`my_list = [i+i for i in ['a','b','c']]`	['aa', 'bb', 'cc']	✓	

2. The list comprehensions in this question are built a little differently than the ones above. How? _____

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`y_even = [y for y in y_values if y % 2 == 0]` (see above for defn of y_values)	[0, 4, 16, 36, 64]	✓	
`y_special = [y for y in range(20) if y % 2 == 0 and y not in my_squares]` (See the top of the worksheet for our definition of `my_squares`.)	[2, 6, 8, 10, 12, 14, 18]	✓	

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`y_special_2 = [y for y in range(20) if y % 2 == 0 if y not in my_squares]` (Compare this statement and its output with the one above. Output should be identical.)	[2, 6, 8, 10, 12, 14, 18]	✓	
`y_new = [k**2 if k%2==0 else k**3 for k in [1,2,3,4,5]]`	[1, 4, 27, 16, 125]	✓	
`has_i = [item for item in some_list if "i" in item]` ~~(Assume~~ some_list ~~is a list of strings. Create one to test this code.)~~ = ['hello', 'ira', 'maria']	['ira', 'maria']	✓	
`small_words = [word for word in sometext.split() if len(word) < 5]` (Assume sometext = a string containing some text, like a paragraph of writing ~~Create one to test this code~~)	['a', 'some', ~~'text'~~, 'like', 'a', 'of'] ← No, includes comma	✗	

3. The list comprehensions in this question are built a little differently than the ones above. How? _____

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`A1 = [i*j for i in range(4) for j in range(3)]`	[0,0,0,0,0,1,2,3, 0,2,4,6]		
`A2 = [[i*j for i in range(4)] for j in range(3)]` Pay particular attention to the difference between this one and the one before it.	[[0,0,0,0], [0,1,2,3], [0,2,4,6]]		
`prods = [i * j for i in [20, 40, 60] for j in [2, 4, 6]]`	[40, 80, 120, 80, 160, 240, 120, 240, 360]		
`rand_nums = [[random.randint(1,10) for _ in range(3)] for _ in range(6)]` (Make sure you have imported `random` first.) P. 4 rand#'s	[[2, 5, 8], [4, 9, 2]]		

4. Your turn: write list comprehensions that generate the desired output below. For each of the following questions, perform the following steps:
   - hand-write your list comprehension on this worksheet
   - type the comprehension into the Python console
   - fix if necessary, and once it works, write a **large check-mark** next to it to signify it worked

1) a list of the values in `y_values` (defined above) that are greater than 20

```
[ y for y in y_values if y > 20 ]
```

2) a list whose elements are the letters in the word 'human' in order.

```
[ l for l in 'human' ]
```

3) a list of the cubes of the numbers from 10 to 20 inclusive

```
[x**3 for x in range(10,21)]
```

4) a 2-dimensional list (table) with 4 rows and 3 columns whose entries are equal to (row # + 1) / (col # + 1).

```
[[(r+1)/(c+1) for c in range(3)] for r in range(4)]
```

5) a list of the multiples of 3, 5, and 7 only that are less than 100 (pos)

```
[ i for i in range(1,100) if i%3==0 if i%5==0 if i%7==0 ]
```
(or use and's)

6) a list of all of the numbers from 1-1000 that have a 3 in them hint: convert the #'s to strings w/in the list comprehsn

```
[ i for i in range(1,1001) if '3' in str(i) ]
```

7) a random list of 10 lower case letters; Hint: use string.ascii_lowercase and random.choice

```
[ random.choice(string.ascii_lowercase) for i in range(10) ]
```

8) one of your choice, write the description here: `[math.sqrt(i) for i in range(5)]`

import math 1st

Dictionary Comprehensions

Dictionary comprehensions work the same as list comprehensions except that both the key and value are specified. We start with a simple example creating a dictionary from a list using the index value of the list items as the dictionary keys.

```
my_list = ['a','b','c']
my_dict = {i:my_list[i] for i in range(len(my_list))}
print(my_dict)
```

```
{0: 'a', 1: 'b', 2: 'c'}
```

Study this code and make sure you understand it.

5. Go through the commands in the first column of the table below, filling out the other columns as you proceed, testing the code in the console or Jupyter:

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`recips = {x:1/x for x in range(1,10)}`	{1:1, 2:0.5, 3:0.3, ..., 9:0.11}	✓	
`lowers = 'abcdefg'` `let_dict = {v:lowers[v] for v in range(len(lowers))}`	{0:'a', 1:'b', ... 6:'g'}	✓	
`vowels = 'aeiou'` `cons = {v:lowers[v] for v in range(len(lowers)) if lowers[v] not in vowels}`	{1:'b', 2:'c', 3:'d', 5:'f', 6:'g'}	✓	

Command(s)	What do you think it will do?	Were you correct?	What does it actually do (if you weren't correct)?
`word = 'apple'` `locs = {l:word.index(l) for l in word}`	{'a': 0, 'p': 1, ~~'p': 1~~, 'l': 3, 'e': 4} ← gets overwritten, unique keys!	✗	
`is_it_vowel = {lowers[i]: (i+1,lowers[i] in vowels) for i in range(len(lowers))}` This one is involved. The dict values are tuples. Of what?	{'a': (1, True), 'b': (2, False) 'c': (3, False), ... 'z': (7, False)}	✓	

6. Your turn: write **dictionary comprehensions** that generate the desired output below. For each of the following questions, perform the following steps:
   - hand-write your dictionary comprehension on this worksheet
   - type the comprehension into the Python console
   - fix if necessary, and once it works, write a **large check-mark** next to it to signify it worked

   1) a dictionary whose keys are integers from 0 to 5 and whose values are True if the key is even and False if not

   { i: i%2==0 for i in range(6) }

   2) given a list of words called `my_list`, define a dictionary whose keys are the first letters of the words in `my_list` and whose values are the lengths of the words in `my_list`. E.g. if `my_list = ['apple', 'banana', 'orange']`, then the dict would be: `{'a': 5, 'b': 6, 'o': 6}`.)

   { w[0]: len(w) for w in my_list }

Creating dictionaries using the *zip* command

Dictionaries can be created by **zipping** two containers (lists, ranges, etc.) together with the `zip` command. For example,

```
list1 = [10, 12, 14]
new_dict = dict(zip(range(3),list1))
print(new_dict)
```

```
{0: 10, 1: 12, 2: 14}
```

7. What do you think the following code will produce? Test it out and if you were incorrect, write down what it did.

```
profits = dict(zip(['Aquas','Hydros'],[350,300]))
```

{'Aquas': 350, 'Hydros': 300}