[Tutorial] Python List & Dictionary Comprehension: Behind the Scenes

discussion posted 6 days ago by [**Kiara-Elizabeth**](https://courses.edx.org/courses/course-v1:MITx+6.00.1x+1T2022/discussion/forum/users/6402998) (Community TA)

**Python List Comprehension 🎉**

Hi! Welcome to this short tutorial on Python List Comprehension. You will learn about the basic syntax of this wonderful feature that Python offers and then we will dive into some examples. Sounds exciting, right? Let’s begin! 👍

First of all… what is list comprehension? It is a **more compact** way to create lists in Python. By using list comprehensions, you can merge several lines into a single line that indicates which elements should be added to the list. Let’s see how this works.

**Without list comprehension:** 👋

Let’s say that you want to iterate over a sequence and only include items that meet specific criteria. You would usually write something like this using loops and conditionals:

list1 = []

for i in range(15):

if i % 2 == 0:

list1.append(i)

**With list comprehensions:** 👍

But... you could make an **equivalent code in just one line** by using list comprehension:

list1 = [i for i in range(15) if i % 2 == 0]

This says: iterate over every integer from 0 to 14 (for i in range(15)) and if the current integer (i) meet this criteria (i % 2 == 0), include the value of i in the final list.

>>> list1 = [i for i in range(15) if i % 2 == 0]

>>> list1

[0, 2, 4, 6, 8, 10, 12, 14]

Let's start by analyzing the basic syntax:

[<value\_to\_include> for <elem> in <sequence> if <condition>]

* First we have the value that will be included in the list. This could be a variable or an expression.
* Then we have a loop because we need to analyze a sequence and determine which values will be included in the list.
* If the item in the sequence meets the condition (the condition evaluates to True), the item is included in the final list.

Note: the if <condition> part is optional. You could simply write: [i for i in range(15)]

>>> list1 = [i for i in range(15)]

>>> list1

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

and the list would be populated with these values without any restrictions or conditions.

This is really awesome, right? Our code is now much more compact! But what if want to use an expression instead of the value i directly.

**Using Expressions:** 👏

Instead of just including the value of i, you could include the result of an expression that uses i or another variable.

For example, we could use i \* 3 or i + 5:

list1 = [i \* 3 for i in range(15) if i % 2 == 0]

code:

>>> list1 = [i \* 3 for i in range(15) if i % 2 == 0]

>>> list1

[0, 6, 12, 18, 24, 30, 36, 42]

In this case, we iterate over the integers from 0 to 14 and select the ones that are even, but we add the values resulting from the expression i \* 3 at each step.

You could also have more complex expressions that work with existing values such as dictionaries:

[value for value in namesDict.values() if value % 2 != 0]

code:

>>> namesDict = {"Nora": 56, "Lulu": 15, "Gino": 31}

>>> oddValues = [value for value in namesDict.values() if value % 2 != 0]

>>> oddValues

[15, 31]

This would iterates over all the values in the dictionary and if the current value is odd, it is included in the list.

This is equivalent to using:

oddValues = []

for value in namesDict.values():

if value % 2 != 0:

oddValues.append(value)

**Dictionary Comprehension: 🎉**

Python also supports dictionary comprehension, so you can create new dictionaries from existing ones but assigning different values.

<variable> = { key:new\_value for (key, value) in <dictionary>.items() }

This iterates over the keys and values in a dictionary and adds the corresponding key and a new value that is the result of an expression.

For example:

>>> grades = {"Nora": 90, "Lulu": 15, "Gino": 60}

>>> doubleGrades = {key: value\*2 for (key, value) in grades.items()}

>>> doubleGrades

{'Nora': 180, 'Lulu': 30, 'Gino': 120}

This iterates over the keys of the grades dictionary and their corresponding values and creates a new dictionary that contains the keys of the original dictionary with their corresponding values doubled.

You can also use a condition in dictionary comprehension:

>>> grades = {"Nora": 90, "Lulu": 15, "Gino": 60}

>>> doubleGrades = {key: value\*2 for (key, value) in grades.items() if value % 2 == 0}

>>> doubleGrades

{'Nora': 180, 'Gino': 120}

This only includes the item if the original value was even. This is why "Lulu" is not included, because the original value was 15.

I really hope that you liked this tutorial. 👍

List and Dictionary comprehensions are really powerful tools to help you write more concise and readable code. 🎉

Please do not hesitate to ask if you have any questions. Your classmates and Community TAs will be very glad to help you. 😀

Estefania.