23.4. List Comprehensions

Python provides an alternative way to do `map` and `filter` operations, called a **list comprehension**. Many programmers find them easier to understand and write. List comprehensions are concise ways to create lists from other lists. The general syntax is:

```
[<transformer_expression> for <loop_var> in <sequence> if <filtration_expression>]
```

where the if clause is optional. For example,

Save & Run Original - 1 of 1 Show in CodeLens

```
things = [2, 5, 9]

yourlist = [value * 2 for value in things]

print(yourlist)

```

```
[4, 10, 18]
```

Activity: 1 -- ActiveCode (ac20_4_1)

The transformer expression is `value * 2`. The item variable is `value` and the sequence is `things`. This is an alternative way to perform a mapping operation. As with `map`, each item in the sequence is transformed into an item in the new list. Instead of the iteration happening automatically, however, we have adopted the syntax of the for loop which may make it easier to understand.

Just as in a regular for loop, the part of the statement `for value in things` says to execute some code once for each item in things. Each time that code is executed, `value` is bound to one item from `things`. The code that is executed each time is the transformer expression, `value * 2`, rather than a block of code indented underneath the for statement. The other difference from a regular for loop is that each time the expression is evaluated, the resulting value is appended to a list. That happens automatically, without the programmer explicitly initializing an empty list or appending each item.

The `if` clause of a list comprehension can be used to do a filter operation. To perform a pure filter operation, the expression can be simply the variable that is bound to each item. For example, the following list comprehension will keep only the even numbers from the original list.

Save & Run Original - 1 of 1 Show in CodeLens

```
def keep_evens(nums):
    new_list = [num for num in nums if num % 2 == 0]
    return new_list

print(keep_evens([3, 4, 6, 7, 0, 1]))

```

```
[4, 6, 0]
```

Activity: 2 -- ActiveCode (ac20_4_2)

You can also combine `map` and `filter` operations by chaining them together, or with a single list comprehension.

Save & Run Original - 1 of 1 Show in CodeLens

```
things = [3, 4, 6, 7, 0, 1]
#chaining together filter and map:
# first, filter to keep only the even numbers
# double each of them
print(map(lambda x: x*2, filter(lambda y: y % 2 == 0, things)))

# equivalent version using list comprehension
print([x*2 for x in things if x % 2 == 0])

```

```
[8, 12, 0]
[8, 12, 0]
```

Activity: 3 -- ActiveCode (ac20_4_3)

Check your understanding

AdAccum-4-1: What is printed by the following statements?

```
alist = [4,2,8,6,5]
blist = [num*2 for num in alist if num%2==1]
print(blist)
```

- A. [4,2,8,6,5]
- B. [8,4,16,12,10]
- C. 10
- D. [10]

Check me | Compare me

✔ Yes, 5 is the only odd number in alist. It is doubled before being placed in blist.

Activity: 4 -- Multiple Choice (question21_4_1)

2. The for loop below produces a list of numbers greater than 10. Below the given code, use list comprehension to accomplish the same thing. Assign it the the variable `lst2`. Only one line of code is needed.

Save & Run | 5/15/2021, 12:01:13 AM - 14 of 14 | Show in CodeLens

```
L = [12, 34, 21, 4, 6, 9, 42]
lst = []
for x in L:
    if x > 10:
        lst.append(x)
print(lst)
lst2 = []
lst2 = [y for y in L if y > 10]
print(lst2)
```

```
[12, 34, 21, 42]
[12, 34, 21, 42]
```

Activity: 5 -- ActiveCode (ac21_4_4)

Result	Actual Value	Expected Value	Notes	
Pass	[12, ..., 42]	[12, ..., 42]	Testing that lst2 is assigned to correct values	Expand Differences
Pass	'map('	'\nL = ...t2)\n\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'filter('	'\nL = ...t2)\n\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'sum('	'\nL = ...t2)\n\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'zip('	'\nL = ...t2)\n\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences

You passed: 100.0% of the tests

3. Write code to assign to the variable `compri` all the values of the key `name` in any of the sub-dictionaries in the dictionary `tester`. Do this using a list comprehension.

Save & Run | 5/14/2021, 11:46:32 PM - 2 of 2 | Show in CodeLens

```
tester = {'info': [{"name": "Lauren", 'class standing': 'Junior', 'major': "Informa

inner_list = tester['info']

compri = [d['name'] for d in inner_list]
print(compri)

```

```
['Lauren', 'Ayo', 'Kathryn', 'Nick', 'Gladys', 'Adam']
```

Activity: 6 -- ActiveCode (ac21_4_5)

Result	Actual Value	Expected Value	Notes	
Pass	['Ada...ick']	['Ada...ick']	Testing that compri has the correct values.	Expand Differences
Pass	'map('	'\ntest...pri)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'filter('	'\ntest...pri)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'sum('	'\ntest...pri)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'zip('	'\ntest...pri)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences

You passed: 100.0% of the tests

You have attempted 7 of 6 activities on this page

23.3. Filter">
ter">
23.5. Zip">
✔ Completed. Well Done!
23.5. Zip">Next Section - 23.5. Zip

| Back to top