Python Comprehensions

CSE 160 Summer 2026

Questions?

Announcements

- **Homework Assignment 5 due Monday, August 17 at 11:59PM**
- **Programming Practice 7 due Wednesday, August 19 at 11:59PM**
- **Resubmission Cycle 4 available!**
  - Homeworks 1-4 eligible
  - Check Ed post for more information.
- **Final Exam information and Practice Finals** now available on course website!
  - Final Exam on Friday, August 21 at 9:40am in HRC 155

Three Ways to Define a List

Explicitly write out the whole thing:

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Write a loop to create it:

```
squares = []
for i in range(11):
    squares.append(i * i)
```

Write a list comprehension:

```
squares = [i * i for i in range(11)]
```

- A list comprehension is a concise description of a list
- A list comprehension is shorthand for a loop

List Comprehensions

Jupyter Hub

General Form:

```
result = [<expression> for <item> in <sequence>]
```

Examples:

```
squares = [i * i for i in range(11)]

tens = [x * 10 for x in range(1, 11)]

hundreds = [i * 10 for i in tens]

letters = [x for x in "snow"]
```

Cubes of the first 10 natural numbers

Goal:

Produce: `[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]`

With a loop:

With a list comprehension:

```
cubes = [<expression> for <item> in <sequence>]
```

Cubes of the first 10 natural numbers (answer)

Goal:

Produce: `[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]`

With a loop:

```
cubes = []
for x in range(10):
    cubes.append(x ** 3)
```

With a list comprehension:

```
cubes = [<expression> for <item> in <sequence>]
```

```
cubes = [x ** 3 for x in range(10)]
```

Powers of 2: (2^0 through 2^{10})

Goal: `[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]`

With a loop:

With a list comprehension:

```
powers = [<expression> for <item> in <sequence>]
```

```
powers =
```

Sli.do

Powers of 2: (2^0 through 2^{10}) (answer)

Goal: `[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]`

With a loop:

```
powers = []
for i in range(11):
    powers.append(2 ** i)
```

With a list comprehension:

```
powers = [<expression> for <item> in <sequence>]
```

```
powers = [2 ** i for i in range(11)]
```

Expressions can include calling a function

General Form:

```
result = [<expression> for <item> in <sequence>]
```

Examples:

```
sqrts = [math.sqrt(i) for i in range(1, 11)]

abs_list = [abs(x) for x in range(-5, 6)]

word_list = ["  happy  ", " Friday   ", "folks "]

no_spaces = [word.strip() for word in word_list]
```

Convert Centigrade to Fahrenheit

```
ctemps = [17.1, 22.3, 18.4, 19.1]
```

- With a loop:

```
ftemps = []
for c in ctemps:
    f = celsius_to_farenheit(c)
    ftemps.append(f)
```

- With a list comprehension:

```
ftemps = [<expression> for <item> in <sequence>]
```

```
ftemps = [celsius_to_farenheit(c) for c in ctemps]
```

The comprehension is usually shorter, more readable, and more efficient

Lengths of elements of a list

Goal: A list containing the length of each string in the list `colors`.

```
colors = ["red", "blue", "purple", "gold", "orange"]
```

Would produce: `[3, 4, 6, 4, 6]`

With a loop:

With a list comprehension:

```
lengths = [<expression> for <item> in <sequence>]
```

Lengths of elements of a list (answer)

Goal: A list containing the length of each string in the list `colors`.

```
colors = ["red", "blue", "purple", "gold", "orange"]
```

Would produce: `[3, 4, 6, 4, 6]`

With a loop:

```
lengths = []
for color in colors:
    lengths.append(len(color))
```

With a list comprehension:

```
lengths = [<expression> for <item> in <sequence>]
```

```
lengths = [len(color) for color in colors]
```

Extract values greater than 10

Goal: Create a list containing ONLY the values from `input_list` that are greater than 10.

With a loop:

```
big_vals = []
for x in input_list:
    if x > 10:
        big_vals.append(x)
```

You can also do this with a list comprehension!

```
big_vals = [x for x in input_list if x > 10]
```

List Comprehensions with Conditionals

Can add conditionals:

```
result = [<expression> for <item> in <sequence> if <condition>]
```

Example:

```
squares = [i * i for i in range(11)]

sq_over_ten = [x for x in squares if x > 10]
```

Or:

```
sq_over_ten = [x * x for x in range(11) if x * x > 10]
```

Even elements of a list

Goal: Given an input list `nums`, produce a list of the even numbers in `nums`

`nums = [3, 1, 4, 1, 5, 9, 2, 6, 5]` should produce: `[4, 2, 6]`

With a loop:

With a list comprehension:

```
evens = [<expression> for <item> in <sequence> if <condition>]
```

Even elements of a list (answer)

Goal: Given an input list `nums`, produce a list of the even numbers in `nums`

`nums = [3, 1, 4, 1, 5, 9, 2, 6, 5]` should produce: `[4, 2, 6]`

With a loop:

```
evens = []
for num in nums:
    if num % 2 == 0:
        evens.append(num)
```

With a list comprehension:

```
evens = [<expression> for <item> in <sequence> if <condition>]
```

```
evens = [num for num in nums if num % 2 == 0]
```

Today's Roadmap

1. ~~List comprehensions~~
2. Dictionary comprehensions

Jupyter Hub

Dictionary Comprehensions

Jupyter Hub

General Form:

```
result = {<key>: <value> for <item> in <sequence>}
```

Examples:

```
squares = {i: i * i for i in range(10)}

cubes = {x: x ** 3 for x in range(1, 11)}

powers = {i: 2 ** i for i in range(11)}
```

Dictionary of lengths of items in a list

Goal: A dictionary containing the length of each string in the list `colors`.

```
colors = ["red", "blue", "purple", "gold"]
```

Would produce: `{"red": 3, "blue": 4, "purple": 6, "gold": 4}`

With a loop:

With a dictionary comprehension:

```
lengths_dict = {<key>: <value> for <item> in <sequence>}
```

Dictionary of lengths of items in a list (answer)

Goal: A dictionary containing the length of each string in the list `colors`.

```
colors = ["red", "blue", "purple", "gold"]
```

Would produce: `{"red": 3, "blue": 4, "purple": 6, "gold": 4}`

With a loop:

```
lengths_dict = {}
for color in colors:
    lengths_dict[color] = len(color)
```

With a dictionary comprehension:

```
lengths_dict = {<key>: <value> for <item> in <sequence>}
```

```
lengths_dict = {color: len(color) for color in colors}
```

Squares of even elements of a list

Goal: Given an input list `nums`, produce a dict of the squares of even numbers in `nums`

```
nums = [3, 1, 4, 1, 5, 9, 2, 6, 5]
```

should produce: `{4: 16, 2: 4, 6: 36}`

With a loop:

With a dictionary comprehension:

```
even_squares = {<key>: <value> for <item> in <sequence> if <condition>}
```

Squares of even elements of a list (answer)

Goal: Given an input list `nums`, produce a dict of the squares of even numbers in `nums`

```
nums = [3, 1, 4, 1, 5, 9, 2, 6, 5]
```

should produce: `{4: 16, 2: 4, 6: 36}`

With a loop:

```
even_squares = {}
for num in nums:
    if num % 2 == 0:
        even_squares[num] = num * num
```

With a dictionary comprehension:

```
even_squares = {<key>: <value> for <item> in <sequence> if <condition>}
```

```
even_squares = {num: num * num for num in nums if num % 2 == 0}
```