

UMpy: Dictionary comprehensions

Goals

1. Awareness/Literacy: Write dictionary comprehensions in place of `for` loops.
2. Review: Work with lists and dictionaries.
3. Review: Write conditional statements that filter elements.
4. Awareness: Use `string.translate()` to remove punctuation from a string.

Glossary

source: <https://docs.python.org/3/glossary.html>

Dictionary comprehension

A compact way to process all or part of the elements in an iterable and return a dictionary with the results.

```
some_dict = {key: val for key, value in iterable}
```

```
some_dict = {key: val for key, value in iterable if condition}
```

```
some_dict = {  
    (key if condition else default_key): (something_if_true if condition  
    else some_other_thing)  
    for key, val in dict_.items()  
}
```

Challenges

The following challenges illustrate how to write dictionary comprehensions in cases where items to be included in the new dictionary are first transformed and/or filtered based on one or more conditional expressions.

Challenge 01

Recall that a dictionary comprehension can specify one or more conditional statements.

```
some_dict = {key: val for key, value in iterable if condition}
```

Write a dictionary comprehension based on a list of West African tuples that returns a dictionary that uses each country name as the key for the associated country code value. Assign the new dictionary to a variable named `country_codes`.

```
data = [  
    ('Benin', 'BEN'),  
    ('Burkina Faso', 'BFA'),  
    ...  
]  
  
country_codes = {  
    'Benin': 'BEN',  
    'Burkina Faso': 'BFA',  
    ...  
}
```

Challenge 02

Write a dictionary comprehension based on a dictionary of US annual inflation rates for the years 2000-2020 that returns a new dictionary of annual inflation rates computed as percentage values. Convert each float to a percentage value, rounding to two (2) decimal places. Assign the new dictionary to a variable named `inflation_rates`.

```
data = {  
    '2020': 0.0125,  
    '2019': 0.0181,  
    '2018': 0.024399999999999998,  
    ...  
    '2002': 0.0159,  
    '2001': 0.028300000000000002,  
    '2000': 0.0338  
}  
  
inflation_rates = {  
    '2020': 1.25,  
    '2019': 1.81,  
    '2018': 2.44,  
    ...  
    '2002': 1.59,  
    '2001': 2.83,  
    '2000': 3.38  
}
```

Challenge 03

Write a dictionary comprehension based on `inflation_rates` that returns a new dictionary of US annual inflation rates for the years 2010-2020 *only*. Assign the new dictionary to a variable named `inflation_rates`.



Employ a conditional expression as a filter.

Challenge 04

Write a dictionary comprehension based on a list of directions in which the key is a specified as a numbered step (eg., 1, 2, 3 ...) and the value is a dictionary comprising two key-value pairs: 'segment' and 'distance_mi'. Assign the new dictionary to a variable named `directions`.

```
data = [
    ['Head north on S State St toward E Washington St.', 0.1],
    ...
    ['Turn right at 1st cross street onto N Mechanic St. Destination on left', 0.0359],
]

directions = {
    '1': {
        'segment': 'Head north on S State St toward E Washington St.',
        'distance_mi': 0.1
    },
    ...
    '11': {
        'segment': 'Turn right at 1st cross street onto N Mechanic St. Destination on left',
        'distance_mi': 0.0359
    }
}
```

Challenge 05

Write a dictionary comprehension based on a dictionary of planet data that includes only those planets categorized as "inner". Assign the new dictionary to a variable named `inner_planets`.

```
data = {
    "mercury" : {'category': 'inner', 'satellites': 0},
    ...
    "neptune" : {'category': 'outer', 'satellites': 14}
}

inner_planets = {
    'mercury': {'category': 'inner', 'satellites': 0},
    ...
    'mars': {'category': 'inner', 'satellites': 2}}
}
```



You can enhance readability by writing dictionary comprehensions that exceed 80-100 characters and/or feature complex conditions or other expressions across multiple lines vertically.

```
some_dict = {
    key: val
    for key, value in iterable
    if condition
    ...
}
```

Challenge 06


Write a dictionary comprehension based on a dictionary of planet data that includes only those planets categorized as "outer" that possess between 10 and 30 satellites (inclusive). Assign the new dictionary to a variable named `outer_planets`.

Challenge 07

Write a dictionary comprehension based on a dictionary of planet data that recategorizes each planet based on the following criteria: inner = terrestrial; outer = jovian and replaces the each dictionary value with a string (e.g., `< planet_name >: < 'terrestrial' | 'jovian' >`). Assign the new dictionary to a variable named `planet_types`.

```
data = {
    "mercury" : {'category': 'inner', 'satellites': 0},
    ...
    "neptune" : {'category': 'outer', 'satellites': 14}
}

planet_types = {
    'mercury': 'terrestrial',
    ...
    'neptune': 'jovian'
}
```

 Employ `if-else` logic to effect the value conversion.

Challenge 08 (Bonus)

Write a dictionary comprehension that provides a *case insensitive* frequency count of the words Amanda Gorman employs in an excerpt of her poem *The Hill We Climb* (2020). Remove all punctuation from the multi-line string (use `string.translate()`) *before* performing writing the dictionary comprehension that performs the frequency count. Assign the new dictionary to a variable named `word_counts`.

```
data = """
When day comes we ask ourselves, where can we find light in this never-
ending shade?
...
again know defeat but because we will never again sow division.
```

```

"""

# Remove punctuation (import string module)
data_cleaned = data.translate(str.maketrans('', '', string.punctuation)) #
remove punctuation

word_count = {
    'sea': 1,
    ...
    'we': 25,
    ...
    'victorious': 1,
    ...
}

```

Recommended reading

Lisa Tagliaferri, "[Understanding dictionary comprehensions in Python 3](#)" (DigitalOcean, Jan 2017).

Very short introduction to the topic. Once you have read Lisa proceed *directly* to Trey Hunner's blog post.

Trey Hunner, "[Python dictionary comprehensions: Explained Visually](#)" (Trey Hunner, 15 Dec 2015).

What I and others like about Trey's blog post is the way he helps the reader familiar with writing `for` loops transition to writing comprehensions. A must read.

Trey Hunner, "[Overusing dictionary comprehensions and generator expressions in Python](#)" (Trey Hunner, 26 Mar 2019).

Warns against comprehension overuse and misuse.

Josh Robin, "[Dictionary Comprehension in Python 3 for Beginners](#)" (Medium, May 2019).

Josh fills in the details covering how to write comprehensions that iterate over dictionary keys and values in tandem. We will cover this topic in our next meeting.

James Timmons, "[When to Use a dictionary comprehension in Python](#)" (Real Python, n.d.)

A more advanced comprehensions tutorial that also discusses the built-in functions `map()` and `filter()`, the new Walrus operator (`:=`) and generators.