07. Python Collections (2)

3ikakke

14, March, 2022

Outline

- Objectives
- Review of what we have learnt so far and map to data science
- Review of collections
- Lambda functions
- Map, Filter, Reduce
- Comprehensions
- Comprehension -> Mapping
- Comprehension -> Filtering
- Comprehension -> Reduce
- Review of Objectives
- Discussion
- Gist of the day
- Q&A

Objectives

- Understand collections even better
- Understand comprehensions
- Understand the concepts of map, filter, and reduce
- Understand lambda functions

Review of what we have learnt so far and map to data science

- Data types

  - Numeric
  - Categorical
- Logic - conditions
- Loops - iterations
- Data
  - Obserevations -> Rows
  - Variables -> Columns
  - See Excel

Review of collections

- Lists: 0 indexed defined by [], can contain duplicate values, not necessarily sorted, can be modified
- Tuples: 0 indexed, defined by (), can contain duplicates, not necessarily sorted, can not be modified
- Dictionaries: key: value pairs, defined by {}, can not contain duplicate keys but may contain duplicate values, not sorted, can be modified
- Sets: not indexed, but iterable, unique, sorted, can be modified

Lambda functions

- Understanding functions - borrwoing from the mathematical concept of functions
- Anonymous functions
- Nameless functions
- Written in single line
- Does not use the `return` key word

```
lambda x: x
foo = lambda x, y: x + y
print(foo(1, 2))
```

Map, Filter, Reduce

- Map
  - Apply some function to every member of a collection or iterable
- Filter
  - Filtering members of a collection or iterable to only return those that match with a Boolean True
- Reduce
  - Iteratively applying a function to members of a collection and carrying this forward till a single return value is realized

Comprehensions

- Let's start with English. Assuming we have a box of chocolates we may say
  - Give me chocolate for every chocolate you pull out of the box
  - chocolate for chocolate in box
  - [chocolate for chocolate in box]
- Comprehension done!

```
box = ['red m&m', 'blue m&m', 'yellow m&m', 'green m&m', 'green m&m', 'red m&m', 'green m&m', 'yellow m&
print([choc for choc in box])
```

Comprehension -> Mapping

- We may take this further by saying
  - Unwrap chocolate for each chocolate you pull out of the box
  - Unwrap chocolate for chocolate in box
  - [unwrap(chocolate) for chocolate in box]
  - ["unwrap"+ chocolate for chocolate in box]
  - [f"unwrap {chocolate}" for chocolate in box]

```
unwrapped = [f"unwrap {chocolate}" for chocolate in box]
#unwrapped = {f"unwrap {chocolate}" for chocolate in box}

print(unwrapped)
```

Comprehension -> Filtering

- Lets try filtering
  - Give me the red chocolates in the box
  - I want red chocolates for each chocolate you pull out of the box
  - chocolate for chocolate in box if red
  - [chocolate for chocolate in box if chocolate == 'red m&m']

```
reds = [chocolate for chocolate in box if chocolate == 'red m&m']
print(reds)

unwraped_reds = [f"unwrap {chocolate}" for chocolate in box if chocolate == 'red m&m']
```

Comprehension -> Reduce

- Lets try getting a single value from a comprehension
  - sum all the scores in the class
  - sum([score for score in scores])
  - min([score for score in scores])

  - max([score for score in scores])

- More useful with iterables that cannot be indexed

```
scores = (34, 28, 77, 90, 65, 44, 58, 100)
print(sum([score for score in scores]))
print(min([score for score in scores]))
print(max([score for score in scores]))
```

Review of Objectives

- Understand collections even better
- Understand comprehensions
- Understand the concepts of map, filter, and reduce
- Understand lambda functions

Discussion

- Your project/portfolio
- Git account
- Live project

Gist of the day

- Get the pdf version
- Get the gist
- I will put up the Jupyter Notebook and Homework

Q&A

Thanks for listening and participating!