Lecture 10: Tabular Data

Wednesday, September 28, 2022 10:20 PM

CSC110 Lecture 10: Tabular Data

Print this handout

Exercise 1: Working with tabular data

1. Here is the sample list data we saw in lecture. (You can copy and paste this directly into the Python console.)

```
import datetime

marriage_data = [
0   [1657, 'ET', 80, datetime.date(2011, 1, 1)],
1   [1658, 'NY', 136, datetime.date(2011, 1, 1)],
2   [1659, 'SC', 159, datetime.date(2011, 1, 1)],
3   [1660, 'TO', 367, datetime.date(2011, 1, 1)],
4   [1661, 'ET', 109, datetime.date(2011, 2, 1)],
5   [1662, 'NY', 150, datetime.date(2011, 2, 1)],
6   [1663, 'SC', 154, datetime.date(2011, 2, 1)],
7   [1664, 'TO', 383, datetime.date(2011, 2, 1)]
]
```

For each of the following expressions, write down what it would evaluate to.

```
>>> len(marriage_data)

        8

>>> marriage_data[5]

  [ 1662, 'NY',...

>>> marriage_data[0][0]



>>> marriage_data[6][2]

      154

>>> len(marriage_data[3])



>>> max([row[2] for row in marriage_data])

   383
```

2. Using the same data from the previous question, write Python expressions for each of the following descriptions:

```
>>> # Number of marriages in 'ET' in February 2011
>>> # (use list indexing to access the correct row)

>>> marriage_data[4][2]

109

                                   ✓
>>> # The date corresponding to ID 1661
>>> # (use list indexing to access the correct row)


>>>


datetime.date(2011, 2, 1)


>>> # The minimum number of marriage licenses issued in a month


>>>
                                                          row[2]

80
                                                          ~~row~~ > 300

>>> # The rows where more than 300 licenses were issued
>>> # (use a filtering list comprehension!)

>>> [ row      for row in marriage_data if row[2] > 300]

[[1660, 'TO', 367, datetime.date(2011, 1, 1)], [1664, 'TO', 383,
        datetime.date(2011, 2, 1)]]
```

Exercise 2: Constraining the "marriage license" data representation

Suppose we have a variable `marriage_data` that is a nested list representing the marriage license data we've been working with in lecture. We have listed below several statements that are *constraints* on the rows in this list. Your task is to translate each of these constraints into an equivalent Python expression (similar to how you translated English preconditions into Python on this week's prep).

Use the variable `marriage_data` to refer to the nested list containing all rows; each of the constraints except the last can be expressed as an expression of the form `all({____ for row in marriage_data})`.

1. Every row has length 4.

```
all({ len(row) == 4        for row in marriage_data})
```

2. Every row's first element (representing the "row id") is an integer greater than 0. (Hint: use `isinstance(___, int)` to return whether a value is an `int`.)

↳ $x \in \mathbb{Z} \wedge x > 0$

```
all({                        for row in marriage_data})
```

3. Every row's second element (representing the civic centre) is one of `'TO'`, `'ET'`, `'NY'`, or `'SC'`.

```
all({                        for row in marriage_data})
```

4. Every row's third element (representing the number of marriage licenses) is an integer greater than or equal to 0.

```
all({                        for row in marriage_data})
```

5. Every row's fourth element is a `datetime.date` value. (You can use `isinstance` here as well.)

```
                          datetime.date
all({ isinstance(row[3], ^ ) for row in marriage_data})
```

6. *At least one row* has `'TO'` as its civic centre.

Exercise 3: Implementing functions on tabular data

Your task here is to implement the following functions that operate on our marriage license data set.

```
def civic_centres(data: list[list]) -> set[str]:
    """Return a set of all the civic centres found in data.

    Preconditions:
        - data satisfies all of the properties described in Exercise 2
    """
  return {row[1] for row in data}
```

```
def civic_centre_meets_threshold(data: list[list], civic_centre: str, num: int) ->
        bool:
    """Return whether civic_centre issued at least num marriage licences every month.

    You only need to worry about the rows that appear in data; don't worry about
    "missing" months.

    Preconditions:
        - num > 0
        - data satisfies all of the properties described in Exercise 2
        - civic_centre in {'TO', 'NY', 'ET', 'SC'}

    HINT: you'll need to use a filtering list comprehension.
    """
    return all({row[2] >= num for row in data if row[1] ==
                                                  civic_centre})
```

```python
def summarize_licences_by_centre(data: list[list]) -> dict[str, int]:
    """Return the total number of licences issued by each civic centre in <data>.

    Returns a dictionary where keys are civic centre names and values are the
    total number of licences issued by that civic centre.

    Preconditions:
        - data satisfies all of the properties described in Exercise 2

    HINT: you will find it useful to write a function that calculates the total
    number of licences issued for a given civic centre as a parameter,
    e.g. total_licenses_for_centre(data, civic_centre).
    """
```

(similar to avg # per civic centre)

Additional exercises

```
def average_licenses_by_centre(marriage_data: list[list]) -> dict[str, float]:
    """Return a mapping of the average number of marriage licenses issued at
each civic centre.

    In the returned mapping:
      - Each key is the name of a civic centre
      - Each corresponding value is the average number of marriage licenses
issued at
        that centre.

    Preconditions:
    - marriage_data satisfies all of the conditions from Exercise 2
    """

    # 1. Compute the civic centres in marriage_data.
    civic_centres = …


    # 2. For each civic centre, find its average number of licenses.

    {cc : get_avg_licenses(marriage_data, cc)  for cc in civic_centres }

return


def get_avg_licenses(marriage_data, cc):
""" Return the avg # licenses for cc."""

    # find the right numbers
    right = [  row[2]             for row in marriage_data if row[1] ==
cc     ]

    # average
    sum(right) / len(right)

 return
```