


CSC110 Lecture 10: Tabular Data

 Print this handout

Exercise 1: Working with tabular data

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

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

```
>>> len(marriage_data)

>>> marriage_data[5]

>>> marriage_data[0][0]

>>> marriage_data[6][2]

>>> len(marriage_data[3])

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

- 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)

>>>

109

>>> # The date corresponding to ID 1662
>>> # (use list indexing to access the correct row)

>>>

datetime.date(2011, 2, 1)

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

>>>

80

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

>>>

[[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})`.

- Every row has length 4.
- 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`.)
- Every row's second element (representing the civic centre) is one of 'TO', 'ET', 'NY', or 'SC'.
- Every row's third element (representing the number of marriage licenses) is an integer greater than or equal to 0.
- Every row's fourth element is a `datetime.date` value. (You can use `isinstance` here as well.)
- 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
    """
```

```
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 comprehension.
    """
```

```
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_licences_for_centre(data, civic_centre).
    """
```

Additional exercises

- Writing constraints. Express the following constraint on `marriage_data` as a Python expression: Every row has a unique ID (the first element).
- Investigating the `datetime.date` data type.

Suppose we want to find the number of marriage licenses that were issued in a particular month. To start, we would like to filter the rows of data by their month, much like we did filtering by civic centre in lecture. But the fourth element of each row is a `date` value; from this value, how do we check its month? In Python, a `date` value is composed of three pieces of data, a year, month, and day, and each of these are called *attributes* of the value. We can access these attributes individually using dot notation once again:

```
>>> import datetime
>>> canada_day = datetime.date(1867, 7, 1)
>>> canada_day.year # The year attribute
1867
>>> canada_day.month # The month attribute
7
>>> canada_day.day # The day attribute
1
```

With this information in mind, implement the following functions:

```
def issued_licences_by_year(data: list[list], year: int) -> int:
    """Return the total number of marriage licences issued in <year>.

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

```
def only_first_days(data: list[list]) -> bool:
    """Return whether every row's fourth element is a datetime.date whose
    <day> attribute is 1.

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

```
def issued_licences_in_range(data: list[list],
                             start: datetime.date, end: datetime.date) -> int:
    """Return the number of marriage licenses issued between start and end, inclusively.

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

    HINT: You can use <, <=, >, and >= to compare date values chronologically.
    """
```