CSC110 Lecture 9: Nested Data

Hisbaan Noorani

September 30, 2020

Contents

1 Ex 1: Working with nested lists

1. Here is the sample list data we saw in lecture.

```
>>> 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)
8

>>> marriage_data[5]
[1662, 'NY', 150, datetime.date(2011, 2, 1)]

>>> marriage_data[0][0]
1657

>>> marriage_data[6][2]
154

>>> len(marriage_data[3])
4
```

```

>>> max([row[2] for row in marriage_data])
383
```

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

```
    >>> # Number of marriages in 'ET' in February 2011 (use list indexing to access the right row)
    >>> marriage_data[4][2]
    109

    >>> # The date corresponding to ID 1662 (use list indexing to access the right row)
    >>> marriage_data[5][3]
    datetime.date(2011, 2, 1)

    >>> # The minimum number of marriage licenses given in a month
    >>> min([row[2] for ro in marriage_data])
    80

    >>> # The rows for February 2011---use a list comprehension with a filter
    >>> [row for row in marriage_data if row[3].year == 2011 and row[3].month == 2]
    [[1661, 'ET', 109, datetime.date(2011, 2, 1)], [1662, 'NY', 150, datetime.date(2011, 2, 1)], [16
```

2 Ex 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. Ever 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).

```
all({row[0] > 0 and isinstance(row[0], int) for row in marriage_data})
```

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

```
all({row[1] in {'TO', 'ET', 'NY', 'SC'} for row in marriage_data})
```

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

```
all({row[2] > 0 and isinstance(row[2], int) for row in marriage_data})
```

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

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

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

```
any({row[1] == 'TO' for row in marriage_data})
```

3 Ex 3: Implementing functions on nested lists

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

```
def civic_centre_dates(data: List[list]) -> Set[datetime.date]:
    """Return a set of all the dates found in data.

    Preconditions:
    - data satisfies all of the properties described in Exercise 2
    """
    return {row[3] 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 list comprehension with a filter.
    """
    filtered_data = [row for row in data if row[1] == civic_centre]
    return all({row[2] >= num for row in filtered_data})
```

```
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).
    """
    return {'TO': calculate_total_marriages(data, 'TO'),
```

```python
            'NY': calculate_total_marriages(data, 'NY'),
            'ET': calculate_total_marriages(data, 'ET'),
            'SC': calculate_total_marriages(data, 'SC')}

def calculate_total_marriages(data: list, civic_centre: str) -> int:
    """Return the total marriages licensed at a given civic centre"""
    return sum([row[2] for row in data if row[1] == civic_centre])
```

4 Ex 4: Investigating **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:

```python
>>> 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:

```python
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
    """
    return sum([row[2] for row in data if row[3].year == year])
```

```python
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
    """
    return all({row[4].day == 1 for row in data})
```

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

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

    HINT: You can use <, <=, >, and >= to compare date values chronologically.
    """
    return sum([row[2] for row in data if start <= row[3] <= end])
```

5 Additional exercises

1. Express the following constraint on `marriage_data`: Every row has a unique ID (the first element).