SI 100B Tutorial: Intro. to HW2 – Flight Analytics

Hongtu Xu
from the SI 100B staff
https://si100b.org/staff

Where to Get the PA?

- The course homepage: https://si100b.org/. Click link in the syllabus section;
- Direct access: http://gitlab.q71998.cn/homework-fall2020/homework2;

Policy on Academic Integrity

“We enforce academic integrity strictly. If you participate in any form of cheating, you will fail this course immediately. Do not try to hack the GitLab grading system in any way. You can view the full version of the code of conduct on the course homepage: https://si100b.org/resource-policy/#policies.”

When Should I Start to Do this HW?

- Now!
- Starting early does not guarantee a high score but starting few days before the due usually means you will fail to obtain most of the score.

What tools do you need?

Custom Data Models

- Basic idea: change the behavior of your class when doing some operations by defining magic method to the class;
- Example:
  - `__len__()`: called by the `len()` function;
  - `__lt__()`: called when comparing two objects using `<`;
  - `__getitem__()`: called when getting items using `[]`;
  - `__setitem__()`: called when setting items using `[]`.
- Read the doc: https://docs.python.org/3.7/reference/datamodel.html.

Iterator

- See tutorial3: https://si100b.org/content/discussion/week3.pdf;
- Defining 2 methods: `__iter__()` and `__next__()`;
- Make your custom class work like the built-in `list`, `dict`, `tuple` etc:
  - Use the `for in` syntax to access the content;
  - Use the list generator.

Handling Exceptions

- See tutorial3: https://si100b.org/content/discussion/week3.pdf;
- Improve robustness and fault tolerance
- User-friendly error message
- In this homework, you may need to raise some exceptions.

Handling the CSV File

- Read and write files:
  - Take care of the mode: `'w'`, `'w+'` or `'r'`? Read the doc!
  - The `with open(…) as f` syntax: save the code to open and close the file.
- Handling the string:
  - Read in:
    - `readline()` or `readlines()`?: Mind the `\n` at the end of line;
    - How to use `split()`, `strip()` and other methods?;
  - Write out:
    - Smart people uses `.join()`.

What are you going to do in this HW?

Task1: Row and Table

The `Row` Object

- **We have implemented it for you;**
- Some number of named columns (keys);
  - An ID: uniquely identify a row;
  - Other data columns.
- Support retrieving data and modify data through `[]` (e.g., `row['AIRLINE']`);
- Support iterating through all the keys;
- Rows are compared by their ID.

Row Example

```python
from analysis import Row

row = Row(["ID", "a", "b", "c"], [4, 1, 2, 3])  # Create a row.
print(f"The row is {len(row)} items long.")  # get the row's length
for col in row:  # iterate through the row
    print(f"Column: {col}, Data: {row[col]}")
```

The row is 4 items long.

Column: ID, Data: 4

Column: a, Data: 1

Column: b, Data: 2

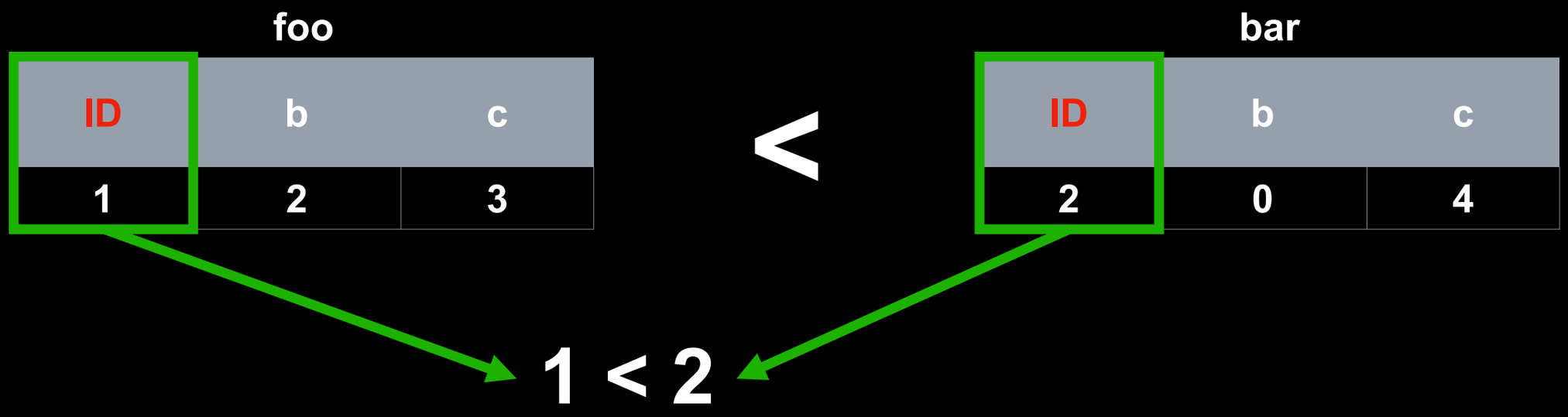
Column: c, Data: 3

ID	a	b	c
4	1	2	3

Row Compare Example

```
1   from analysis import Row
2
3   foo = Row(["ID", "b", "c"], [1, 2, 3])
4   bar = Row(["ID", "b", "c"], [2, 0, 4])
5   print("If `foo` is less than `bar`?", foo < bar)
```

If `foo` is less than `bar`? True



The `Table` Object

- Contains zero or more rows;
- Every row has the same set of columns (attribute);
  - Columns could not be changed;
- Read in from a file or construct from multiple `Row` objects;
- Indexing by the ID;
- Design how to store them by yourself.
- Rows are sorted by their ID.

Table Example

```
for row in table:
    for col in row:
        print(f"Row ID: {row.get_id()}, Column: {col}, Data: {row[col]}")
```

example.csv

ID	AIRLINE	FLIGHT_NUMBER
0	GS	98
1	AA	2336
2	SS	840

Table

Row ID: 0, Column: AIRLINE, Data: GS
Row ID: 0, Column: FLIGHT_NUMBER, Data: 98
Row ID: 0, Column: ID, Data: 0
Row ID: 1, Column: AIRLINE, Data: AA
Row ID: 1, Column: FLIGHT_NUMBER, Data: 2336
Row ID: 1, Column: ID, Data: 1
Row ID: 2, Column: AIRLINE, Data: SS
Row ID: 2, Column: FLIGHT_NUMBER, Data: 840
Row ID: 2, Column: ID, Data: 2

Task2: Query on Single Table

The query cont’d

- Queries are `dict` in Python;
- 'condition' contains a list of dictionaries:
  - Each dictionary contains 'key', 'value', 'operator';
    - Possible operators are ==, >, <, >=, <= and !=
- 'filename': the data source for you query;
- You need to filter the row by comparing row's key and value using operator where key is on the left of operator;
- Every two conditions are connected with and (i.e., in conjunctive normal form);

query example

```
query = {
    "condition": [
        {
            'key': 'YEAR',
            'value': '2015',
            'operator': '<'
        },
        {
            'key': 'AIRLINE',
            'value': 'AA',
            'operator': '!='
        }
    ],
    "filename": "query_example.csv",
}
```

From query_example.csv

Filter all rows that satisfy row['YEAR'] < 2015 and row['AIRLINE'] != 'AA'.

Remember to deal with the type of the value, convert it if it's a number.

query example

query_example.csv

ID	YEAR	AIRLINE	FLIGHT_NUMBER
0	2011	GS	98
1	2012	AA	2336
2	2013	SS	840
3	2014	CA	258
4	2015	QS	135
5	2015	NK	451
6	2015	NK	972
7	2015	AA	1323

From query_example.csv

Filter all rows that satisfy row['YEAR'] < 2015 and row['AIRLINE'] != 'AA'.

result

ID	YEAR	AIRLINE	FLIGHT_NUMBER
0	2011	GS	98
2	2013	SS	840
3	2014	CA	258

Task3: Data Exportation

Data Exportation

- Persistent storage is very important for DBMS;
- Export the data from a table to a file on disk:
  - 'columns': a list of columns names specified by the user to identify the columns to be exported.
  - 'filename': the location where the file should be located.
- Your exported table should also be a CSV one.

Data Exportation Example

Table

Row0

	AIRLINE	FLIGHT_NUMBER	ID
Row0	GS	98	0
Row1	AA	2336	1
Row2	SS	840	2

export.csv

AIRLINE	FLIGHT_NUMBER	ID
GS	98	0
AA	2336	1
SS	840	2

Bonus: Data Aggregation

The aggregate operation

- This task is a bonus task. Finishing this task will give you an additional 20% of score in this assignment.
- Some column-wise operation on a single table;
  - e.g., calculate the average number of the column;
- 2 aggregate functions: `AVG`, `MAX`;
- 3 new key in the query: 'column', 'function', '`group_by`'.

The aggregate operation Cont’d

- 3 Steps:
  - Apply filters in 'conditions' to the table like in single table query;
  - Group rows by the column as specified in '`group_by`' : Put the rows with the same value at the column in '`group_by`' together in groups;
  - Apply the aggregate function to other columns on a per-group basis.

Happy Coding!*

*Do remember to start early!