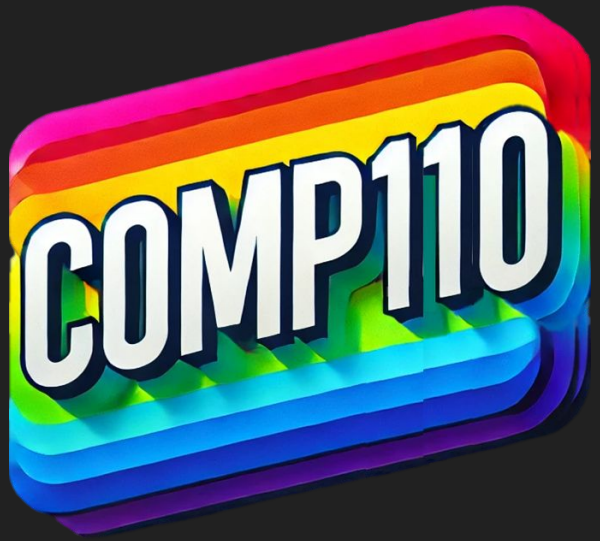

Analyzing Data from a CSV in a Jupyter Notebook (Part 3)

Reminders

Re: Assignments:

- **EX09: Linked List Utility Functions** due at 11:59pm on LDOC (April 27)
  - Late submission deadline: 4pm on April 30th (start of the final exam)
  - You can work with a partner!!

Re: Final Exam:

- **4–7pm on Thursday, April 30**
  - Makeup final exam on Friday, May 1 in SN014
    - If you qualify for the makeup, please complete our internal form ASAP. You will get a confirmation email next week!

Apply to be a TA for COMP110 in Fall 2026!

Review:

`read_csv_rows`

`read_csv_rows` reads an entire CSV of data and returns it as a list of rows, where each row represents a `dict[str, str]`.

- Function Name: `read_csv_rows`
- Parameter:
  - `str` path to CSV file
- Return Type: `list[dict[str, str]]`

Data in our .csv:

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

keys

values

```
{'year': '2018',
 'temp': '48.24',
 'result': 'no shadow'}
```

(append to the list)

The current contents of our `list[dict[str, str]]`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}]
```

Data in our .csv:

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

keys

values

```
{'year': '2019',
 'temp': '46.63',
 'result': 'shadow'}
```

(append to the list)

The current contents of our `list[dict[str, str]]`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}]
{'year': '2019', 'temp': '46.63', 'result': 'shadow'}]
```

Data in our .csv:

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

keys

values

```
{'year': '2020',
 'temp': '52.50',
 'result': 'shadow'}
```

(append to the list)

The current contents of our `list[dict[str, str]]`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'},
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'}]
```

Data in our .csv:

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

keys

values

```
{'year': '2021',
 'temp': '45.86',
 'result': 'no shadow'}
```

(append to the list)

The current contents of our `list[dict[str, str]]`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'}]
{'year': '2021', 'temp': '45.86', 'result': 'no shadow'}]
```

Data in our .csv:

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

keys

values

```
{'year': '2022',
 'temp': '52.36',
 'result': 'shadow'}
```

(append to the list)

The current contents of our `list[dict[str, str]]`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'},
{'year': '2021', 'temp': '45.86', 'result': 'no shadow'},
{'year': '2022', 'temp': '50.36', 'result': 'shadow'}]
```

read_csv_rows

Before calling read_csv_rows (data in our .csv):

```
year,temp,result
2018,48.24,no shadow
2019,46.63,shadow
2020,52.50,shadow
2021,45.86,no shadow
2022,50.36,shadow
```

After calling read_csv_rows (Python objects in our .ipynb):

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'},
{'year': '2021', 'temp': '45.86', 'result': 'no shadow'},
{'year': '2022', 'temp': '50.36', 'result': 'shadow'}]
```

`read_csv_rows` (solution)

```
def read_csv_rows(filename: str) -> list[dict[str, str]]:
    """Read the rows of a CSV into a 'table'."""
    result: list[dict[str, str]] = []

    # Open a handle to the data file
    file_handle = open(filename, "r", encoding="utf8")

    # Prepare to read the data file as a CSV rather than just strings.
    csv_reader = DictReader(file_handle)

    # Read each row of the CSV line-by-line
    for row in csv_reader:
        result.append(row)

    # Close the file when done, to free its resources.
    file_handle.close()

    return result
```

`column_values` function definition & documentation

```
def column_values(table: list[dict[str, str]], column: str) -> list[str]:
    """Produce a list[str] of all values in a single column."""
    result: list[str] = []

    for row in table:
        item: str = row[column]
        result.append(item)

    return result
```

- Purpose: Produce a `list[str]` of all values in a single `column` whose name is the second parameter.
- Function Name: `column_values`
- Parameters:
    1. `list[dict[str, str]]` - a list of rows representing a _table_
    2. `str` - the name of the column (key) whose values are being selected
- Return Type: `list[str]`

`column_values` function

Example input:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'},
{'year': '2021', 'temp': '45.86', 'result': 'no shadow'},
{'year': '2022', 'temp': '50.36', 'result': 'shadow'}]
```

"temp"

Returns:

```
['48.24','46.63','52.50','45.86','50.36']
```

`columnar` function definition

```python
def columnar(row_table: list[dict[str, str]]) -> dict[str, list[str]]:
    """Transform a row-oriented table to a column-oriented table."""
    result: dict[str, list[str]] = {}

    first_row: dict[str, str] = row_table[0]
    for column in first_row:
        result[column] = column_values(row_table, column)

    return result
```

- Purpose: *Transform* a table represented as a list of rows (e.g. `list[dict[str, str]]`) into one represented as a dictionary of columns (e.g. `dict[str, list[str]]`).
- Function Name: `columnar`
- Parameter: `list[dict[str, str]]` - a "table" organized as a list of rows
- Return Type: `dict[str, list[str]]` - a "table" organized as a dictionary of columns

`columnar` function purpose

Before calling `columnar`:

```
[{'year': '2018', 'temp': '48.24', 'result': 'no shadow'}
{'year': '2019', 'temp': '46.63', 'result': 'shadow'},
{'year': '2020', 'temp': '52.50', 'result': 'shadow'},
{'year': '2021', 'temp': '45.86', 'result': 'no shadow'},
{'year': '2022', 'temp': '50.36', 'result': 'shadow'}]
```

After calling `columnar`:

```
{'year': ['2018', '2019', '2020', '2021', '2022'],
'temp': ['48.24', '46.63', '52.50', '45.86', '50.36'],
'result': ['no shadow', 'shadow', 'shadow', 'no shadow', 'shadow']}
```

With a partner, discuss:

1. What are some of the benefits/drawbacks of column-wise data (in general, and in this example in Python)?

```python
def select(
    input_data_table: dict[str, list[str]], selected_columns: list[str]
) -> dict[str, list[str]]:
    """Returns only the selected columns table into table."""
    result: dict[str, list[str]] = {}

    for column_name in input_data_table:
        if column_name in selected_columns:
            result[column_name] = input_data_table[column_name]

    return result

dt: dict[str, list[str]] = {"year": ["2018", "2019"], "temp": ["48.24", "46.63"], "result": ["no shadow", "shadow"]}
cols: list[str] = ["temp", "result"]

dt_selected: dict[str, list[str]] = select(dt, cols)
print(dt_selected)
```

```python
def select(
    input_data_table: dict[str, list[str]], selected_columns: list[str]
) -> dict[str, list[str]]:
    """Returns only the selected columns table into table."""
    result: dict[str, list[str]] = {}

    for column_name in input_data_table:
        if column_name in selected_columns:
            result[column_name] = input_data_table[column_name]

    result["temp"].append("0")

    return result

dt: dict[str, list[str]] = {"year": ["2018", "2019"], "temp": ["48.24", "46.63"], "result": ["no shadow", "shadow"]}
cols: list[str] = ["temp", "result"]

dt_selected: dict[str, list[str]] = select(dt, cols)
print(dt_selected)
```

Consider the **new line** that was added to our function body. Discuss:

1. What would this line of code do?
2. Which variables/parameters would be mutated as a result of this line?

NOTE: DO NOT actually include this line in your `select` function in EX09!