

SI 506 Lecture 17

Topics

1. `csv` module `DictReader` and `DictWriter` classes
2. Dictionaries and the accumulator pattern
3. Challenges

Vocabulary

- **Dictionary.** An associative array or a map, wherein each specified value is associated with or mapped to a defined key that is used to access the value.

Data

Today's data contained in the file `whc-countries-2022.csv` is a listing of UNESCO World Heritage sites.

Source: [UNESCO World Heritage Convention](#)

! Forty-three (43) Heritage Sites (including one endangered site) that span multiple countries have been removed from the data set in order to simplify querying. In addition, columns not relevant to today's discussion such as French translations of certain string fields have also been removed in order to reduce the size of the data set.

1.0 The `csv` module (`csv.DictReader()`, `csv.DictWriter()`)

The `csv` module also provides two classes for *decoding* CSV row data into dictionaries and *encoding* dictionaries into CSV row data.

The `DictReader()` class returns a reader-like object that maps row data to a `dict` whose keys are provided by the optional `fieldnames` parameter.

```
def read_csv_to_dicts(filepath, encoding='utf-8', newline='',
delimeter=','):
    """Accepts a file path, creates a file object, and returns a list of
    dictionaries that represent the row values using the csv.DictReader().

    Parameters:
        filepath (str): path to file
        encoding (str): name of encoding used to decode the file
        newline (str): specifies replacement value for newline '\n'
                        or '\r\n' (Windows) character sequences
        delimeter (str): delimiter that separates the row values

    Returns:
        list: nested dictionaries representing the file contents
        """

    with open(filepath, 'r', newline=newline, encoding=encoding) as
```

```

file_obj:
    data = []
    reader = csv.DictReader(file_obj, delimiter=delimiter)
    for line in reader:
        data.append(line) # OrderedDict()
        # data.append(dict(line)) # convert OrderedDict() to dict

    return data

```

Likewise, the `DictWriter()` class returns a writer-like object that maps dictionaries into row data using the `fieldnames` parameter to determine the order in which each dictionary's key-value pairs are written to each row in the target file. The writer includes a `writeheader()` method for writing the CSV's header row (1st row) based on the passed in `fieldnames` elements and a `writerow()` method for writing each dictionary's key-value data as CSV row data.

```

def write_dicts_to_csv(filepath, data, fieldnames, encoding='utf-8',
newline=''):
    """
    Writes dictionary data to a target CSV file as row data using the
    csv.DictWriter().
    The passed in fieldnames list is used by the DictWriter() to determine
    the order
    in which each dictionary's key-value pairs are written to the row.

    Parameters:
        filepath (str): path to target file (if file does not exist it
will be created)
        data (list): dictionary content to be written to the target file
        fieldnames (seq): sequence specifying order in which key-value
pairs are written to each row
        encoding (str): name of encoding used to encode the file
        newline (str): specifies replacement value for newline '\n'
or '\r\n' (Windows) character sequences

    Returns:
        None
    """

    with open(filepath, 'w', encoding=encoding, newline=newline) as
file_obj:
        writer = csv.DictWriter(file_obj, fieldnames=fieldnames)

        writer.writeheader() # first row
        writer.writerows(data)
        # for row in data:
        #     writer.writerow(row)

```

Say you were interested in retrieving the first Chinese site(s) that were added to the UNESCO [World Heritage List](#) and writing the data to a new CSV file. First, retrieve the heritage sites by calling the function

`read_csv_to_dicts`. The return value is a list of dictionaries.

```
# Get UNESCO World heritage sites
sites = read_csv_to_dicts('whc-countries-2022.csv')
```

Once you have acquired the data you will need to check the following key-value pairs in each dictionary:

- 'states_name_en'
- 'date_inscribed'

This is an "accumulation" problem. However, while the country is known (e.g., China) the earliest inscription date is *unknown*. This date (a year) will need to be discovered when looping over the sites. Decrementing the year value until the earliest year is uncovered will require a "tracking" variable located outside the loop to which the (declining) "date_inscribed" year value is assigned.

You've worked with this pattern before. What's new in the example below is the use of the `datetime` module to assign a start date that is always equal to the current year--no matter what year this code is run it will find the target country site(s) with the earliest inscription date.

```
china_sites = []
year = datetime.today().year # Return current year
for site in sites:
    date_inscribed = int(site['date_inscribed']) # convert
    if site['states_name_en'].lower() == 'china' and date_inscribed <
year:
        china_sites.clear() # reset
        china_sites.append(site)
        year = date_inscribed
    elif site['states_name_en'].lower() == 'china' and date_inscribed ==
year:
        china_sites.append(site)
    else:
        continue # else optional but explicit
```

After retrieving the earliest inscribed Chinese sites, you can write the data out to a file by calling the function `write_dicts_to_csv` and passing to it the following required arguments:

- 'stu-china-earliest.csv' (filepath)
- `china_sites` (the data)
- `china_sites[0].keys()` (fieldnames used to reconstitute the "header" row)

```
# Write to file
write_dicts_to_csv('stu-china-earliest.csv', china_sites,
china_sites[0].keys())
```

2.0 Dictionaries and the accumulator pattern

When exploring data and compiling descriptive statistics (e.g., counts, mean, min, max values) or grouping data, consider using a dictionary to hold the values.

For example, if you needed a count of the number of Chinese heritage sites inscribed each year recorded in the list you could employ a dictionary to hold the counts with each year serving as a key to which the annual count is mapped.

```
{
    '< Year A >': < count >,
    '< Year B >': < count >,
    ...
}
```

For the Chinese heritage sites you could write the following code:

```
china_counts = {}
for site in sites:
    if site['states_name_en'].lower() == 'china':
        year = site['date_inscribed'] # str
        if year not in china_counts.keys():
            china_counts[year] = 1 # seed
        else:
            china_counts[year] += 1 # increment

# Write to file
filepath = 'stu-china-counts.csv'
# WARN: must pass a list of dictionaries
write_dicts_to_csv(filepath, [china_counts], china_counts.keys())
```

BONUS If you check the file `stu-china-counts.csv` you'll notice that the data is not ordered by year. This reflects the lack of strict ordering in the original UNESCO Heritage Sites list. To reorder the Chinese site counts by year before writing the data to a CSV file create a new dictionary employing either the built-in function `dict()` or a dictionary comprehension.

💡 Sorting using an anonymous `lambda` function is actually out of scope for this week but seeing how its done should spark your curiosity about how to use lambda functions. For a useful article on lambdas see Andre Burgaud, ["How to Use Python Lambda Functions"](#) (Real Python, June 2019).

```
# Employ the built-in sorted() function and a lambda function that sorts
# on the key
# Sort by key (x[0]); sort by value (x[1])
# Then convert the list returned by sorted() with the built-in dict()
# function
china_counts = dict(sorted(china_counts.items(), key=lambda x: x[0]))
```

or

```
# Preferred: Employ a dictionary comprehension along with sorted() and a
lambda function
# Sort by key (x[0]); sort by value (x[1])
china_counts = {k: v for k, v in sorted(china_counts.items(), key=lambda
x: x[0])}
```

Challenge 01

Task: Retrieve UNESCO Heritage Sites located in the United States that are categorized as either "Natural" or "Mixed" and then write the data to a CSV file.

1. In `main` create an empty "accumulator" list named `usa_sites`. Loop over the `sites` list. In the loop block write a conditional statement that identifies heritage sites located in the USA that are categorized as either "Natural" or "Mixed". Append each site that meets the conditions to your accumulator list.

Key	Value(s)
undp_code	'usa'
category	'Natural' or 'Mixed'

2. After exiting the loop, call the function `write_dicts_to_csv` and pass it the following arguments:
 1. "stu-usa-nat_mix.csv" (filepath)
 2. accumulator list of dictionaries (data)
 3. the keys from one of the nested dictionaries (fieldnames)



The "keys" that you pass as an argument is actually a `dict_keys` "view" object

Then review the file output.

Challenge 02

Task: Create a dictionary that holds counts of UNESCO Heritage Sites by region.

1. In `main` create an empty dictionary named `region_counts`. Loop over `sites` and accumulate site counts by region.
2. Uncomment the `print()` and `pp.pprint()` functions and check your work. The dictionary *must* contain the following key-value pairs.

```
{
    'Asia and the Pacific': 273,
    'Europe and North America': 516,
    'Arab States': 88,
    'Africa': 92,
    'Latin America and the Caribbean': 143
}
```

BONUS: Reorder the key-value pairs by *value* in descending order by creating a new dictionary named `region_counts` employing either the built-in function `dict()` or a dictionary comprehension.



In your expression pass the following arguments to the built-in function `sorted()`:

```
# Sort by key (x[0]); sort by value (x[1])
# reverse=True (descending order)
sorted(region_counts.items(), key=lambda x: x[1], reverse=True)
```

Challenge 03

Task: Retrieve UNESCO Heritage Sites designated as endangered per [Article 11, paragraph 4](#) of the UNESCO [Convention Concerning the Protection of the World Cultural and Natural Heritage](#). Create a list of new dictionaries comprising a subset of the available key-value pairs and then write the data to a CSV file.

1. In `main` create an empty "accumulator" list named `endangered_sites`. Loop over the `sites` list. In the loop block write a conditional statement that identifies heritage sites considered endangered (i.e., `'endangered': '1'`). Append each site that meets the conditions to your accumulator list constructing a new dictionary that maps the appropriate values to the following keys:

1. 'id_no'
2. 'category'
3. 'name_en'
4. 'region_en'
5. 'states_name_en'



As noted in the previous lecture there are several ways to create a dictionary and assign it key-value pairs.

2. After exiting the loop, call the function `write_dicts_to_csv` and pass it the following arguments:

1. "stu-endangered.csv" (filepath)
2. accumulator list of dictionaries (data)
3. the `dict_keys` object obtained from one of the nested dictionaries (fieldnames)

Then review the file output.

Challenge 04

Task: Create a dictionary that holds counts of endangered UNESCO Heritage Sites by region. Then loop over the dictionary and replace each regional count with the corresponding percentage value rounded to the second decimal place.

1. In `main` create an empty dictionary named `endangered_counts`. Loop over `endangered_sites` and accumulate site counts by region ("region_en").

BONUS: Reorder the key-value pairs by *value* in descending order by creating a new dictionary named `endangered_counts` employing either the built-in function `dict()` or a dictionary

comprehension.



In your expression pass the following arguments to the built-in function `sorted()`:

```
# Sort by key (x[0]); sort by value (x[1])
# reverse=True (descending order)
dict(sorted(endangered_counts.items(), key=lambda x: x[1],
reverse=True))
```

2. Uncomment the `print()` and `pp.pprint()` functions and check your work. The dictionary *must* contain the following key-value pairs.

```
{
    'Arab States': 21,
    'Africa': 14,
    'Asia and the Pacific': 6,
    'Latin America and the Caribbean': 6,
    'Europe and North America': 4
}
```

3. Currently, three (3) of the Ukrainian Heritage Sites are now located in a conflict zone. Update the "Europe and North America" key-value pair count to reflect this new reality.
4. Call the appropriate `dict` method that returns `endangered_sites` values (an *iterable*) and pass the expression to a `built-in function` that adds each of the iterable's items and returns the total amount. Assign the return value to the variable named `count`.
5. Loop over the `endangered_sites` items (keys and values). For each key-value pair encountered replace the value with the corresponding percentage value rounded to the 2nd decimal place computed from the following equation:

$$< \text{value} > / < \text{count} > * 100 \text{ (rounded to the 2nd decimal place)}$$


Be prepared to convert string values to integers.

6. Uncomment the `print()` and `pp.pprint()` functions and check your work. The mutated dictionary *must* contain the following key-value pairs.

```
{
    'Arab States': 38.89,
    'Africa': 25.93,
    'Asia and the Pacific': 11.11,
    'Latin America and the Caribbean': 11.11,
    'Europe and North America': 12.96
}
```

