SI 506 Lecture 18

Topics

1. JSON
    1. JSON data structures
    2. JSON value types
    3. JSON example
2. `json` module
    1. Reading JSON files (`json_load()`)
    2. Writing to a JSON file (`json_dump()`)
    3. Challenge 01
3. Nested loops
    1. Challenge 02
    2. Challenge 03

Vocabulary

- **Nested Loop**. A `for` or `while` loop located within the code block of another loop.

Reference

Boomark the following w3schools reference page:

1. w3schools, "Python Dictionary Methods"

Data

The New York Times provides an Article Search API (Application Programming Interface) that permits keyword searching and retrieval of JSON representations of NY Times articles.

The file `nyt-articles-20221031.json` contains metadata about 300 articles published by the Science news desk under the general subject "Psychology and Psychologists".

1.0 JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format for exchanging information between systems.

JSON consists of two basic data structures and several value types.

1.1 JSON data structures

1. An *unordered* set of name-value pairs known as an *object* and denoted by curly braces (`{}`).
2. An *ordered* list of values known as an *array* and denoted by square brackets (`[]`).

1.2 JSON value types

1. string
2. number
3. boolean
4. array `[]`
5. object `{}`
6. null

1.3 JSON example

Below is an example JSON representation of a NYT article that can be found in the `nyt-articles-20221031.json` file.

💡 Certain name-value pairs have been removed from the JSON document below in the interests of brevity. For instance only the top 3 keywords (out of 10) have been included in the example.

```
{
  "abstract": "Ten years ago, psychologists proposed that a wide range of people would suffer anxiety and grief over climate. Skepticism about that idea is gone.",
  "web_url": "https://www.nytimes.com/2022/02/06/health/climate-anxiety-therapy.html",
  "source": "The New York Times",
  "headline": {
    "main": "Climate Change Enters the Therapy Room",
    "kicker": null,
    "content_kicker": null,
    "print_headline": "Anxiety Over Climate Change Lands on the Therapist’s Couch"
  },
  "keywords": [
    {
      "name": "subject",
      "value": "Anxiety and Stress",
      "rank": 1,
      "major": "N"
    },
    {
      "name": "subject",
```

```
        "value": "Psychology and Psychologists",
        "rank": 2,
        "major": "N"
      },
      {
        "name": "subject",
        "value": "Global Warming",
        "rank": 3,
        "major": "N"
      }
    ],
    "pub_date": "2022-02-06T10:00:12+0000",
    "document_type": "article",
    "news_desk": "Science",
    "section_name": "Health",
    "byline": {
      "original": "By Ellen Barry",
      "person": [
        {
          "firstname": "Ellen",
          "middlename": null,
          "lastname": "Barry",
          "qualifier": null,
          "title": null,
          "role": "reported",
          "organization": "",
          "rank": 1
        }
      ],
      "organization": null
    },
    "type_of_material": "News",
    "word_count": 1940
  }
```


2.0 `json` module

Like the `csv` module the Python standard libary's `json` module provides enhanced functionality for working with JSON files. JSON is a lightweight data interchange format for exchanging information between systems.

To use the `json` module you must import it:

```
import json
```

There are four `json` module functions; two of which are of particular interest to us:

Function	Purpose

Function	Purpose
`json.load()`	*Deserializes* (decodes) a text or binary file that contains a JSON document to a `dict` or `list`.
`json.dump()`	*Serializes* (encodes) an object as a JSON formatted stream to be stored in a file.
`json.loads()`	*Deserializes* (decodes) a string, bytes, or bytearray containing a JSON document to a `dict` or `list`.
`json.dumps()`	*Serializes* (encodes) an object to a JSON formatted string.

Since you will also be working with JSON documents between now and the end of the semester implmenting a function that can read a JSON document as well one that can write a JSON document to a file will prove useful.

2.1 Reading JSON files (`json_load()`)

The function `read_json()` reads a JSON document per the provided filepath, calls the json module's `json.load()` function in order to *decode* the file data as a `dict` or a `list` (of dictionaries), before returning the decoded data to the caller.

```
def read_json(filepath, encoding='utf-8'):
    """Reads a JSON document, decodes the file content, and returns a list
or dictionary if
    provided with a valid filepath.

    Parameters:
        filepath (str): path to file
        encoding (str): name of encoding used to decode the file

    Returns:
        dict/list: dict or list representations of the decoded JSON
document
    """

    with open(filepath, 'r', encoding=encoding) as file_obj:
        return json.load(file_obj)
```

2.2 Writing to a JSON file (`json_dump()`)

The function `write_json()` accepts a dictionary or a list of dictionaries, calls the json module's `json.dump()` function in order to *encode* the passed in data as JSON before writing the encoded data to the target file.

```
def write_json(filepath, data, encoding='utf-8', ensure_ascii=False,
indent=2):
    """Serializes object as JSON. Writes content to the provided filepath.

    Parameters:
        filepath (str): the path to the file
        data (dict)/(list): the data to be encoded as JSON and written to
the file
        encoding (str): name of encoding used to encode the file
        ensure_ascii (str): if False non-ASCII characters are printed as
is; otherwise
                            non-ASCII characters are escaped.
        indent (int): number of "pretty printed" indention spaces applied
to encoded JSON

    Returns:
        None
    """

    with open(filepath, 'w', encoding=encoding) as file_obj:
        json.dump(data, file_obj, ensure_ascii=ensure_ascii,
indent=indent)
```

Challenge 01

Task: Read in the NY Times JSON article, extract articles written in 2022, and write the articles that meet the filter condition to a file as JSON.

1. In `main` call the function `read_json` and provide it with the filepath `nyt-articles-20221031.json` in order to retrieve NY Times Science Desk articles filtered on the subject "Psychology and Psychologists". Assign the return value to a variable named `articles`.

2. Loop over `articles` and in the loop block write an `if` statement that identifies articles with a publication year of 2022. Append each 2022 article to a list (name your choice).

   💡 Review the JSON file `nyt-article-example.json` for the appropriate publication date name-value pair in order to derive the dictionary key name to use in your `if` statement.

   ❗ Each article contains an ISO-8601 date formatted string as the following example illustrates:

   ```
   2022-02-06T10:00:12+0000
   ```

   💡 In your `if` statement utilize a `str` method to access the year portion (i.e., "2022") of the string. Alternative approaches could involve use of the `datetime` module or the third-party library `dateutil`, but these are out-of-scope for this challenge.

3. After exiting the loop, call the function `write_json` and write the "accumulator" list encoded as JSON to a file named `stu-nyt-articles-2022.json`.

3.0 Nested loops

A nested loop refers to a loop located within the code block of another loop. During each iteration of the "outer" loop, the "inner" loop will execute, completing all its iterations (unless terminated early) *prior* to the outer loop commencing its next iteration, if any.

```
for < element > in < sequence >:
    # indented block
    for < element > in < element (itself a sequence) >
        < statement A >
        < statement B >
        ...
```

The following example illustrates the concept. Loop over the nested lists and print the result of summing the list elements multipled by each individual element.

```
nums = [
        [1, 2, 3, 4, 5],
        [10, 20, 30, 40, 50],
        [100, 200, 300, 400, 500],
        [1000, 2000, 3000, 4000, 5000]
    ]

    for i in nums:
        for j in i:
            print(sum(i) * j) # sum the list element then multiply by each
element
```

Returning to the NYT articles, what if you were asked to identify articles in the `articles` list that possess one or more keywords? Note, that each JSON document representing a NY Times article contains a list of keyword objects. For example a 2018 article by Paula Span entitled "Dementia Is Getting Some Very Public Faces" includes the following "keywords" JSON list:

```
"keywords": [
  {"name": "subject", "value": "Alzheimer's Disease", "rank": 1, "major":
"N"},
  {"name": "subject", "value": "Elderly", "rank": 2, "major": "N"},
  {"name": "subject", "value": "Dementia", "rank": 3, "major": "N"},
  {"name": "subject", "value": "Psychology and Psychologists", "rank": 4,
"major": "N"},
  {"name": "subject", "value": "Disabilities", "rank": 5, "major": "N"},
  {"name": "subject", "value": "Celebrities", "rank": 6, "major": "N"},
```

```
  {"name": "persons", "value": "O'Connor, Sandra Day", "rank": 7, "major":
"N"}
]
```

If you needed to return articles in the `articles` list that where tagged with either "Dementia" and/or "Alzheimer's Disease" you could write a function that performs a keyword look up (e.g., `has_keywords(keywords, filters)` and then call it from inside a `for` loop in order to identify dementia-related articles. Alternatively, we could implement a nested loop.

❗ The data set contains five (5) keyword "name" values: "subject", "glocations", "organizations", "persons", and "creative_works" so filtering on "subject" is required.

```
# Get all articles touching on Dementia
dementia = []
for article in articles:
    for keyword in article['keywords']:
        if keyword['name'] == 'subject' and keyword['value'] in
("Alzheimer's Disease", 'Dementia'):
            dementia.append(article)
            break # avoid appending duplicates due to either/or membership
check

# Write to file
write_json('stu-nyt-dementia-articles.json', dementia_articles)
```

In the example above, the outer loop iterates over the `articles` list. The inner loop iterates over each article's "keywords" list. If the keyword is a subject keyword and the keyword value is either "Alzheimer's Disease" or "Dementia" a match is obtained and the article is appended to the list `dementia_articles`. Note the use of the `break` statement inside the inner loop. Since an article can contain both keyword objects it is imperative that no additional loop iterations occur after the first match is obtained in order to avoid duplicate append operations.

💡 # Dementia is a generic term that covers a range of neurological conditions impacting the brain including Alzheimer's Disease.

3.1 Challenge 02

Task: Return a dictionary of *unique* keyword values grouped by the first character of each keyword and write the results to a JSON file.

1. In `main`, create an empty `dict` named `subjects`.
2. Implement a nested loop that iterates over each article's "keywords" list (i.e., loop over the articles and for each article loop over its keywords).

```
{
    'abstract": '...',
    ...,
    'keywords': [
        {
            'name': 'subject',
            'value': 'Anxiety and Stress',
            ...,
        },
        ...
    ],
    ...
}
```

3. Inside the nested loop block, write an `if` statement that encompasses the following two conditions:

    1. The keyword dictionary's "name" value equals "subject"
    2. The keyword dictionary's "value" is not a member of `subjects`

4. If *both* conditions return `True` create two variables inside the `if` block and assign them the following values:

    1. `key`: access the **first character** of the "value" string and assign a **capitalized** version of it to `key`.

    2. `val`: assign the "value" string to `val`.

5. While inside the `if` block assign key-value pairs to the `subjects` dictionary. Each key-value pair must be structured as follows:

```
# '1st character of "value" capitalized': [ < value_01 >, < value_02
>, ...]
{'A': ['Anatomy and Physiology', 'Advertising and Marketing', ...]}
```

Implement the following filtering "rules" with an `if-else` statement:

1. Add a new key-value pair *if and only if* the key does not exist among the `subjects` dictionary's keys.

❗ When adding a new key-value pair assign a list comprising *a single element* (`val`) to the new key as the value.

2. If the key exists among the `subjects` dictionary's keys, append `val` to the list value but only if the string assigned to `val` has not been added previously. In other words, no duplicates permitted.

6. After exiting the loop uncomment the `write_json` function and run your file. Confirm that the JSON file `stu-nyt-subjects.json` was written to your current working directory.

7. BONUS. Uncomment the "Sort keys" dictionary comprehension and the accompanying call to the function `write_json()` and run your file again. Confirm that the JSON file `stu-nyt-subjects_sorted.json` was written to your current working directory.

3.2 Challenge 03

Task: Create a dictionary that links individual articles to the keyword subjects assigned to each by the NYT staff. You will create two sets of nested loops, one outer nested loop and one inner nested loop (four loops in total).

1. In `main`, create an empty `dict` named `subject_articles`.

2. Implement a nested loop:

   - Loop I (outer): iterate over the `subjects` dictionary's key-value pairs
   - Loop II (inside loop I): iterate over the "subject" elements in the value (a `list`)

3. Inside the "subject" inner loop implement another nest loop:

   - Loop III (inside loop II): iterate over the the `articles` list of article dictionaries
   - Loop IV (inside loop III): iterate over each article's "keywords" list (as you did in the previous challenge)

4. Insert **between** loop III and loop IV a dictionary named `story` comprising the following key-value pairs:

   - headline_main
   - web_url
   - pub_date

   Access the current article and assign its main headline, web URL, and publication date values to the appropriate `story` keys. You will assign this "thin" version of the article dictionary to `subject_articles`.

5. In the innermost "keywords" loop write an `if` statement evaluates the return value of the function named `has_subject`. Pass the current "keyword" value (a `dict`), the string "subject" and the loop II "subject" loop variable.

6. Inside the `if` statement, implement the same `if-else` logic utilized in the previous challenge to populate the "accumulator" dictionary `subject_articles`. Below is the structure of the key-value pairs that you must add to `subject_articles`:

   ```
   {'< Loop two subject >': [< story_01 >, < story_02 >, ...]}
   ```

   The `if-else` logic must determine whether or not the current "subject" value is a key among the `subject_article` keys and either create a new key-value pair or append the current `story` dictionary to a matching key's list value.

The goal is to populate `subject_articles` with key-value pairs such as the example below:

```
'Artificial Intelligence': [
 {
   'headline_main': 'Something Bothering You? Tell It to Woebot.',
   'web_url': 'https://www.nytimes.com/2021/06/01/health/artificial-
intelligence-therapy-woebot.html',
   'pub_date': '2021-06-01T06:00:11+0000'
 },
 {
   'headline_main': 'Need a Hypothesis? This A.I. Has One',
   'web_url': 'https://www.nytimes.com/2020/11/24/science/artificial-
intelligence-ai-psychology.html',
   'pub_date': '2020-11-24T13:47:01+0000'
 },
 {
   'headline_main': 'Why Stanford Researchers Tried to Create a
‘Gaydar’ Machine',
   'web_url': 'https://www.nytimes.com/2017/10/09/science/stanford-
sexual-orientation-study.html',
   'pub_date': '2017-10-09T16:44:15+0000'
 }
```

],

```
7. After exiting the loop uncomment the `write_json` function and run your
file. Confirm that the
 JSON file `stu-nyt-subject_articles.json` was written to your current
working directory.

8. BONUS. Uncomment the "Sort keys" dictionary comprehension and the
accompanying call to the function
 `write_json()` and run your file again. Confirm that the JSON file
 `stu-nyt-subjects_articles_sorted.json`was written to your current
working directory.
```