Discussion 6: Midterm Review and CSV

SI 206: Data-Oriented Programming

Instructor: Dr. Barbara (Barb) Ericson
GSI: Kexuan (Michael) Huang
IA: Cristina & Jade

School of Information
University of Michigan

Fall 2023

Reminders

- Midterm 1 during lecture time (Wed/Thur) in the lecture room
- Can bring one double-side cheat sheet with notes
- No outside resource (Google, ChatGPT, GitHub Copilot, Runestone...)

Deadlines

- Nothing due this Friday
- Project 1 due next Friday (10/13)

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Table of Contents

Table of Contents

Function

- Defined with the keyword def
- Must be called to execute

Function

```python
def print_list(my_list):
    for elm in my_list:
        print(elm)

print_list([1, 2, 3])
```

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Conditional

- Execute code if a Boolean expression is true or false
- Keywords: if, elif, else
- Logical operators: not, and, or.

Conditionals

```python
score = 100
if 95 <= score <= 100:
    print('A+')
elif score >= 60:
    print('Pass')
else:
    print('Fail')
```

String

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

- String methods:
  - `str.lower()`: converts all uppercase characters in a string into lowercase characters and returns it.
  - `str.upper()`: converts all lowercase characters in a string into uppercase characters and returns it.
  - `str.capitalize()`: converts the first character of a string to an uppercase letter and all other alphabets to lowercase.
  - `str.startswith(prefix)`: returns True if a string starts with the specified prefix. If not, it returns False
  - `str.find(sub_str)`: returns the index of first occurrence of the substring (if found). If not found, it returns -1.
- String indexing and slicing

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

String

Assume `my_string = 'i love PYTHON'`

String

```
my_string.lower()        # return 'i love python'
my_string.upper()        # return 'I LOVE PYTHON'
my_string.capitalize()   # return 'I love python'
my_string.find('ov')     # return 3
my_string[2]             # return 'l'
my_string[-5:-3]         # return 'YT'
```

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

List

- List holds items in order
- List methods:
  - `list.append(item)`: adds an item to the end of the list.
  - `list.extend(list_2)`: adds all the elements of an iterable (list, tuple, string etc.) to the end of the list.
  - `list.pop(index)`: removes the item at the specified index. The method also returns the removed item.
  - `list.reverse()`: reverses the elements of the list.
  - `list.sort()`: sorts the items of a list in ascending or descending order.
- List indexing and slicing

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

List

Assume:

```
list_1 = ['a', 'b', 'c']
list_2 = ['d', 'e']
```

List

```
list_1.append('d')             # ['a', 'b', 'c', 'd']
list_1.extend(list_2)          # ['a', 'b', 'c', 'd', 'e']
list_1.pop(1)                  # ['a', 'c']
list_1.reverse()               # ['c', 'b', 'a']
list_1.sort(reverse=True)      # ['c', 'b', 'a']
list_1[-1]                     # 'c'
list_1[1:]                     # ['b', 'c']
```

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Loops

- For loop
  - For each item in a collection:
    - `for element in list:`
    - `for key in dict:`
    - `for key in dict.keys():`
    - `for value in dict.values():`
    - `for key, value in dict.items():`
  - For value in range:
    - `for i in range(len(my_list)):`
- While loop: only executes while the Boolean expression is true

Tuple

- Immutable (unchangeable)
- Tuple indexing (the same as what we do for List)
- Unpacking a Tuple

Tuple

```
my_tuple = (1, 2, 3)
my_tuple[1] = 4      # thorw an error
my_tuple[1]          # return 2

a, b, c = my_tuple   # unpack a tuple
print(a)             # 1
print(b)             # 2
print(c)             # 3
```

Dictionary

- Key: Value pairs

Dictionary

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_dict['d'] = 4    # create a new key-value pair
my_dict.get('c')    # return 3
my_dict.get('e', 0) # return 0

# print key-value pairs in loop
for key, value in my_dict.items():
    print(key, value)
```

Turtle

- Turtle methods
  - `forward(distance)`
  - `backward(distance)`
  - `color(str)`
  - `goto(x, y)`
  - `left(degree)`
  - `right(degree)`
  - `pendown()`
  - `penup()`
  - `setheading((degree))`
- Screen methods
  - `screensize(number)`
  - `bgcolor(str)`
  - `exitonclick()`

Terminal

Terminal Commands:

- `cd`: Change Directory
- `pwd` / `chdir`: Print working (current) directory
- `ls` / `dir`: List content of directory

Git and GitHub

Git Commands:

- `git clone`: make a copy of your remote repository on your local computer
- `git add`: add file to your staging area
- `git commit -m "message"`: make a snapshot of your local repository
- `git push`: send your committed local changes to remote repository (GitHub)
- `git status`: display the information for current stage

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Class and Object

- Class: defines what all objects of the class know (attributes) and can do (methods)
- Attribute: Variable which is defined inside a class
  - class attribute: the variable defined in the class but outside the methods.
  - object attribute: the variable defined with self. (usually inside `__init__()`)
- Object: an instance of a class
- Inheritance: "is a type of" (dog is a type of animal)
- Association: "has a" (animal has a name)

Class and Object

Class and Object

```python
class MyClass:
    a = 'This is a class attribute' # This is a class attribute

    def __init__(self, b):
        self.b = b # This is an object attribute

    def my_method(self, c):
        # attributes can be accessed inside the class with 'self' keyword
        print(self.a)
        print(self.b)
        # parameters can only be accessed inside this method
        print(c)
```

Unit Test

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Unit Test

```
import unittest

class TestAll(unittest.TestCase): # make a subclass of unittest.TestCase
    def setUp(self): # setUp() will be executed before test cases
        self.my_variable = 'some values'

    def test_my_function(self):   # start each method with "test_"
        self.assertEqual(my_function(), "expected output")
        self.assertAlmostEqual(my_function(), 0.02 , places=2)
        self.assertTrue(my_function())
        self.assertFalse(my_function())

unittest.main() # run all tests
```

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Table of Contents

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

CSV Format

- CSV (Comma Separated Values) files are a simple and lightweight way to store structured data.
- CSV are a common format for sharing datasets.
- a CSV file represents data as a series of rows and columns, much like an Excel spreadsheet or a matrix.
- Header row: the first row of a CSV usually contains the name of each column. Most CSV files have one, but this is not required.
- Each line of a CSV represents a row. Usually the columns are separated from each other using commas "," within each row.
- Other separators such as tabs "\t" and pipes "|" can be used, but commas are by far the most common.

CSV File Example

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

forestfires.csv

```
month,day,temp,RH,area
jul,tue,18.0,42,0.36
sep,tue,21.7,38,0.43
aug,wed,23.3,31,0.55
```

- Here the first line is header row
- Each row represents a set of data
- No need for additional spaces between commas.

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Newline Characters \n

demo.txt

```
this is the first row
this is the second row
```

- When we open a CSV file in a program like Excel or an IDE like VSCode, the rows are automatically placed onto their own line for readability.
- However computers don't have eyes, so they need to use a special character called a newline to know where one line should end and one should begin.
- A newline character is represented in Python as \n and it counts as a single character: `len('\n') == 1` would return True
- The above file equals "this is the first row\nthis is the second row"

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Table of Contents

Discussion 6 Exercise

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

- Go to Canvas → Assignment → Discussion 6 and clone the GitHub Repo https://classroom.github.com/a/RH4n_meN
- We will be working with a dataset of Forest Fires

forestfires.csv

- month: month of the year: 'jan' to 'dec'
- day: day of the week: 'mon' to 'sun'
- temp: temperature in Celsius degrees: 2.2 to 33.30
- RH: relative humidity in
- area: the burned area of the forest (in hectares): 0.09 to 1090.84

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Discussion 6 Exercise

Your task

- Our task is to read in the CSV file and builds a dictionary called `data_dict`
- Fix the bugs in the `build_data_dict()` method
- Implement all methods of the FireReader class
- Don't change test cases

```
self.data_dict = {
        'month': ['jul', 'sep', 'sep', ...],
        'day':   ['tue', 'tue', 'mon', ...],
        'temp':  [18.0, 21.7, 21.9, ...],
        'RH':    [42, 38, 39, ...],
        'area':  [0.36, 0.43, 0.47, ...]
}
```