

Midterm
Review

Function
Conditional
String
List
Loops
Tuple
Dictionary
Turtle
Terminal
Git and GitHub
Class and Object
Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

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

Midterm
Review

Function
Conditional
String
List
Loops
Tuple
Dictionary
Turtle

Terminal

Git and GitHub
Class and Object
Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

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)

Table of Contents

Midterm Review

Function
Conditional
String
List
Loops
Tuple
Dictionary
Turtle
Terminal
Git and GitHub
Class and Object
Unit Test

Working with CSV files

CSV format
Newline Character

Practice Problem

- ① Midterm Review
- ② Working with CSV files
- ③ Practice Problem

Table of Contents

Midterm Review

Function
Conditional
String
List
Loops
Tuple
Dictionary
Turtle
Terminal
Git and GitHub
Class and Object
Unit Test

Working with CSV files

CSV format
Newline Character

Practice Problem

- 1 Midterm Review
- 2 Working with CSV files
- 3 Practice Problem

Function

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

Function

```
1  def print_list(my_list):  
2      for elm in my_list:  
3          print(elm)  
4  
5  print_list([1, 2, 3])
```

Midterm
Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format

Newline Character

Practice
Problem

Conditional

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

Conditionals

```
1 score = 100
2 if 95 <= score <= 100:
3     print('A+')
4 elif score >= 60:
5     print('Pass')
6 else:
7     print('Fail')
```

String

Midterm Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with CSV files

CSV format

Newline Character

Practice Problem

- 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

String

Midterm
Review

Function
Conditional

String

List
Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

Assume `my_string = 'i love PYTHON'`

String

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


List

Midterm
Review

Function
Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format

Newline Character

Practice
Problem

- 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

List

Assume:

```
list_1 = ['a', 'b', 'c']
```

```
list_2 = ['d', 'e']
```

List

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

Midterm
Review

Function
Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

Loops

Midterm
Review

Function
Conditional

String
List

Loops

Tuple
Dictionary
Turtle

Terminal
Git and GitHub
Class and Object
Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

- **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

```
1 my_tuple = (1, 2, 3)
2 my_tuple[1] = 4      # thorw an error
3 my_tuple[1]          # return 2
4
5 a, b, c = my_tuple   # unpack a tuple
6 print(a)             # 1
7 print(b)             # 2
8 print(c)             # 3
```

Dictionary

- Key: Value pairs

Dictionary

```
1 my_dict = {'a': 1, 'b': 2, 'c': 3}
2 my_dict['d'] = 4      # create a new key-value pair
3 my_dict.get('c')      # return 3
4 my_dict.get('e', 0)   # return 0
5
6 # print key-value pairs in loop
7 for key, value in my_dict.items():
8     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()`

Midterm
Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format

Newline Character

Practice
Problem

Terminal

Midterm Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with CSV files

CSV format

Newline Character

Practice Problem

Terminal Commands:

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

Git and GitHub

Midterm Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with CSV files

CSV format

Newline Character

Practice Problem

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

Class and Object

Midterm Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with CSV files

CSV format

Newline Character

Practice Problem

- **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

```
1  class MyClass:
2      a = 'This is a class attribute' # This is a class attribute
3
4      def __init__(self, b):
5          self.b = b # This is an object attribute
6
7      def my_method(self, c):
8          # attributes can be accessed inside the class with 'self' keyword
9          print(self.a)
10         print(self.b)
11         # parameters can only be accessed inside this method
12         print(c)
```

Unit Test

Unit Test

```
1  import unittest
2
3  class TestAll(unittest.TestCase): # make a subclass of unittest.TestCase
4      def setUp(self): # setUp() will be executed before test cases
5          self.my_variable = 'some values'
6
7      def test_my_function(self):    # start each method with "test_"
8          self.assertEqual(my_function(), "expected output")
9          self.assertAlmostEqual(my_function(), 0.02 , places=2)
10         self.assertTrue(my_function())
11         self.assertFalse(my_function())
12
13  unittest.main() # run all tests
```

Midterm
Review

Function
Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format
Newline Character

Practice
Problem

Table of Contents

Midterm Review

[Function](#)[Conditional](#)[String](#)[List](#)[Loops](#)[Tuple](#)[Dictionary](#)[Turtle](#)[Terminal](#)[Git and GitHub](#)[Class and Object](#)[Unit Test](#)

Working with CSV files

[CSV format](#)[Newline Character](#)

Practice Problem

① Midterm Review

② Working with CSV files

③ Practice Problem

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

forestfires.csv

```
1 month,day,temp,RH,area
2 jul,tue,18.0,42,0.36
3 sep,tue,21.7,38,0.43
4 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.

Newline Characters \n

demo.txt

```
1  this is the first row
2  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"`

Table of Contents

Midterm Review

Function
Conditional
String
List
Loops
Tuple
Dictionary
Turtle
Terminal
Git and GitHub
Class and Object
Unit Test

Working with CSV files

CSV format
Newline Character

Practice Problem

- 1 Midterm Review
- 2 Working with CSV files
- 3 Practice Problem

Discussion 6 Exercise

- 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

Midterm

Review

Function

Conditional

String

List

Loops

Tuple

Dictionary

Turtle

Terminal

Git and GitHub

Class and Object

Unit Test

Working with
CSV files

CSV format

Newline Character

Practice
Problem

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 the test cases

```
1 self.data_dict = {  
2     'month': ['jul', 'sep', 'sep', ...],  
3     'day':   ['tue', 'tue', 'mon', ...],  
4     'temp':  [18.0, 21.7, 21.9, ...],  
5     'RH':    [42, 38, 39, ...],  
6     'area':  [0.36, 0.43, 0.47, ...]  
7 }
```