CS1520 Recitation: Week 5 Web storage / Python / Flask(0) ...

Jeongmin Lee

Web storage

Using the Web Storage API

The Web Storage API provides **mechanisms**

- **browsers can securely store key/value pairs**

SOURCE: MOZILLA DOCS

Using the Web Storage API

The keys and the values are always strings.

Question: What happens if the key is an integer ?

Using the Web Storage API

The keys and the values are always strings.

Question: What happens if the key is an integer ?

A: Integer key is automatically converted to string, like what objects do.

Types

sessionStorage

- maintains a separate storage area for each given origin
- available for the **duration of the page session**
- as long as the browser is open,
  including page reloads and restores

SOURCE: MOZILLA DOCS

Types

sessionStorage

- maintains a separate storage area for each given origin
- available for the **duration of the page session**
- as long as the browser is open,
  including page reloads and restores

localStorage

- does the same thing
- **persists** even when the browser is closed and reopened

SOURCE: MOZILLA DOCS

Browser Support!

Newer versions of most browsers support Web Storage.

Although!!! You need to be sure.

Question: How would you check ?

```
function storageAvailable(type) {
    try {
        var storage = window[type],
            x = '__storage_test__';
        storage.setItem(x, x);
        storage.removeItem(x);
        return true;
    }
    catch(e) {
        //Print Not supported!
        //Other reasons possible ? YES!
        //Get error codes as e.code
        //Get error names as e.name
    }
}
```

window[type] ? What is this ?

SOURCE: MOZILLA DOCS

```
function storageAvailable(type) {
    try {
        var storage = window[type],
            x = '__storage_test__';
        storage.setItem(x, x);
        storage.removeItem(x);
        return true;
    }
    catch(e) {
        //Print Not supported!
        //Other reasons possible ? YES!
        //Get error codes as e.code
        //Get error names as e.name
    }
}
```

window[type] ? What is this ?

A: The type of storages;

sessionStorage and localStorage

SOURCE: MOZILLA DOCS

Get the storage first!

```
var sess_storage = window[“sessionStorage”]
```

```
var local_storage = window[“localStorage”]
```

SOURCE: MOZILLA DOCS

Setting values in storage

Storage.setItem(*key, value*) is used both to
- create new data items
- (if the data item already exists) update existing values.

- This takes two arguments
  — the key of the data item to create/modify
  — the value to store in it.

SOURCE: MOZILLA DOCS

Getting values from storage

Storage.getItem(*key*) method is used to get a data item from storage.

Deleting data records

Web Storage provides a couple of ways to remove data.

Storage.removeItem(*key*) takes a single argument

— the key of the data item you want to remove

— and removes it from the storage object for that domain.

Storage.clear() takes no arguments, and simply empties the entire storage object for that domain.

CS1520 Recitation:

Python

...

Jeongmin Lee

Plan for Today

- Functions
- String functions
- Tuples
- Dictionaries
- List Manipulation

Note that this slide is largely based on the Python Cheat Sheet distributed at http://www.cogsci.rpi.edu/~destem/igd/python_cheat_sheet.pdf

Indentations

Indentation matters in Python

- Usually 4 space char or tab
- It creates boundary for function / if-else statement / for loops

Functions

Functions

1. Syntax

```
def function_name ( parameters ):

    # function contents here

    return [expression]
```

Functions

1. Syntax

parameters can be omitted. But still you need parenthesis

```
def function_name ( parameters ):

    # function contents here

    return [expression]
```

you must have indentation (recommend four spaces)

you can also omit return if no returning value exists

Functions

Example

```
def helloworld ():

    print('hello world')


helloworld()
```

you can call a function like this

Functions

Example

```
def sum(a,b):
    return a+b


sum(2,3)
```

String Functions

String Functions

1. Creation

- the_str = “Hello World”
- the_str = ‘Hello World’
- the_str = “““Hello World””” <- quite often used to write multi-line comments

String Functions

2. Accessing

- the_str[4]
  - returns ‘o’

String Functions

3. Splitting

- the_str.split(‘ ’)
  - returns [‘Hello’, ‘World’]

String Functions

3. Splitting

- the_str.split(‘ ’)
  - returns [‘Hello’, ‘World’]
- the_str.split(‘r’)
  - returns [‘Hello Wo’, ‘ld’]
- Notice that ‘r’ is removed from both sides of output

String Functions

4. Joining a list to a string

- words = [‘Welcome’, ‘to’, ‘CS1520’, ‘recitation’]
- ‘ ’.join(words)
  - returns ‘Welcome to CS1520 recitation’
- ‘ZZZ’.join(words)
  - returns ‘WelcomeZZZtoZZZCS1520ZZZrecitation’

String Functions

5. String formatting

- this_string = “there”
- print(“Hello {}!”.format(this_string))
  - returns “Hello there!”

String Functions

5. String formatting (continue)

- When multiple items to print, put them into a tuple
  - this_string = “there”
  - this_day = 6
  - this_month = “October”

String Functions

5. String formatting (continue)

- When multiple items to print, just pass them
  - this_string = “there”
  - this_day = 6
  - this_month = “October”
- print(“Hello {}! at {}th day of {} ”.format(this_string, this_day, this_month))
  - returns “Hello there! at 6th day of October”

Python Tuples

Python Tuples

- A tuple consists of values separated by commas.
- They are useful for ordered pairs and returning several values from a function.

Python Tuples

1. Creation

- emptyTuple = ()
- a = (“spam”,) <- Note the comma
- b = (“spam”)
- a == b
    - returns False!
    - a: tuple, b: string

Python Tuples

2. Accessing

- c = (“spam”, 12, 4)
- c[1]
  - returns 12

Python Tuples

- Tuple is immutable, while list isn't.
  - a = (1,2)
  - a[0] = 4 // returns error

Python Dictionaries

Python Dictionaries

A dictionary is a set of key:value pairs. All keys must be unique.

1. Creation

- empty_dic = {}
- new_dic = {‘October’:10, ‘July’:7, ‘nine’:9}

2. Accessing

- new_dic[‘July’] returns 7

Python Dictionaries

3. Add new entry

- new_dic[‘entry_key’] = ‘entry_value”

4. Deleting an entry

- del new_dic[‘July’]

Python Dictionaries

5. Finding

- new_dic.has_key(‘z’)
  - return False
- new_dic.keys()
  - return [‘October’, ‘nine’]
- new_dic.values()
  - return [10, 9]

Python Dictionaries

6. Iteration on loop

- for key, value in new_dic.iteritems():
  - print(key)
  - print(value)

Python List Manipulation

Python List Manipulation

List is one of the most important data structure in Python. It is very flexible and have many built-in functions.

1. creation

- thelist = [1, 2, 4, ‘a’, ‘b’, ‘c]

2. accessing

- thelist[3] returns ‘a’

Python List Manipulation

3. Slicing

- thelist[1:3] returns [2,4]

thelist = [1, 2, 4, ‘a’, ‘b’, ‘c’]

Python List Manipulation

3. Slicing

- thelist[1:3] returns [2,4] thelist = [1, 2, 4, ‘a’, ‘b’, ‘c’]
- thelist[2:] returns [4,’a’,’b’,’c’] thelist = [1, 2, 4, ‘a’, ‘b’, ‘c’]

Python List Manipulation

3. Slicing

- thelist[1:3] returns [2,4]
- thelist[2:] returns [4,'a','b','c']
- thelist[:3] returns [1,2,4]

thelist = [1, 2, 4, 'a', 'b', 'c']

thelist = [1, 2, 4, 'a', 'b', 'c']

thelist = [1, 2, 4, 'a', 'b', 'c']

Python List Manipulation

3. Slicing

- thelist[1:3] returns [2,4]
- thelist[2:] returns [4,'a','b','c']
- thelist[:3] returns [1,2,4]
- thelist[-1] returns ['c']

thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']

Python List Manipulation

3. Slicing

- thelist[1:3] returns [2,4]
- thelist[2:] returns [4,'a','b','c']
- thelist[:3] returns [1,2,4]
- thelist[-1] returns ['c']
- thelist[4:-1] returns ['b','c']

thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']
thelist = [1, 2, 4, 'a', 'b', 'c']

Python List Manipulation

4. Length

- len(thelist)
  - returns 6

Python List Manipulation

5. Sort

- sorted(thelist)
  - returns sorted (new) list

Python List Manipulation

5. Sort

- sorted(thelist)
  - returns sorted (new) list
- thelist.sort()
  - returns nothing. It sorts the list itself.

Python List Manipulation

5. Sort (continue)

- Sort with inverse order (default is ascending order)
  - => reverse=True

Python List Manipulation

5. Sort (continue)

- Sort with inverse order (default is ascending order)
  - => reverse=True
- sorted(thelist, reverse=True)
  - returns [‘c’, ‘b’, ‘a’, 4, 2, 1]

Python List Manipulation

5. Sort (continue)

- Sort with inverse order (default is ascending order)
  - => reverse=True
- sorted(thelist, reverse=True)
  - returns [‘c’, ‘b’, ‘a’, 4, 2, 1]
- thelist.sort(reverse=True)
  - change the list’s contents to [‘c’, ‘b’, ‘a’, 4, 2, 1]

Python List Manipulation

6. Add new element

- thelist.append(‘z’)
  - thelist: [1,2,4, ‘a’, ‘b’, ‘c’, ‘z’]
- New element is added at the end of the list

Python List Manipulation

7. Add new element with specific position

- thelist.insert(2, ‘x’)
  - 2: specific position
  - ‘z’: new element
  - thelist: [1,2, ‘x’, 4, ‘a’, ‘b’, ‘c’, ‘z’]

Python List Manipulation

8. Return and remove

- thelist.pop()
  - return the last element and remove the element
  - return ‘z’
  - thelist: [1,2, ‘x’, 4, ‘a’, ‘b’, ‘c’]

Python List Manipulation

8. Return and remove (continue)

- thelist.pop(0)
  - return and remove the element at specific position
  - return 1
  - thelist: [2, ‘x’, 4, ‘a’, ‘b’, ‘c’]

Python List Manipulation

9. List concatenation

- thelist + [0]
  - returns [2, ‘x’, 4, ‘a’, ‘b’, ‘c’, 0]

Python List Manipulation

10. Find the index of an item

- thelist.index(‘x’)
  - returns 1
  - thelist: [2, ‘x’, 4, ‘a’, ‘b’, ‘c’, 0]
- When multiple same items are exist, it returns first item’s position

CS1520 Recitation:

Install Flask (with VirtualENV)

...

Jeongmin Lee

Installing Flask

A Big Picture

What you need to install/do for running Flask is as follows, in the order written:

1. Install Python 3.6
2. Install PIP (download get-pip.py, `python get-pip.py`)
3. Install Virtual Environment (`pip install virtualenv`)
4. Create a new environment (`python -m virtualenv new_env -p ...`)
5. Activate the new environment (`source bin/activate <mac/linux>; script/activate.bat <windows>`)
6. Install Flask (`pip install Flask`)
7. Create new Flask app (next recitation)
8. Run the new app (`python first.py`)
9. Open a browser and see (http://localhost:5000)

Start with Virtual ENV

1. Install Virtual Environment first

- pip install virtualenv

Start with Virtual ENV

** For windows

1-1. Download get-pip.py file and run it

- https://bootstrap.pypa.io/get-pip.py

1-2. Setup pip and then virtual env

- pip install --upgrade pip setuptools
- pip install virtualenv

Start with Virtual ENV

2. Create new environment with Python 3.6

- `python -m virtualenv cs1520_flask -p /Library/Frameworks/Python.framework/Versions/3.6/bin/python`
  (example for my setting)

Start with Virtual ENV

You can get your own python interpreter location by this

- OS X / Linux:
  - Open a terminal and type this and hit enter
  - `which python`
  - `or`
  - `which python3` `or` `which python3.6`

Start with Virtual ENV

You can get your own python interpreter location by this

- Windows:
  - Open a terminal and get into a python that you can use as with version 3.6. and type this and run line-by-line
    - `import sys`
    - `sys.executable`

    then, something like this will be on your screen

Start with Virtual ENV

```
>>> import sys
>>> sys.executable
'C:\\Python26\\python.exe'
```

- and you can use ‘C:\\Python26\\...’ with -p argument in Virtual env command.

Start with Virtual ENV

3. Get into the folder and activate environment

- cd/cs1520_flask
- source bin/activate (OS X/Linux)
- script/activate.bat (Windows)

4. Now, we are in the new environment

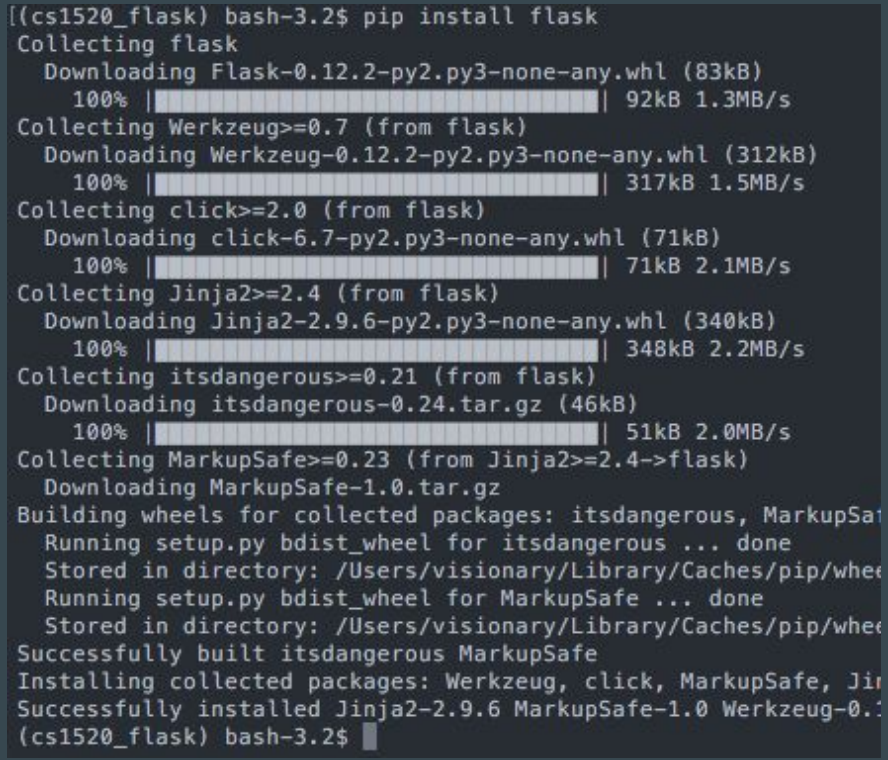
```
bash-3.2$
bash-3.2$ source bin/activate
(cs1520_flask) bash-3.2$
```

Get Flask

5. Install Flask

- pip install flask

Note that this installation is only on current environment. Not your system's Python.



Run First Application

- save it as first.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

- run >> python first.py

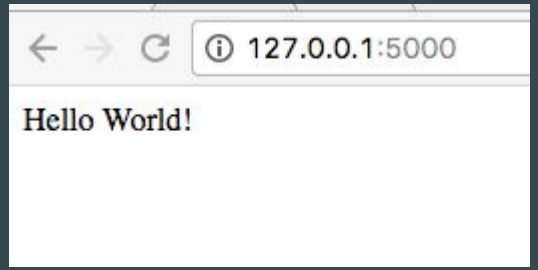
Run First Application

- save it as first.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```



- run >> python first.py

```
(cs1520_flask) bash-3.2$ python first.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Questions?