module_assignment

January 27, 2018

In [5]:
```
class Authentication:
    def __init__(self):
        # instantiate an instance variable
        self.user_dict = {}
    def register_user(self, uname, passwd):
        if uname in self.user_dict:
            print("Username exists! Try a new one.")
            return False
        else:
            self.user_dict[uname] = passwd
            print("Registration successful" )
            return True
```

In [6]:
```
def data_entry(auth):
    # registering 3 users
    auth.register_user('jdoe', '$234^%$') # Jane Doe
    auth.register_user('sburry', '456@#&^') # Sam Burry
    auth.register_user('mfisc', '%6&#$@#') # Mike Fischer
    auth.register_user('nhay', 'ildfu45') # Nicky Hailey
    auth.register_user('bobama', 'klj43509jafd') # Barack Obama
    auth.register_user('bgates', '^&%kjsfd934@#$') # Bill Gates
    auth.register_user('mcuban', '9&4rl#nsf') # Mark Cuban

# Main program
auth = Authentication()
data_entry(auth)
```

```
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
```

In [7]:
```
# Question 1:
# Inherit the Authentication class to create a new child class called AuthenticationIO
```

```
        # add a new method called write_info()
        # which writes all the usernames and passwords to a CSV file (never recommended in rea
        # the filename should be "userinfo.csv"
        # It should have 2 columns: Username, Password
        # After writing to file successfully, print "Write to file successful!
        import csv

        class AuthenticationIOcsv(Authentication):
            def write_info(self):
                fname='userinfo.csv'
                # fill in your code
                # remove pass after your implementation is complete
                # Python's "with" will close the file.
                with open(fname, 'w') as f:
                    f.write("Username,Password\n")
                    for u in self.user_dict:
                        f.write("{},{}\n".format(u, self.user_dict[u]))
                        print("Write to file successful!")
        # Main Program
        auth = AuthenticationIOcsv()
        data_entry(auth)
        # writing to file
        auth.write_info()
```

```
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Write to file successful!
Write to file successful!
Write to file successful!
Write to file successful!
Write to file successful!
Write to file successful!
Write to file successful!
```

```
In [10]: # Question 2:
         # Read and print the contents of the CSV file "userinfo.csv"

         fname = "userinfo.csv"

         with open(fname) as f:
             # fill in your code
             # remove pass after your implementation is complete
             print(f.readlines())
```

```
['Username,Password\n', 'jdoe,$234^%$\n', 'sburry,456@#&^\n', 'mfisc,%6&#$@#\n', 'nhay,ildfu45\n
```

```
In [12]: # Question 3:
         # Inherit the Authentication class to create a new child class called AuthenticationIO
         # add a new method called write_info()
         # which writes all the usernames and passwords to a json file (never recommended in re
         # the filename should be "userinfo.json"
         # It should have Username as the key, Password as the value
         # After writing to file successfully, print "Write to file successful!"
         import json
         from pprint import pprint
         class AuthenticationIOjson(Authentication):

             def write_info(self):
                 fname = 'userinfo.json'
                 items_str = json.dumps(self.user_dict)
                 with open(fname, 'w') as f:
                     f.write(items_str)

                     # Main Program
         auth = AuthenticationIOjson()
         data_entry(auth)
         # writing to file
         auth.write_info()
```

```
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
Registration successful
```

```
In [13]: # Question 4:
         # Read and print the contents of the json file "userinfo.json"
         import json
         fname = 'userinfo.json'
         with open(fname, 'r') as f:
             info = json.load(f)
             pprint(info)
```

```
{'bgates': '^&%kjsfd934@#$',
 'bobama': 'klj43509jafd',
 'jdoe': '$234^%$',
 'mcuban': '9&4rl#nsf',
 'mfisc': '%6&#$@#',
 'nhay': 'ildfu45',
```

```
 'sburry': '456@#&^'}
```

```
In [15]: # Question 5
         # Given a string,strip all the white spaces on both sides of the string
         # Then, capitalize first letter of all words
         # if first character is not an alphabet leave it as it is
         # and lower case the rest of the characters in each word
         tweet = "      Its a happy day in bloomington #happy"

         # strip all the white spaces and split the string to individual words
         words = tweet.strip('').split()
         formatted_words = []
         for w in words:
             # check if the first character is an alphabet
             if w.isalpha():
                 formatted_words.append(w.capitalize())
             else:
                 formatted_words.append(w)

         # join all the words in formatter_words to create a single string
         formatted_tweet = ' '.join(w for w in formatted_words)

         print(formatted_tweet)
```

```
Its A Happy Day In Bloomington #happy
```

```
In [1]: import re
        addr = "2706 10th Street, Bloomington, IN - 47408"
        zip_expr = r'\d\d\d\d\d'
        street_expr = r'^\d+ \d+\w* (Street|St|st), [A-Z]\w*'
        state_expr = r'[A-Z][A-Z]'

        zip_regex = re.compile(zip_expr)
        zip_match = zip_regex.search(addr)
        if zip_match:
            print("Found the address {}".format(zip_match.group()))
        else:
            print("No match!")

        state_regex = re.compile(state_expr)
        state_match = state_regex.search(addr)
        if state_match:
            print("Found the state {}".format(state_match.group()))
        else:
            print("No match!")
```

```
        street_regex = re.compile(street_expr)
        street_match = street_regex.search(addr)
        if street_match:
            print("Found the street {}".format(street_match.group()))
        else:
            print("No match!")
```

```
Found the address 47408
Found the state IN
Found the street 2706 10th Street, Bloomington
```