Week 1: In Your Interface

Script with updated class and test calls:

```python
# Patrick Johnson          3/16/2020 #
# SWDV 630 3W 20/SP2          Week 1 #
#####################################
# Expansion of Teams class provided in assignment:
# 1) Add the __contains__ protocol and show whether or not  'Tim' and
#     'Sam' are part of our team.
# 2) Add the __iter__ protocol and show how you can print each member
#     of the classmates object.
# 3) Determine if the class classmates implements the __len__ method.

class Teams:
    def __init__(self, members):
        self.__myTeam = members

    def __len__(self):
        return len(self.__myTeam)

    def __contains__(self, member):
        return member in self.__myTeam

    def __iter__(self):
        self.__i = 0
        return self

    def __next__(self):
        if self.__i < len(self.__myTeam):
            nextItem = self.__myTeam[self.__i]
            self.__i += 1
            return nextItem
        else:
            raise StopIteration

def main():
    classmates = Teams(['John', 'Steve', 'Tim'])

    print('1. Tim in classmates:','Tim' in classmates)
    print('   Sam in classmates:','Sam' in classmates)

    print('2. Iterate over classmates:')
    for classmate in classmates:
        print(classmate)

    print('3. Length of classmates:',len(classmates))

main()
```

Output:

```
1. Tim in classmates: True
   Sam in classmates: False
2. Iterate over classmates:
John
Steve
Tim
3. Length of classmates: 3
```

Responses:

1. Add the `__contains__` protocol and show whether or not 'Tim' and 'Sam' are part of our team.
   This is demonstrated in the code, with the output that Tim is on the team but Sam is not.
2. Add the `__iter__` protocol and show how you can print each member of the classmates object.
   Also demonstrated in the output, the for loop printed out the three names.
3. Determine if the class `classmates` implements the `__len__` method.
   Well, `classmates` is an object, but yes, as an instance of the `Teams` class this was provided in the initial code and is shown working in the output.
4. Explain the difference between interfaces and implementation.
   An interface is a description of the functionality of an object, such as the parameters, returns, and effects for a function, the attributes and method signatures for an OOP class, or the arguments and output for a program. The implementation is the actual code that process the inputs and generates the output. In order to use an object, the developer can evaluate the interface to determine if the object will work and how it will need to be used, without worrying about the implementation. This also allows the implementation to be modified and improved without affecting usage so long as the interface is maintained.
5. Using both visual and written descriptions, think through the interface-implementation of a large scale storage system. In many systems today, we have the ability to store information from a single application to a variety of storage devices - local storage (hard drive, usb), the cloud and/or some new medium in the future. How would you design an interface structure such that all of the possible implementations could store data effectively.
   It would need to cover the basics for presenting information, such as listing directory contents, total and available capacity, and file information. Because some of the media may have long latencies, it should be able to provide file details in batches which can be filtered if requested, such as by file extension or modified date. The interface should encourage caching by the client, supporting multiple levels of file locking and change notification, and it should allow block-level updates. There will need to be a confirmation for writes and updates, and the interface should support grouping multiple actions together into an atomic set so that if one fails the rest are rolled back. The interface shouldn't offer medium-specific information, such as hard drive rotation speed that would be irrelevant on other media and is of little value to most users. By meeting the minimum storage interface requirements the interface can be implemented as an abstraction to all manners of storage without those accessing the interface needing to worry about the details.

GitHub Link:

https://github.com/Maryville-SWDV-630/ip-1-PJohnson9