ECE695DL: Homework 1

Tulasi Haritsa

February 2021

Solution by: Tulasi Haritsa

```
import random
import string
import os


# Function to clear the terminal output
def clear():
    os.system('cls')


# Base class
# Response to Question 1: Class creation | name_sort_type added later to complete
question 5
class People:
    def __init__(self, first_names, middle_names, last_names,
    name_sort_type='first_name_first'):
        self.first_names = first_names
        self.middle_names = middle_names
        self.last_names = last_names
        self.name_sort_type = name_sort_type

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

    def __next__(self):
        iter_var = self.iter_var
        limit = len(self.first_names)
        if iter_var >= limit:
            raise StopIteration
        if self.name_sort_type == 'first_name_first':
```

```
            text = self.first_names[iter_var] + ' ' + self.middle_names[iter_var] +
            ' ' + self.last_names[iter_var]
            print(text, end="")
        elif self.name_sort_type == 'last_name_first':
            text = self.last_names[iter_var] + ' ' + self.first_names[iter_var] + ' '
            + self.middle_names[iter_var]
            print(text, end="")
        elif self.name_sort_type == 'last_name_with_comma_first':
            text = self.last_names[iter_var] + ', ' + self.first_names[iter_var] +
            ' ' + self.middle_names[iter_var]
            print(text, end="")
        else:
            raise Exception('Invalid name sort type')
        self.iter_var += 1
        return iter_var

    def set_sort_type(self, name_sort_type='first_name_first'):
        if name_sort_type not in ['first_name_first', 'last_name_first',
        'last_name_with_comma_first']:
            raise Exception('Invalid name sort type')
        self.name_sort_type = name_sort_type

    def __call__(self):
        sorted_last_names = self.last_names.copy()
        sorted_last_names.sort()
        return sorted_last_names


# Subclass - inherits the parent class People
class PeopleWithMoney(People):
    def __init__(self, first_names, middle_names, last_names, wealth):
        super().__init__(first_names, middle_names, last_names)
        self.wealth = wealth

    def __iter__(self):
        super().__iter__()
        return self

    def __next__(self):
        var = super().__next__()
        print(' ', self.wealth[var])

    def __call__(self):
        dict_create = {}
        for i in range(len(self.wealth)):
            dict_create[str(self.wealth[i])] = i
```

```
        sorted_wealth = self.wealth.copy()
        sorted_wealth.sort()
        for i in sorted_wealth:
            person = dict_create.get(str(i))
            print()
            print(self.first_names[person], self.middle_names[person],
            self.last_names[person],
                  self.wealth[person], end="")


# Response to Question 2: Creating arrays of names
random.seed(0)  # Setting the random generator seed to 0 for reproducibility
name_len = 5
num_names = 10
first_names_arr = []
middle_names_arr = []
last_names_arr = []

for m in range(num_names):
    first_names_arr.append(''.join(random.choice(string.ascii_lowercase) for i in
    range(name_len)))
    middle_names_arr.append(''.join(random.choice(string.ascii_lowercase) for j in
    range(name_len)))
    last_names_arr.append(''.join(random.choice(string.ascii_lowercase) for k in
    range(name_len)))

# Response to Question 3: Creating an instance of class People
people_obj = People(first_names_arr, middle_names_arr, last_names_arr)

# Response to Question 4: Creating an iterator to run through the names
# Commented out to prevent duplicate outputs
# i = iter(people_obj)
# for j in range(num_names):
#    next(i)
#    print()
# print()
# clear()

# Response to Question 5: Displaying the various types of name sorting
for sort_type in ['first_name_first', 'last_name_first',
'last_name_with_comma_first']:
    people_obj.set_sort_type(sort_type)
    i = iter(people_obj)
    for j in range(num_names):
        next(i)
        print()
```

```
    print()

# Response to Question 6: Printing the last names - sorted in ascending
alphabetical order
print('\n'.join(people_obj()))
print()

# Response to Question 7: Printing the last names - sorted in ascending
alphabetical order
wealth_gen = random.sample(range(1000), num_names)
peopleWithMoney_obj = PeopleWithMoney(first_names_arr, middle_names_arr,
last_names_arr, wealth_gen)
i = iter(peopleWithMoney_obj)
for j in range(num_names):
    next(i)

peopleWithMoney_obj()
```