CS 515 Homework 7

April 2022

Homework 7: The Date Class
Due: 4/13/2022 at 11:59pm

Introduction

For this homework, you will expanding on the Date class you started in lab. The file you will be submitting is called **hw7.py**, so I recommend copying over the code you already wrote as a starting point, though you will have to change some things.

As a recap, the Date class has three attributes:

- An attribute holding the month: self.month
- An attribute holding the day: self.day
- An attribute holding the year: self.year

The Date class should have an __init__ method and a __str__ method. As we've discussed in class,the double underscores before and after these method names indicate that these methods are special methods that Python knows to look for. In the case of __init__, this is the method that Python looks for when making a new Date object. In the case of __str__, this is the method that Python looks for when it needs to represent the object as a string.

Now, implement a few methods for the Date class from scratch. Be sure to add a docstring/method specification to each of the methods you write!

Please be sure to name your functions exactly as specified for TA grading.

DO NOT USE THE BUILT IN DATE CLASS FOR ANY OF THESE OPERATIONS!!!

Please submit this homework as a single file containing all functions, called hw7.py

1 Question 1: Verify the constructor and string methods work (5 points)

Implement the __init__(self, month, day, year) method and __str__(self) method. For __str__ method, it should construct and return a string with the month, day, and the year, formatted nicely to have exactly two digits places for the month and two digit places for the day. e.g.,

```
>>> d = Date(11, 4, 2021)
>>> str(d)
'11/04/2021'
```

2 Question 2: Create the equal operator (5 points)

__init__ and __str__ are not the only double underscore functions a class can overwrite. Do some research to find out how to make sure the == works for two dates. Two dates are equal if their month, day, and year are the same.

```
>>> d1 = Date(1, 1, 2000)
>>> d2 = Date(1, 1, 2000)
>>> d3 = Date(1, 1, 2001)
>>> d1 == d2
True
>>> d1 == d3
False
```

3 Question 3: Write a leap year function (10 points)

Add the method isLeapYear(self) to your Date class. The method should return True if the current object represents a leap year, return False otherwise. You may Google to check the definition of a leap year.

```
>>> d1 = Date(1,1,1900)
>>> d2 = Date(1,1,2020)
>>> d3 = Date(1,1,2000)
>>> d1.isLeapYear()
False
>>> d2.isLeapYear()
True
>>> d3.isLeapYear()
True
```

4 Question 4: Write a before function (10 points)

Add the method isBefore(self, d2) to your Date class. This method should return True if the calling object is a calendar date **before** the input named d2 (which will always be an object of type Date). If self and d2 represent the same day, this method should return False. Similarly, if self is after d2, this should return False.

```
>>> d1 = Date(1,1,2000)
>>> d2 = Date(1,1,2001)
>>> d1.isBefore(d2)
True
>>> d2.isBefore(d1)
False
>>> d1.isBefore(d1)
False
```

5 Question 5: Write a tomorrow function (10 points)

Add the method tomorrow(self) to your Date class. This method should **NOT RETURN ANYTHING!** Rather, it should **change** the calling object so that it represents one calendar day after the date it originally represented. This means that self.day will definitely change. What's more, self.month and self.year might change.

You may define the list DIM = [31,28,31,30,31,30,31,31,30,31,30,31] at the very top of your python file to help you determine how many days there are in any particular month (self.month).

```
>>> d = Date(12, 30, 2010)
>>> str(d)
12/30/2010
>>> d.tomorrow()
>>> str(d)
12/31/2010
>>> d = Date(2, 28, 2011)
>>> d.tomorrow()
>>> str(d)
03/01/2011
>>> d.tomorrow()
>>> str(d)
03/02/2011
```

6 Question 6: Write a day of the week function (20 points)

Write a function dayOfWeek(self) that gives the day of the week (Monday, Tuesday, Wednesday...) of the calling object. This function should **NOT** use any other external classes. It must be done by your own calculation using math. You can Google to look for the algorithm.

```
>>> d = Date(11, 4, 2021)
>>> d.dayOfWeek()
'Thursday'
```

7 Question 7: Create a days apart function (20 points)

Write a function daysApart(self, other) that gives the absolute value of the amount of days apart between the calling object and other.

```
>>> d1 = Date(1, 1, 2021)
>>> d2 = Date(1, 10, 2021)
>>> d1.daysApart(d2)
9
>>> d2.daysApart(d1)
9
```

8 Question 8: Create a Sub-Class QuantumDate (20 points)

A QuantumDate is a subclass of date with a few different properties from a normal Date. Please create a class QuantumDate in the same file and make it a subclass of Date. Use correct subclass design principles to make this subclass. A QuantumDate has the following differences from a normal Date. All other properties remain the same.

- When printed, it prints a randomly chosen month, day, and year.
- It is never a leap year, no matter what.
- There is no "tomorrow" of a QuantumDate, it is the same day as when it is created.
- It is always the day of the week after when it should be.
- QuantumDate has an additional method randomize(self) which shuffles the month, day, and year. The year must be within 10 years of the original year, the month and day have no restriction as long as they are valid.

Again, please submit this homework as a single file containing all functions, called hw7.py