stimated time

30-60 minutes

Level of difficulty

Easy/Medium

Objectives

- improving the student's skills in defining classes from scratch;
- defining and using instance variables;
- defining and using methods.

Scenario

We need a class able to count seconds. Easy? Not as much as you may think as we're going to have some specific expectations.

Read them carefully as the class you're about write will be used to launch rockets carrying international missions to Mars. It's a great responsibility. We're counting on you!

Your class will be called `Timer`. Its constructor accepts three arguments representing **hours** (a value from range [0..23] - we will be using the military time), **minutes** (from range [0..59]) and **seconds** (from range [0..59]).

Zero is the default value for all of the above parameters. There is no need to perform any validation checks.

The class itself should provide the following facilities:

- objects of the class should be "printable", i.e. they should be able to implicitly convert themselves into strings of the following form: "hh:mm:ss", with leading zeros added when any of the values is less than 10;
- the class should be equipped with parameterless methods called `next_second()` and `previous_second()`, incrementing the time stored inside objects by +1/-1 second respectively.

Use the following hints:

- all object's properties should be private;
- consider writing a separate function (not method!) to format the time string.

Complete the template we've provided in the editor. Run your code and check whether the output looks the same as ours.

Expected output

```
23:59:59
```

```
00:00:00
```

```
23:59:59
```

Code:

```
class Timer:
    def __init__( ??? ):
        #
        # Write code here
        #

    def __str__(self):
        #
        # Write code here
        #

    def next_second(self):
        #
        # Write code here
        #

    def prev_second(self):
        #
        # Write code here
        #


timer = Timer(23, 59, 59)
print(timer)
timer.next_second()
print(timer)
timer.prev_second()
print(timer)
```

Estimated time

30-60 minutes

Level of difficulty

Easy/Medium

Objectives

- improving the student's skills in defining classes from scratch;
- defining and using instance variables;
- defining and using methods.

Scenario

Your task is to implement a class called `Weeker`. Yes, your eyes don't deceive you - this name comes from the fact that objects of that class will be able to store and to manipulate days of a week.

The class constructor accepts one argument - a string. The string represents the name of the day of the week and the only acceptable values must come from the following set:

```
Mon Tue Wed Thu Fri Sat Sun
```

Invoking the constructor with an argument from outside this set should raise the `WeekDayError` exception (define it yourself; don't worry, we'll talk about the objective nature of exceptions soon). The class should provide the following facilities:

- objects of the class should be "printable", i.e. they should be able to implicitly convert themselves into strings of the same form as the constructor arguments;
- the class should be equipped with one-parameter methods called `add_days(n)` and `subtract_days(n)`, with **n** being an integer number and updating the day of week stored inside the object in the way reflecting the change of date by the indicated number of days, forward or backward.
- all object's properties should be private;

Complete the template we've provided in the editor and run your code and check whether your output looks the same as ours.

Expected output

```
Mon
```

```
Thu
```

```
Sun
```

```
Sorry, I can't serve your request.
```

Code:

```
class WeekDayError(Exception):
    pass


class Weeker:
    #
    # Write code here
    #

    def __init__(self, day):
        #
        # Write code here
        #

    def __str__(self):
        #
        # Write code here
        #

    def add_days(self, n):
        #
        # Write code here
        #

    def subtract_days(self, n):
        #
        # Write code here
        #


try:
    weekday = Weeker('Mon')
    print(weekday)
    weekday.add_days(15)
    print(weekday)
    weekday.subtract_days(23)
    print(weekday)
    weekday = Weeker('Monday')
except WeekDayError:
    print("Sorry, I can't serve your request.")
```

Estimated time

30-60 minutes

Level of difficulty

Easy/Medium

Objectives

- improving the student's skills in defining classes from scratch;
- defining and using instance variables;
- defining and using methods.

Scenario

Let's visit a very special place - a plane with the Cartesian coordinate system (you can learn more about this concept here: https://en.wikipedia.org/wiki/Cartesian_coordinate_system).

Each point located on the plane can be described as a pair of coordinates customarily called **x** and **y**. We expect that you are able to write a Python class which stores both coordinates as float numbers. Moreover, we want the objects of this class to evaluate the distances between any of the two points situated on the plane.

The task is rather easy if you employ the function named `hypot()` (available through the *math* module) which evaluates the length of the hypotenuse of a right triangle (more details here: https://en.wikipedia.org/wiki/Hypotenuse) and here: https://docs.python.org/3.7/library/math.html#trigonometric-functions.

This is how we imagine the class:

- it's called `Point`;
- its constructor accepts two arguments (**x** and **y** respectively), both default to zero;
- all the properties should be private;
- the class contains two parameterless methods called `getx()` and `gety()`, which return each of the two coordinates (the coordinates are stored privately, so they cannot be accessed directly from within the object);
- the class provides a method called `distance_from_xy(x,y)`, which calculates and returns the distance between the point stored inside the object and the other point given as a pair of floats;
- the class provides a method called `distance_from_point(point)`, which calculates the distance (like the previous method), but the other point's location is given as another `Point` class object;

Complete the template we've provided in the editor and run your code and check whether your output looks the same as ours.

Expected output

```
1.4142135623730951
```

```
1.4142135623730951
```

Code:

```
import math


class Point:
    def __init__(self, x=0.0, y=0.0):
        #
        # Write code here
        #

    def getx(self):
        #
        # Write code here
        #

    def gety(self):
        #
        # Write code here
        #

    def distance_from_xy(self, x, y):
        #
        # Write code here
        #

    def distance_from_point(self, point):
        #
        # Write code here
        #


point1 = Point(0, 0)
point2 = Point(1, 1)
print(point1.distance_from_point(point2))
print(point2.distance_from_xy(2, 0))
```

Estimated time

30-60 minutes

Level of difficulty

Medium

Objectives

- improving the student's skills in defining classes from scratch;
- using composition.

Scenario

Now we're going to embed the `Point` class (see Lab 3.4.1.14) inside another class. Also, we're going to put three points into one class, which will let us define a triangle. How can we do it?

The new class will be called `Triangle` and this is the list of our expectations:

- the constructor accepts three arguments - all of them are objects of the `Point` class;
- the points are stored inside the object as a private list;
- the class provides a parameterless method called `perimeter()`, which calculates the perimeter of the triangle described by the three points; the perimeter is a sum of all legs' lengths (we mention it for the record, although we are sure that you know it perfectly yourself.)

Complete the template we've provided in the editor. Run your code and check whether the evaluated perimeter is the same as ours.

Below you can copy the `Point` class code we used in the previous lab:

Check

Expected output

```
3.414213562373095
```

Code:

```
import math


class Point:
    #
    # The code copied from the previous lab.
    #


class Triangle:
    def __init__(self, vertice1, vertice2, vertice3):
        #
        # Write code here
        #

    def perimeter(self):
        #
        # Write code here
        #


triangle = Triangle(Point(0, 0), Point(1, 0), Point(0, 1))
print(triangle.perimeter())
```