Programming for Business Computing

In-class Practices: Classes

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Problem 1: is_earlier_than()

- Let's build a class **Time**:

```
class Time():

    # hour, minute, and second are integers
    # check the values of h, m, and s before input into the constructor
    def __init__(self, h, m, s):
        self.hour = h
        self.minute = m
        self.second = s

    def print_normally(self):
        print('%d:%d:%d' %(self.hour, self.minute, self.second), end = "")
```

只要呼叫class，此constructor就會自動被呼叫

object的(.)東西

此函式則需要利用dot來呼叫

```
t1 = Time(9, 10, 0)
t1.print_normally()
```

Problem 1: `is_earlier_than()`

- Please implement a member function **`is_earlier_then(self, t)`**:
  - Return true if and only if the invoking object's time is earlier than that of the input **`Time`** object **`t`**.

```
def is_earlier_than(self, t):
```

- An example program:

```
t1 = Time(12, 30, 40)
t2 = Time(13, 10, 5)
if t1.is_earlier_than(t2):
    print("t1 is earlier than t2")
else:
    print("t1 is not earlier than t2")
```

Problem 1: `is_earlier_than()`

- Given three time moments, find the earliest one.
- Input:
  - One line with nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i.
  - Separated by white spaces.
- Output:
  - The earliest time printed by **`print_normally()`**.
- Sample input/output:

Input:
```
14 0 0 15 4 13 13 8 2
```

Output:
```
13:8:2
```

Input:
```
14 0 0 12 4 13 15 8 2
```

Output:
```
12:4:13
```

Problem 2: `print_nicely()`

- Please implement a member function
  - Print out 08:09:06 if **`hour`** is 8, **`minute`** is 9, and **`second`** is 6.
  - You may define other member functions if helpful.
- An example program:

```
def print_nicely(self):
```

```
if t1.is_earlier_than(t2):
    t1.print_nicely()
else:
    t2.print_nicely()
```

Problem 2: `print_nicely()`

- Given three time moments, print out the earliest one nicely.
- Input:
  - One line with nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i.
  - Separated by white spaces.
- Output:
  - The nice format of the earliest time.
- Sample input/output:

Input:
```
14 0 0 15 4 13 13 8 2
```

Output:
```
13:08:02
```

Input:
```
4 0 0 2 4 13 5 8 2
```

Output:
```
02:04:13
```

Problem 3: display format

- Sometimes we want to print out a time string in the 12-hour format.
- Modify your **`Time`** to provide the two display formats.
  - Under the 12-hour format, print out a space and append "AM" or "PM" at the end.
  - The default format is 24 hours.
  - The values are still stored in one single system. Only the display format differs.
  - If we use one format for one time string, we should do that for **all** time strings.
  - A **static** member variable may help.

```
t1 = Time(14, 30, 0)
t2 = Time(14, 25, 5)

t1.print_nicely()
Time.set_hourIn12(False)
t2.print_nicely()
```

Problem 3: display format

- Given three time moments, print out the earliest one nicely in the given format.
- Input:
  - Line 1: nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i. Separated by white spaces.
  - Line 2: 12 or 24 meaning the display format.
- Output:
  - The nice format of the earliest time in the given format.

Problem 3: display format

- Sample input/output:

Input:
```
14 0 0 13 4 13 15 8 2
12
```

Output:
```
01:08:02 PM
```

Input:
```
4 0 0 2 4 13 5 8 2
24
```

Output:
```
02:04:13
```

Problem 4: Event

- Let's implement a class **Event**:

```
class Event():

    def __init__(self, name, t1, t2):

        self.name = name
        self.start = t1
        self.end = t2

    def print_nicely(self):

       # print the event in a nice way

    def set_name(self, n):

       # let self.name becomes n
```

```
name = 'PBC'
t1 = Time(9, 10, 0)
t2 = Time(12, 10, 0)
e1 = Event(name, t1, t2)
e1.print_nicely()

print()

name2 = 'PBC!!'
e1.set_name(name2)
e1.print_nicely()
```

Problem 5: shallow and deep copy

- Try the following two programs and explain the difference.

```
name1 = 'PBC'
t1 = Time(9, 10, 0)
t2 = Time(12, 10, 0)
e1 = Event(name1, t1, t2)
e1.print_nicely()
print()

e2 = e1
name2 = 'Calculus'
e2.set_name(name2)
e2.print_nicely()
print()

e1.print_nicely()
```

```
import copy

name1 = 'PBC'
t1 = Time(9, 10, 0)
t2 = Time(12, 10, 0)
e1 = Event(name1, t1, t2)
e1.print_nicely()
print()

e2 = copy.deepcopy(e2)
name2 = 'Calculus'
e2.set_name(name2)
e2.print_nicely()
print()

e1.print_nicely()
```

Problem 6: Schedule

- Try to create a class **Schedule** to store **Event** objects in a list.
  - Constructor: initialize an instance variable as the event list.
  - **add_event(self, e)**: add e into the event list.
  - **print_events(self)**: print the events.

```
class Schedule():

    def __init__(self):

        # implement this

    def add_event(self, e):

        # implement this

    def print_events(self):

        # implement this
```