Object Oriented Programming In Python

Answers for Homework Tasks: Week 1

Completed programs are available to download.

Homework Task 1: Date and Time *Date and time are quite complex. For example, a date can be a day number or a month number and a day number. A time can use a 12-hour or 24-hour clock and summer time or GMT. We may also want to compare two dates/time (before or after) or subtract one from the other (to get a time interval).*

- *Suggest a simple abstract interface (i.e. functions) that could be used to create dates and times and show them in different formats. Do not include every possible complication!*
- *Suggest how the date/time might be represented, behind your interface.*
- *Implement (some of) the interface as a module (i.e. in the from of NearlyPerson.py)*

A simple version handles Day, Month, Hour and Minute, but not years

- Hours and minutes a from 00:00 (midnight) to 23:59
- Formats: 19 March, 3:31pm or 19/03, 15:31hrs
- Assume that February always has 28 days and the year 365 days: a more complex version includes years and distinguishes leap years.

An issue is how to handle errors, since some dates do not exist. For example 31 June, 25:01hrs or 32 March, 13:62pm. Our approach is:

- Always create a data/time when asked, whatever the input
- Provide a function that must be used to check if a new date/time is valid. If it is not valid, an error message is returned and the other functions may fail. If the error message is empty, the date/time is valid and the other functions will work ok.

Method/Function	Description
dt = newDatetime(mm, dd, hh, nn) dt = newDate(mm,dd)	mm: month number 1 to 12 dd: day number: 1 to 31, depending on month hh: 0 to 23; nn: 0 to 59
errorMsg(dt)	Returns an empty string if the date/time is valid; otherwise one of the following errors: “Month mm does not exist”, “No day DD in month mmm”, “Hour HH does not exist”, “Minute NN does not exist”
displayShort(dt)	Display as ‘19/03, 15:31hrs’; hours and minutes are not shown if no time was entered
displayLong(dt)	Display as ‘19 March, 3:31pm’; hours and minutes are not shown if no time was entered
interval = between(dt1, dt2)	Returns the interval between two date time: as a string: DD:HH:MM after/befoe

The difference between two dates is an ‘interval’, thus 19/3, 12:00hrs is 2:03:15 before 21 March, 3.15pm. The interval (underlined in the previous sentence) could be a separate abstraction, but we have just used a string.

Including the 12 hour clock time (with am/pm) in on of the formats adds a lot of work as this is much more complex than often realised: see https://en.wikipedia.org/wiki/12-hour_clock

Representation: the representation must handle both valid and invalid dates and also whether a time or just a date was entered. On design is we use a tuple:

- Invalid date/time: (False, *errmsg string*)
- Valid date/time: (True, (day of year – 0 to 364, minute of day 0 to 1439))
- Valid date only: (True, (day of year – 0 to 364, -1))

Whatever representation is used, we must be able to convert between minutes from start of year and month/day/hour/minute.

Implementation

An example implementation is available: DateTime-outline.py just has the interface; DateTime.py is completed (and also includes test cases).

Homework Task 2: Functional Decomposition

Your students have learnt how to define functions but some struggle to choose which functions will be helpful.

- *Suggest two forms of exercise that could help them practice using functional decomposition.*
- *Given an example exercise in at least one of these forms*

Types of Exercise

On this course, we try to use the read, modify, create paradigm. In this approach, students read a given program, made small modifications and then use their understanding to create new programs.

We can adapt this approach to exercises that aim to encourage use of functional decomposition.

Idea 1: Given a program using functions, describe how the functions are used. One way is to draw a call hierarchy (a tree – also called a call graph), showing which function is called from which. See https://en.wikipedia.org/wiki/Call_graph and also http://pycallgraph.slowchop.com/ (not tried). (Variation: provide two programs that behave in the same way but use a different decomposition.)

Idea 2: Give a problem description, and ask students to invent a list of functions and then write (only) a main program to solve the problem. Each function must have a description, so that other students could write each function without having seen the original problem. (Variation: actually write the functions. Hard to set up in practice.)

Idea 3: Give a problem description and set of useful functions. The task is to use the provided functions to solve the problem. (Variation: give the call graph.)

Example: write a program for two players to play Os and Xs. The program draws the board, gets the next position (checking it is not taken) until a line is made or the board is full.

Two variants are provided:

1. Contains the main program and the functions, but the functions are not implemented. This version could be an answer to an 'idea 2' exercise, or to set an 'idea 1' exercise.
2. Contains the complete implementation and could be used to set an 'idea 3' exercise (by removing the main and some other functions).