CIS 210, Fall 2016
Introduction to Computer Science

Main Menu

- Class home page
- Piazza
- How to Succeed
- Schedule
- Assignments
- References
- Exams

Assignment 3, part 1: Printing a calendar

This assignment is due at 5pm on Friday, October 14. Use Canvas to turn calpr.py in electronically.

Purpose

Sometimes we can't write a single loop that processes a whole collection. We may need to treat the first and/or the last elements separately from those in the middle. Such algorithms are often called “fencepost algorithms”, because there is a first fencepost, then several full sections of fence separated by fenceposts, then a last fencepost. This project is to think through a fencepost algorithm: printing the first (partial) week of a month, then the full weeks, then the last (partial) week.

Pair Assignment

You are encouraged to use *pair programming* to complete this assignment. Work together with one classmate.

Before writing code at the computer, you should work together and independently on the design. Each of you should be able to clearly explain how the program or a part of the program will work. When you are convinced that you both understand how the code will work, then and only then are you ready to write the code.

Printing a calendar

Calendars are traditionally organized into weeks, beginning (at least in the U.S.) on Sunday. The first and last week may be partial, as in October 2013:

```
Su Mo Tu We Th Fr Sa
      1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Requirements

Your program will take two integers, a month number (1-12) and a year number (1-2100) from the command line. It will then print a calendar in (exactly) the format shown above. For example:

```
> python3 calpr.py 11 2013
 Su Mo Tu We Th Fr Sa
                1  2
  3  4  5  6  7  8  9
 10 11 12 13 14 15 16
 17 18 19 20 21 22 23
 24 25 26 27 28 29 30
```

We will use the datetime module of the Python library to determine which day of the week a month starts on. We will not use the calendar module ... because it has a function to completely lay out the calendar, and the point is for us to write that ourselves.

To avoid additional complexity, we will treat February as always having 28 days, even in leap years, and all months having their current lengths. (We are ignoring alternative and earlier versions of the calendar, and some historical adjustments. For example, in 1712 February had 30 days, but we will treat it as if it had 28.)

Notes and hints on the calendar printer

I have provided quite a bit of starter code; your task is to figure out the logic of breaking a month down into a starting week, full weeks in the middle, and (possibly) a partial ending week. I have provided some pseudocode to help, but it will take careful thinking and design (and, almost certainly, a few rounds of testing and revision) to turn it into code that works correctly.

The initial loop that I have provided demonstrates how to print some parts of a line without moving to a new line. This involves using a “keyword argument” `end=""`. It also uses the `format` function to make each day number print right-justified in a field of exactly 3 characters.

Please name your program file `calpr.py`. Consistent naming makes life a little easier for your grader, and helps him return graded homeworks to you faster. Follow the format above precisely.

Grading

35 points possible

- 15: Program runs and consistently produces correct output for any month and year from January 1800 to December 2525.
- 10: Follows CIS 210 coding guidelines, including author identification and header .
- 10: Clarity. The program should not only be consistent with the requirements and approach described here, but it should be very easy to read the program and verify its consistency with the spec.