Assignment 4: Sept 14 - Oct 4

Hand-in deadline: Sunday, Oct 4

Note: This assignment will be peer-reviewed after the hand-in deadline, both for INF3331 and INF4331 students. More information about the peer review process will be published on the lecture on the 22. September.

Important: The scripts of this assignment should be saved in a subdirectory named `week4` of your INF3331 github repository.

4.1 Retrieve web page (2 pt)

It is very much recommended that you read all the assignments before you start. Many of the assignment are linked in such a way that good planing can drastically reduce the amount of work used to solve this assignment.

Write a Python function that given a supplied URL returns a string with the HTML content. For this task you have two options. In either case, it is assumed that you read the documentation to operate the module(s).

Alternative 1

Use `cURL` from command line. To retrieve prompt content in Python, you can use the following command:

```
from subprocess import Popen, PIPE
stdout_string = Popen(["prog", "arg1", "..."], stdout=PIPE).communicate()[0]
```

On Linux the cURL is already installed, while on Windows and Mac the program can be installed from

`http://curl.haxx.se/download.html`

Alternative 2

Use the native python module `urllib`

4.2 Find link to location (5 pt)

On the following page you will find `yr.no`'s overview page for weather stations in Norway:

`http://fil.nrk.no/yr/viktigestader/noreg.txt`

Write a function that provided a valid location name then the returns a list of (English) URLs to XML files that matches name of said location. It should have the following properties:

- First attempt to return all with matching `Stadnamn`.
- If no such match exists, retry replacing `Stadnamn` with `Kommune`.

- Lastly, try `Fylke`.
- If empty string is provided, return links for all entries in list.
- If no match, return an empty list.

Strip list such that only unique entries in the list exists. Your code should support wildcards `*`. For example the string `"Idd*"` should match both `Idd` and `Idd kirke`

Note that it will be assumed that the program fetches the file `noreg.txt` from the internet using the tool created in 4.1.

4.3 Retrieve weather information (5 pt)

Following the links provided from the `noreg.txt` document to XML weather data provided by `yr.no`. The data contains a tag group `<forecast>` and subgroup `<tabular>`. The tabular data are 6 hour weather summary going back in time.

Create a function that retrieve weather information using regular expression.

The information should include name of place, weather summary (`symbol name`), amount of rain (`precipitation value`), wind speed (`winSpeed mps`), temperature (`temperature value`), and a time stamp from (`time from`) and to (`time to`).

4.4 Buffer all internet activity (5 pt)

To ensure that we do not overload `yr.no`'s servers, we want to minimize the amount of online activities. This can be done by creating a lazy evaluation class. To illustrate what and how lazy evaluation works, observed the following example:

```
class Lazy:

  def __init__(self, func):
    self.func = func
    self.buffer = {}

  def __call__(self, arg):
    if arg in self.buffer:
      return self.buffer[arg]
    retval = self.func(arg)
    self.buffer[arg] = retval
    return retval
```

In practice it can work as follows:

```
>>> def func(x):
...    print "calculating", x
```

```
...    return x+1

>>> lazyfunc = Lazy(func)

>>> print lazyfunc(4)
calculating 4
5
>>> print lazyfunc(4)
5
>>> print lazyfunc(5)
calculating 5
6
```

In other words, the function only evaluates if the argument is unique. Otherwise it uses a buffered result.

Expand this example to work on the your code such that all online retrievals are buffered. Also, if the amount of time since last retrieval is long enough for a new (six hour) time period to kick in, the current buffer becomes obsolete and a new period should be retrieved.

The buffering should be saved to disk (using either `open` or `cPickle`). Restarting the program should not have an effect on the buffering result.

Include an extra optional time stamp argument in the constructor. Uses this instead of system time if provided. This will be used for testing in 4.6.

4.5 Create weather forecast (5 pt)

Create a function that takes three arguments: `place`, `hour` and `minute`. From this it will print out the forecast at all places that matches `place` using 4.2 at the next occurrence of the time `hour:minute`. In other words, if `13:30` is provided and the time is `12:30`, then return the time period `12:00-18:00` Today. However, if the time is `14:30`, return the same time period, but for Tomorrow.

The function should return a fitting summary of the weather using the information available for each location found. For example:

```
>>> print weather_update("Hannestad", 13, 0)
10.09.15 13:00
Hannestad: Partly cloudy, rain:0 mm, wind:0.9 mps, temp:8 deg C
```

One line per location.

Hint

Use the python module `time` to convert between Epoch time (number of seconds from 1.1.1950) and `struct_time`, (time on a human readable format). Epoch time is often much easier to perform calculate with.

4.6 Testing the code (5 pt)

Write appropriate tests for 4.1 through 4.5. Depending on the problem choose either Pydoc-test or Py.test. At least one of each test must be included.

4.1 Download the page for reference:

`http://www.islostarepeat.com/`

Check if your program creates the same document.

4.2 Check if `Hannestad` creates the link

`http://www.yr.no/place/Norway/Østfold/Sarpsborg/Hannestad/forecast.xml`

4.3 Ensure that temperature in Hannestad now is a valid numerical value between -50 and 50.

4.4 Create a dummy function that prints out a message before returning, much like the example in 4.4. Use the print out to confirm that the buffering works.

4.4 Use the time stamp argument in 4.4 to test if the files expiration works through the following two tests. Create a dummy file with expired time stamp and ensure that it is replaced. And opposite, create a dummy file with unexpired time stamp and ensure that it isn't replaced.

4.5 Ensure (again) that the temperature in Hannestad at (next) 13:00 is valid between -50 and 50.

4.7 Extreme places in Norway (3 pt)

Write a script, using your own tools that finds Norways hottest and coldest location at (next) 13:00.

Notes

Only return the first 100 entries, if the number of results is too high. Yr.no do not mind that we use their information, but would have licence that specifies that we should not over use their bandwidth.

Total points: 30