121COM: Introduction to Computing

Academic Year 2015/16

LabSheet 7

For use in labs the week beginning Monday 9th November 2015

Author: Dr Matthew England (Coventry University)

C Lab Exercises 1 - Exception Management

Q1 Sketch Flow Charts for `Distribute2.py`, `Distribute3.py` and `Distribute7.py` given in Worksheet 7.

Q2 Write a function `formattedDate`. This should take three integers as input which represent the day, month and year and print the date with month in words and appropriate English suffix on the day. E.g.

```
>>> formattedDate(24, 7, 1965)
"24th July 1965"
>>> formattedDate(31, 8, 2015)
"31st August 2015"
```

The function should raise appropriate errors if the input is not suitable. For example:

```
>>> formattedDate(31, 4, 1965)
Value Error:  In month 4 the days are 1-30
```

You do not need to check for leap years.

Q3 Write code to calculate the Ackermann Functions from Week 5 Slides A. Use a dictionary in the main namespace to store and use previously computed values. You may done this as a LabSheet6 extension.

Add a line to the start of your function with `print("Hello")` and then run the code for $(m, n) = (2, 2)$. The `Hello`s will be printed too quickly to count. Instead, redirect the output to a text file. How many lines the text file has shows how often the function was called. See how this increases with m and n. How high can you go before you reach the maximum recursion depth?

I Lab Exercises 2 - English Generation

The `try-except` syntax can be a useful tool for debugging: Identify the part of your code causing the error; make this the code block following `try` and have the code following `except` print the state of the program (i.e. relevant values) at the point the error occurs. This may help identify the problem.

Q5 Download the text file `OliverTwist.txt`. This is the book Oliver Twist by Charles Dickens as a text file. The book is out of copyright and so able to be distributed this way. It was obtained from `http://www.textfiles.com/etext/` where many other such books can be found.

The aim of this question is to have Python generate random sentences in English. The basic steps are below. Hint: start with just the first few lines of the book as a test.

(a) Open the file, read the contents into a string and then close it. Remember that you should use a `try-finally` statement to ensure the file closes properly.

(b) Split the string into words.

(c) Create a dictionary whose keys are the words; and values a list of those words that follow the key word in the text.

(d) Create a random English sentence by: selecting a random word to start with; then adding on one of the words which follow it in the text; and so on for a set amount of words.

Once you have this working improve the code by:

- Having the process continue until it reaches a word ending with punctuation synonymous with the end of the sentence (i.e. one of the characters `.?!`)
- Having the process always start with a word which begins with a capital letter.

Extensions: The output can be improved further by doing the following:

- Remove the new line characters.
- Remove the small-print at the start of the book (the first 251 lines)
- Remove the chapter headings - identified as words all in capitals.

Can you suggest any other ways to improve the output (i.e. get output closer to real English sentences)?

Try your code with a different book (find one from the website). What happens if you combine books?

Extended Task

Look at the code you wrote for the APIs we used previously: The Music APIs in Week 2; Quandl in Week 3; Crime Statistics in Week 6. Edit your code to build-in appropriate Exception Management. Your code should try to catch errors you can anticipate (e.g. one relating to a song not found on Spotify or an incorrect database code in Quandl). For those errors that should rise to the user create meaningful error messages. Aim for systems that can be used easily by someone not accustomed to Python programming.

Build new functions to access an API you have not met before. Try to find one that gives data is json format so you can use the same approach as the other tasks. The following all offered a free json service at the time of writing (October 2015): Some possibilities:

- The Guardian Newspaper: `http://open-platform.theguardian.com/explore/`
- Twitter: `https://dev.twitter.com/rest/public`
- Google Books: `https://developers.google.com/books/?hl=en`
- Open Weather: `http://openweathermap.org/api`

You can play with any other suitable API.