Michael Sanchez

CPS 3320: Python Programming

Robert Domanski

4/8/20

Project Assignment #2 Write Up

**Central Purpose Pillow**: aka PIL (Fork), a library based on the PIL, Python Imaging Library. Pillow adds image processing to Python. Pillow has support for various photo file formats/extensions. From the Documentation Overview, some of the few possible uses include Image Archives, Image Display, and Image Processing. Image Archiving includes thumbnail creation, file format conversion, print images and etc. Image Display includes TK PhotoImage, BitmapImage, and Windows DIB interface. Image processing covers point operations, filtering, color space conversions, image resizing, rotation, arbitrary affine transforms and histogram method to pull some statistics out of an image.

**Describe Useful or Interesting ways to use(2 practical examples)**: Pillow’s core focus is around image processing/manipulation. A quick Disclaimer, I’m not an expert in photo editing but photoshop (& other Photo editors) may be superior for photo editing for those that have the “Skills & Practice”. This brings me to the first useful/interesting way to use Pillow, a simple Thumbnail generator. It would work by using one image as the “Background”, merging at least one other image on top and finally support text being added in. Of course for it to be useful to the user, it would ask for the images and text to be used for the thumbnail. Another use for Pillow might also involve file conversion, of course the code for converting to simple jpeg exists but it could be made better such as allowing the user to continually enter files to be converted (recursion) or enter files as a list to be converted(loops) and also allow for images to be manipulated before being save as a new file.

**Overview of Functions**: Pillow includes various modules and functions, to keep this as short as possible I will be going over the functions shown in Pillow’s Documentation tutorial which is linked down below. Pillow includes functions such as an *open(), save()* and *show()* to load, save and show images, respectively. Functions to gather image attributes from *format* (Image source), *size* (2-tuple containing width and height), *mode*(numbers/names of bands in image & pixel type and depth). Functions to edit/manipulate photos, *resize(), rotate(), transpose(), convert(), point(), paste() and filter().* There are also many more functions and few other modules included in the Pillow Library.

**Documentation Link**: <https://pillow.readthedocs.io/en/stable/index.html>

**Arrow**: Arrow is a python library that aims to create a more user-friendly option to dealing with time related tasks. It’s core purpose it to deal with the many problems/complications faced when dealing with python’s standard library’s time-related functions. Such Complications include, too many modules & data types, and time zones/timestamp conversions being unpleasant. Arrow boasts features such as Time zone Awareness, Formats and Parses Strings automatically, shift method, and humanizes time.

**Uses**: Arrow is focused on making the process of using time-related modules & functions to be more user friendly. So the best use for the Arrow Library would be creating/enhancing programs that involve time related tasks and maybe even making use of the included humanize functions. One interesting use could be to create a list that contains the multiple time zones for a user inputted datetime, answering the question of when a certain thing happened in one time zone compared to other time zones. May not initially sound very useful but may be helpful with more external data, in which we could compare what two events occurred at the same time in various parts of the world or it could help create a timeline in which events are recorded in UTC. A second potential use could be to create a program that gathers dates/times from string and maybe compare to how many days/years/minutes/seconds have passed since then. Another use could compare how many days till a certain time/date arrives and return it through a humanize function.

**Overview of Functions**: Arrow includes a wide amount of functions. A *now()* function to get current time, *get()* function to get data from a timestamp, time zone aware datetime, directly or parsed from a string. Functions that return properties such as *datetime, timestamp, naïve, tzinfo, date, time* and *year, day, month, etc.* Functions *replace(),*replaces any value within the Arrow data even the time zone without affecting other data *and shift()*, moves one attribute forward or backwards. The *format()* turns a Arrow datetime into a formatted string. The *to()* function converts dates from one time zone to another. The *humanize()* function creates a string that compares one time relative to another, and presents it such as “In 1 hour”, “An hour ago” , “Just Now” and etc. The Last few functions cover ranges & spans such as *span(), floor(), ceil() and span\_range()* which return their respective unit of time measurement(s).

**Documentation Link**: <https://arrow.readthedocs.io/en/latest/>

**TextBlob**: TextBlob provides a simplified API for those who want to explore common natural language processing. TextBlob contains many features from noun phrase extraction to sentiment analysis to word inflection to word & phrase frequencies. TextBlob is built upon NLTK & Pattern and makes use of Google Translate & WordNet. Overall the core focus of TextBlob is providing Python users with an simplified API that makes using common text data processing easier to code & use.

**Uses**: TextBlob builds up on NLTK, so most text related programs would be a good way to use TextBlob. TextBlob may also be good for achieving certain more efficiently rather than programming it through the NLTK library. One Interesting use could be to use TextBlob’s sentiment analysis to compare polarity and subjectivity between two speeches, text, or writings. One good example of this, could be to use TextBlob’s sentiment analysis to analyze poems to see whether the poem is negative or positive, subjective or objective. This could lead to creating a dataset of poems created in a certain era and obtain the average polarity and subjectivity of the time frame. Another useful use for TextBlob could be using spellcheck to correct user input before it is used and/or making use of the translator to create general(translations from google can be so-so at times) UI text that matches user’s language regardless of the code/strings being written in English.

**Overview of Functions**: TextBlob contains numerous functions that deal with common text-processes. TextBlob uses a TextBlob object and works likes a python string using some familiar functions such as *upper(), find(),* substring syntax [:], comparisons and can be concatenated. *tags* and *noun\_phrases* functions gather Part-of-Speech tags and noun phrases respectively. The *sentiment* functions provides a tuple contains a TextBlob’s polarity and subjectivity scores. *words* and *sentences* functions can break a TextBlob into a list of the words/sentences. The *translate()* function can translate a string from one language to another. TextBlob also contains a *word\_counts[], words.count() and noun\_phrases.count()*  that gather the frequency of a word/noun\_phrases, by default it is not case sensitive. TextBlob contains many other functions, and most of these are gathered from TextBlob’s tutorial within its documentation, which is linked below.

**Documentation Link**: <https://textblob.readthedocs.io/en/dev/>