Python TIPS – Assignments

VERY SIMPLE EXERCISES

Numbers

1. Define a function max() that takes two numbers as arguments and returns the largest of them. Use the if-then-else construct available in Python. (It is true that Python has the max() function built in, but writing it yourself is nevertheless a good exercise.)
2. Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.

String

1. Define a function that computes the *length* of a given list or string. (It is true that Python has the len() function built in, but writing it yourself is nevertheless a good exercise.)
2. Write a function that takes a character (i.e. a string of length 1) and returns True if it is a vowel, False otherwise.
3. Write a function translate() that will translate a text into "rövarspråket" (Swedish for "robber's language"). That is, double every consonant and place an occurrence of "o" inbetween. For example, translate("this is fun") should return the string "tothohisos isos fofunon".

List

1. Define a function sum() and a function multiply() that sums and multiplies (respectively)all the numbers in a list of numbers. For example, sum([1, 2, 3, 4]) should return 10, multiply([1, 2, 3, 4]) should return 24.
2. Define a function reverse() that computes the reversal of a string. For example, reverse("I am testing") should return the string "gnitset ma I".
3. Define a function is_palindrome() that recognizes palindromes (i.e. words that look thesame written backwards). For example, is_palindrome("radar") should return True.
4. Write a function is_member() that takes a value (i.e. a number, string, etc) x and a list of values a, and returns True if x is a member of a, False otherwise. (Note that this is exactly what the in operator does, but for the sake of the exercise you should pretend Python did not have this operator.)
5. Define a function overlapping() that takes two lists and returns True if they have at leastone member in common, False otherwise. You may use your is_member() function, or the in operator, but for the sake of the exercise, you should (also) write it using two nested for-loops.
6. Define a function generate_n_chars() that takes an integer n and a character c and returnsa string, n characters long, consisting only of c:s. For example, generate_n_chars(5,"x") should return the string "xxxxx". (Python is unusual inthat you can actually write an expression 5 * "x" that will evaluate to "xxxxx". For the sake of the exercise, you should ignore that the problem can be solved in this manner.)
7. Define a *procedure* histogram() that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following:

    ****

    ****

    ****

8. The function max() from exercise 1) and the function max_of_three() from exercise 2)will only work for two and three numbers, respectively. But suppose we have a much larger number of numbers, or suppose we cannot tell in advance how many they are? Write a function max_in_list() that takes a *list* of numbers and returns the largest one.

9. Write a program that maps a list of words into a list of integers representing the lengths of the corresponding words.
10. Write a function find_longest_word() that takes a list of words and returns the length ofthe longest one.
11. Write a function filter_long_words() that takes a list of words and an integer n andreturns the list of words that are longer than n.
12. Write a version of a palindrome recognizer that also accepts phrase palindromes such as"Go hang a salami I'm a lasagna hog.", "Was it a rat I saw?", "Step on no pets", "Sit on apotato pan, Otis", "Lisa Bonet ate no basil", "Satan, oscillate my metallic sonatas", "I roamed under it as a tired nude Maori", "Rise to vote sir", or the exclamation "Dammit, I'm mad!". Note that punctuation, capitalization, and spacing are usually ignored.
13. A *pangram* is a sentence that contains all the letters of the English alphabet at least once, for example: *The quick brown fox jumps over the lazy dog*. Your task here is to write a function to check a sentence to see if it is a pangram or not.
14. "99 Bottles of Beer" is a traditional song in the United States and Canada. It is popular tosing on long trips, as it has a very repetitive format which is easy to memorize and can take a long time to sing. The song's simple lyrics are as follows:

    99 bottles of beer on the wall, 99 bottles of beer.

    Take one down, pass it around, 98 bottles of beer on the wall.

    The same verse is repeated, each time with one fewer bottle. The song is completed when the singer or singers reach zero.

    Your task here is write a Python program capable of generating all the verses of the song.

Dictionary

1. Represent a small bilingual lexicon as a Python dictionary in the following fashion {"merry":"god", "christmas":"jul", "and":"och", "happy":gott", "new":"nytt", "year":"år"} and use it to translate your Christmas cards from English into Swedish. That is, write a function translate() that takes a list of English words and returns a list of Swedish words.
2. Write a function char_freq() that takes a string and builds a frequency listing of the characters contained in it. Represent the frequency listing as a Python dictionary. Try it with something like char_freq("abbabcbdbabdbdbababcbcbab").
3. In cryptography, a *Caesar cipher* is a very simple encryption techniques in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with hisgenerals. *ROT-13* ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. In Python, the key for ROT-13 may be represented by means of the following dictionary:

    ```
    key = {'a':'n', 'b':'o', 'c':'p', 'd':'q', 'e':'r', 'f':'s', 'g':'t', 'h':'u', 'i':'v', 'j':'w', 'k':'x', 'l':'y', 'm':'z', 'n':'a', 'o':'b',
    'p':'c', 'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k', 'y':'l', 'z':'m', 'A':'N', 'B':'O', 'C':'P', 'D':'Q', 'E':'R',
    'F':'S','G':'T', 'H':'U', 'I':'V', 'J':'W', 'K':'X', 'L':'Y', 'M':'Z', 'N':'A', 'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F',
    'T':'G', 'U':'H', 'V':'I', 'W':'J', 'X':'K', 'Y':'L','Z':'M'}
    ```

    Your task in this exercise is to implement an encoder/decoder of ROT-13. Once you're done, you will be able to read the following secret message:

    Pnrfne pvcure? V zhpu cersre Pnrfne fnynq!

    Note that since English has 26 characters, your ROT-13 program will be able to both encode and decode texts written in English.

Regular Expression

1. Define a simple "spelling correction" function correct() that takes a string and sees to itthat 1) two or more occurrences of the space character is compressed into one, and 2) inserts an extra space after a

period if the period is directly followed by a letter. E.g. correct("This is very funny and cool.Indeed!") should return "This is very funny and cool. Indeed!" Tip: Use regular expressions!

2. The *third person singular* verb form in English is distinguished by the suffix -*s*, which is added to the stem of the infinitive form: *run -> runs*. A simple set of rules can be given asfollows:
    a. If the verb ends in *y*, remove it and add *ies*
    b. If the verb ends in *o*, *ch*, *s*, *sh*, *x* or *z*, add *es*
    c. By default, just add *s*

    Your task in this exercise is to define a function make_3sg_form() which given a verb ininfinitive form returns its third person singular form. Test your function with words like *try*, *brush*, *run* and *fix*. Note however that the rules must be regarded as heuristic, in the sense that you must not expect them to work for all cases. Tip: Check out the stringmethod endswith().

3. In English, the *present participle* is formed by adding the suffix -*ing* to the infiniteform: *go -> going*. A simple set of heuristic rules can be given as follows:
    a. If the verb ends in *e*, drop the *e* and add *ing* (if notexception: *be*, *see*, *flee*, *knee*, etc.)
    b. If the verb ends in *ie*, change *ie* to *y* and add *ing*
    c. For words consisting of consonant-vowel-consonant, double the finalletter before adding *ing*
    d. By default just add *ing*

    Your task in this exercise is to define a function make_ing_form() which given a verb ininfinitive form returns its present participle form. Test your function with words such as *lie*, *see*, *move* and *hug*. However, you must not expect such simple rules to work for allcases.

Higher order functions and list comprehensions

1. Using the higher order function reduce(), write a function max_in_list() that takes a *list* ofnumbers and returns the largest one. Then ask yourself: why define and call a new function, when I can just as well call the reduce() function directly?
2. Write a program that maps a list of words into a list of integers representing the lengths of the corresponding words. Write it in three different ways: 1) using a for-loop, 2) usingthe higher order function map(), and 3) using *list comprehensions*.
3. Write a function find_longest_word() that takes a list of words and returns the length ofthe longest one. Use only higher order functions.
4. Using the higher order function filter(), define a function filter_long_words() that takes alist of words and an integer n and returns the list of words that are longer than n.
5. Represent a small bilingual lexicon as a Python dictionary in the following fashion {"merry":"god", "christmas":"jul", "and":"och", "happy":gott", "new":"nytt", "year":"år"} and use it to translate your Christmas cards from English into Swedish. Usethe higher order function map() to write a function translate() that takes a list of English words and returns a list of Swedish words.
6. Implement the higher order functions map(), filter() and reduce(). (They are built-in butwriting them yourself may be a good exercise.)

Simple exercises including I/O

1. Write a version of a palindrome recogniser that accepts a file name from the user, readseach line, and prints the line to the screen if it is a palindrome.
2. According to Wikipedia, a *semordnilap* is a word or phrase that spells a *different* word or phrase backwards. ("Semordnilap" is itself "palindromes" spelled backwards.) Write a semordnilap recogniser that accepts a file name (pointing to a list of words) from the userand finds and prints all pairs of words that are semordnilaps to the screen. For example, if "stressed" and "desserts" is part of the word list, the

the output should include the pair"stressed desserts". Note, by the way, that each pair by itself forms a palindrome!

3. Write a *procedure* char_freq_table() that, when run in a terminal, accepts a file name from the user, builds a frequency listing of the characters contained in the file, and printsa sorted and nicely formatted character frequency table to the screen.

4. The *International Civil Aviation Organization (ICAO) alphabet* assigns code words to the letters of the English alphabet acrophonically (Alfa for A, Bravo for B, etc.) so that critical combinations of letters (and numbers) can be pronounced and understood by those who transmit and receive voice messages by radio or telephone regardless of their native language, especially when the safety of navigation or persons is essential. Here is aPython dictionary covering one version of the ICAO alphabet:

   ```
   d = {'a':'alfa', 'b':'bravo', 'c':'charlie', 'd':'delta', 'e':'echo', 'f':'foxtrot','g':'golf', 'h':'hotel', 'i':'india',
   'j':'juliett', 'k':'kilo', 'l':'lima', 'm':'mike', 'n':'november', 'o':'oscar', 'p':'papa', 'q':'quebec', 'r':'romeo',
   's':'sierra', 't':'tango', 'u':'uniform', 'v':'victor', 'w':'whiskey','x':'x-ray', 'y':'yankee', 'z':'zulu'}
   ```

   Your task in this exercise is to write a procedure speak_ICAO() able to translate any text (i.e. any string) into spoken ICAO words. You need to import at least two libraries: os and time. On a mac, you have access to the system TTS (Text-To-Speech) as follows: os.system('say ' + msg), where msg is the string to be spoken. (Under UNIX/Linux and Windows, something similar might exist.) Apart from the text to be spoken, your procedure also needs to accept two additional parameters: a float indicating the length of the pause between each spoken ICAO word, and a float indicating the lengthof the pause between each word spoken.

5. A hapax legomenon (often abbreviated to hapax) is a word which occurs only once in either the written record of a language, the works of an author, or in a single text. Define a function that given the file name of a text will return all its hapaxes. Make sure your program ignores capitalization.

6. Write a program that given a text file will create a new text file in which all the lines from the original file are numbered from 1 to n (where n is the number of lines in the file).

7. Write a program that will calculate the average word length of a text stored in a file (i.e the sum of all the lengths of the word tokens in the text, divided by the number of word tokens).

8. Write a program able to play the "Guess the number"-game, where the number to be guessed is randomly chosen between 1 and 20. (Source: http://inventwithpython.com)This is how it should work when run in a terminal:

   ```
   >>> import guess_number

   Hello! What is your name?

   Torbjörn

   Well, Torbjörn, I am thinking of a number between 1 and 20.Take a guess.

   10

   Your guess is too low.

   Take a guess.

   15

   Your guess is too low.

   Take a guess.

   18

   Good job, Torbjörn! You guessed my number in 3 guesses!
   ```

9. An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; e.g., orchestra = carthorse, A decimal point = I'm a dot in place. Write a Python program that, when started 1) randomly picks a word w from given list of words, 2) randomly permutes w (thus creating an anagram of w), 3) presents the anagram to the user, and 4) enters an interactive loop in which the user is invited to guess the original word. It

may be a good idea to work with (say) colour words only. The interaction with the program may look like so:

>>> import anagram

Colour word anagram: onwbr

Guess the colour word!

black

Guess the colour word!

brown

Correct!

10. In a game of Lingo, there is a hidden word, five characters long. The object of the game is to find this word by guessing, and in return receive two kinds of clues: 1) the characters that are fully correct, with respect to identity as well as to position, and 2) the characters that are indeed present in the word, but which are placed in the wrong position. Write a program with which one can play Lingo. Use square brackets to mark characters correct in the sense of 1), and ordinary parentheses to mark characters correct in the sense of 2).

    Assuming, for example, that the program conceals the word "tiger", you should be able to interact with it in the following way:

    >>> import lingo snake

    Clue: snak(e)

    fiest

    Clue: f[i](e)s(t)

    times

    Clue: [t][i]m[e]s

    tiger

    Clue: [t][i][g][e][r]

SOMEWHAT HARDER EXERCISES

1. A *sentence splitter* is a program capable of splitting a text into sentences. The standardset of heuristics for sentence splitting includes (but isn't limited to) the following rules:

    Sentence boundaries occur at one of "." (periods), "?" or "!", except that

    a. Periods followed by whitespace followed by a lower case letter are not sentenceboundaries.

    b. Periods followed by a digit with no intervening whitespace are not sentenceboundaries.

    c. Periods followed by whitespace and then an upper case letter, but preceded by any of a short list of titles are not sentence boundaries. Sample titles include *Mr.*, *Mrs.*, *Dr.*, and so on.

    d. Periods internal to a sequence of letters with no adjacent whitespace are notsentence boundaries (for example, *www.aptex.com*, or *e.g*).

    e. Periods followed by certain kinds of punctuation (notably comma and moreperiods) are probably not sentence boundaries.

    Your task here is to write a program that given the name of a text file is able to write its content with each sentence on a separate line. Test your program with the following shorttext: *Mr. Smith bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it. Did he mind? Adam Jones Jr. thinks he didn't. In any case, this isn't true... Well, with a probability of .9 it isn't.* The result should be:

    Mr. Smith bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it.

Did he mind?

Adam Jones Jr. thinks he didn't.

In any case, this isn't true...

Well, with a probability of .9 it isn't.

2. An *anagram* is a type of word play, the result of rearranging the letters of a word orphrase to produce a new word or phrase, using all the original letters exactly once; e.g., *orchestra = carthorse*. Using the word list at http://wiki.puzzlers.org/pub/wordlists/unixdict.txt, write a program that finds the sets of words that share the same characters that contain the most words in them.

3. Your task in this exercise is as follows:

   a. Generate a string with N opening brackets ("[") and N closing brackets ("]"), in some arbitrary order.

   b. Determine whether the generated string is *balanced*; that is, whether it consistsentirely of pairs of opening/closing brackets (in that order), none of which mis-nest.

   Examples:

[]	OK	][	NOT OK
[][]	OK	][][	NOT OK
[[][]]	OK	[]][[]	NOT OK

4. A certain children's game involves starting with a word in a particular category. Each participant in turn says a word, but that word must begin with the final letter of the previous word. Once a word has been given, it cannot be repeated. If an opponent cannotgive a word in the category, they fall out of the game. For example, with "animals" as the category,

   Child 1: dog

   Child 2: goldfish

   Child 1: hippopotamus

   Child 2: snake

   ...

   Your task in this exercise is as follows: Take the following selection of 70 English Pokemon names (extracted from Wikipedia's list of Pokemon) and generate the/a sequence with the highest possible number of Pokemon names where the subsequentname starts with the final letter of the preceding name. No Pokemon name is to be repeated.

   audino bagon baltoy banette bidoof braviary bronzor carracosta charmeleon cresselia croagunk darmanitan deino emboar emolga exeggcute gabiteg girafarig gulpin haxorus heatmor heatran ivysaur jellicent jumpluff kangaskhan kricketune landorus ledyba loudred lumineon lunatone machamp magnezone mamoswine nosepass petilil pidgeottopikachu pinsir poliwrath poochyena porygon2 porygonz registeel relicanth remoraid rufflet sableye scolipede scrafty seaking sealeo silcoon simisear snivy snorlax spoink starly tirtouga trapinch treecko tyrogue vigoroth vulpix wailord wartortle whismur wingull yamask

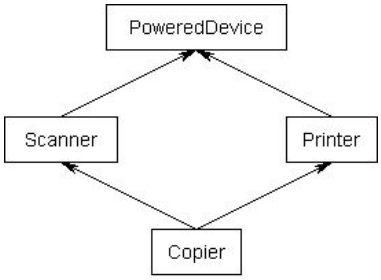
5. An *alternade* is a word in which its letters, taken alternatively in a strict sequence, and used in the same order as the original word, make up at least two other words. All lettersmust be used, but the smaller words are not necessarily of the same length. For example,a word with seven letters where every second letter is used will produce a four-letter word and a three-letter word. Here are two examples:

   "board": makes "bad" and "or". "waists": makes "wit" and "ass".

   Using the word list at http://wiki.puzzlers.org/pub/wordlists/unixdict.txt, write a program that goes through each word in the list and tries to make two smaller words using every second letter. The smaller words must also be members of the list. Print the words to the screen in the above fashion.

Classes

1. Refer the Source code at http://ece.arizona.edu/~edatools/Python/Animals.py. Re-write this program without using static methods. What are the advantages and disadvantages ofthe new structure compared to the original? Note: The show() methods must provide essentially the same functionality as in the original program. Each method provides a display of the characteristics of its class in a format unique to that class. The methods must be callable from any instance of any class, or from no instance at all. The displayedcharacteristics for any class must include the characteristics of its ancestors.

2. 

   implement the class hierarchy with methods of their own.

   Below are few methods:

   1. Scan Document -> Scanner
   2. Print Document -> Printer
   3. TurnON

3. Create shape class as base and various shapes as derived. Use run time polymorphism touse various shapes
   a. Square
   b. Circle
   c. Rectangle
   d. Triangle

Exceptions

1. Create a your own application that throws exception when an invalid email is entered in atextbox and handle the exception

MultiThreading

1. Implement the producer - consumer problem in python using multi threading

Database connectivity

1. Implement an employee details maintenance system,using mysql/MS Access in back endand store/retrieve values from the database

CGI

1. Save some simple html code such as

   <HTML>

   <HEAD>

```
<TITLE> Hello World</TITLE>
</HEAD>
<BODY>
<H1>Greetings</H1>
</BODY>
</HTML>
```

as a file in your www directory on ella. Look at it through a browser.

2. Save some simple cgi code such as

```
#!/usr/bin/env python
import cgi
print "Content-Type: text/html\n"
print """
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
</HEAD>
<BODY>
<H1>Greetings</H1>
</BODY>
</HTML>"""
```

in your cgi directory. Run it on the command-line to check the syntax. Change file permissionsto executeable ("chmod 700 filename" or "chmod u+x filename"). Look at it through your browser.

Networking

GUI Programming

Five mini programming projects for the Python beginner

1. Dice Rolling Simulator

The Goal: Like the title suggests, this project involves writing a program that simulates rolling dice. When the program runs, it will randomly choose a number between 1 and 6. (Or whatever other integer you prefer — the number of sides on the die is up to you.) The program will print what that number is. It should then ask you if you'd like to roll again. For this project, you'll need to set the min and max number that your dice can produce. For the average die, that means a minimum of 1 and a maximum of 6. You'll also want a function that randomly grabs a numberwithin that range and prints it.

Concepts to keep in mind:

- Random
- Integer
- Print
- While Loops

A good project for beginners, this project will help establish a solid foundation for basic concepts. And if you already have programming experience, chances are that the concepts usedin this project aren't completely foreign to you. Print, for example, is similar to Javascript's console.log.

2. Guess the Number

The Goal: Similar to the first project, this project also uses the random module in Python. The program will first randomly generate a number unknown to the user. The user needs to guess what that number is. (In other words, the user needs to be able to *input* information.) If the user'sguess is wrong, the program should return some sort of indication as to how wrong (e.g. The number is too high or too low). If the user guesses correctly, a positive indication should appear.You'll need functions to check if the user input is an actual number, to see the difference between the inputted number and the randomly generated numbers, and to then compare the numbers.

Concepts to keep in mind:

- Random function
- Variables
- Integers
- Input/Output
- Print
- While loops
- If/Else statements

Jumping off the first project, this project continues to build up the base knowledge and introduces user-inputted data at its very simplest. With user input, we start to get into a little bitof variability.

3. Mad Libs Generator

The Goal: Inspired by Summer Son's Mad Libs project with Javascript. The program will first prompt the user for a series of inputs a la Mad Libs. For example, a singular noun, an adjective, etc. Then, once all the information has been inputted, the program will take that data and place them into a premade story template. You'll need prompts for user input, and to then print out thefull story at the end with the input included.

Concepts to keep in mind:

- Strings

- Variables
- Concatenation
- Print

A pretty fun beginning project that gets you thinking about how to manipulate user inputted data.Compared to the prior projects, this project focuses far more on strings and concatenating. Have some fun coming up with some wacky stories for this!

4. TextBased Adventure Game

The Goal: Remember *Adventure*?

Well, we're going to build a more basic version of that. A complete text game, the program will let users move through rooms based on user input and get descriptions of each room. To create this, you'll need to establish the directions in which the user can move, a way to track how far the user has moved (and therefore which room he/she is in), and to print out a description. You'llalso need to set limits for how far the user can move. In other words, create "walls" around the rooms that tell the user, "You can't move further in this direction."

Concepts to keep in mind:

- Strings
- Variables
- Input/Output
- If/Else Statements
- Print
- List
- Integers

The tricky parts here will involve setting up the directions and keeping track of just how far the user has "walked" in the game. I suggest sticking to just a few basic descriptions or rooms, perhaps 6 at most. This project also continues to build on using user inputted data. It can be a relatively basic game, but if you want to build this into a vast, complex word, the coding will get substantially harder, especially if you want your user to start interacting with actual objectswithin the game. That complexity could be great, if you'd like to make this into a longtermproject. *Hint hint.

5. Hangman

The Goal: Despite the name, the actual "hangman" part isn't necessary. The main goal here is tocreate a sort of "guess the word" game. The user needs to be able to input letter guesses. A limitshould also be set on how many guesses they can use. This means you'll need a way to grab a word to use for guessing. (This can be grabbed from a pre-made list. No need to get too fancy.) You will also need functions to check if the user has actually inputted a single letter, to check if the inputted letter is in the hidden word (and if it is, how many times it appears), to print letters, and a counter variable to limit guesses.

Concepts to keep in mind:

- Random
- Variables
- Boolean
- Input and Output
- Integer

- Char
- String
- Length
- Print

Likely the most complex project on this list (well, depending on just how intense you went with the adventure text game), the Hangman project compiles the prior concepts and takes them a stepfurther. Here, outcomes are not only determined based on user-inputted data, that data needs to be parsed through, compared, and then either accepted or rejected. If you want to take this project a step further, set up a hangman image that changes!

TOPIC SPECIFIC ASSIGNMENTS

Heap:

Implement Min and Max heap

Implement a base class called Heap which has the basic properties and methods of a Heap.

Derive MaxHeap and MinHeap classes from Heap class.

Basic Heap operations to be implemented:

1. insert
2. delete
3. dearch
4. min / max
5. delete_min / delete_max
6. sort

You should implement all the supporting functions as well. Calculate and convince yourself the complexity of each operation

General instructions:

1. Use abstract methods if needed.
2. Use getters and setters wherever needed.
3. Use class variables if needed
4. Use the right access specifiers Private/Protected/Public

BST (Binary Search Tree) & BBST (Balanced BST):

Implement BST and its basic operations

Implement a class for BST with all its basic properties and methods

Basic BST properties to be implemented

1. insert
2. delete
3. search
4. min / max
5. sort

Derive BBST from the base class BST and implement all the basic operations.Compare BST and BBST with respect to all the basic operations' complexity

General instructions:

1. Use abstract methods if needed.
2. Use getters and setters wherever needed.
3. Use class variables if needed
4. Use the right access specifiers Private/Protected/Public

OOP: Project Weekly Billing

Overview

Data : Weekly Timesheet of the Team (csv file)

Config : Create a manual config file where the user changes the settings. (INI File / JSON File)

Report : Report should be generated as HTML and JSON file

Implementation

Create Project and Employee classes

Project (+ are properties and – are methods)

- \+ employees - All the employees working in the project
- \+ tags - All the tags related to the project
- \+ billing_amount_in_inr - Total billing amount for the project
- \+ blling_amount_in_usd
- \+ total_hours_spent
- – calculate_acitivity_summary() → Table with Rows as Tags and Columns asEmployees
- – calculate_employee_summary() →Table of Employee vs hours_spent
- – write_report_to_json()*
- – write_report_to_html()**
- – display_bar_chart() → Shows employee vs billing amount

Employee (+ are properties and - are methods)

- \+ projects - All the projects the employee is a part of
- \+ tags - All the tags used by the employee
- \+ billing_rate_in_usd
- \+ total_hours_spent
- \+ total_billing_amount
- – calculate_acitivity_summary() → Dictionary - {tag: hours_spent}
- – calculate_project_summary() → Dictionary - {project: hours_spent}
- – write_report_to_json()*
- – write_report_to_html()**
- – display_bar_chart() → Shows Project vs Billing amount

General instructions

1. Read how can a custom object be serialized into json
2. Handle errors wherever necessary. The program should not crash at any cost. It shouldgracefully exit
3. * Understand the information present and come up with your own json format whichprovides a summary of project/employee
4. ** No need to implement in the first version.

Performance tuning

Optimize the below code to take only 100ms.

Currently it takes ~14 seconds

```
''' Gradient synthesis functions'''


from math import sqrt
import time
import cv2
import numpy as np
def get_grad_img (bg_size, colors, points):
   '''
   Generate gradient synthesized image with given colors
   on given locations
   Inputs:
       bg_size = Width and Height of the image
       color = Color to be spread
       points = Points from which each color should be spread
       is_l2 = If True L2 distance measure will be used for
              spreading color L1 distance otherwise
   Returns:
       bg_img = Background image with the gradients
   '''
   width, height = bg_size
   bg_img = np.zeros((height, width, 3 ), np.uint8)
   colors = [(clr[ 0 ] / 255 , clr[ 1 ] / 255 , clr[ 2 ] / 255 ) for clr in colors]

   for pxl_x in range(width):
       for pxl_y in range(height):
       # for circular gradient
       dists = [ 1 / (sqrt((pt[ 0 ] - pxl_x) ** 2 + (pt[ 1 ] - pxl_y) ** 2 ) +
                    0.000000001 ) for pt in points]

       # Find the inverse distance as we need the color to be present
       # at the 0 distance location and fade away when the distance increases

       max_dist = sum(dists)

       norm_dists = np.array(dists) / max_dist


       # Weight each color with the calculated weight and add them to

       # get the final color for the location

       norm_r, norm_g, norm_b = np.sum(np.transpose(colors) * norm_dists, axis= 1 )
       r_val, g_val, b_val = [int( 256 * v) for v in (norm_r, norm_g, norm_b)]
       bg_img[pxl_y, pxl_x] = r_val, g_val, b_val
   bg_img = cv2.GaussianBlur(bg_img, ( 5 , 5 ), 0.2 )
   return bg_img


if __name__ == '__main__' :
   CLRS = [[ 117 , 139 , 155 ], [ 153 , 95 , 83 ], [ 226 , 250 , 252 ]]
   PTS = [[ 0 , 80 ], [ 0 , 400 ], [ 750 , 480 ]]

   start_time = time.time()
   BG_IMG = get_grad_img(( 960 , 480 ), CLRS, PTS)
   print( 'Time taken: {} seconds' .format(round(time.time() - start_time, 4 )))
```

Flask: File upload and display the size

Create a flask application which has following options:

1. Simple html with options to
    a. Upload a file
    b. Button to view size
    c. Button to view the date modified
    d. Button to view number of lines
    e. Button to view all the three properties
2. On click of any button the details should be displayed as simple text (no html markup needed)
3. Application should run be accessible by all machines in the network on port 9876

EXE creation:

Create a windows executable for the flask application project with the following options

1. Should be able to run on any windows machine with same bit processor (ie., if you have 32 bit machine, your exe should be able to run on any 32 bit machine)
2. Double click to start the application
3. Modify the flask application with an option tostop the server from browser (ie., a button on the webpage to stop the server)