SOLUTION CP1200 – Practical 9

Aims

1. Practise working with strings
2. Write programs using dictionaries

Operations On Strings

Write a program that gets a string from the user (e.g. "banana split"), and then prints out the following (if you find yourself taking more than a few minutes for any of these, continue on, and come back and try later – you have no more than 25 minutes for these 7 questions):

1. The string reversed ("tilps ananab")
2. The first three characters ("ban")
3. The string without the letter "a" ("bnn split")
4. The string in uppercase ("BANANA SPLIT")
5. The last three characters of the string in uppercase and reversed ("TIL")
6. The string with an extra space between all the characters ("b a n a n a s p l i t")
7. The string with the words reversed ("split banana")

SOLUTION – See Python file

File Line Adder-Upper

Problem For You To Fill In The Blanks

We're going to write a program that will read a file like (comma separated values):

```
3,4,5
1,2
4,5,6,7
```

and output the sum of the numbers on each line, so, for example:

```
Total for line 1 is 12.
Total for line 2 is 3.
Total for line 3 is 22.
```

Copy and complete the following code. You only need to add code to the **addUpLine** function.

```
def main():
    filename = input("Enter the filename: ")
    fileIn = open(filename, 'r')
    i = 1
    for line in fileIn:
        total = addUpLine(line)
        print("Total for line ", i, " is ", total, ".", sep="")
    fileIn.close()

def addUpLine(line):
    # split the line into parts
    parts = line.split()
    # add up all the parts
    total = 0
```

```
    for number in parts:
        total += int(number)
    # return the total
    return total

main()
```

Dictionary Warm-up

Write a program that uses a constant dictionary (use ALL_CAPS) to store the Australian state abbreviations and names - e.g. QLD is Queensland. Then ask the user for their 'short' state and print the full state name. The program should have error checking, be case insensitive and loop until an empty state is entered, like:

```
Enter short state: nz
Invalid short state
Enter short state: qld
QLD is Queensland
Enter short state: NSw
NSW is New South Wales
Enter short state:
```

As always, don't start by writing the whole thing in one go (even if you could). Create a small dictionary with two states and make sure you can print the correct full name for a short name (without a loop) - then add error checking, then the loop, then complete the dictionary - testing after each of those iterations.

SOLUTION – See Python file

Autocorrector Program

Noboddy knows how to spell these days [sic]. Let's help them out by writing a program to correct common misspellings. The program will read from a file called "corrections.txt" that contains content like the following:

```
teh,the
tje,the
dis,this
lyk,like
```

The program should construct a dictionary that maps incorrect spellings to correct spelling, so correctionsDict["teh"] would equal "the", and so on (i.e. the keys are incorrect spellings, the values are the matching correct spellings - notice keys must be unique, but values can be duplicated).

After the program has built the dictionary, it should ask the user for a line of text to correct (it will quit when they just hit enter). The program will then output the corrected line.

Sample output:

```
Enter a line of text to correct: Everybody should speak lyk dis
You should have written: Everybody should speak like this
Enter a line of text to correct: What is teh time
You should have written: What is the time
```

Enter a line of text to correct: ***<ENTER>***
Go forth and spell correctly!

You're welcome to try to write this program from scratch (try to make good use of functions), otherwise download **autocorrect_to_complete.py** from LearnJCU and complete the file. (You'll need to use the string **replace** method, and remember that this doesn't change the string, it returns a **changed version!**)

SOLUTION – See Python file

Word Counter

Write a program to count the occurrences of words in a text file. The program should ask the user for a filename, then open and read the file and print the counts of how many of each word are in the file.
The output should look something like (depending on the file contents):

```
and : 12
but : 1
concordance : 2
dancing : 17
```

(It's an interesting piece of text, this.)
Hints: use a dictionary where the keys are the words and the values are the counts; when you find a word, check if it's in the dictionary...
Notice that the sample output is sorted. Make your program do this (sort the words), but only **after** you have it working.

SOLUTION – See Python file