CM1103 Lab Worksheet Week 8

Formatted Output *[Think Python Version 2 - Section 14.3]*

1. The following code will print out a floating point value. Enter the code and see what it does.

   ```
   d = 50506.2124524342352525242
   print("The␣value␣of␣d␣is␣{0:f}".format(d))
   ```

   (a) Modify the code above so that `{0:f}` is replaced by `{0:.2f}`. What is the purpose of adding the "`.2`"?

   (b) Modify the previous code to use the formatting string `{0:10.2f}`. What is the purpose of the `10`?

   (c) The format code `x` converts the value to hexadecimal format.

      i. Write code that prompts the user to enter an integer and converts it to hexadecimal.

      ii. Write code that prompts the user to enter an integer and converts it to binary.

2. Enter the following code:

   ```
   a = "animal"
   b = "horse"
   print("My␣favourite␣{}␣is␣{}".format(a,b))
   ```

   Try the code, replacing the first `{}` with the following:

   (a) {:d}

   (b) {:.2s}

   (c) {:10s}

3. Complete the following function

   ```
   def writeCurrency(currency, amount):
      print(???)
   ```

   so that it prints a given amount of money in the correct format. Two examples are given below.

   ```
   >>> writeCurrency("$",50.2567)
   $50.26
   >>> writeCurrency("£",500.0)
   £500.00
   ```

   Modify the function so that it returns the correct string representation instead.

Dictionaries *[Think Python Version 2 - Section 14.3]*

4. The following code will create a dictionary that relates a set of names (the keys) to a set of numbers (the values).

```
d = {"Tom" : 500, "Stuart" : 1000, "Bob" : 55, "Dave" : 21274}
```

(a) Provide code that prints the number of items in the dictionary, a list of the keys, and a list of the values. [Hint: `dir()` will give a list of the functions in `dictionary`.]

(b) When using a dictionary it is possible to add and delete items from it. Provide code to delete the entry for `"Bob"` and add an entry for `"Phil"`.

(c) We can also check if items are present in the dictionary. Enter the following code and check if the results are what you expect:

```
"Dave" in d
"Peter" in d
500 in d
```

(d) What is the difference between the code:

```
for key, value in d.items():
   print(key, value)
```

and:

```
for i in d:
   print(i)
```

5. Download the file `towns.csv` from Learning Central which contains data showing the population of various towns in the UK.

(a) Read and store the data in a dictionary `d` which uses the towns as keys.

(b) Write code which outputs this dictionary in the following format:

```
          Coventry : 303475
         Edinburgh : 430082
             Derby : 229407
             Leeds : 443247
        Birmingham : 970892
         Liverpool : 469017
           Belfast : 276459
         Leicester : 330574
           Cardiff : 292150
    Stoke-on-Trent : 259252
Newcastle upon Tyne : 189863
          Bradford : 293717
       Southampton : 234224
        Nottingham : 249584
        Manchester : 394269
     Wolverhampton : 251462
           Reading : 232662
 Kingston upon Hull : 301416
            London : 7172091
```

```
      Bristol : 420556
    Sheffield : 439866
       Dudley : 194919
     Plymouth : 243795
      Glasgow : 629501
  Northampton : 189474
```

[Hint: what does the expression `width = max([len(x) for x in d])` represent?]

(c) Modify the code so that it only outputs those towns with population between 500,000 and 1,000,000.

(d) Modify the code from (b) so that it prints the results in the order of decreasing population.

6. Using the code given in the previous examples and your knowledge from previous exercises, write a function that will count the number of times each letter occurs in a passage of text. Your program should then output the list of all letters and the number of times each of them occurs. Code is given below to start you off.

```
def countLetters(text):
   letterList={}
   for letter in text:
      if letter not in letterList:
         ??? set the value of letter in the dictionary to 1 ???
      else:
         ??? incremement the value of letter in the dictionary by 1???
   ??? Return dictionary ???:
```

Regular Expressions

7. The function shown below will print “Matched!” if the text passed to it consists only of one or more lower case letters (a-z). It does this using regular expressions. To do this the **re.match** function is used. This function will return *None* if no matches are found.

```
import re
def expressionTest(text):
   result=re.match("[a-z]+",text)
   if result != None:
      print "Matched!"
   else:
      print "Not␣Matched!"
```

(a) Enter the above code into Python.

(b) Call the `expressionTest` function with the data shown below and check if the results are what you expect.

```
expressionTest("Hello")
expressionTest("&")
expressionTest("Python␣is␣cool")
expressionTest("goodbye")
expressionTest("test1")
```

(c) Did any of the results differ from what you expected? If so which one and why do you think this happened?

8. The regular expression *[a-z]+@[a-z]+.[a-z]{3}* will match to many simplistic email addresses. In short it means that an email address consists of:

   - *One or more lower case letters (a-z).*
   - *An @.*
   - *One or more lower case letters (a-z).*
   - *A period (.).*
   - *Three final lower case letters.*

   Finish the function shown below so that it returns true if an email address matches the format which is described above.

```
import re
def matchEmail(email):
   result= ??? Check if the email address matches the regular expression
   ??? If result is None then return false
   ??? Otherwise then return true.
```

For more information on regular expression you can look at the tutorial at
http://www.regular-expressions.info/tutorial.html

Formative assessment

The question in this section will help to prepare you for the coursework that will be set in week 9.

Download the file `Week7Template.py`, and complete the function as indicated below. Try to ensure your code passes all of the doctests.

9. Suppose we have data stored in a csv file, where each row represents a record consisting of a name and some numerical values, e.g.:

   ```
   bob,5,2,4,
   ```

   Complete the function `sumRows` so that it reads a file of this format and returns a dictionary whose keys specify the names and whose values are the sum of numerical values in the corresponding row. For example, the record above would result in an entry `'bob': 11.0`. Empty or non-numerical fields (other than the names in the first column) should be ignored.

   If the optional parameter `header` is `True`, your code should ignore the first line of the file.

   Suppose instead the data was listed by column instead of by row (as in the file `columns.csv`). Complete the function `sumColumns` to return a similar dictionary, indexed by the column headings, with values giving the sum of each column.

Advanced

10. Develop a function called *postCodeChecker*. This function should do the following.

    (a) Read from a file a list of postcodes.

    (b) Check all the postcodes in the file against the regular expression *[A-Z]{2}[0-9]{1,2}[0-9][A-Z]{1,2}*

    (c) If a postcode does not match the regular expression a warning should be printed for the user.

    Code to start you off is provided below and an file of example data *postcodes.txt* is available on Learning Central.

```
import re
def matchPostCodeChecker():
   ??? Read file into a list
   for postcode in ???
      #This next line strips the returns off the input file.
      postcode=postcode.replace("\n","")
      ??? Does Postcode match the expression
      ??? If Postcode does not match print warning.
```

Challenge: *Dermatoglyphics*

My iPad is unlocked using a four digit passcode that needs to be entered using an on-screen keypad. ***Given that I spend most of the day eating doughnuts and crisps, and I'm extremely lazy and untidy,*** which of the following schemes would give me the most secure passcode:

- choosing a code with four distinct digits (e.g. 2416);
- choosing a code with three distinct digits (e.g. 3151);
- choosing a code with two distinct digits (e.g. 8228);
- choosing a code with a single digit (e.g. 9999).

Rank these schemes in order, and justify your answer either mathematically or with Python code.

Email AllenSM@cardiff.ac.uk with your answer and the code you wrote to solve the problem, with the subject line "CM1103 week 8 challenge" by midnight on Wednesday in Week 9 – a winner will be picked at random from the correct answers **but** you'll get one extra entry for particularly noteworthy, concise or interesting code.