Introduction to programming – ASSIGNMENT 9

General instructions:

- Type in and test your programs using Python Idle.
- Bring in your answers to the next assignment (i.e. demonstration) session in memory stick, or save them to a web folder accessible in class. Alternatively, you can bring written answers with you, though this is not recommended.
- Remember to comment your code – this does NOT mean, that every single line should be commented. Comment the important parts in your program.
- Prepare to present your solution to the class.

1.

Write a program that queries the user for products and prices, one item at a time, and writes them to file ”products.txt”. Each product and the price is written at the same line, separated with a comma. When the user enters an empty string as a product name, the program terminates.

Example run:

```
Enter product name: Apples
Enter price in Euros: 12
Enter product name: Oranges
Enter price in Euros: 9
Enter product name:
Thank you, bye!
```

The file “products.txt” after the example run:

```
Apples, 12
Oranges, 9
```

2.

Write a function **`calculateDigits(),`** which reads a web page from address

http://users.utu.fi/ertaka/prog/digits.txt

...and calculates and returns the total number of digits (0-9) in that file.

For example, if the contents of the file would look like this:

```
a2b7c3d4e8fghi6abcdsss
```

...the function would return 6, since there are a total of 6 digits in the file.

3.

Download a file *matrix.txt* from Moodle. The file consists of integer numbers representing a matrix. Each line represents a row, and the items in that row are separated with spaces. Hence an example row in file could look something like this:

```
11 43 2 5 9 111 4 2 8
```

Write a function **`rowAverageSum(),`** which reads the matrix from a file, calculates the average for each row, and finally returns the sum of all those averages.

4.

Write a function **`parseComments(programFile),`** which gets a name of a Python program file as an argument. The function reads the program and returns a list with all comments from that program in it.

For example, for program “average.py”:

```
# Calculate the avg of list
def listAvg(lst):
   # variable for avg
   avg = 0.0
   # iterate through list
   for item in lst:
      avg = avg + item
   # return average
   return avg / len(lst)
```

...the example run would look like this:

```
>>> lst = parseComments("average.py")
>>> print lst
['Calculate the avg of list', 'variable for avg', 'iterate through list',
'return average']
```

NOTE: For following questions you need the file “persons.dat”, which can be downloaded from Moodle. File holds a list of persons with name, age, height and weight associated for each person in that order; a single line in file looks something like this:

```
Eric Example, 23, 185, 79
```

5.

Write a function **`parsePersons(),`** which reads the persons file and creates a dictionary from the persons in that file. In dictionary, the name is used as a key, and the value is a tuple with age, height and weight as *integers* in that order. Finally, the dictionary is returned.

6. ** *Expert assignment (double points)*

Write a program, that queries the user for a criteria (name, age, weight or height) and the sorts the persons in “persons.dat” file in ascending natural order using that criteria. For example, if the user wants to sort the file by age, the youngest person should be the first in file and the oldest last etc.

Hint: it is probably easiest to read the whole file into a list, sort it and then write it back to file. To sort items in list by any criteria, see the model answers for last assignments.

Example run:

```
Select the criteria to sort the file by:
1. Name
2. Age
3. Weight
4. Height
Sort by what criteria: 2
Sorted by age!
```