Programmeringsteknik (TIN214, EEN110)

8 June 2021

Contact: Krasimir Angelov
krasimir@chalmers.se, tel. 031 772 10 19

Note: Solutions must be uploaded to Canvas as a single Python file. It is not necessary to comment on your code, it is more important that the code is clean and easy to read.

All help materials are allowed, but everyone is expected to work alone. Cooperation is not allowed and cheating may lead to suspension from the university.

The grading schema is G/U. You need at least 15 points to get grade G.

For questions contact me primarily via the e-mail. The phone number is only for backup.

Question 1

The following variable:

```
products = {
  "äpple":
     {"price": 27.90
     },
  "vispgrädde":
     {"price": 21.50
     ,"discount": 2
     },
  "jordgubbar":
     {"price": 57.90
     ,"discount": 3
     }
}
```

contains information about the product list in a shop. It is assigned to a dictionary where the keys are the product names and the values are other dictionaries which describe the pricing of the corresponding product. The nested dictionary can only have two keys: `"price"` and `"discount"`. The price is what you normally pay for the product, but if there is also a discount then you deduce that many percentages from the final sum paid for the product.

Implement the function `shopping()` which repeatedly asks the customer for product name and quantity. Based on the product name and the quantity it computes the sum to be paid after deducting the potential discount. After each interaction, the program prints the sum for the current product and the total sum accumulated so far. The interaction stops when the customer enters an empty string for the product name. On the other hand, if the customer enters a product name that is not listed in the above variable, then the program must print a reasonable message without failing.

An example interaction follows:

```
>>> shopping()
Product? äpple
Quantity? 2
Sum:   55.8                          # no discount applied
Total: 55.8

Product? mjölk
Sorry we don't have mjölk            # error message printed

Product? vispgrädde
Quantity? 1
Sum:   21.07                         # 2% discount
Total: 76.87                         # total = 55.8+21.07

Product?                             # no product name -> exit
```

(7 points)

Question 2

Implement the function:

```
def load_products(file_name):
    ...
    return ...
```

which reads the file with the given name and returns a data structure similar to the one shown in Question 1, e.g. the one with product names, prices and discounts. The structure of the file should be as follows:

- Each line describes a single product.
- The first word in the line is the product name. For simplicity, we assume that all products have one-word names.
- The second word is a real number and represents the price.
- If there is a third word, then it is also a real number and represents the discount. If there is no third word, then you should not put any discount key in the dictionary.

Note that the file format is defined in such a way that it is enough to use the split() method to split each line into words. You don't need to do any complicated tokenizations.

The following is an example file content which represents exactly the data structure from Question 1.

```
äpple 27.90
vispgrädde 21.50 2
jordgubbar 57.90 3
```

(10 points)

Question 3

Implement the class `Cooccurrences` which must allow counting how often a word appears together with another word in a sentence.

The class must have the following components:

1. An `__init__` method which sets the initial state of the object.
2. A method:

   ```
   def addSentence(sentence):
       …
   ```

   which adds another sentence to the state of the object. The sentence itself is a list of words.
3. A method:

   ```
   def get(word):
       …
   ```

   which given a word returns a list of other words that appear together with that word. The list must contain each word only once.

The following is an example for how the class can be used:

```
>>> occ = Cooccurrences()
>>> occ.addSentence(["the", "boy", "saw", "the", "girl"])
>>> occ.addSentence(["the", "girl", "loved", "flowers"])
>>> print(occ.get("boy"))
["the", "saw", "girl"]
>>> print(occ.get("girl"))
["the", "boy", "saw", "loved", "flowers"]
>>> print(occ.get("the"))
["the", "boy", "saw", "girl", "loved", "flowers"]
>>> occ.addSentence(["the", "cat", "likes", "the", "girl"])
>>> print(occ.get("girl"))
["the", "boy", "saw", "loved", "flowers", "cat", "likes"]
```

(13 points)