CSSE1001

Semester 2, 2009
Assignment 1
20 marks

Due Friday 4 September, 2009, 5pm

1 Introduction

The Just Bicycles bike shop builds a product line of bicycles from parts. The shop keeps information about parts and their product line in two text files.

The first file (`parts.txt`) contains information about the parts stocked in the store. Each (non-empty) line of the file contains three pieces of information separated by commas. The first is the part ID, the second is the name of the part and the third is the cost of the part in cents. For example, one line of the file is

```
WH239,  Mountain Bike Wheel,            5000
```

The part ID is `WH239`, the part name is `Mountain Bike Wheel` and the cost is $50.00. Note that each line of the file may contain any number of spaces and tabs between the items on the line.

The second file (`products.txt`) contains information about the bicycles sold by the store. Each line of the file is a comma separated list. The first item is the product ID, the second item is the product name, and the rest of the list is made up of pairs where each pair consists of a part ID and the number of this part needed to build this bicycle. Each pair uses colon as a separator. Here is an example line of this file (two lines are used below in the display in order to fit on the page).

```
bike201,        Mountain Bike,
     WH239:2,TR202:2,TU277:2,FR201:1,FB201:1,BB201:1,GS201 : 1
```

The product ID is `bike201`, the product name is `Mountain Bike` and the bike is built using two parts with ID `WH239`, two parts with ID `TR202`, ..., and one part with ID `GS201`. Again note that any number of spaces and tabs may be used between items (both comma separated items and colon separated items).

In this assignment you will write Python code to provide a simple interactive environment to aid staff in querying information from the two files.

The staff wish to be able to do the following.

- Given a product name find the ID of that product.
- Given a product ID find the name of the product.
- Given a product ID find the cost of the product based on the cost of the parts.
- Given a product ID find the parts, and the number of each part needed to build the product.

Your job is to write a simple interface that first reads in the parts and products files and then interacts with the user in order to extract the above information.

The interaction repeatedly asks the user for a command, processes the command and prints out the results.

The valid commands are as follows.

- `e` - Exit the program.
- `i product_name` - Display the ID of the given `product_name`.
- `n product_id` - Display the name of the product with ID `product_id`.
- `c product_id` - Display the cost of parts for the product with ID `product_id` in dollars.
- `p product_id` - Display the parts list for the product with ID `product_id`.

Below is an example of what is expected (`Command:` is the prompt produced by the system; the rest of the line is the user input - the command).

```
>>> interact()
Command: n bike201
Mountain Bike
```

```
Command: i Mountain Bike
bike201
Command: i Ford Car
No such item
Command: n csse1001
No such item
Command: x
Unknown command
Command: c bike201
  920.00
Command: p bike201
[('WH239', 2), ('TR202', 2), ('TU277', 2), ('FR201', 1),
 ('FB201', 1), ('BB201', 1), ('GS201', 1)]
Command: e
>>>
```

The output above for the command `p bike201` has been split over two lines in order to fit on the page - it is not required to do that for the assignment. Apart from that, you should get exactly the same behavior as above.

For the assignment you may assume that, if the command is a valid command, then the command is followed by the correct number of arguments.

The next section describes the tasks required in order to achieve the above behavior.

2 Assignment Tasks

For each function that you write you need to provide a suitable comment giving a description, the type and any preconditions. You should use the triple-quote commenting style.

2.1 Download files

The first task is to download the parts and products files `parts.txt` and `products.txt` which are the two files discussed earlier.

We suggest you create a folder in which to write your solution and put these files in that folder.

The file `assign1.py` is for your assignment. Add your name and student number in the space provided. **Do not otherwise modify this comment block and add your code after this block** When you have completed your assignment you will submit the file `assign1.py` containing your solution to the assignment.

2.2 Write the code

Finally, write your solution to the assignment making sure you have included suitable comments. There are several functions you need to write and these are described below.

2.2.1 Load Parts

`load_parts(filename)` takes the name of a file containing parts information and returns a dictionary that associates each part ID with a pair consisting of the name of the part and the cost of the part in cents.

For example (this is not exactly how the output will appear in IDLE):

```
>>> load_parts('parts.txt')
{'GS301': ('Racing Bike Gear Set', 9000),
 'BB101': ('Road Bike Back Brakes', 4000),
 'TU377': ('Racing Bike Tube', 6000),
 'GS201': ('Mountain Bike Gear Set', 6000),
 'TU277': ('Mountain Bike Tube', 2000),
 'FR201': ('Mountain Bike Frame', 60000),
 'FB301': ('Racing Bike Front Brakes', 6000),
 'TU177': ('Road Bike Tube', 4000),
 'BB201': ('Mountain Bike Back Brakes', 4000),
 'FB101': ('Road Bike Front Brakes', 4000),
 'WH139': ('Road Bike Wheel', 10000),
 'WH339': ('Racing Bike Wheel', 20000),
 'TR302': ('Racing Bike Tire', 6000),
 'BB301': ('Racing Bike Back Brakes', 6000),
 'TR102': ('Road Bike Tire', 4000),
```

```
 'FB201': ('Mountain Bike Front Brakes', 4000),
 'GS101': ('Road Bike Gear Set', 6000),
 'WH239': ('Mountain Bike Wheel', 5000),
 'FR301': ('Racing Bike Frame', 200000),
 'TR202': ('Mountain Bike Tire', 2000)}
```

NOTE: In the example above `load_parts` returns the dictionary and this result is printed by the interpreter. The function DOES NOT print out anything. This is also true of all the functions below except `interact`.

When reading information from files please use `open(filename, 'U')` in order to get operating system independent processing of newlines.

2.2.2 Get Components

`get_components(parts)` takes a list of strings representing pairs of part IDs and number of this part (colon separated) and returns a list of pairs consisting of a part ID and a number.

For example:

```
>>> get_components(['WH239:2','TR202:2','TU277:2','FR201:1',
                    'FB201:1','BB201:1','GS201 : 1'])
[('WH239', 2), ('TR202', 2), ('TU277', 2), ('FR201', 1),
('FB201', 1), ('BB201', 1), ('GS201 ', 1)]
>>>
```

2.2.3 Load Products

`load_products(filename)` takes the name of a file containing product information and returns a dictionary that associates each part ID with a pair consisting of the name of the product and a list of pairs, each consisting of the name of the part and the number of this part required to build the product. For example:

```
>>> load_products('products.txt')
{'bike301': ('Racing Bike', [('WH339', 2), ('TR302', 2),
                             ('TU377', 2), ('FR301', 1),
                             ('FB301', 1), ('BB301', 1),
```

```
                              ('GS301', 1)]),
 'bike201': ('Mountain Bike', [('WH239', 2), ('TR202', 2),
                               ('TU277', 2), ('FR201', 1),
                               ('FB201', 1), ('BB201', 1),
                               ('GS201', 1)]),
 'bike101': ('Road Bike', [('WH139', 2), ('TR102', 2),
                           ('TU177', 2), ('FR101', 1),
                           ('FB101', 1), ('BB101', 1),
                           ('GS101', 1)])}
>>>
```

2.2.4 Get Product ID

`get_product_id(product_dict, name)` takes, as arguments, a product dictionary, as produced by `load_products`, and a string, `name`, and returns the ID of the product with the given name from the dictionary, if it exists, and returns `None` otherwise.

For example:

```
>>> products = load_products('products.txt')
>>> get_product_id(products, 'Racing Bike')
'bike301'
>>> get_product_id(products, 'Ford Car')
>>>
```

2.2.5 Get Product Name

`get_product_name(product_dict, id)` takes, as arguments, a product dictionary, as produced by `load_products`, and a string, `id`, and returns the name of the product with the given ID from the dictionary, if it exists, and returns `None` otherwise.

For example:

```
>>> products = load_products('products.txt')
>>> get_product_name(products, 'bike301')
'Racing Bike'
>>> get_product_name(products, 'csse1001')
>>>
```

2.2.6 Get Parts

`get_parts(product_dict, id)` takes, as arguments, a product dictionary, as produced by `load_products`, and a string, `id`, and returns the parts list of the product with the given ID from the dictionary, if it exists, and returns `None` otherwise.

For example:

```
>>> products = load_products('products.txt')
>>> get_parts(products, 'bike301')
[('WH339', 2), ('TR302', 2), ('TU377', 2), ('FR301', 1),
 ('FB301', 1), ('BB301', 1), ('GS301', 1)]
>>> get_parts(products, 'csse1001')
>>>
```

2.2.7 Compute Cost

`comput_cost(product_dict, parts_dict, id)` takes, as arguments, a product dictionary, as produced by `load_products`, a parts dictionary as produced by `load_parts`, and a string, `id`, and returns the cost (in cents) of all the parts of the product with the given ID, if it exists, and returns `None` otherwise.

For example:

```
>>> parts = load_parts('parts.txt')
>>> products = load_products('products.txt')
>>> compute_cost(products, parts, 'bike301')
285000
>>> compute_cost(products, parts, 'csse1001')
>>>
```

2.2.8 The Top-Level Interface

`interact()` is the top-level function that defines the interface as described in the previous section. It will make use of all of the above functions.

2.2.9 Hints

For file processing: `open, partition, strip, split`
For user interaction: `raw_input`
For dictionary processing: use `[]` for updating the dictionary and `get` for accessing information.

3 Marking Criteria

We will mark your assignment according to the following criteria.

Criteria	**Mark**
Comments:	3
- comments that are complete, consistent, clear and succinct	3
- comments with some minor problems	2
- comments with major problems	1
- work with little or no academic merit	0
Code:	17
- code that is mostly complete, correct, clear and succinct	12 - 17
- code with a number of problems	6 - 11
- code that is clearly incomplete, incorrect, too complex or hard to understand	1 - 5
- work with little or no academic merit	0
Total marks	20

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out that code and we will mark that.

Please read the section in the course profile about plagiarism.

4 Assignment Submission

You must submit your completed assignment electronically through the website: `http://submit.itee.uq.edu.au`

Please read `http://submit.itee.uq.edu.au/student-guide.pdf` for in-

formation on using electronic submission.

You should electronically submit your copy of the file `assign1.py` (use this name - all lower case).

You may submit your assignment multiple times before the deadline - only the last submission will be marked.

Late submission of the assignment will not be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on-time, you should contact the lecturer in charge and be prepared to supply appropriate documentary evidence. You should be prepared to submit whatever work you have completed at the deadline, if required. Requests for extensions should be made as soon as possible, and preferably before the assignment due date.