## Coffee Tasting Challenge

TextNow recently opened a retail website to sell imported coffee. We’ve done quite well, but we can do better! We’d like to use our historic data to build a recommendation engine.

Our sample set includes N people and M types of coffee, and each person has rated approximately half of the possible coffee types. The file is tab delimited, and each line contains a user id, a descriptive name for the coffee, and the person’s rating:

Input:  
 person\_id coffee\_name rating

Example:  
5 Organic Fair Trade Decaf Honey Burst Mexican 2  
5 Fair Trade Decaf AA Indian 3  
5 Fair Trade Caturra Balinese 1  
5 Fair Trade Swiss Water Peruvian 1  
5 Organic Sidamo Panamanian 1  
6 Organic Fair Trade Decaf Honey Burst Mexican 3  
6 Fair Trade Decaf AA Indian 2  
6 Organic Fair Trade Swiss Water Costa Rican 1  
6 Paradise Valley Panamanian 1  
6 Decaf Caturra Guatemalan 4

## 1 Write a coffee-name parser

Your first task is to write a coffee-name parser. It should extract properties based on the coffee’s descriptive name. We’ve identified five properties:

boolean Decaf  
boolean Organic  
boolean Fair trade  
String Adjective  
String Country

For example, given "Organic Decaf Mandeheling Bolivian", it should extract:

Decaf True  
Organic True  
Fair trade False  
Adjective Mandheling  
Country Bolivia

To help get you started, we’ve provided a sample framework written in python. Using this framework, you’ll implement the parser as a classmethod:

// example:  
c = Coffee.fromname("Fair Trade Decaf Black Satin Panamanian");

If you choose to use the provided python framework, notice that we have included a command-line tool to assist in testing. Please configure the tool such that you can parse coffee names from the command line; for example:

// Command line example:  
$ python coffee.py parse 'Fair Trade Decaf Black Satin Panamanian'  
Fair Trade Decaf Black SAtin Panamanian

Decaf true  
 Organic false  
 Fair trade true  
 Adjective Black Satin  
 Country Panama

Be sure to convert the adjectival form of a country to the noun form. Specifically, you’d change Panamanian to Panama. Below is a table of countries that are used in the data set.

"Balinese", "Bali"  
"Bolivian", "Bolivia"  
"Brazilian", "Brazil"  
"Costa Rican", "Costa Rica"  
"Dominican", "Dominican Republic"  
"Salvadorean", "El Salvador"  
"Ethiopian", "Ethiopia"  
"Guatemalan", "Guatemala"  
"Indian", "India"

"Kenyan", "Kenya"  
"Malian", " Mali"  
"Mexican", "Mexico"  
"Panamanian", "Panama"  
"Peruvian", "Peru"  
"Sumatran" "Sumatra"

2 Summarize the Data

Let’s summarize the data set! Using the parser that you just wrote, please iterate through the input file and create a summary table. You should expose this function through your command-line tool:

$ python coffee.py summarize coffee\_ratings.txt  
Total people 140  
Total coffee types 50  
Decaf

True 20

False 30  
Organic

True 10

False 40  
Fair trade

True 5

False 45  
Adjective

Bright 15

Supremo 35  
Country

India 10  
 Peru 40

3 Build a Recommendation Engine

Let’s build the recommendation engine! Your job is to recommend three new coffee types to each person. The output should include the user’s id, the coffee’s name, and the predicted rating:

$ python coffee.py recommend coffee\_ratings.txt  
9 Decaf Black Satin Balinese 4  
9 Black Satin Balinese 4  
9 Organic Longberry Mexican 4  
10 Honey Burst Domincan 5  
10 Organic Fair Trade AA Guatemalan 4  
10 Organic Fair Trade Decaf Swiss Water Malian 4

Bonus!

Congratulations! Coffee enthusiasts everywhere will be grateful for your hard work. If you’re still amped on caffeine and feeling peppy, consider these bonus tasks:

* Display the summary from summarize visually.
* An error occurred in which either 1 or 2 people would be associated with the same ID number. In other words, we can no longer be certain that an ID represents only one person. Fortunately, we do know that this affected ≤ 25% of the data in bonus\_ratings.txt. Design a recommendation scheme to process this data.