POINTS INTERNATIONAL

Sales Tax Calculator

Prepared for: Point International, Engineering team
Prepared by: Shayan Yousefian, Development Engineer Candidate
April 18, 2019

4 pages document

SUMMARY

Objective

Create a sales tax calculator that process a list of items in a cart and create a receipt with all sale taxes included.

Goals

Basic sales tax is 10%, applicable on all goods except books, food and medical products which are exempt. Import duty is an additional sales tax applicable on all imported goods at a rate of 5%, with no exemptions.

Solution

The project is written in Python 3.7.3, no framework used. For a web-application, Django would've been a good choice. All code is in a virtual environment, for isolating the required packages in the scope of this project. Unit tests have created to test each module and their key functions. Code documentation can be extracted using packages like Sphinx to create an HTML documentation or in other formats. In the scope of this test project, use of Sphinx has been skipped, but as standard procedure to write a clean code, documentation is added for classes and methods of the project, in addition to explanation for more complicated parts of the code's logic.

Project Outline

Project is consist of 4 modules: tax, product, cart, and interface. Each responsible of one aspect of the project, that will be explained later on, tests for each module and some configuration root files described below:

Root files

- In the root folder, "**constant.py**" carries all constant strings, "imported", "Sales Tax" and "Total". Because in a well-written code no constant number or variable should be "hard-coded". This would make the code extendable and re-usable (except obviously range limits and empty values, such as number 0, empty string or null value).
- "**settings.py**" holds configuration variables, related to basic functioning of the whole project. The one variable that someone can edit for the use of their own machine is the "root" which has the system file path to the root of the project. If left unchanged, command "os.path.dirname(__file__)" will set the variable automatically.
- "**requirements.txt"** contains list of Python packages needed to be installed to run the project, which in the case of current implementation it is empty, because no third-party packages has been used.
  - Note, running command "pip install -r requirements.txt" in your shell, will install all packages automatically.
- "**runner.py**" is the file the user should run to see the project running. Results will be printed in the console.

Tax module:

Tax module handles basic properties of tax calculation, in “core.py”, class Tax declared which has the two fundamental methods of tax calculation: calculate_sale_tax() and round_up_05()

Tax class has two attribute, basic tax and import duty which can be set later based on the type of product. Then calculated tax amounts will be round up to the closest 0.005 using round_up_05() method. Rounding is done using “Decimal(amount).quantize(Decimal(‘.1’), rounding=ROUND_UP)” tool. But there are some limits in computing floating numbers in Python and in computer in general.

The trick to rounding up to nearest 0.05 is to doubling the amount, rounding up to nearest 0.1 and then dividing the result by 2. But in Python, doubling up the Decimal(0.1 * 2) results in Decimal('0.2000000000000000111022302462515654042363166809082031 25')

Therefore, a round down to 0.00001 has been applied just to the doubled amount itself. A thorough test in “tests/tax/test_core.py” validated the function for various number of inputs.

Tax types set the two sales tax percentages based on their definition. Type basic, has basic tax of 10 percent, While Exempt has 0. Imported has 5 percent while domestic has 0 import duty.

Then tax categories use mix of these 4 types. BasicDomestic, ExemptImported, etc. Each product has a tax profile from this list.

Product module:

Each product has a tax profile, which calculates its tax amount and the cost of product (tax included) based on the product’s price. Each tax profile has different settings of taxes, therefore, tax calculation is different for products of different kinds (like books and imported cosmetic products).

Product categories like Books has predefined tax profile. But instead of creating two categories for imported and domestic products. Products are given an “imported” flag that will automatically change their tax profile from domestic to imported while keeping their basic or exempt profile.

All tax calculations from now on, is done with precision of 2 floating points. Again, because arithmetic calculation of float values are not always accurate. (For example: 4.99 + 1.9 = 6.890000000000001 while 6.89 is correct)

Cart module:

Item, is an encapsulation of product and count of them. In case multiple number of same product are saved in the cart.

Receipt is simply a collection of items, which has a total sales tax amount and total cost ready to be printed for the final result. Receipts can be printed using str method or convert to JSON by dictionary output.

Interface module:

Interface module, parse text files and JSON files to populate receipts.

parser.py has two methods, parse_item() and parse_item_from_json() they both return a list of tuples of (count, imported/or not, product's name, category string, and price). This list of tuples can be used by the receipt module to populate the receipt object, calculating and summarizing sale taxes.
parse_item() parse a text line to extract information about the item, product's name, if it is imported or not, its price and its count in the shopping cart. This method of input, by design has some flaws, for example if a product's title has the word "imported" in it, like in a book's title, this method will fail because it uses same keyword (imported) to determine whether the product is imported or not (for import duty tax purposes)
parse_item_from_json() has an easier job, dealing with standardized JSON inputs, but it returns the same format.

input.py prepares files to read, one method for text files each item in one line, and one method for JSON files. They call parser methods to parse the raw input they read from the input files.
File directory is fetched from the setting variable described earlier in the documentation.

Instructions to Run

Activate the virtual environment using the command:
"source venv/bin/activate"
Then run the main runner by:
"python3 runner.py"

References

Please visit the Github page of this project for further development details.
GitHub: https://github.com/shayan-ys/sales-tax-calculator