Prof. Dr. techn. Wolfgang Nejdl
Franziska Schoger

September 25th 2023

Advanced Topics in AI
Programming Exercise 0

Organization

- This exercise is introductory and doesn't need to be submitted.
- You may participate in this and upcoming exercises as a group of 2 persons.
- If you encounter any bugs, please report them to *schoger@l3s.de*

Disclaimer

We are reusing a project from UC Berkeley[1]. The whole documentation can be found at https://inst.eecs.berkeley.edu/~cs188/fa23/projects/proj0. The required code is also available at https://inst.eecs.berkeley.edu/~cs188/fa23/assets/projects/tutorial.zip

Python Installation

To participate in the programming exercises, you need a **python3** environment of the version python 3.6 or higher. Additionally you will need `pip` or `conda`. You can check via `python -V` or `python3 -V` your python version and with `conda -V` or `pip -V` / `pip3 -V` your conda or pip version.
If you need to install python, I personally suggest to use conda, which automatically comes with python. Installation instructions for Windows, Mac and Linux can be found here.
If there is a `tkinter` import error, it's likely because Python is atypical, and from Homebrew. Uninstall that and install python from Homebrew with `tkinter` support, or use the another installer.
When running the autograder you might encounter an exception on modern python versions. The cause of this is that the method `cgi.escape` has been deprecated sine 2011 and removed with Python 3.8. To fix this you can either use an earlier version (3.6 works fine) or make the following two changes to the autograder: Replace `import cgi` with `from html import escape`. Replace `message = cgi.escape(message)` with `message = escape(message)`.

[1] http://ai.berkeley.edu

Project Structure & Autograding

If you download the exercise files, you will see that it contains many files. Most of them are not relevant to you and you can just ignore them. Files you will edit:

- `addition.py`
- `buyLotsOfFruit.py`
- `shopSmart.py`

Additional interesting files:

- `shop.py`

Do not change any other files and do not change any of the provided function names as this will create problems for the autograder and you will not be able to check your progress.

To check your progress in solving the exercise run

```
python autograder.py
```

If you need a basic introduction to python, please check here.

Question 1: Addition

Open the file `addition.py` and look at the definition of `add`:

```
def add(a, b):
    "Return the sum of a and b"
    "*** YOUR CODE HERE ***"
    return 0
```

When you run the autograder it calls this function with different values for a and b, but the code always returns 0. Modify the code so that it returns the sum of a and b instead of 0.
Run the autograder using

```
python autograder.py -q q1
```

All the tests should pass now. If you have probles you can add `print()` statements in the function to help you debugging.

Question 2: buyLotsofFruits

Implement the `buyLotsOfFruit(orderList)` function in `buyLotsOfFruit.py` which takes a list of `(fruit,numPounds)` tuples and returns the cost of your list. If there is some fruit in the list which doesn't appear in `fruitPrices` it should print an error message and return None. Please do not change the `fruitPrices` variable.

Run `python autograder.py -q q2` until question 2 passes all tests and you get full marks. Each test will confirm that `buyLotsOfFruit(orderList)` returns the correct answer given various possible inputs. For example, `test_cases/q2/food_price1.test` tests whether the cost of `[('apples', 2.0), ('pears', 3.0), ('limes', 4.0)]` is 12.25.

Question 3: shopSmart

Fill in the function `shopSmart(orderList,fruitShops)` in `shopSmart.py`, which takes an `orderList` (like the kind passed in to `FruitShop.getPriceOfOrder`) and a list of `FruitShop` and returns the `FruitShop` where your order costs the least amount in total. Don't change the file name or variable names, as otherwisethe auto-grader will have problems.

Run `python autograder.py -q q3` until question 3 passes all tests and you get full marks. Each test will confirm that `shopSmart(orderList,fruitShops)` returns the correct answer given various possible inputs.