National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2024/2025

Tutorial 10
Inheritance & Exceptions

Release date: 4th November 2024
Due: 10th November 2024, 23:59

General Restrictions

- No importing packages unless explicitly allowed to do so.
- You are only allowed to use data structures taught so far.

Questions

1. For this question, we want to model three different classes: `Product`, `DiscountProduct`, and `Cart`.
    - A `Product` is initialised with a `name` and `price`. It supports three methods: `get_name()` which returns the `name`, `base_price()` which returns the `price` of a `Product`, and `get_price()` that returns the price of a `Product`, inclusive of 9% GST, rounded to two decimal places.
    - A `DiscountProduct` is also a `Product`, but at a discounted price. It is initialised with a name, price (before discount), and the discount rate (in percent) to be applied. Unfortunately, GST does not get discounted. Hence, `get_price()` should use `base_price` from `Product` and return the discounted price after tax (i.e., discounted price + tax on original price).
    - A `Cart` is initialised with no inputs, and supports two methods:
        - `add_item()`, which takes in a `Product`, and returns a string following these rules:
            - If the product is a valid `Product`, it is added in the cart, and the string `"Adding <product name> to cart."` is returned.
            - Otherwise, the cart remains unchanged and the string `"Invalid item."` is returned.
        - `checkout()`, which takes in no arguments, and returns a dictionary with `Product` names as the keys and total cost of `Products` in cart as the corresponding values. `Products` with the same name should have their costs added together.

          Important: When adding multiple `Products` of the same type, you should calculate the total price by using the price **after tax** (i.e., the total price is **not** calculated by multiplying the amount by unit price before tax, then incrementing the tax).

    Implement `Product`, `DiscountProduct`, and `Cart`.

Sample Execution:

```
>>> spam = Product('tulip', 4.2)
>>> jam = Product('jam', 5.35)
>>> eggs = Product('eggs', 1.5)
>>> ham = DiscountProduct('ham', 7.2, 15)
>>> spam.get_price()
4.58
>>> eggs.base_price()
1.5
>>> ham.get_price() # (7.2 * 0.85) + (7.2 * 0.09)
6.77
>>> shopping_list = [eggs, ham, eggs]
>>> c = Cart()
>>> c.add(eggs)
'Adding eggs to cart.'
>>> c.add(ham)
'Adding ham to cart.'
>>> c.add(shopping_list)
'Invalid item.'
>>> c.add(eggs)
'Adding eggs to cart.'
>>> c.add(spam)
'Adding tulip to cart.'
>>> c.checkout()
{'eggs': 3.28, 'tulip': 4.58, 'ham': 6.77}
```

2. What is the output of the following code snippet?

```
s = 'abcde'
try:
    t = s[0:-2:1]
    print(t)
    t[0] = s[1]
    print(t)
except TypeError:
    t = t + s[1]
    print(t)
except IndexError:
    t = t + s[0]
    print(t)
finally:
    print(t + s[-1])
    print(t[6])
```