

Exercise Sheet 3

Exercise 1 - Exceptions

Write a function called `oops` that explicitly raises an `IndexError` exception when called. Then write another function that calls `oops` inside a `try/except` statement to catch the error. What happens if you change `oops` to raise `KeyError` instead of `IndexError`? Also try to print a Python stack trace by using the `print_exc()` function in the standard `traceback` module (see the Python library reference for details)

Exercise 2 - Operator Overloading

Write a class called `Mylist` that “wraps” a Python list: it should overload most list operator and operations: `+`, indexing, iteration, slicing, and list methods such as `append` and `sort`. Also provide a constructor for your class that takes an existing list (or a `Mylist` instance) and copies its components into an instance member. Experiment with your class interactively. Things to explore:

- Why is copying the initial value important here?
- Can you use an empty slice (e.g. `start[:]`) to copy the initial value if its a `Mylist` instance?
- Is there a general way to route list method calls to the wrapped list?
- Can you add a `Mylist` and a regular list? How about a list and a `Mylist` instance?
- What type of object should operations like `+` and slicing return; how about indexing?

Exercise 3 - Kangaroo classes

This exercise is a cautionary tale about one of the most common, and difficult to find, errors in Python. Write a definition for a class named `Kangaroo` with the following methods:

- An `__init__` method that initializes an attribute named `pouch_contents` to an empty list.
- A method named `put_in_pouch` that takes an object of any type and adds it to `pouch_contents`.
- A `__str__` method that returns a string representation of the `Kangaroo` object and the contents of the pouch.

Test your code by creating two `Kangaroo` objects, assigning them to variables named `kanga` and `roo`, and then adding `roo` to the contents of `kanga`'s pouch.

Open the file `BadKangaroo.py` in the course repository. It contains a solution to the previous problem with one big, nasty bug. Find and fix the bug.

Hint: you can use `pylint` to highlight the bug for you

Exercise 4 - Composition

Simulate a fast-food ordering scenario by defining four classes:

1. `Lunch`: a container and controller class
2. `Customer`: the actor that buys food
3. `Employee`: the actor that a customer orders from
4. `Food`: what the customer buys

To get you started, here are the classes and methods you'll be defining:

```
class Lunch:
    def __init__(self)           # make/embed Customer and Employee
    def order(self, foodName)    # start a Customer order simulation
    def result(self)             # as the Customer what kind of Food it has

class Customer:
    def __init__(self)           # initialize my food to None
    def placeOrder(self, foodName, employee) # place order with an Employee
    def printFood(self)          # print the name of my food

class Employee:
    def takeOrder(self, foodName) # return a Food, with requested name

class Food:
    def __init__(self, name)     # store food name
```

The order simulation works as follows:

1. The Lunch class's constructor should make and embed an instance of Customer and Employee, and export a method called order. When called, this order method should ask the Customer to place an order, by calling its placeOrder method. The Customer's placeOrder method should in turn ask the Employee object for a new Food object, by calling the Employee's takeOrder method
2. Food objects should store a food name string (e.g. "burritos"), passed down from Lunch.order to Customer.placeOrder, to Employee.takeOrder, and finally to Food's constructor. The top-level Lunch class should also export a method called result, which asks the customer to print the name of the food it received from the Employee (this can be used to test your simulation)
3. Note that Lunch needs to either pass the Employee to the Customer, or pass itself to the Customer, in order to allow the Customer to call Employee methods.