FIT1008 – Intro to Computer Science
Solutions for Tutorial 6

Semester 1, 2018

Exercise 1

The call to `silly()` will print "'pong" twice. Here's why:

```
def silly():
    x = 'ping'      # here x is 'ping'
    def g():        # and here g is defined
        print(x)    # and it's true that this x means the enclosing scope's x
    x = 'pong'      # but here we change the enclosing scope's x to mean 'pong'
    def f(x):
        print(x)
    g()             # so by the time we call g(), the enclosing scope's x
    f(x)            # is the same than when we call f(x), that is, 'pong'

silly()             # therefore, this will print "pong" twice, on two separate lines
```

Exercise 2

What did Juan do wrong? More like what did Juan do right, right?
First, when he defined his class thus:

```
class PoolMember:
    poolname = "FIT1008 students community pool"
    name = ""
    age = 0
    gender = None

    def __init__(self, name, age, gender):
        PoolMember.name = name
        PoolMember.age = age
        PoolMember.gender = gender
```

He made `name`, `age` and `gender` properties of the class. The initialiser modifies the class property instead of making a variable for each instance of the class.
Thus, all members of the pool will share the same name, age and gender. Not only that but the name, age and gender of all pool members will be the ones for the latest person we've added. If a grannie joins the pool, suddenly all users will be elderly females; if a new baby boy enrol, suddenly all users will become newborn males. That's why the admin user details are actually Emilia's.
The second mistake Juan made was to modify the pool's name in his own instance. In doing so, he created an instance variable `poolname`

that was only accessible through the admin user, via the qualified access `admin.poolname`.

This would be the correct way of writing the class:

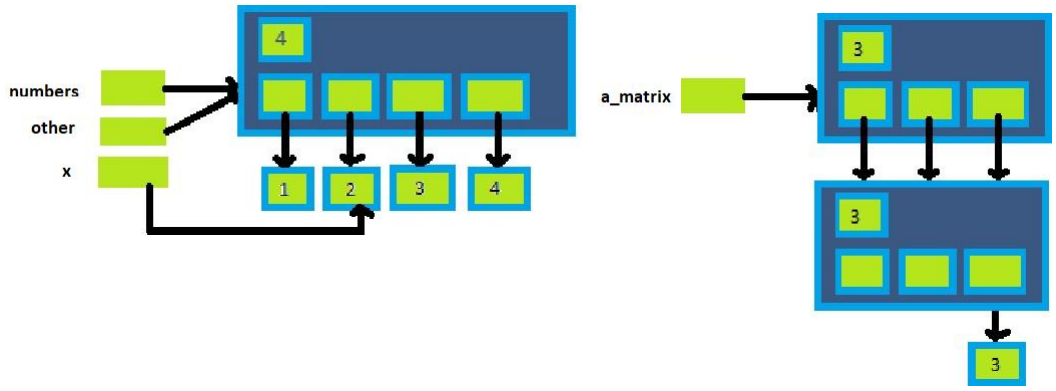
```
class PoolMember:
    poolname = "FIT1008 students community pool"
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
```

And this is how you can change the name of the pool and check that everything is fine:

```
>>> admin = PoolMember('Juan', 18, 'male')
>>> PoolMember.poolname = "MONASH all-inclusive community pool"
>>> admin.poolname
'MONASH all-inclusive community pool'
>>> supervisor = PoolMember('Emilia', 26, 'female')
>>> supervisor.poolname
'MONASH all-inclusive community pool'
>>> admin.name
'Juan'
```

Exercise 3

Figure 1: Memory diagram for Excercise 3



Exercise 4

There are different options. One option is to maintain a list of customers and a list of coffees.

1. CoffeeShop - maintains a list of customers and a list of coffees.
2. Customer - contains Customer attributes and methods. Attributes include name, phone number and number of points. The methods might include one that updates the number of points, each time a coffee is bought.
3. Coffee - contains Coffee attributes and methods. Attributes include type, price, and day of the week in which the coffee is discounted.
4. List - will be defined in lectures later, it includes attributes the_array and length, and methods _empty, is_full, add, delete, size.

The student could think about some issues with this model. For example, if we wish to search for a customer, what if the name of a customer is not unique? Should we ideally have a customer ID instead?
Later we will learn about dictionaries and hash tables, and the student may wish to think about an implementation with dictionaries. When searching for a customer 'record' by name, would using a dictionary speed things up? By how much?