What is a class?

A class is a representation of a real-world entity
- Defines data, plus methods to work on that data
- You can hide data from external code, to enforce encapsulation

Domain classes
- Specific to your business domain
- E.g. BankAccount, Customer, Patient, MedicalRecord

Infrastructure classes
- Implement technical infrastructure layer
- E.g. NetworkConnection, AccountsDataAccess, IPAddress

Error classes
- Represent known types of error
- E.g. Error, BankError, CustomerError

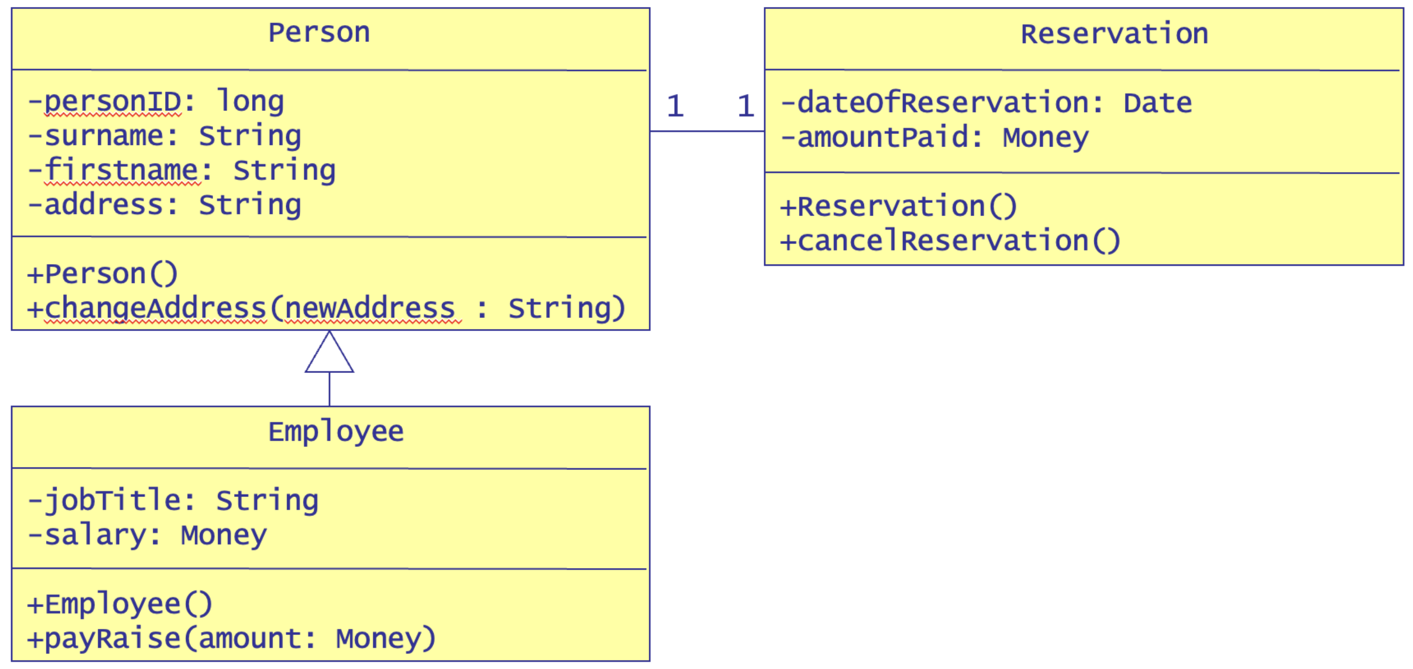
Etc.

What is an object?

An object is an instance of a class
- Created (or "instantiated") by client code
- Each object is uniquely referenced by its memory address (no need for primary keys, as in a database)

Object management
- Objects are allocated on the garbage-collected heap
- An object remains allocated until the last remaining object reference disappears
- At this point, the object is available for garbage collection
- The garbage collector will reclaim its memory sometime thereafter

During OO analysis and design, you map the real world into candidate classes in your application



```
class ClassName:
    #
    # Define attributes (data and methods)
    #
```

To create an instance (object) of the class:
- Use the name of the class, followed by parentheses
- Pass initialization parameters if necessary (see later)
- You get back an object reference, which points to the object in memory

You can define methods in a class
i.e. functions that operate on an instance of a class

In Python, methods must receive an extra first parameter
- Conventionally named self
- Allows the method to access attributes in the target object

You can implement a special method named __init__()
- Called automatically by Python, whenever a new object is created
- The ideal place for you to initialize the new object!
- Similar to constructors in other OO languages

Typical approach:
- Define an __init__() method, with parameters if needed
- Inside the method, set attribute values on the target object
- Perform any additional initialization tasks, if needed

One of the goals of OO is encapsulation
- Keep things as private as possible

However, attributes in Python are public by default
- Client code can access the attributes freely!

To make an object's attributes private:
- Prefix the attribute name with two underscores, __

Class-wide variables belong to the class as a whole
- Allocated once, before usage of first object
- Remain allocated regardless of number of objects

To define a class-wide variable:
- Define the variable at global level in the class

To access the class-wide variable in methods:
- Prefix with the class name

Typical uses for class-wide methods:
- Get/set class-wide variables
- Factory methods, responsible for creating instances
- Instance management, keeping track of all instances

The @classmethod and @staticmethod decorators can be applied to class-wide methods