CSCA48
Ramp-Up Session

January 19, 2018

Object Oriented Programming

- Functions
- `my_function(data)`
- global function,
- get data and pass to the function

- Object Oriented approach
- `my_object.method(data)`
- the object has its own methods and data

Terminology

- Class
  - The type of an object (i.e int, str, bool)
    - `>>> type('hello')  ----> <class 'str'>`
- Object
  - An instance of a class
    - `>>> s = 'hello'  # s is an instance of string class`
- Method
  - Like a function, but it belongs to a class
    - `>>> s.find('e')`
- Attribute

- Create a class
  - `class MyClassName():`
  - brackets are used for inheritance
- Define a method
  - `def method_name(self, arg1,arg2…):`
  - Every method gets implicitly a “copy” of itself when calling the method
  - This “copy” represents the object itself
- Create a object
  - `my_object = MyClassName()`
- Access the methods
  - `my_object.method_name(data1,data2…)`

Built-in Methods

- `__init__(self, arg1,ar2…):`
  - The constructor method
  - Initialize the object and its attributes
  - Automatically called when you create the object
  - Class attributes are defined in this method

- `__str__(self):`
  - Return a string representation of this class object
  - You can define the output string by yourself

Unified Modelling Language (UML)

- Commonly used in software engineering
- REASONABLE design is important

- You are running a restaurant. Your restaurant name and business hours available to the public.
- As the restaurant owner, you should be able to accept table reservation via phone call or online booking. Once a reservation is confirmed, a table should be reserved by the system with customer specified information. You can also cancel the reservation by customer's request or 30 minutes after the reservation time start.
- If a customer would like to reserve a table, they must provide information including date, time and number of people in the request in order to successfully reserve a table.

- Association
  - One object holds another object as a variable or class attribute **(has-a relationship)**
  - A Dog has-a Toy
- Composition
  - One object is made up of many other kind of objects **(part-of relationship)**
  - Usually interact with composite object, no direct access to the object components
  - A Room is part-of a Building
- Inheritance
  - One class is a specific case of another general class **(is-a relationship)**
  - A Student is-a Person

Inheritance

- When we say classB inherite from classA
  - code: `class ClassB(ClassA):`
- What we are saying is the ClassB is-a subclass of ClassA.
  - ClassB can **use** all ClassA's methods
  - ClassB can **overwrite** ClassA's methods (same method name) with its own implementation

UML Diagram

- 5 components in a UML diagram

- Class name
- Attributes
  - Attribute with underscore = private attribute
  - Show type of all attributes
- Methods
  - Method with underscore = private method
  - Show type contract of all methods

- Relationship
  - Association
  - Composition
  - Inheritance
- Cardinality (or Multiplicity)
  - Use numbers to represent the relation of the two classes

- You are running a restaurant. Your restaurant name and business hours available to the public.
- As the restaurant owner, you should be able to accept table reservation via phone call or online booking. Once a reservation is confirmed, a table should be reserved by the system with customer specified information. You can also cancel the reservation by customer's request or 30 minutes after the reservation time start.
- If a customer would like to reserve a table, they must provide information including date, time and number of people in the request in order to successfully reserve a table.

Unittest

- https://github.com/MeowsterEric/CSCA08-A48/blob/master/CSCA48/00_Review/unittest_example.py

Abstraction & ADT

- We want to hide the implementation details from outside users/code
- Makes it easier to change code in the future

- Understand ADT from 2 perspectives:
  - User: does not care how ADT is implemented, as long as they can use them to do their work
  - Developer: provide interfaces that allow user to use the ADT, and hide implementation details as much as possible