DT265/B CMPU4060: Lab 11

DIT School of Computing | Dr Andrew Hines

Instructions

Classes, Objects, Methods and Attributes

This lab will revise and expand on some of the object oriented programming topics introduced in our lectures on creating classes and objects. There is an online tutorial available provides further examples on the topic and may be useful for revision:
http://www.python-course.eu/python3_object_oriented_programming.php Today we are going to code objects for airplanes and a control tower to manage the take off sequence.

Problem

We need to have an airplane that is identified from other airplanes by its flight number. The airplane needs fuel to fly and can be fuelled up to capacity but needs at least 1000 to fly. Before flying the pilot must request clearance from the tower and the tower will check the flight number against a list of flights before granting clearance. The tower grants clearance and the pilot then performs a preflight check and if the airplane is properly fuelled and has clearance from the tower it must take off.

Design Tasks

Read and review the problem above. Write down the classes you will need to create and the attributes and methods that the classes will need. Remember nouns, verbs, adjectives can often correspond to Classes, Methods, and Attributes. Draw a flow chart of the takeoff procedure/process. Then move onto the example below and complete the coding execises section to solve the problem.

Example Code to get started

We will start by creating an Airplane class. The basic class code is in the Aircraft.py listing below. The Airplane has three attributes:
`__fuel`
`maxFuel`
`__fuelCheck`

The __ before the fuel and fuelCheck mark them as private attributes, meaning you can?t access them directly in the object, only through the objects methods. Why do we do this? So that the __fuelCheck attribute is only set when a given amount of fuel has been added and so that the amount of __fuel is always a valid amount.
What do we mean by a valid amount? Well, if I set the fuel value directly, I could make it -500 or 100000. As the minimum amount of fuel cannot logically be less than zero or greater than the tank capacity, the method addFuel(volume) ensures these boundaries are maintained.
e.g. I could have written the `addFuel(self, volume)` method with one line:
`self.__fuel = self.__fuel + volume`
But this would not stop incorrect amounts of fuel on the plane or tell me how much was leftover after topping up the plane.

Listing 1: Aircraft.py

```python
class Aircraft:
    """
    Aircraft class: An Airplane has to be fuelled before it can take off
    """
    __fuel = 0 # private attribute containing current fuel in aircraft
    maxFuel = 24000
    __fuelCheck = False  # this is a Boolean flag for a pre-flight check.
    MIN_FUEL = 1000 # minimum amount of fuel for takeoff

    def __init__(self, planeType = '747'):
        self.planeType = planeType

    def fuelCheck(self):
        if self.__fuel < self.MIN_FUEL:
            print("Fuel Check Failed: Current fuel below safe limit:", self.__fuel,
                  " less than ", self.MIN_FUEL)
            self.__fuelCheck = False
        else:
            print("Fuel Check Complete:", self.__fuel)
            self.__fuelCheck = True

    def takeOff(self):
        if self.__fuelCheck == True:
            print("Cleared for Takeoff! Fasten your seat-belt!")
        else:
            print("Take off Failed: Please complete pre-flight check first")
            print(self.fuelCheck())

    def printStatus(self):
        print("Current fuel:", self.__fuel)

    def addFuel(self, volume):
        unusedFuel = 0

        if volume<0:
            print("No syphoning fuel!")
        elif self.__fuel + volume <= self.maxFuel:
            self.__fuel = self.__fuel + volume
        elif self.__fuel + volume > self.maxFuel:
            self.__fuel = self.maxFuel
            unusedFuel = volume - self.__fuel

        return unusedFuel
```

Exercises

1. (*5 points*) Create the test.py file (code below) to test out the `Aircraft` class by creating and calling methods on example object instances.

2. (*5 points*) Create a new class called `Tower` that has a list of flight numbers as an attribute and a method `updateFlightList(aFlightList)` used to update the flight list and ensure all flight numbers in the list are valid.

3. (*5 points*) Add attributes to the `Aircraft` class for `flightNumber` (a string, e.g. "EI124") and `__flightClearance` (a boolean `True` or `False`).

4. (*5 points*) Add another method to the `Aircraft` class called `preFlightCheck()`. This method should check the `__fuelCheck` and `__flightClearance` are both set to true or warn you not to take off yet.

5. (*5 points*) Add a method `requestFlightClearance(anAirplane)` to the Tower class that takes an `Aircraft` object checks its flight number against its list of flights and gives clearance to take off it the flight is in the list.

6. (*5 points*) Update the `test.py` to create a `dublinTower` instance of the `Tower` class, add two flight numbers to the towers flight list and then create an Aircraft, assign a flight number, fuel it, get clearance from the tower and take off!

Listing 2: testAircraftCode.py

```
from Aircraft import *


myjumbo= Aircraft('747')
print("About to start preparing a ", myjumbo.planeType, " for takeoff")
myjumbo.addFuel(30)
myjumbo.printStatus()
myjumbo.fuelCheck()
myjumbo.takeOff()

print("---")

myairbus= Aircraft('A330')
print("About to start preparing a ", myairbus.planeType, " for takeoff")
myairbus.addFuel(2000)
myairbus.printStatus()
myairbus.fuelCheck()
myairbus.takeOff()
```

```python

print("---")

myBoeing= Aircraft('737')
print("About to start preparing a ", myBoeing.planeType, " for takeoff")
fuelInTruck = 50000
fuelInTruck = myBoeing.addFuel(fuelInTruck)
myBoeing.printStatus()
myBoeing.fuelCheck()
myBoeing.takeOff()
print("Fuel Truck still has:", fuelInTruck)
```

Program Output:

```
About to start preparing a  747  for takeoff
Current fuel: 30
Fuel Check Failed: Current fuel below safe limit: 30  less than  1000
Take off Failed: Please complete pre-flight check first
Fuel Check Failed: Current fuel below safe limit: 30  less than  1000
None
---
About to start preparing a  A330  for takeoff
Current fuel: 2000
Fuel Check Complete: 2000
Cleared for Takeoff! Fasten your seat-belt!
---
About to start preparing a  737  for takeoff
Current fuel: 24000
Fuel Check Complete: 24000
Cleared for Takeoff! Fasten your seat-belt!
Fuel Truck still has: 26000
```