W18 Reading Unit 7

June 26, 2016

1 Classes

1.1 Classes Introduction

To solve hard problems effectively, we need powerful and specialized data structures

```
For inits and strs, to lists, sets and dicts.

we've seen how these different types can help us solve problems effectively
```

Now we'll see how you can cerate your own classes/types in Python
Why create your own types?

```
Can be more complicated than built in types

Can be tailored to specific tasks

Don't jusr store data-classes can interact with each other and perform extensve computations

In fact, most large scale software development is object oriented, meaning most of what developers actu
```

what exactly is a type, and how it relate to objects?

```
a class/type is a template for making objects. It defines methods and other attributes shared by all ob

an object is an instance of class. Each instance shares (many of) the same behaviors, but is filled wit
```

Attibutes

```
when we create a class, we have to decide what outward behaviors we want its instances to share

these outward behaviors are what we call attributes

    they include methods as well s other data

 e.g. a student object might contain:

     Methods: s.add_calss(), s.get_gpa(),...

     Data: s.first_name, s.last_name ,...
```

1.2 Classes

```
In [1]: class Drone:
            """A representation of a drone aircraft"""
```

```
In [4]: ?Drone
        # to check what this is
```

```
In [8]: d1 = Drone()
        d2 = Drone()
        print("d1 has type", type(d1), "d2 has type", type(d2))
```

```
d1 has type <class '__main__.Drone'> d2 has type <class '__main__.Drone'>
```

1.2.1 Data Attributes

```
In [22]: # first way is to do it manually
         Drone.power_system = "Battery"
         # this is a class attribute, we give the whole class Drone an attribute
         # called "Battery"
```

```
In [23]: # Therefore, all the instances of the class will have the attribute Battery
         # under power_system
         print(d1.power_system)
         print(d2.power_system)
         print(Drone.power_system)
```

```
Battery
Battery
Battery
```

```
In [24]:  d1.power_system = "Gasoline"
         # now we give d1 an attribute called power_system
```

```
In [56]: print(d1.power_system)
         print(d2.power_system)
         print(Drone.power_system)
```

```
dream
battery
battery
```

```
In [27]: # now let's give the objects some more attributes called altitude
         d1.altitude = 100
         d2.altitude = 150
```

```
In [29]: print(d1.altitude)
         print(d2.altitude)
```

```
100
150
```

```
In [30]: dir(d1)
         # this will check what are the attributes the this object have
```

```
Out[30]: ['__class__',
          '__delattr__',
          '__dict__',
```

```
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'altitude',
 'power_system']
```

In [32]:
```
# let's check some of the built in attributs
#(notice these built in attributrs start and end with __ )
d1.__class__
```

Out[32]:
```
__main__.Drone
```

In [34]:
```
# let's check another built in attributs, the dictionary
d1.__dict__
```

Out[34]:
```
{'altitude': 100, 'power_system': 'Gasoline'}
```

1.2.2 Using the Class Definition

In [36]:
```
class Drone:    #defines a class

    power_ststem = "battery"    # this is a class attribute because it is
                                # insied the class definition
    def fly(self):
        return "The frone is flying"
```

In [37]:
```
d = Drone()   # create a drone instance
print(d.fly())
```

```
The frone is flying
```

In [44]:
```
class Drone:    #defines a class

    power_system = "battery"    # this is a class attribute because it is
                                # insied the class definition
    def fly(self):              # this self means the instance specificly
                                # we can therefore, use it to call the class
                                # attributes
        return "The " + self.power_system + "-power drone is flying"
```

```
In [47]: d = Drone()   # create a drone instance
         print(d.fly())
```

```
The battery-power drone is flying
```

```
In [50]: d1 = Drone()
         d2 = Drone()
         d1.power_system = "dream"
         # notice we never create a d2.power_system
         print(d1.fly())   # we have the def function
         print(d2.fly())   # python will check to see if there is a class attribute if
                           # does not have a instance attribute
```

```
The dream-power drone is flying
The battery-power drone is flying
```

```
In [55]: class Drone:    #defines a class

             power_system = "battery"    # this is a class attribute because it is
                                         # insied the class definition
             def fly(self):              # this self means the instance specificly
                                         # we can therefore, use it to call the class
                                         # attributes
                 print( "The " + self.power_system +
                        "-power drone is flying at an altitude of " + str(self.altitude))

             def ascend(self,change):    # whatever the argument is we need to shif
                                         # it after the self!
                                         # also, notice, you dont need to use self,
                                         # you can use whatever you want, just change
                                         # everything from self to what you want.
                 self.altitude += change

         d = Drone()
         d.altitude = 0
         d.fly()
         d.ascend(100)
         d.fly()
```

```
The battery-power drone is flying at an altitude of 0
The battery-power drone is flying at an altitude of 100
```

1.2.3 Initializing a class

```
In [60]: class Drone:    #defines a class

             def fly(self):
                 print( "The drone is flying at", self.altitude,"feet.")

             def ascend(self,change):
                 self.altitude += change

         d1 = Drone()
         #d1.altitude = 0
         d1.fly()
         d1.ascend(100)
         d1.fly()
```

```
---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-60-45665b218e11> in <module>()
      9 d1 = Drone()
     10 #d1.altitude = 0
---> 11 d1.fly()
     12 d1.ascend(100)
     13 d1.fly()


<ipython-input-60-45665b218e11> in fly(self)
      2
      3     def fly(self):
----> 4         print( "The drone is flying at", self.altitude,"feet.")
      5
      6     def ascend(self,change):


AttributeError: 'Drone' object has no attribute 'altitude'
```

```
In [67]: class Drone:    #defines a class

             def __init__(self,altitude = 0):   #initializing the instance
                                                 # set = 0 gives it a default value
                 self.altitude = altitude

             def fly(self):
                 print( "The drone is flying at", self.altitude,"feet.")

             def ascend(self,change):
                 self.altitude += change

         d1 = Drone(100)
         #d1.altitude = 0
         d1.fly()
         d1.ascend(100)
         d1.fly()
         d2 = Drone()
         #d2.altitude = 0
         d2.fly()
```

```
The drone is flying at 100 feet.
The drone is flying at 200 feet.
The drone is flying at 0 feet.
```

1.2.4 Countring with Data Attributes

```
In [2]: class Drone:    #defines a class

            num_drones = 0
```

```
        def __init__(self,altitude = 0):   #initializing the instance
                                            # set = 0 gives it a default value
            self.altitude = altitude
            self.ascend_count = 0
            Drone.num_drones += 1
        def fly(self):
            print( "The drone is flying at", self.altitude,"feet.")

        def ascend(self,change):
            self.altitude += change
            self.ascend_count += 1

    d1 = Drone(100)
    print("Number of drones: ",Drone.num_drones)
    print("d1 ascend count", d1.ascend_count)
    d1.fly()
    d1.ascend(100)
    d1.ascend(100)
    print("d1 ascend count", d1.ascend_count)
    d1.fly()
    d2 = Drone()
    d2.fly()
    print("Number of drones: ",d1.num_drones)
    d2.fly()
    print("Number of drones: ",d1.num_drones)
```

```
Number of drones:  1
d1 ascend count 0
The drone is flying at 100 feet.
d1 ascend count 2
The drone is flying at 300 feet.
The drone is flying at 0 feet.
Number of drones:  2
The drone is flying at 0 feet.
Number of drones:  2
```

1.2.5 Direct Access versus Getters and Setters

```
In [3]: class Drone:

            num_drones = 0

            def __init__(self,altitude = 0):
                self.altitude = altitude
                self.ascend_count = 0
                Drone.num_drones += 1

            def fly(self):
                print( "The drone is flying at", self.altitude,"feet.")

            def ascend(self,change):
                self.altitude += change
                self.ascend_count += 1

        d1 = Drone(100)
```

```
        print("The Drone's altitude is",d1.altitude)
        d1.altitude = 300
        print("The Drone's altitude is",d1.altitude)
```

```
The Drone's altitude is 100
The Drone's altitude is 300
```

In [7]:
```
        # Getters and Setters method
        class Drone:

            num_drones = 0

            def __init__(self,altitude = 0):
                self.altitude = altitude
                self.ascend_count = 0
                Drone.num_drones += 1

            def fly(self):
                print( "The drone is flying at", self.altitude,"feet.")

            def ascend(self,change):
                self.altitude += change
                self.ascend_count += 1

            def get_altitude(self):
                return self.altitude

            def set_altitude(self,new_altitude):
                self.altitude = new_altitude

        d1 = Drone(100)
        print("The Drone's altitude is",d1.get_altitude())
        d1.set_altitude(300)
        print("The Drone's altitude is",d1.get_altitude())
```

```
The Drone's altitude is 100
The Drone's altitude is 300
```

In [9]:
```
        # Getters and Setters method

        class Drone:

            num_drones = 0

            def __init__(self,altitude = 0):
                self.altitude = altitude
                self.ascend_count = 0
                Drone.num_drones += 1

            def fly(self):
                print( "The drone is flying at", self.altitude,"feet.")

            def ascend(self,change):
                self.altitude += change
                self.ascend_count += 1
```

```
            def get_altitude(self):
                return self.altitude

            def set_altitude(self,new_altitude):
                if new_altitude < 0:
                    raise Exception("Drone cannot have a negative altitude")
                self.altitude = new_altitude


        d1 = Drone(100)
        print("The Drone's altitude is",d1.get_altitude())
        d1.set_altitude(-1)
        print("The Drone's altitude is",d1.get_altitude())
```

```
The Drone's altitude is 100
```

```
        ---------------------------------------------------------------------------

        Exception                                 Traceback (most recent call last)

        <ipython-input-9-9a373ec4926a> in <module>()
         28 d1 = Drone(100)
         29 print("The Drone's altitude is",d1.get_altitude())
    ---> 30 d1.set_altitude(-1)
         31 print("The Drone's altitude is",d1.get_altitude())


        <ipython-input-9-9a373ec4926a> in set_altitude(self, new_altitude)
         22     def set_altitude(self,new_altitude):
         23         if new_altitude < 0:
    ---> 24             raise Exception("Drone cannot have a negative altitude")
         25         self.altitude = new_altitude
         26


        Exception: Drone cannot have a negative altitude
```

```
In [11]: # Getters and Setters method

         class Drone:

             num_drones = 0

             def __init__(self,altitude = 0):
                 self.altitude = altitude
                 self.ascend_count = 0
                 Drone.num_drones += 1

             def fly(self):
                 print( "The drone is flying at", self.altitude,"feet.")

             def ascend(self,change):
```

```
            self.altitude += change
            self.ascend_count += 1

        def get_altitude(self):
            return self.altitude

        def set_altitude(self,new_altitude):
            if new_altitude < 0:
                raise Exception("Drone cannot have a negative altitude")
            self.altitude = new_altitude


    d1 = Drone(100)
    print("The Drone's altitude is",d1.get_altitude())
    d1.set_altitude(300)
    d1.altitude = -10    ################ How can we prevent this ?
    print("The Drone's altitude is",d1.get_altitude())
```

```
The Drone's altitude is 100
The Drone's altitude is -10
```

1.2.6 Hidden attributes

In [16]:
```
    # we solve the problem by using hidden attributes
    # if I want to make an attribute into a hidden one, add to __

    class Drone:

        num_drones = 0

        def __init__(self,altitude = 0):
            self.__altitude = altitude
            self.ascend_count = 0
            Drone.num_drones += 1

        def fly(self):
            print( "The drone is flying at", self.__altitude,"feet.")

        def ascend(self,change):
            self.__altitude += change
            self.ascend_count += 1

        def get_altitude(self):
            return self.__altitude

        def set_altitude(self,new_altitude):
            if new_altitude < 0:
                raise Exception("Drone cannot have a negative altitude")
            self.__altitude = new_altitude


    d1 = Drone(100)
    print("The Drone's altitude is",d1.get_altitude())
    d1.set_altitude(300)
    print("The drone's altitude is ", d1.altitude) # this will however gives an
```

```
                                                    # error
print("The Drone's altitude is",d1.get_altitude())
```

```
The Drone's altitude is 100
```

```
---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-16-4f4ba7cd2320> in <module>()
 30 print("The Drone's altitude is",d1.get_altitude())
 31 d1.set_altitude(300)
---> 32 print("The drone's altitude is ", d1.altitude) # this will however gives an
 33                                                    # error
 34 print("The Drone's altitude is",d1.get_altitude())


AttributeError: 'Drone' object has no attribute 'altitude'
```

1.2.7 Overriding Hidden Attributes

In [20]:
```
# we solve the problem by using hidden attributes
# if I want to make an attribute into a hidden one, add to __

class Drone:

    num_drones = 0

    def __init__(self,altitude = 0):
        self.__altitude = altitude
        self.ascend_count = 0
        Drone.num_drones += 1

    def fly(self):
        print( "The drone is flying at", self.__altitude,"feet.")

    def ascend(self,change):
        self.__altitude += change
        self.ascend_count += 1

    def get_altitude(self):
        return self.__altitude

    def set_altitude(self,new_altitude):
        if new_altitude < 0:
            raise Exception("Drone cannot have a negative altitude")
        self.__altitude = new_altitude


d1 = Drone(100)
print("The Drone's altitude is",d1.get_altitude())
d1._Drone__altitude = 300   #_classname__attribute  OVERRIDING!!
```

```
        print("The drone's altitude is ", d1._Drone__altitude) #_classname__attribute
        print("The Drone's altitude is",d1.get_altitude())
```

```
The Drone's altitude is 100
The drone's altitude is  300
The Drone's altitude is 300
```

1.2.8 Class Odds and Ends

Properties

```
In [22]: class Drone:
             def __init__(self,altitude = 0):
                 self.altitude=altitude
                 self.ascend_count = 0
             def fly(self):
                 print("The drone is flying at", self.altitude,"feet.")
             def ascend(self,change):
                 self.altitude += change
                 self.ascend_count += 1
         d1 = Drone(100)
         print("The Drone's altitude is", d1.altitude)
         d1.altitude = 300
         print("The Drone's altitude is", d1.altitude)
```

```
The Drone's altitude is 100
The Drone's altitude is 300
```

```
In [24]: class Drone:
             def __init__(self,altitude = 0):
                 self.__altitude = altitude
                 self.ascend_count = 0

             def fly(self):
                 print("The drone is flying at", self.__altitude,"feet.")

             def ascend(self,change):
                 self.__altitude += change
                 self.ascend_count += 1

             def get_altitude(self):
                 return self.__altitude

             def set_altitude(self, new_altitude):
                 if new_altitude < 0:
                     raise Exception("Drone cannot have a negative altitude.")
                 self.__altitude = new_altitude

             altitude = property(get_altitude,set_altitude)

         d1 = Drone(100)
         print("The Drone's altitude is", d1.altitude)

         d1.altitude = 300
         print("The Drone's altitude is", d1.altitude)
```

```
The Drone's altitude is 100
The Drone's altitude is 300
```

```
In [25]: d1.altitude = -10
```

```
---------------------------------------------------------------------------

Exception                                 Traceback (most recent call last)

<ipython-input-25-0611421cea68> in <module>()
----> 1 d1.altitude = -10


<ipython-input-24-cddab28ae739> in set_altitude(self, new_altitude)
     16     def set_altitude(self, new_altitude):
     17         if new_altitude < 0:
---> 18             raise Exception("Drone cannot have a negative altitude.")
     19         self.__altitude = new_altitude
     20


Exception: Drone cannot have a negative altitude.
```

```
In [28]: class Drone:
             def __init__(self,altitude = 0):
                 self.__altitude = altitude
                 self.ascend_count = 0

             def fly(self):
                 print("The drone is flying at", self.__altitude,"feet.")

             def ascend(self,change):
                 self.__altitude += change
                 self.ascend_count += 1

             @property
             def altitude(self):
                 return self.__altitude

             @altitude.setter
             def altitude(self, new_altitude):
                 if new_altitude < 0:
                     raise Exception("Drone cannot have a negative altitude.")
                 self.__altitude = new_altitude


         d1 = Drone(100)
         print("The Drone's altitude is", d1.altitude)

         d1.altitude = 300
         print("The Drone's altitude is", d1.altitude)

         d1.altitude = -10
```

```
The Drone's altitude is 100
The Drone's altitude is 300
```

```
---------------------------------------------------------------------------

Exception                                 Traceback (most recent call last)

<ipython-input-28-219880794b0f> in <module>()
 28 print("The Drone's altitude is", d1.altitude)
 29
---> 30 d1.altitude = -10


<ipython-input-28-219880794b0f> in altitude(self, new_altitude)
 18     def altitude(self, new_altitude):
 19         if new_altitude < 0:
---> 20             raise Exception("Drone cannot have a negative altitude.")
 21         self.__altitude = new_altitude
 22


Exception: Drone cannot have a negative altitude.
```

1.3 Class Methods and Static Methods

```
In [31]: class Drone:

             def __init__(self,altitude = 0):
                 self.altitude = altitude
                 self.ascend_count = 0

             def fly(self):
                 print("The drone is flying at", self.altitude,"feet.")

             def ascend(self,change):
                 self.altitude += change
                 self.ascend_count += 1

             @classmethod
             def print_class(cls):
                 print(cls)
         d1 = Drone(100)
         d1.print_class()
```

```
<class '__main__.Drone'>
```

```
In [32]: Drone.print_class()
```

```
<class '__main__.Drone'>
```

```
In [36]: class Drone:
             __num_drones = 0
```

```
            def __init__(self,altitude = 0):
                self.altitude = altitude
                self.ascend_count = 0
                Drone.__num_drones +=1

            def fly(self):
                print("The drone is flying at", self.altitude,"feet.")

            def ascend(self,change):
                self.altitude += change
                self.ascend_count += 1
            @classmethod
            def get_num_drones(cls):
                return cls.__num_drones
        d1 = Drone(100)
        print(d1.get_num_drones())
        d2 = Drone(200)
        print(d2.get_num_drones())
```

```
1
2
```

```
In [37]: class Drone:
            __num_drones = 0

            def __init__(self,altitude = 0):
                self.altitude = altitude
                self.ascend_count = 0
                Drone.__num_drones +=1

            def fly(self):
                print("The drone is flying at", self.altitude,"feet.")

            def ascend(self,change):
                self.altitude += change
                self.ascend_count += 1
            @classmethod
            def get_num_drones(cls):
                return cls.__num_drones

            @staticmethod
            def feet_from_meters(meters):
                return meters * 3.28084

        d1 = Drone(100)
        d1.altitude = Drone.feet_from_meters(200)
        print(d1.altitude)
```

```
656.168
```