Class 7 - Classes

[W200] MIDS Python Bridge Course Summer 2018

Course Content | moving into **OOP**

Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Midterm review (10% of grade)

Mid Semester Survey!

- These surveys are a way you as students can give direct feedback to the administration and instructors.
- We read each one and change the course based on your comments!
- Examples of changes that came about because of student feedback:
  - Hiring a TA!
  - Ensuring grading and feedback is given in a timely manner
  - Course structure and lecture areas (numpy & pandas)
- SURVEY LINK:
  - Posted in chat!

Assignment Review | Week 6

Refresher:

1. Pseudocode for scrabble?
2. Scrabble implementation
3. PEP 8 reading

Week 7 | Polls

Discuss: What was the hardest part of HW6?

Poll: How long did you spend on this week’s assignment?

Poll: what were your times for the scrabble assignment

Homework 5 Grading

- Overall: Good work!
- Printing inside functions:
    - Generally not done - functions should return the answer rather than print from inside
    - Reasoning: The user can't turn the printing off or modify the output if they want to print something different. If the function returns the value the user/programmer can decide how to use it
    - One way to do both - make a Flag for the printing (some functions use a verbose flag as an argument to signal if the user wants the printing to happen or not)
    - If you printed inside a function for scrabble homework - don't worry about changing it but keep it in mind for future functions!

Homework 5 Grading

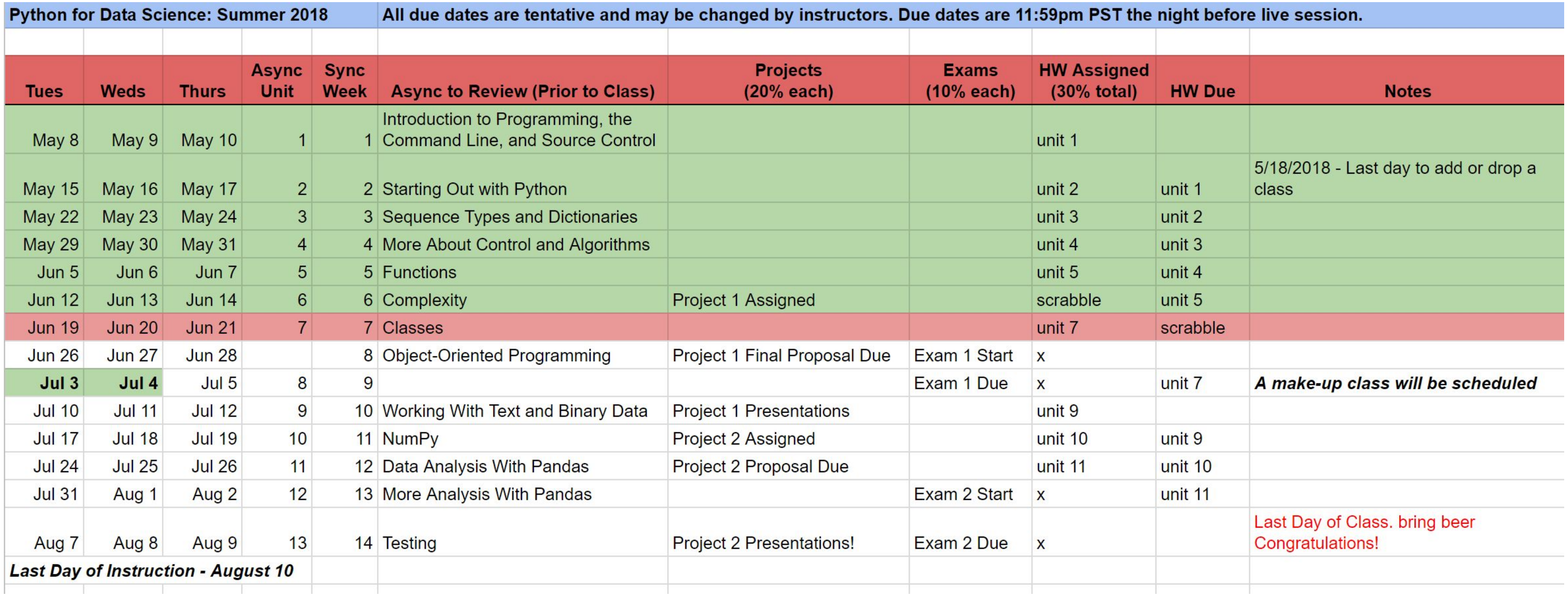
- Error checking inside functions:
    - Also usually not done; functions have docstring comments that tell a programmer what inputs the function requires
    - Functions are used by programmers; there is some expectation that a programmer will be able to read the docstring and figure out what to send to a function.
    - Reasoning: Error checking on every argument on every function adds a lot of lines of code + processing time
- BL: Need to error check a user’s inputs but generally not a programmers

Homework 7: Classes

- There are 3 programming questions:
  - Deck of Cards, Galton’s Box, Sorting Marbles
  - Please do any 2 of the 3

Reminders | Course Schedule

https://docs.google.com/spreadsheets/d/11DxadnNwyFaJl
PYLUJSPUlNGCtTenBCR4yaR1CbFBKg



Python for Data Science: Summer 2018 **All due dates are tentative and may be changed by instructors. Due dates are 11:59pm PST the night before live session.**

Tues	Weds	Thurs	Async Unit	Sync Week	Async to Review (Prior to Class)	Projects (20% each)	Exams (10% each)	HW Assigned (30% total)	HW Due	Notes
May 8	May 9	May 10	1	1	Introduction to Programming, the Command Line, and Source Control			unit 1		
May 15	May 16	May 17	2	2	Starting Out with Python			unit 2	unit 1	5/18/2018 - Last day to add or drop a class
May 22	May 23	May 24	3	3	Sequence Types and Dictionaries			unit 3	unit 2	
May 29	May 30	May 31	4	4	More About Control and Algorithms			unit 4	unit 3	
Jun 5	Jun 6	Jun 7	5	5	Functions			unit 5	unit 4	
Jun 12	Jun 13	Jun 14	6	6	Complexity	Project 1 Assigned		scrabble	unit 5	
Jun 19	Jun 20	Jun 21	7	7	Classes			unit 7	scrabble	
Jun 26	Jun 27	Jun 28		8	Object-Oriented Programming	Project 1 Final Proposal Due	Exam 1 Start	x		
Jul 3	**Jul 4**	Jul 5	8	9			Exam 1 Due	x	unit 7	***A make-up class will be scheduled***
Jul 10	Jul 11	Jul 12	9	10	Working With Text and Binary Data	Project 1 Presentations		unit 9		
Jul 17	Jul 18	Jul 19	10	11	NumPy	Project 2 Assigned		unit 10	unit 9	
Jul 24	Jul 25	Jul 26	11	12	Data Analysis With Pandas	Project 2 Proposal Due		unit 11	unit 10	
Jul 31	Aug 1	Aug 2	12	13	More Analysis With Pandas		Exam 2 Start	x	unit 11	
Aug 7	Aug 8	Aug 9	13	14	Testing	Project 2 Presentations!	Exam 2 Due	x		Last Day of Class. bring beer Congratulations!

Last Day of Instruction - August 10

Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Midterm review (10% of grade)

The Project | Proposal (Due before next class)

Describe your project concept

Pseudocode your major classes and functions

1. Briefly describe the purpose of each class
2. List expected functions belong to each class
3. List inputs and outputs for each function

Instructors will “approve” your draft proposal

Coding is **iterative**. Your final code may not match the proposal exactly

Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Midterm review (10% of grade)

Classes (types) | ready to go vs custom

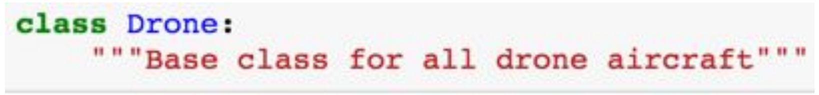
- We are familiar with base python classes.
  - **ints** and **strs**, to **lists**, **sets** and **dicts**.
- What are Classes?
  - Templates conferring a shared form
  - Instantiation uses a class definition to make a distinct **object**
  - Objects of a common class(type) contain distinct data

Classes | ready to go vs custom

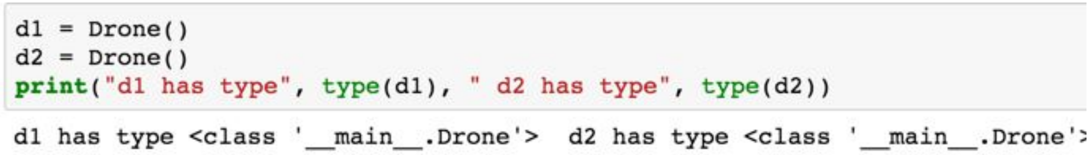
- Why create your own types?
    - Keep the “data” (attributes) with the “functions” (methods)
    - Extend the language
    - Can be tailored to hold new data or execute new tasks
    - Don’t just store data – objects interact:
        - Execute internal functions (class methods)
        - Manage other objects
            - Creation
            - Modification
            - Execution
            - interaction

Class construction | the basics

- Now we can form a **base class**
- **Instantiate** individual objects from the **base**
- Modify **attributes** for all instances
- Modify **attributes** of individual instances

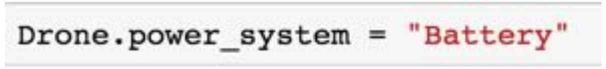


```
class Drone:
    """Base class for all drone aircraft"""
```

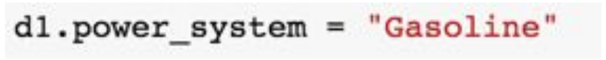


```
d1 = Drone()
d2 = Drone()
print("d1 has type", type(d1), " d2 has type", type(d2))
```

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



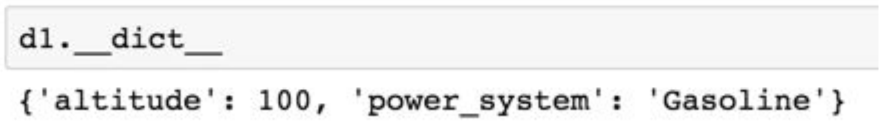
```
Drone.power_system = "Battery"
```



```
d1.power_system = "Gasoline"
```

Class information |

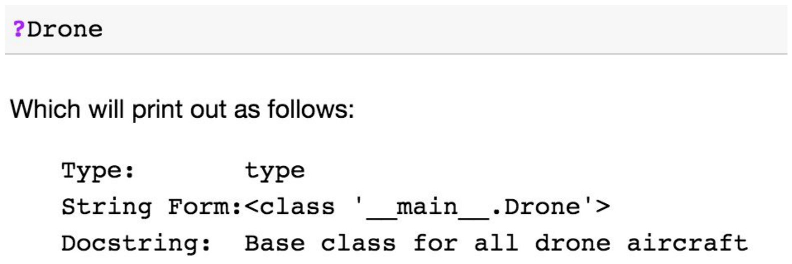
- dir(d1) # class methods
- d1.__dict__ # attribute information



```
d1.__dict__
```

```
{'altitude': 100, 'power_system': 'Gasoline'}
```

- ?Drone # class documentation



```
?Drone
```

Which will print out as follows:

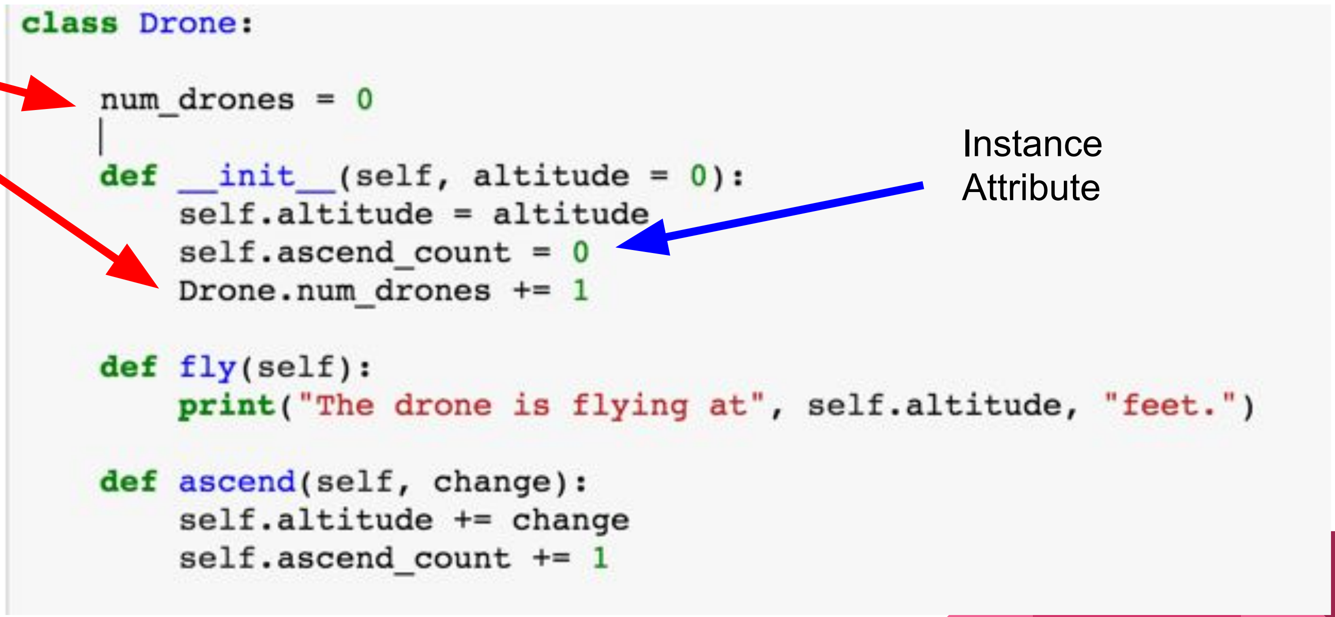
```
Type:        type
String Form:<class '__main__.Drone'>
Docstring:   Base class for all drone aircraft
```

Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Midterm review (10% of grade)

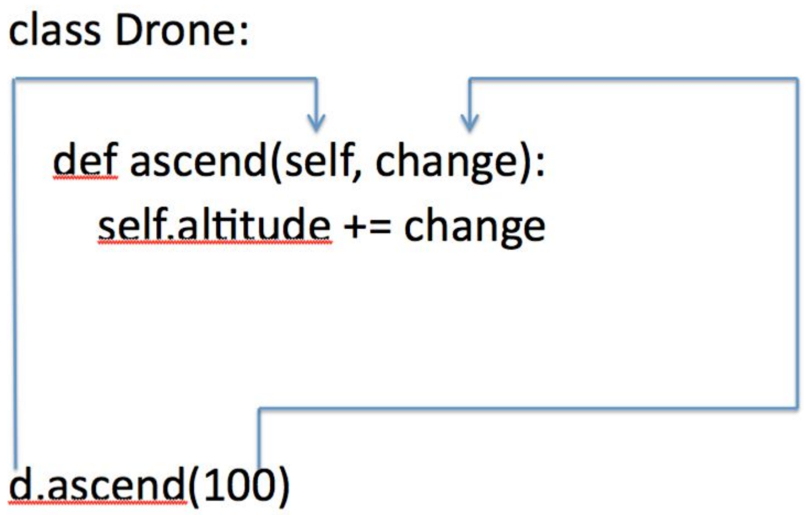
Attributes | class vs. individual



```
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
```

Methods | class-specific functions

- The **method** "ascend" is a like a function bound to objects of the class Drone
- You call ascend on **instance d**, a type **Drone object**
- The first argument (self) is required and binds the method and result to the instance d



```
class Drone:

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


d.ascend(100)
```

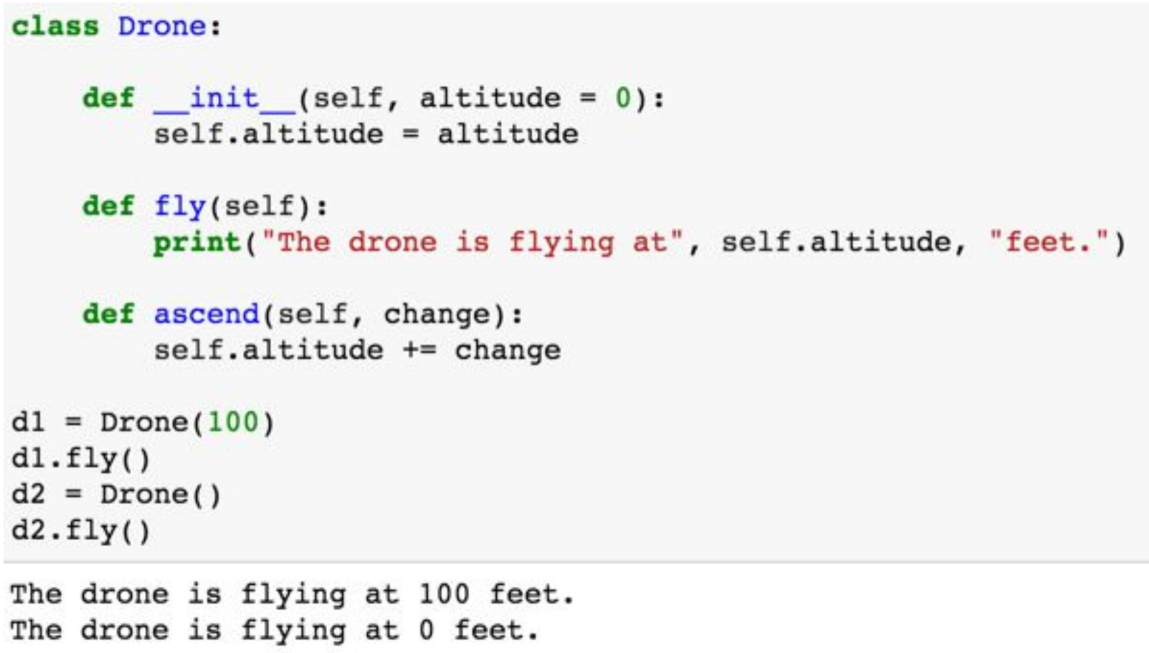
Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Initialize | require attributes at instantiation

- Instantiation runs the **__init__** method

- Altitude is established at initialization and has a default value



```
class Drone:

    def __init__(self, altitude = 0):
        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.fly()
d2 = Drone()
d2.fly()
```

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

Week 7 | Agenda

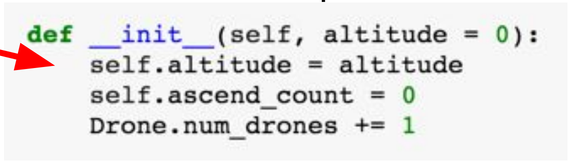
Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Get and set | require attributes at instantiation

- More explicit than direct attribute access

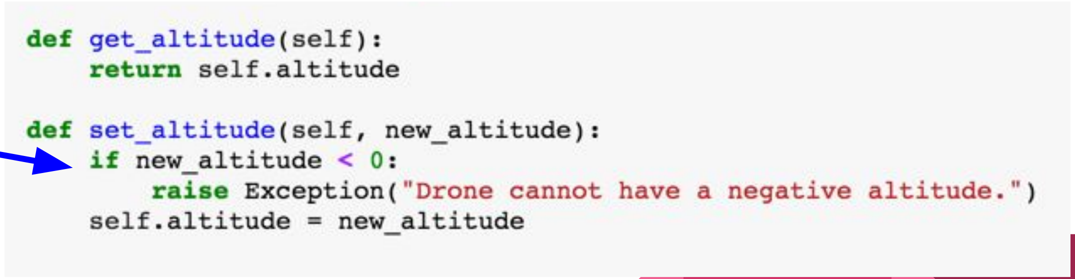
- We can add code into the get and set method

Direct a attribute specification



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

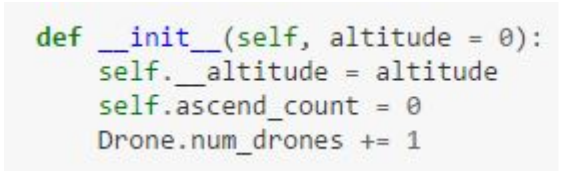
Using get and set for attributes



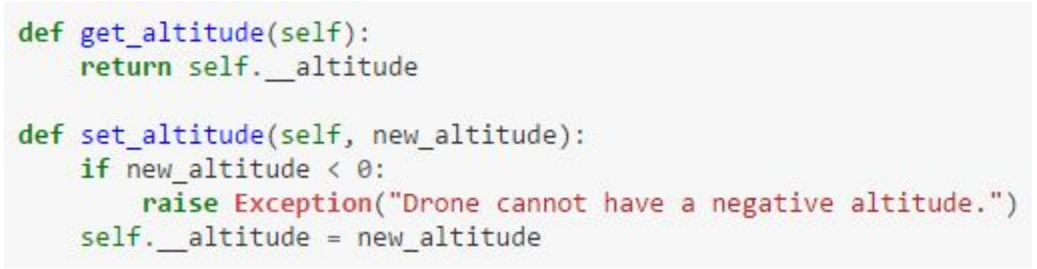
```python
def get_altitude(self):
    return self.altitude

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

Hidden names | access, modify

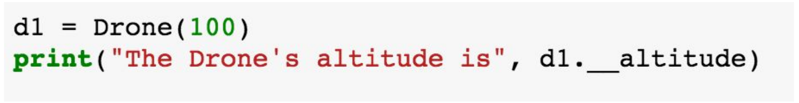
- You can use the "__" prefix to "require" programmers to use your setter and getter methods
- It uses set_altitude automatically
- They can override it via: `d1._Drone__altitude`



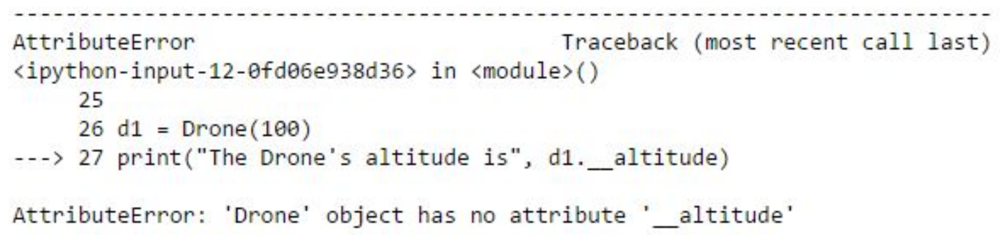
```python
def __init__(self, altitude = 0):
    self.__altitude = altitude
    self.ascend_count = 0
    Drone.num_drones += 1
```



```python
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
```



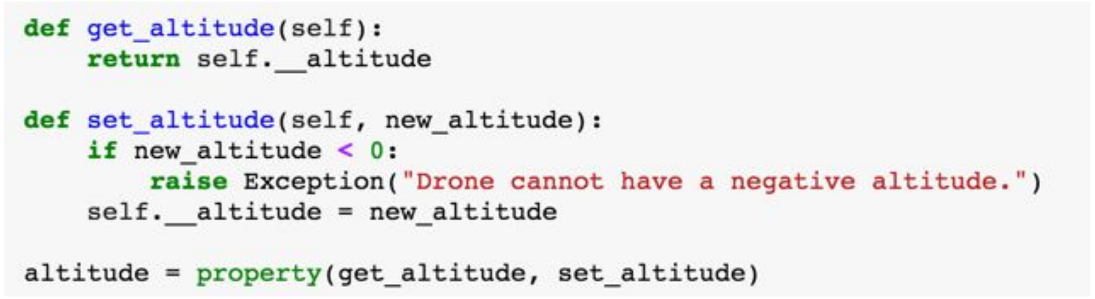
```python
d1 = Drone(100)
print("The Drone's altitude is", d1.__altitude)
```



```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-0fd06e938d36> in <module>()
     25
     26 d1 = Drone(100)
---> 27 print("The Drone's altitude is", d1.__altitude)

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

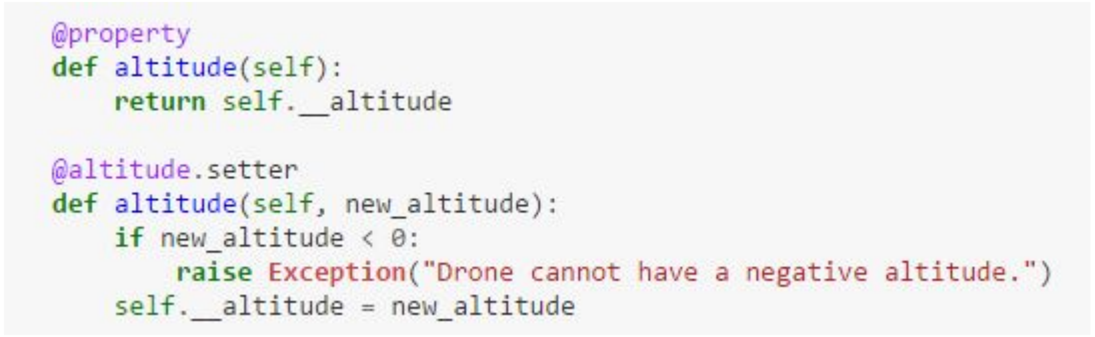
Properties and decorators |

- Properties allow you to apply a setter and getter “after the fact”



```python
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)
```

- Decorators start with @ and flag certain functions. You can use them to flag properties.
  - “set” is implicit



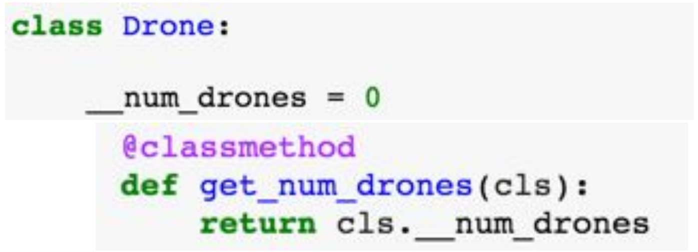
```python
@property
def altitude(self):
    return self.__altitude

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

Other method types | declared with decorators

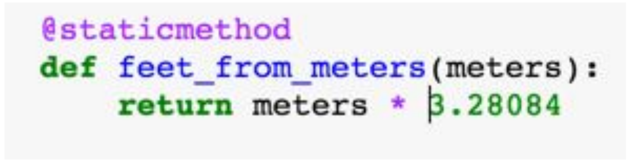
- These decorators don’t do anything except **tell us what to expect** from the method

- **Class methods** affect class - level attributes

- **Static methods** do not affect attributes



```python
class Drone:

    __num_drones = 0
    @classmethod
    def get_num_drones(cls):
        return cls.__num_drones
```



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

Week 7 | Agenda

Homework Review and Admin
Project 1 Proposal
Classes (Objects) Structure and Purpose
Attributes and Methods
Initialization (and “self”)
Getters, Setters and Decorators
Project 1 Breakout and Recap

Class | A quick discussion

- You will learn about “Inheritance” and “Polymorphism” this week. The plan you’ve created today may need to be modified to take advantage of these two concepts.

Inheritance - Allows a “child” class to inherit attributes and functions from a “parent” class. The child class can be customized, but you can change all children classes at once by modifying the parent.

Polymorphism - Allows a function to work on multiple types of object. Different classes can share the same interface, which allows a single function to accept multiple types of object.

Class | Breakout 1 discuss your plan in words

- Read the the first part of this:
  - http://web.archive.org/web/20160816041541/http://learnpythonthehardway.org/book/ex43.html
- Think of your classes;
  - objects as nouns
  - methods as verbs
  - How will objects interact

Class | Breakout 1 discuss your plan in words

- Think about managing classes:
    - Do you need classes that organize /score object interactions?
    - (e.g., a ‘battle engine’ object? A ‘scoreboard’ object?)

- Think about your user:
    - What will they be tasked with
    - What data will they be able to get

- Critique, question, respond ...

The Project | Your Mission

Create a small, object-oriented program of your choosing:

Examples:

- An ATM
- A flower shop
- An adventure game
- Something relating to your everyday work

The Project | Code

Python 3 code, 300-500 lines (750 max)

All code should be well commented!

Must use Object Oriented design and classes

Demonstrate various flow controls and data types

Robust to common user errors and exceptions

The Project | Your Mission

The user will interact with your program via Terminal/Shell

Three documents due before your class on 3/13 or 3/15:

1. Proposal (10%)
2. Code(s) (80%)
3. Reflective Summary (10%)

You will demo your progress in a breakout room (3/6 or 3/8)

You may only use Python libraries that come installed with Anaconda

The Project | Proposal

Describe your project concept

Pseudocode your major classes and functions

1. Briefly describe the purpose of each class
2. List expected functions belong to each class
3. List inputs and outputs for each function

Instructors will “approve” your draft proposal

Coding is **iterative**. Your final code may not match the proposal exactly

The Midterm | Content

All work done in a Jupyter Notebook

Covers Units 1 - 6

Many questions are theory based (short answer)

Also some coding problems

Designed to be completed in a couple of hours

The Project | Questions

The Project | Reflection

Submit a 1-page reflection with your code

Instructors will read your reflection before grading your project

Tell us how to use your project!

Discuss challenges you faced and how you overcame them

The Project | Demo

As time allows, show 1-2 examples of strong projects from last semester.

Midterm Review

Live Q & A using Poll Features