LESSON 10 Classes II

Inheritance

- Create as an example a WeightedDiceSet class that has a different roll() implementation using random.choices. ← This is 3.6 only!!! See last page!
- Use super() with __init__ to call DiceSet init

i.e. random.choices(~~cod~~ range(1, sides+1), weights=weights, k=number)

then use sum()

mention explicit super usage, i.e. super(WeightedDiceSet, self)

Multiple Inheritance and MRO

```
class ~~Bar~~ Foo:
    def spam(self):
        print('foo spam')

Class Bar(Foo):
    def spam(self):
        print('bar spam')
        super().spam()

class Baz(Foo):
    def spam(self):
        ...

class Quux(Bar, Baz)
    def spam(self):
        ...
```

make a Quux, call Spam, see what happens.

Check Quux.mro()

Class and static methods ← leave decorators as "magic" for now, explain briefly higher-order functions.

Add a custom constructor to DiceSet much like datetime.now() ← show example.

```
@classmethod
def tds(cls):          ← make a set of 3 six-sided dice.
    return cls(3, 6)
```

Also explain @staticmethod, explain reduced need in Python but possible convenience (module namespacing).

Property Decorator

Do an example with number of dice being verbose!

```
@property
def number(self):
    print("fetching # of dice")
    return self._number

@number.setter                          also implement
def number(self, value):                @number.deleter
    print("setting # to {}".format(value))
    self._number = value
```

Double Underscore variables (I dont use these

make a __foo variable in DiceSet, look at it in a WeightedDiceSet, etc...

Intro to the dunders

Make sure to highlight the language reference

↳ first do __repr__ and __str__ for diceset.
(if we missed this)

Next go over __add__ for diceset, only allow adding DiceSet objects with the same base and sides.

↳ Discuss implications for Weighted Dice Set, ~~add~~ see what happens when they are added.

~~Also do __iadd__~~

Finally explore __getitem__ and __setitem__ to allow access to the weights on a Weighted Dice Set as & subscripts.

Note: random.choices() is python 3.6+, use this helper function instead

```
def _weighted_choice(weights):
    '''
    Return a random list index from list of weights by weight
    '''
    Val = random.random * sum(weights)
    for idx, weight in enumerate(weight)!
          if val < weight:
               return idx
          val -= weight
     return idx
```

$\nwarrow$

or write this as a method of WeightedDiceSet