# Lesson 9 Classes I

Start with example 'datetime' class.

- ↳ show example of object construction
- ↳ attribute access (day, month, ...)
- ↳ method access ( strftime( '%A %y-%m-%d') )
- ↳ type() on various things to understand them

Two focuses today! ① How to make classes (custom types)

② Why one could want to ~~do~~ do this in python, not a strictly OO language?

↑ We will move back and forth

① Making a minimal class 'Widget'

```
class Widget!
    ''' a minimal class '''
```

↑ No go over constructing widgets, adding attributes, modifying / accessing attributes, and deleting them. Also do bool(), str(), show that int() doesnt work, math operations fail, etc..

② A motivating example to learn more - dice sets

Lets make a 'dice' module with the following functions

```
def roll ( number, sides , base = 1 )
    ↑ roll 'number' dice with 'sides' sides and
      the numbering on the sides increasing by 'base'
          ( give examples)
```

def roll_many ( attempts, number, sides, base = 1 )

" do many rolls of a set and add them up

def count_attempts ( value, attempts, number, sides, base = 1, max tries = 1000000 )

" roll until 'value' has been attained or raise an exception after max tries.

Now implement these as functions using random

---

Introduce class methods by implementing the above in a 'Dice Set' class. (mention camel case!)

leave out count_attempts this is homework!

- First do this without an '__init__' just explain first parameter (usually self).
- Then add an __init__ explain object initialization.
- Mention setter/getter issue, why not needed in python generally.

---

Class attributes and resolution order

- Add a class attribute to widget class show how ~~the~~ an object attribute will 'shadow' this when accessing the attribute.

---

Scope: mention that a class declaration creates a class and namespace.

---

go over __str__ and __repr__ special methods!!