U07JGLLKF : <@U5VEXNQE7> did you mean like chaining methods?
U5VEXNQE7 : Yes.
U5VEXNQE7 : I've looked into @property methods @staticmethods
U5VEXNQE7 : Just can't figure it out
U07JGLLKF : `@staticmethod` just means a function doesn't require an instance of itself to operate on
U07JGLLKF : the key to chaining methods is that you have to return a copy of `self` on chainable methods
U5VEXNQE7 : What do you mean
U07JGLLKF : ```class Foo:
    def __init__(self):
        self.data = [] # whatever your initial data here would be
        return self
    def Function(self, data):
        self.data = do_thing_with_data(data)
        return self
    def toJSON(self):
        return json.dumps(self.data)
```

U07JGLLKF : something like that would have basic chainable class init and `Function` methods
U5VEXNQE7 : like just the fuction needs something like this?'''
class CLASS(object):
    _x = None
    def __init__(self):
        pass
    def something_to_do(self, var1, var2):
        x = var1 + var2
        return x
    @property
    def toFloat(self):
        return float(_x)
    '''

U1BP42MRS : ```class Foo:
    def bar(self):
        # do work
        return self

    def baz(self):
        # do other stuff
        return self

    @property
    def json(self):
        return self.__dict__

Foo().bar().baz().json
```

U5VEXNQE7 : Awesome!
U5VEXNQE7 : I didn't try that ... why didn't I try that?
U5VEXNQE7 : Hmm this didn't work. ```class CLASS(object):
    _x = None
    def __init__(self):
        pass
    def something_to_do(self, var1, var2):
        x = var1 + var2
        return x
    @property
    def toFloat(self):
        return float(_x)

print(CLASS().something_to_do(1,2).toFloat)```

U5VEXNQE7 : ```Traceback (most recent call last): File "Untitled 2.py", line 15, in <module>
    print(CLASS().something_to_do(1,2).toFloat)
AttributeError: 'int' object has no attribute 'toFloat'
```

U07JGLLKF : your `return x` needs to be `return self`
U07JGLLKF : because `return x` is just returning the result of `var1 + var2`, which is just an int
U5VEXNQE7 : ```class CLASS(object):    _x = None
    def __init__(self):
        pass
    def something_to_do(self, var1, var2):
        _x = var1 + var2
        return self
    @property
    def toFloat(self):
        return float(_x)
print(CLASS().something_to_do(1,2).toFloat)

```

U5VEXNQE7 : ```Traceback (most recent call last): File "Untitled 2.py", line 15, in <module>
    print(CLASS().something_to_do(1,2).toFloat)
  File "Untitled 2.py", line 12, in toFloat
    return float(_x)
NameError: name '_x' is not defined
```

U07JGLLKF : ahh, so `_x` in that scope doesn't exist
U07JGLLKF : that will need to be `self._x = var1 + var2`
U5VEXNQE7 : Wow .. ok .. that one I knew .. just too focused on other things ..
U5VEXNQE7 : Just completely missed it like a noob.
```