APS106

UNIVERSITY OF TORONTO

More OOP! Encapsulation and Examples

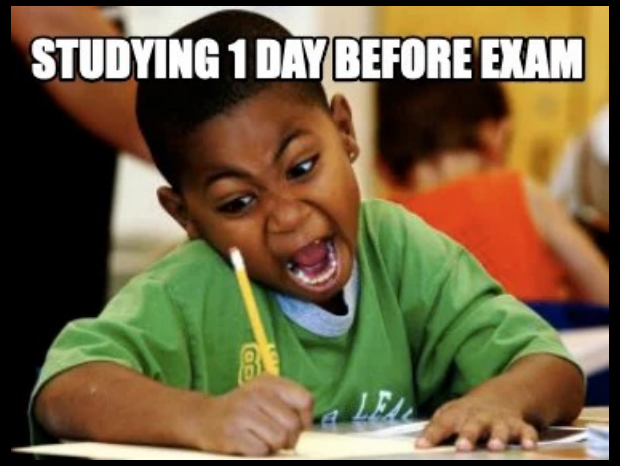
Week 7 | Lecture 2 (7.2)

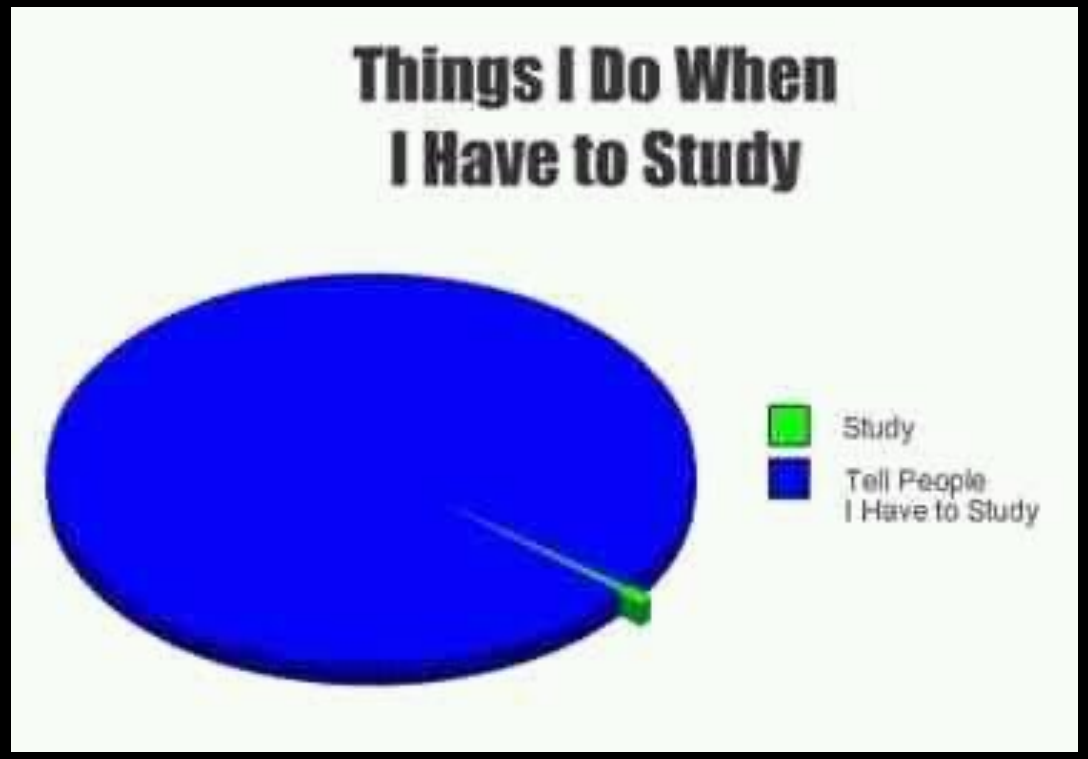
if nothing else, write #cleancode

Today's Content

- Lecture 7.2
  - More OOP! Encapsulation and Examples

Exam: June 24, 2:00 PM
Wallberg Building (WB) 116



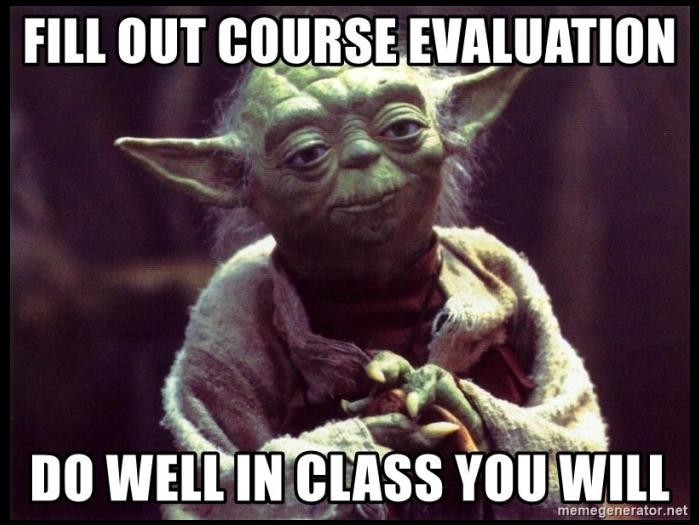


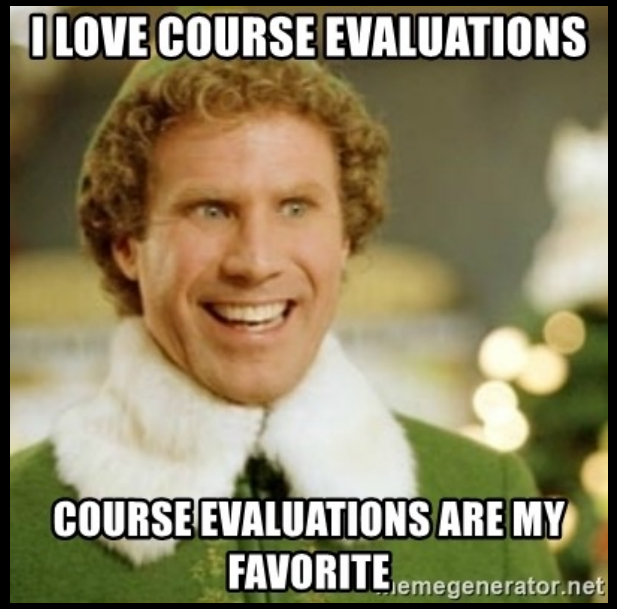
Coffee Break!

- Extra help hours!
  - TIP FOR UNIVERSITY SUCCESS: Put in calendar, treat as a scheduled class, and bring one question you are confused about
- Link is same as Lecture Zoom Room
- Wednesday 1-3 PM

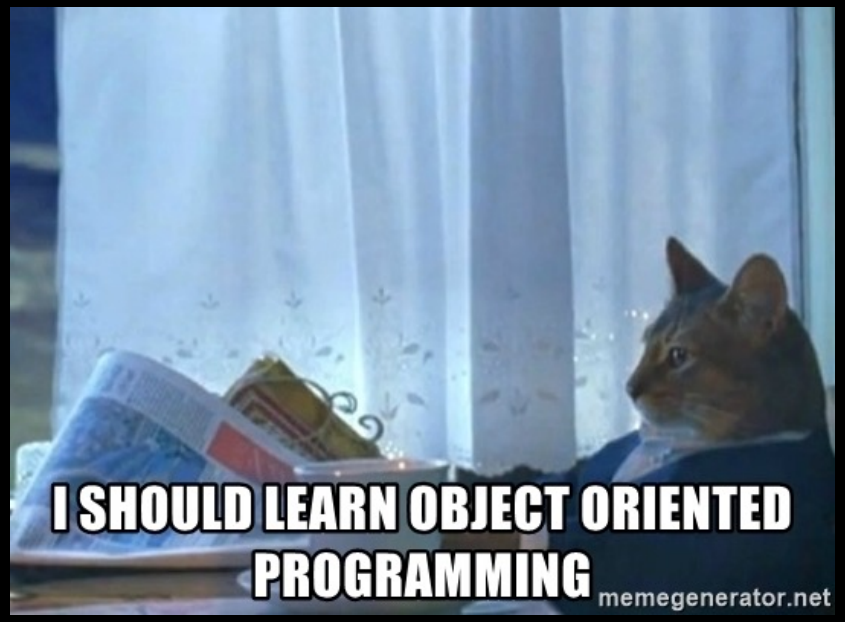
Course Evaluations

- Extremely helpful! Every comment is read by the instructor and their department
- You can help determine who teaches this course and how it's taught in the future!
- Please take the time to complete the course evaluation by June 21, 2022 (it's only 5 minutes)
- Quercus ->Course Evals tab on left ->
  - https://q.utoronto.ca/courses/48756/external_tools/294





Why OOP?



- Models our real-life thinking
  - Sandwich – ingredients, freshness, etc.
  - Car – model, year, fuel level, forward, reverse etc.
  - Movie – actors, director, genre, rating, etc.
  - Cat – weight, name, colour, scratch, meow, sleep, etc.

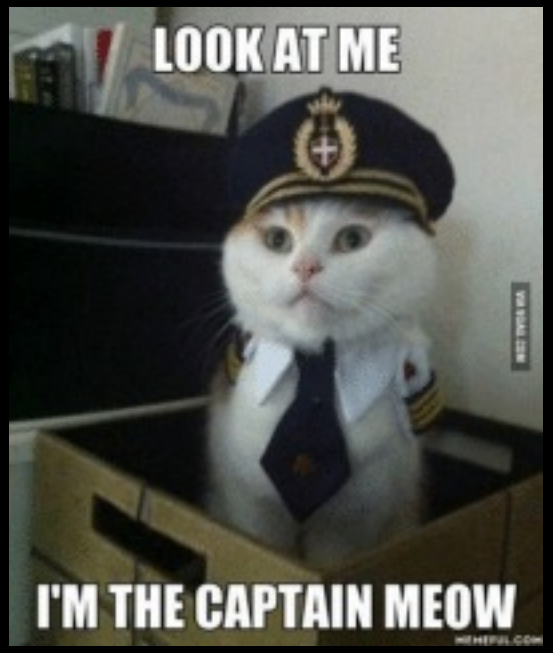
- Why do we use it in programming?
  - Default values for a new object (initialization)
  - Properly set-up, predictable behaviours of different objects within the same class

Why OOP?

- The reason for this dot operator convention is an implicit metaphor:
- The syntax for a function call, Cat.meow(kitty) suggests that the function is the active agent. It says something like, *"Hey meow function from the Cat class! Here's a Cat object (named kitty) for you to meow with."*
- In OOP, the objects are the active agents. A method invocation like kitty.meow() says *"Hey kitty! Please meow."*

```
my_point = Point()
Point.print_point(my_point)
my_point.print_point()
```

```
(0, 0)
(0, 0)
```

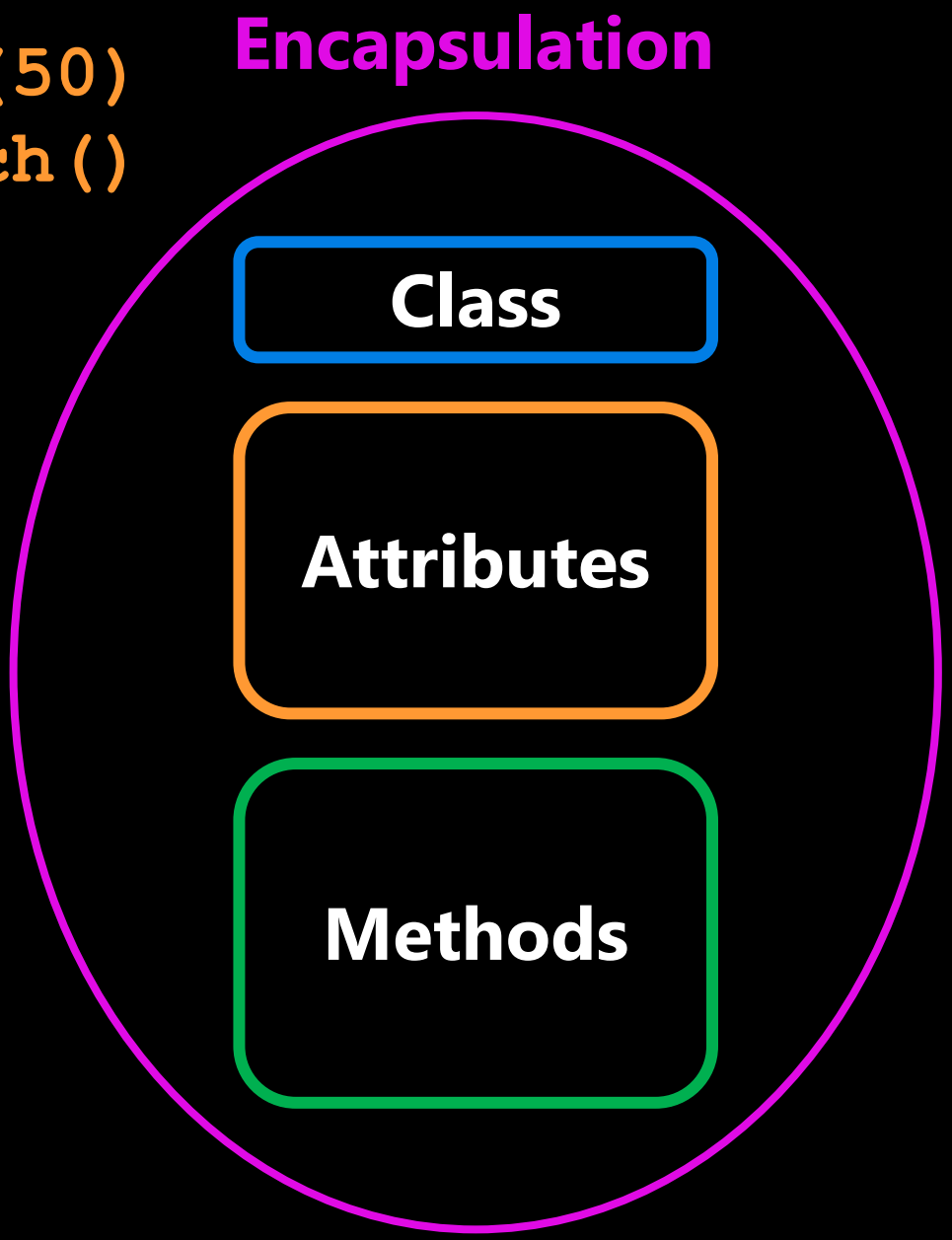



```
kitty = Cat()
Cat.meow(kitty)
kitty.meow()
```

Encapsulation

```
seb.forward(50)
kitty.scratch()
```

- The core of object-oriented programming is the organization of the program by **encapsulating** related data and functions together in an object.
- To encapsulate something means to enclose it in some kind of container.
- In programming, encapsulation means keeping **data** and the **code** that uses it in one place and hiding the details of exactly how they work together.



Encapsulate it!

- For example, each instance of class `file` keeps track of which file on the disk it is reading/writing and where it currently is on that file.
- The class hides the details of how it is done, so we (as programmers) can use it without needing to know how it is implemented

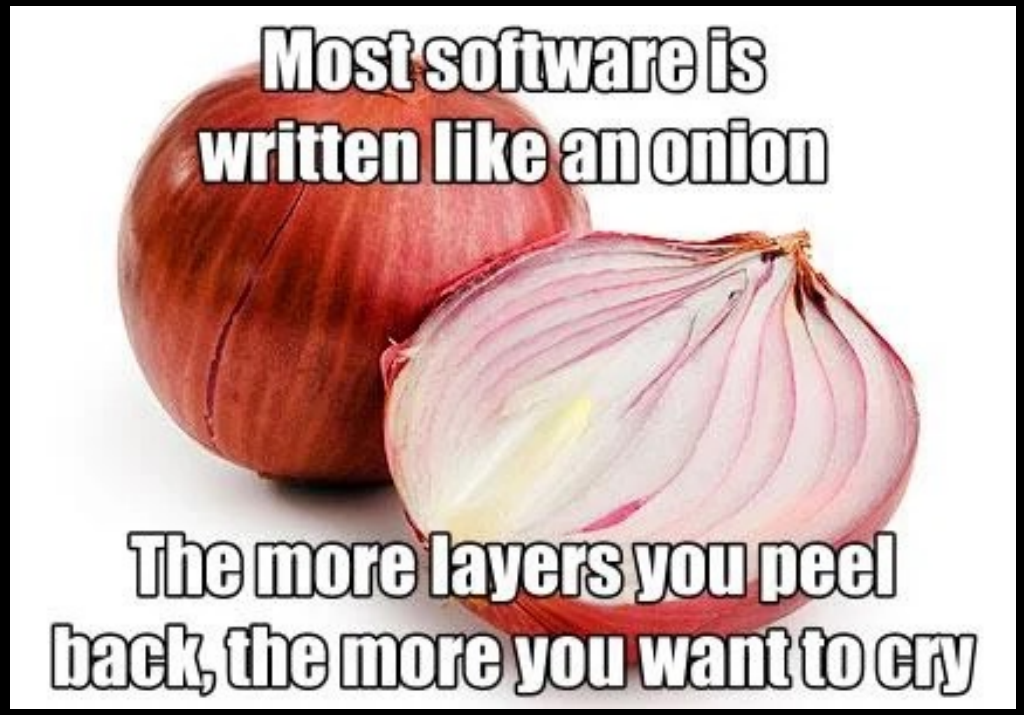
```
f = open('test.txt','w')
f.write('hola')
f.close()
```


```
print(f)
```

```
<_io.TextIOWrapper name='test.txt' mode='w' encoding='UTF-8'>
```

Let's Code!

- Let's look at how this works in Python!
  - Point Class Recap
  - Rectangle Class



Open your notebook

Click Link:
1. Encapsulation

Printing Information

- It would be nice to not have to write a print statement when we want to display attribute information:

```
>>> p = Point(3,4)
>>> print(p)
<__main__.Point object at 0x7fd8100778b0>
```

```
>>> p = Point(3,4)
>>> print(p.x, p.y)
3 4
```

- Is there a better way to encapsulate this? How about a method?

```
>>> p = Point(3,4)
>>> p.to_string()
'(3,4)'
>>> r = p.halfway(Point(5,12))
>>> r.to_string()
'(4,8)'
```

Printing Information

- If we had a method, we just need to format the attributes into a string:

```
class Point:

"""A class that represents and manipulates 2D points"""

    def to_string(self):
        """
        (self) -> None
        Prints the (x, y) coordinate for the point.
        """
        print('(' + str(self.x) + ', ' + str(self.y) + ')')
```

```
>>> p = Point(3,4)
>>> p.to_string()
'(3,4)'
>>> r = p.halfway(Point(5,12))
>>> r.to_string()
'(4,8)'
```

- We could make a similar method for any class (ex: Square)

Printing Information

- If every class we made had a to_string() method, great!
  - Easy to remember
  - All formatting specific to printing an object would be encapsulated
  - But they won't all have one…

- What if other people made their own method for printing and called it something else?
  - Then it would no longer be obvious how to quickly display information
  - Is it called to_string? tostring? 2stringz? print_point? plz_print_me?

Python has a clever trick for this!

Printing Information

- If we call our new method __str__ instead of to_string, Python will use our code whenever it needs to convert Point to a string

```
class Point:

"""A class that represents and manipulates 2D points"""
    def __init__(self, x=0 , y=0):
        ...
```

```
    def __str__(self):
        return ('(' + str(self.x) + ',' + str(self.y) + ')')
```

```
>>> p = Point(3,4)
>>> print(p)
'(3,4)'
```

That's better!
When we call `print(obj)`, then `obj.__str__()` is called to find out what string to print

Let's Code!

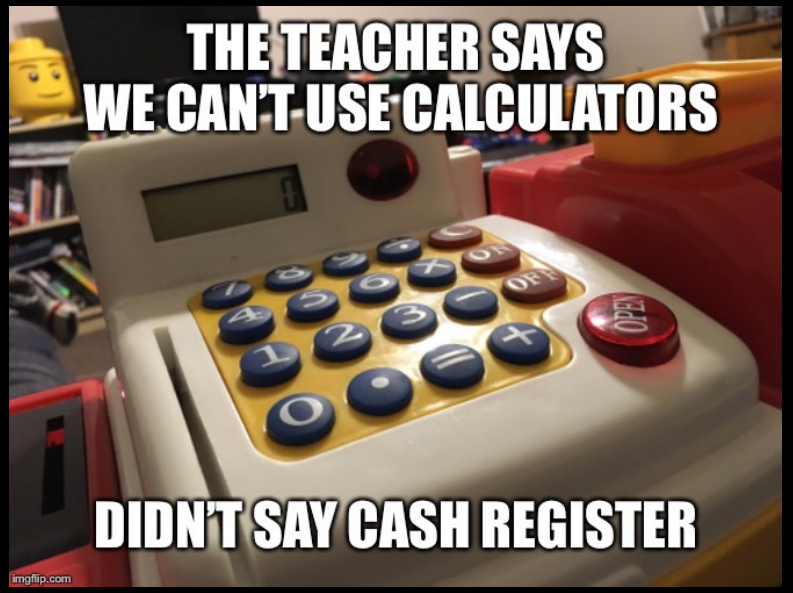
- Let's look at how this works in Python!
  - __str__ method

Open your notebook

Click Link:
2. Printing Objects

Let's Build a Cash Register!

- We want to build a cash register program. What features do we want?
  - Track number of:
    - Loonies
    - Toonies
    - $5 bills
    - $10 bills
    - $20 bills
  - Accepts cash
  - Removes cash
  - Calculates value of contents
  - Print the entire contents



Breakout Session!

- Let's look at how this works in Python!
  - Building a cash register
    - Attributes
    - Methods
      - Including __str__ method

Open your notebook

Click Link:
3. Building a Cash Register

APS106

UNIVERSITY OF TORONTO

More OOP! Encapsulation and Examples

Week 7 | Lecture 2 (7.2)

if nothing else, write #cleancode