CSCA08 Fall 2017

Week 9 - OOP Continued

Brian Harrington & Marzieh Ahmadzadeh

University of Toronto Scarborough

November 6 - 10, 2017

Review of OOP

- Create a class
- Define methods
- Create objects of that type
- self

Documentation

- Class Level Docstrings: Simple
- Method Docstrings: very similar to function docstrings, except no examples

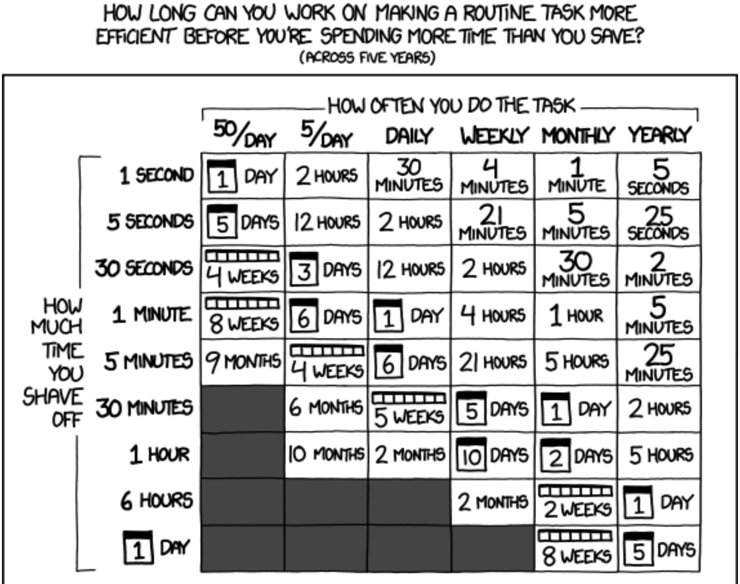
UML

- Unified Modelling Language
- Very common in software engineering
- We will only be using a very stripped down version
- Allow us to work on the design without needing to worry about the next steps

How can classes be related

- Association
  - One object holds another object as a variable
  - *has-a* relationship
  - Example: A `Dog` *has-a* `Toy`
- Composition
  - One object is made up of many other objects
  - Usually only interact with composite object, no direct access to components
  - *part-of* relationship
  - Example: A `Room` is *part-of* a `Building`
- Inheritance
  - One class is a specific case of a more general class
  - *is-a* relationship
  - Examples: A `Student` is-a `Person`

Break



HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS

Inheritance in Python

- When we say `class ClassA(ClassB):`, what we're saying is the ClassA is a **subclass** of ClassB, or that ClassA **inherits** from ClassB.
  - `ClassA` can use all of `ClassB`'s methods
  - or `ClassA` can **overwrite** `ClassB`'s methods with its own
  - Basic idea: if we call `ClassA.method_a()`, check for `method_a` in `ClassA`. If you can't find it, then check in `ClassB`