ACIT 2515

Object Oriented Programming in Python – Week 1

Instructor: Mike Mulder

Agenda – Week 1

- Introductions
- Course Logistics
- Development Toolset
- Object Oriented Programming (OOP) Overview
- Lab 1
  - Programming Exercise: A First Python Class

Instructor

Instructor

- Mike Mulder, P.Eng., CSD/CSM/CSPO
- Interests:
  - Software Requirements, Architecture and Design
  - Programming – Python, Java, Ruby
  - Software Project Management
  - Software Security and Data Privacy
- Office Hours:
  - By Appointment or Discord
- E-mail – mmulder10@bcit.ca (subject: ACIT2515)

Students

- Submit your homework to learn.bcit.ca -> ACIT2515-> Activities -> Assignments
- Check email regularly at my.bcit.ca and course news on learn.bcit.ca
- Ask questions
- Contact your instructor by email (mmulder10@bcit.ca) or through Discord
- Has previously taken ACIT 1515, COMP 1516 or equivalent

Please take <5 minutes now to complete the Survey on D2L.

Activities -> Surveys -> First Class Survey

Course Delivery

- Course Site: https://learn.bcit.ca/d2l/home/848512
- Live classes ("meetings") every Tuesday from 6-9pm
  - Approximately 1 hour for quiz and lecture
  - Lab for the remainder of class – we'll walk through the first part of the lab together
- PDF slides will be posted to the Course Site
- Quizzes will be held online at the beginning of each class
- Homework will be submitted (or demoed) every week

Course Goals – ACIT 2515

- Introduces object-oriented programming principles and techniques.
- Topics include object-oriented programming concepts, such as classes, objects, methods, inheritance, encapsulation, and polymorphism.
- This course also covers techniques for software design and reuse.

This course follows the ACIT 1515 (Scripting for IT) course, and is taught using Python 3. **It assumes a basic understanding of Python syntax.**

Course Outline

#	Week	Topics	Quiz	Lab	Assignment
1	May 10	• (OOP) Intro • Course Development Tools • First Python Class		Lab 1	
2	May 17	• Classes and Objects • Unit Testing	Quiz 1	Lab 2	
3	May 24	• Encapsulation • Abstraction	Quiz 2	Lab 3	
4	May 31	• Inheritance and Composition • Polymorphism	Quiz 3	Lab 4	

Course Outline

#	Week		Quiz	Lab	Assignment
5	June 7	• Python Built-in Objects and Data Structures • Debugging	Quiz 4	Lab 5	Assignment 1 Due
6	June 14	• Object Oriented Design Patterns • More Debugging	Quiz 5	Lab 6	
7	June 21	**Midterm Exam**	In-Class		
8	June 28	Web API Design and Implementation • RESTful API • JSON	Quiz 6	Lab 7	

Course Outline

#	Week		Quiz	Lab	Assignment
9	July 5	Object-Relational Mapping (ORM)	Quiz 7	Lab 8	Assignment 2 Due
10	July 12	GUI Design and Implementation	Quiz 8	Lab 9	
11	July 19	Review and Wrap Up	Quiz 9	Lab 10	Assignment 3 Due
12	July 26	**Final Exam**	In-Class		

No late quizzes, labs or assignments will be accepted.

Evaluation Criteria

Quizzes	10%	Given at the start of each lecture, except the first
Labs	20%	
Assignments	20%	There are 3 assignments
Midterm	20%	
Final	30%	Covers all topics from the course
TOTAL	**100%**	**Students must pass a combination of both the midterm and final exams in order to pass the course.**

The Minimum Passing Grade for this course is 60%

Attendance

- Attendance is mandatory for all classes.
- Students who miss more than two classes without documented medical reason will be assigned a failing grade.
- Students who cannot attend due to illness must notify the instructor via email (or Discord) ***prior*** to the start of class.

Medical documentation must use the approved BCIT medical form: http://www.bcit.ca/files/healthservices/pdf/studentmedicalcertificate.pdf

Class Structure

There is a mandatory reading or video posted on the weekend prior to the next lecture.

Before Class:

- Complete the at-home reading/video on the upcoming class topic
- Post any questions/comments on the topic to Discord (either to one of the channels or directly to Mike Mulder)

General Class Agenda:

- Quick review of last week's topic(s)
- Quiz on at-home reading
- Mini-lecture on the day's topic(s)
- Lab on the day's topic(s)

Labs will be due on the date indicated in the lab write-up. No late lab submissions will be accepted. Some of the labs will require a demonstration.

Expectations of the Student

Basic expectations:

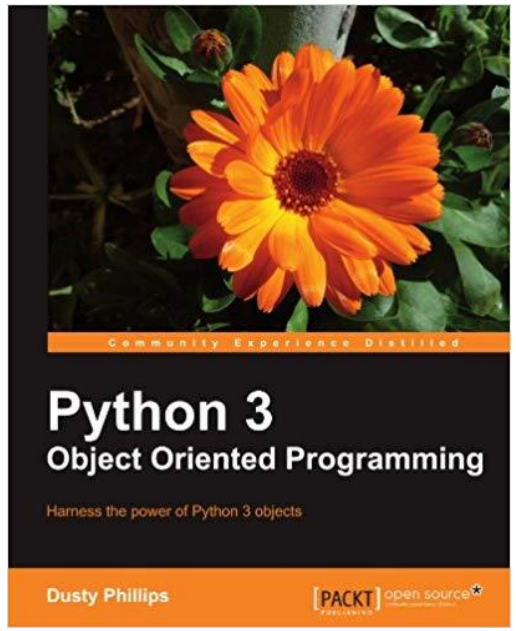
- You already have a working knowledge of the Python programming language
- You can build and run basic Python scripts
- You bring your laptop to every class
- You monitor D2L for class announcements

This class is light on lectures and heavy on hands-on labs. Make sure you ask questions when you need clarification or help on a topic. This can be done in-class, during office hours, by e-mail or on Discord.

The labs are not step-by-step and require you to devise your own solutions or research possible solutions. Again, make sure you get clarification as needed.

Learning Resources

- **There is no required textbook.** Readings and videos for the activities outside of the classroom will be sourced from online or BCIT provided resources.



Optional Textbook

Python 3 Object Oriented Programming (Dusty Phillips)

Available for free through Safari online (see URL below)

Login Here First: https://go.oreilly.com/bcit
Book: https://learning.oreilly.com/library/view/python-3-object-oriented/9781789615852/

Assignments and Exams

3 Assignments that build on each other resulting a full-stack application (database, API and GUI) at the end of the term:

- Assignment 1 – Week 5
- Assignment 2 – Week 9
- Assignment 3 – Week 11

Midterm Exam

- Covers topics from Weeks 1 to 6.

Final Exam

- Covers topics from the entire course, with emphasis on Weeks 8 to 11.

Course Engineering Tools

ACIT 2515

Communication Tools

- E-mail (mmulder10@bcit.ca)
  - ACIT2515 in the subject line
- Discord – Invite link is posted in D2L
  - Discussion Channel
    - Readings
    - Labs
    - Assignments
  - Labs
  - Remote Office Hours

Development Tools

- Python 3 (3.9 or latest)
  - Download: https://www.python.org/downloads/
- Pycharm IDE or Visual Studio Code

Artifact Management Tools

- D2L (https://learn.bcit.ca/d2l/home/848512)
  - Announcements
  - Lecture Notes
  - Labs
  - Assignments
  - Lab Submissions
- Normally in software development we would use a tool like Subversion or Git for this, but for marking it's easier to use D2L.

Other Tools

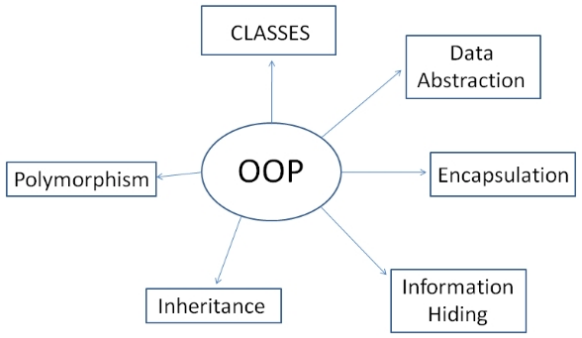
- We will setup other tools later in the term, such as:
  - Postman – for manually testing RESTful APIs
  - SQLite Browser – for viewing the structure and content of SQLite databases

Object Oriented Programming Overview

ACIT 2515

ACIT 2515

Object Oriented Programming (OOP)



Python 3

Programming Language Popularity - Tiobe Index April 2022

Apr 2022	Apr 2021	Change	Programming Language	Ratings	Change
1	3	▲	Python	13.92%	+2.88%
2	1	▼	C	12.71%	-1.61%
3	2	▼	Java	10.82%	-0.41%
4	4		C++	8.28%	+1.14%
5	5		C#	6.82%	+1.91%

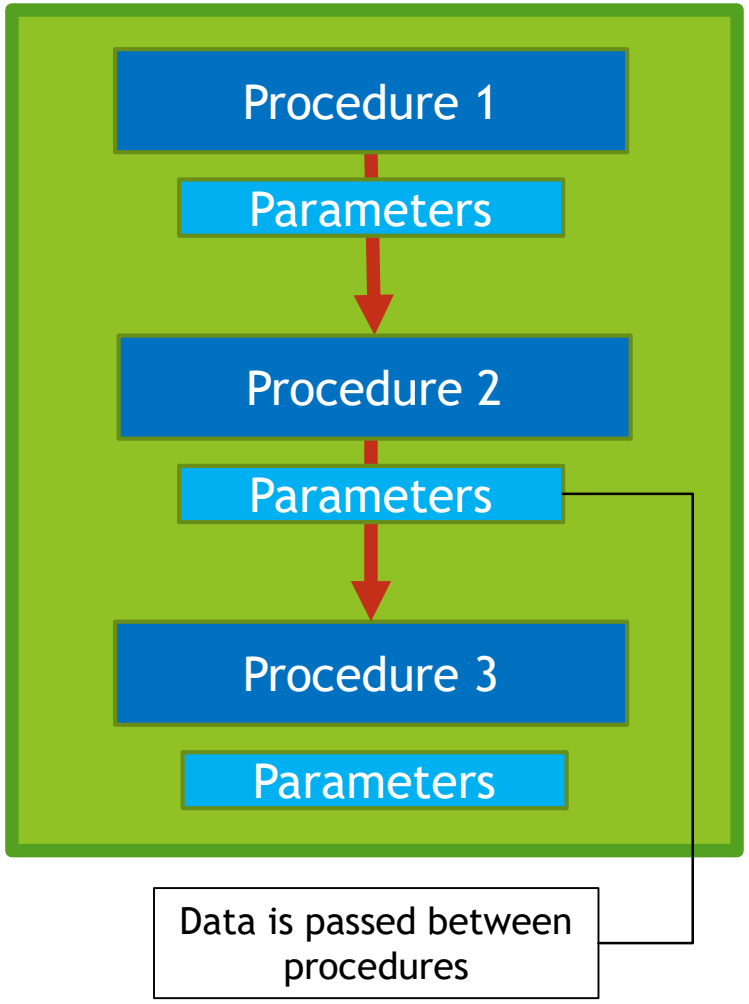
Source: Tiobe Index

Object Oriented Programming (OOP)

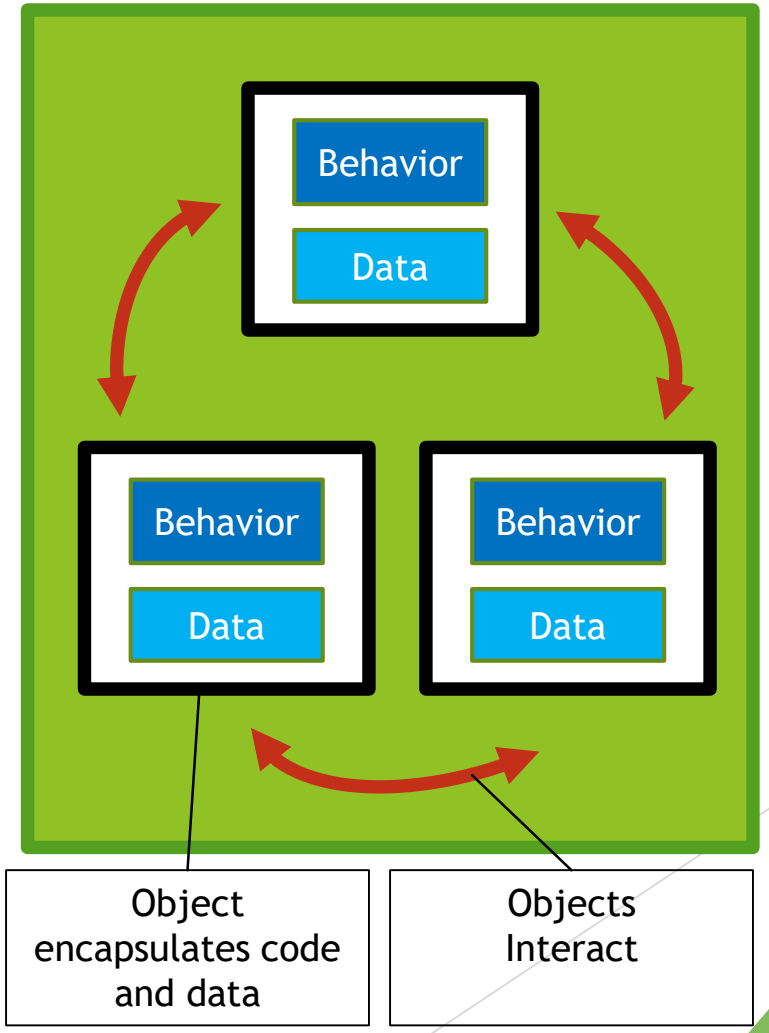
- What is it?
  - A programming language model organized around **objects** rather than "actions" and data rather than logic.
  - Objects have **attributes** (i.e., data) and **behavior**
  - Objects may correspond to:
    - Real-world entities (i.e., shopping cart for online retailer)
    - Abstract entities (i.e., measurement translation service)
  - Four Pillars:
    - Abstraction
    - Encapsulation
    - Inheritance
    - Polymorphism

Procedural Programming vs. OOP

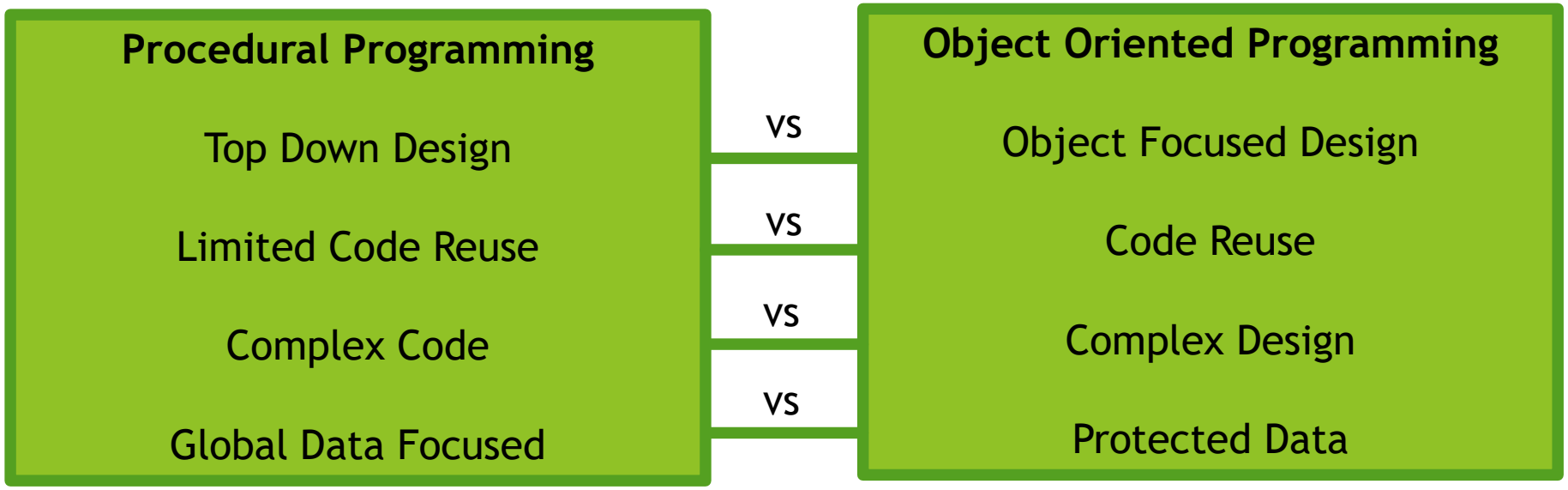
Procedural Programming



Object Oriented Programming



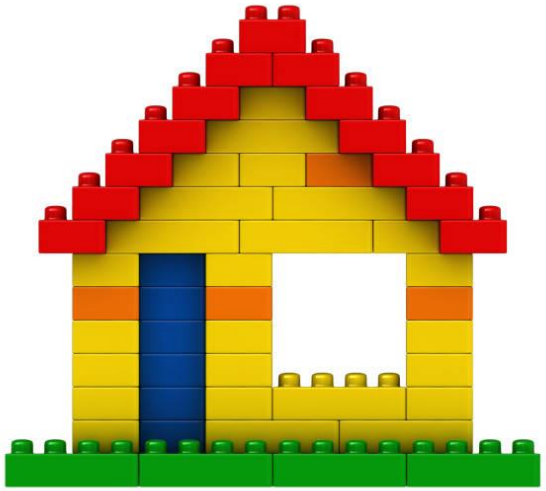
Procedural Programming vs. OOP



Object Oriented Analysis/Design (OOA/OOD)

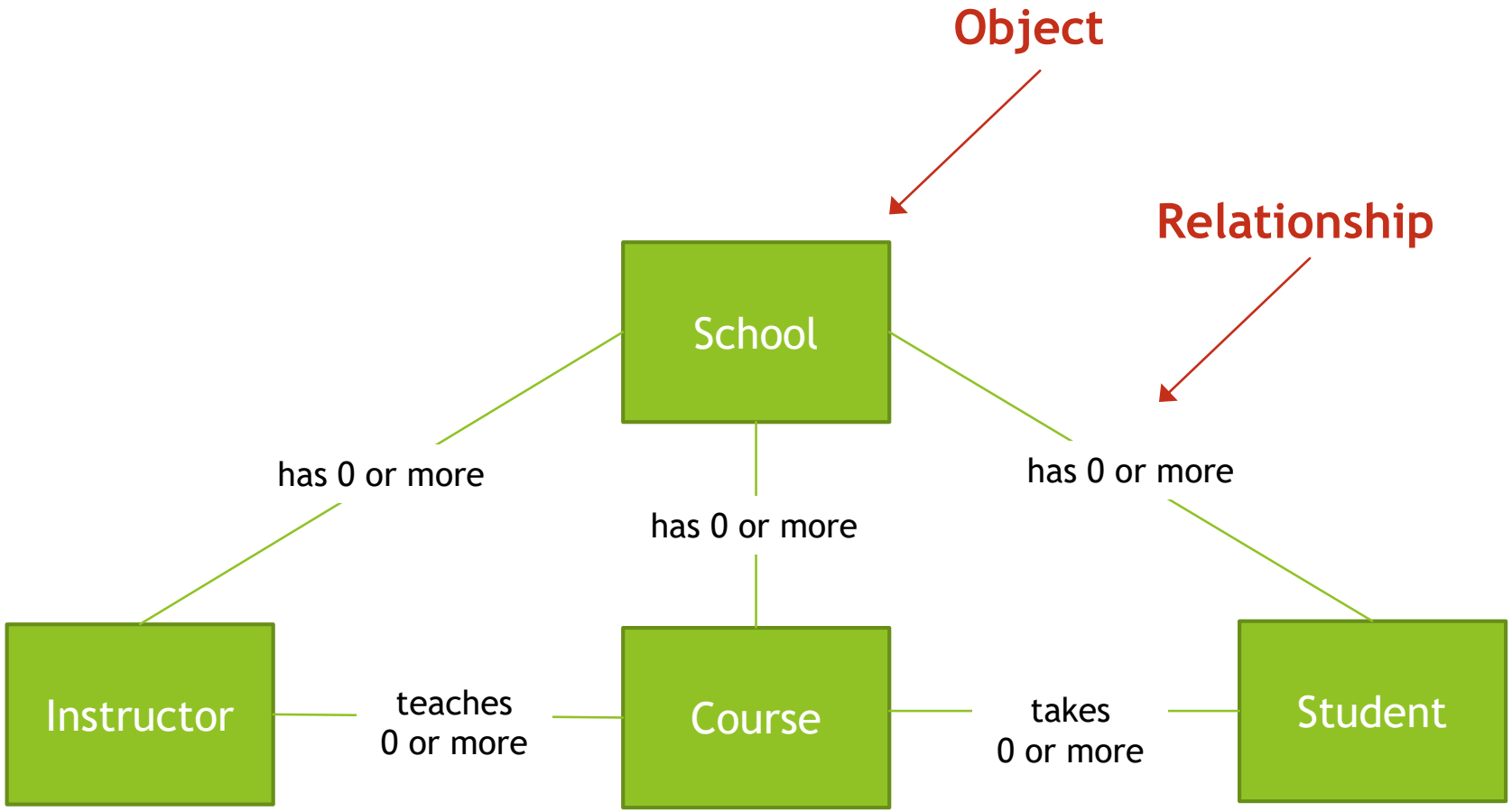
Building block approach:

- Identify the objects that are needed in the software application
- Determine the interactions and/or relationships between the objects
- For each object, determine the the attributes and behaviors

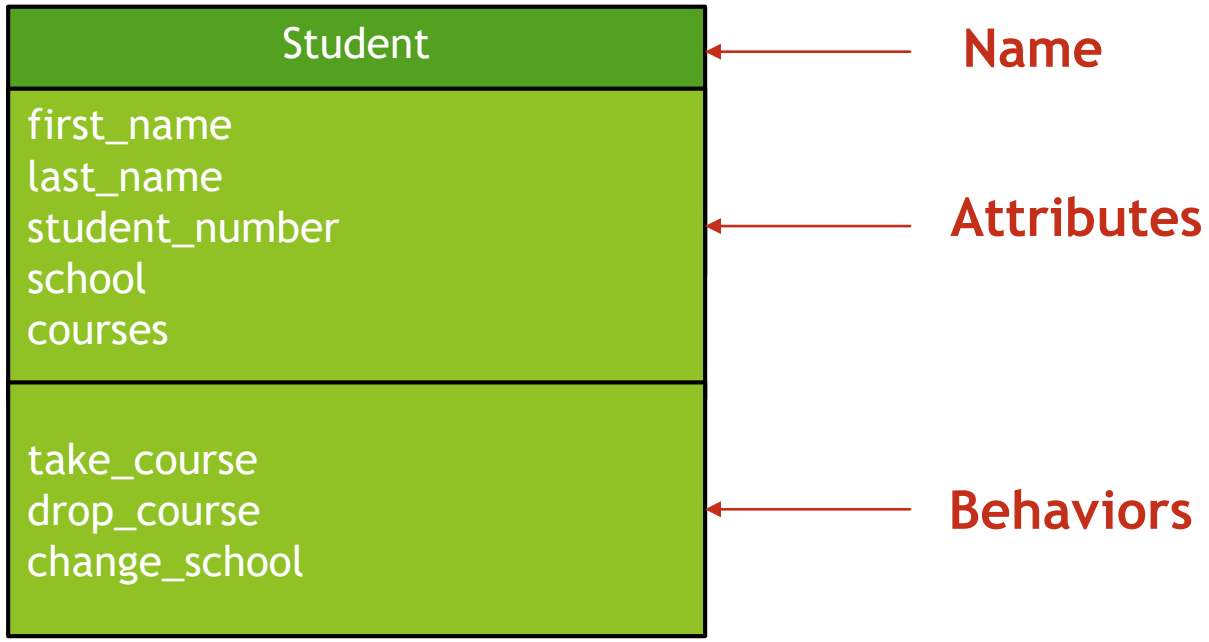


OOA – Identify objects and relationships from end user's perspective (i.e., the business perspective)
OOD – Identify additional objects, interactions and constraints from a technical perspective
OOP – Implementation of the design using OO practices (abstraction, encapsulation, inheritance, polymorphism)

OOA/OOD – Objects and Relationships



OOA/OOD – Attributes and Behaviors

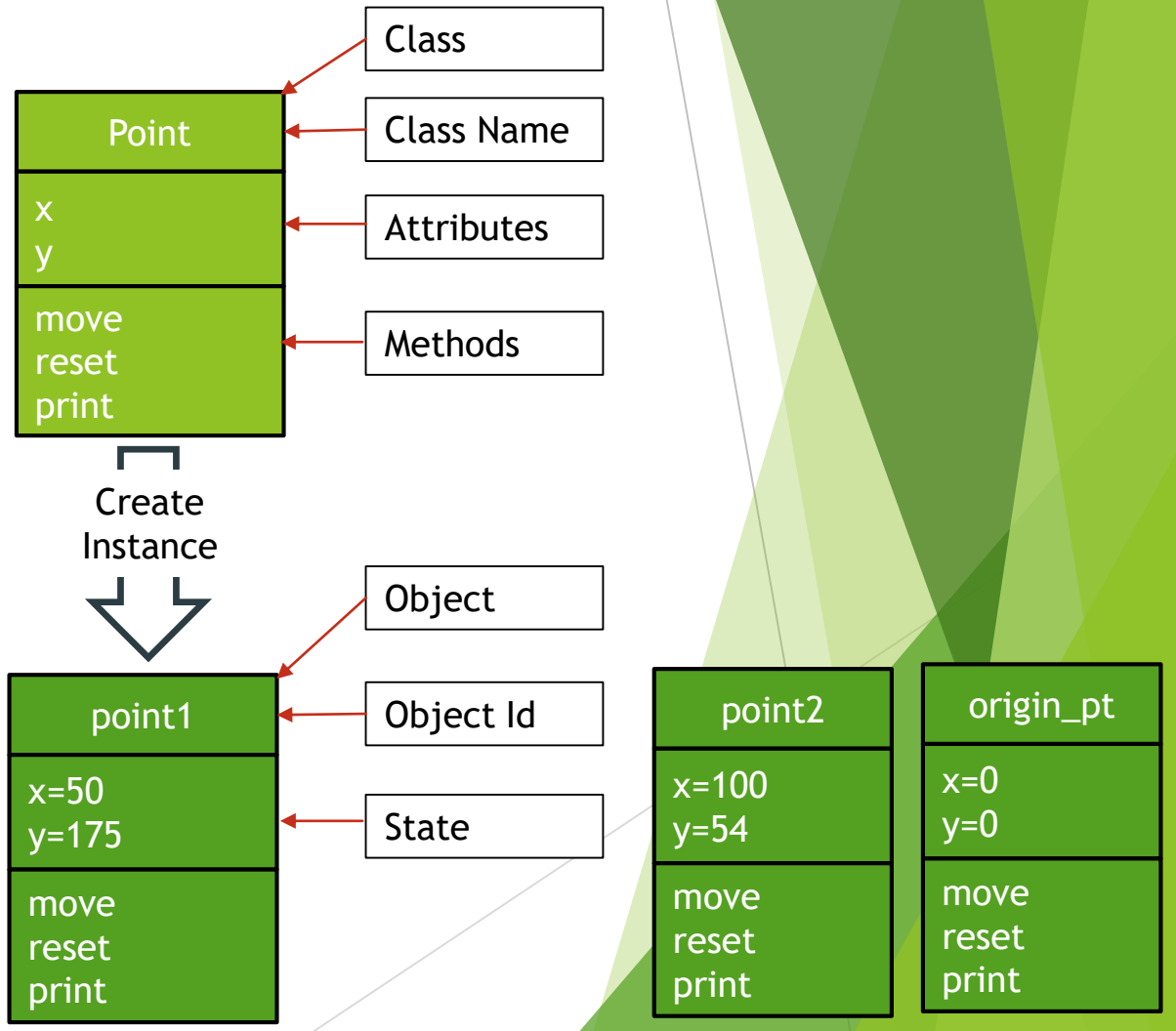


OOP/OOD – Benefits and Drawbacks

- Benefits
  - Reuse and Recycling – Within and across software applications
  - Design - Can force better upfront planning and design for larger projects
  - Testability – At the object level
  - Extensibility – Adding new data and/or behavior to objects
- Drawbacks
  - Can be an overhead for small projects (i.e., extra upfront design)
  - Can lead to over-engineering (i.e., to few classes, too many classes, overuse of design patterns)

Some Definitions

- Class
  - Defines a general category (i.e., book, bank account)
  - Blueprint (or template) for creating an object
- Attributes (or Fields)
  - Defined data attributes of a class
- Methods (or Message)
  - Defined behaviors (or capabilities) of a class
- Object or Instance
  - A specific instance of a class
- State
  - The current values of the attributes in an object



Some More Definitions

- Constructor
  - Method that is called when an object is created
- Instance Variables
  - Variables that contain values specific to an instance of an object
  - The attributes of the object
- Visibility
  - Whether the method or attribute is public or private
  - Public - can be used or accessed from clients (i.e., external users of an object)
  - Private – can only be used or accessed within the object

A Python Example

```
# Example of a Point Class
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y

    def move(self, x, y):
        self._x = x
        self._y = y

    def reset(self):
        self.move(0, 0)

    def print_details(self):
        print(self._x, self._y)
```

Class

Constructor

Instance Variables

Methods

Object Id

Class Name

```
# Example of using the Point Class
point1 = Point(50, 75)
point2 = Point(5, 10)
point1.move(35, 57)
point2.reset()
point1.print_details()
point2.print_details()
```

Creating Objects

Invoking Methods

```
Output:
35 57
0 0
```

__init__ is the constructor of an object in Python

self is a reference to the object that the method is being invoked on.

Python Best Practices - Basics

Comments

DocString

```
class Point:
    """ Point Class Description """

    def __init__(self, x, y):
        """ Constructor Description """
        self._x = x
        self._y = y

    def move(self, x, y):
        """ Method Description """
        self._x = x
        self._y = y

    def reset(self):
        """ Method Description """
        self.move(0, 0)

    def print_details(self):
        """ Method Description """
        print(self._x, self._y)
```

Naming

- Class Name - CapitalizedWords (aka CamelCase)
  - BankAccount
- Attributes - lower_case_with_underscores
  - account_balance
- Methods - lower_case_with_underscores
  - get_account_balance

Visibility

All attributes and methods in a Python class are publicly accessible.

Convention is to use an underscore in front of the name of an attribute or method to indicate it is private.

Example: _calculate_account_balance

Programming Demo

Let’s create a class in Python using PyCharm.

Lab and Next Week

Lab 1

Python Class

- Please read the instructions carefully and ask questions
- Discussion is encouraged, but each student must hand in their own individual work
- Submit your zipfile to D2L on or before the due date (May 16th):
  - Activities -> Assignments -> Lab1

For Next Week…

- Reading will be posted to D2L
- Quiz 1:
  - Will cover OO Definitions and Assigned Reading
- Topics:
  - More on Classes and Objects
  - Unit Testing and Test Driven Development
  - Lab 2