|  |  |
| --- | --- |
|  | Concept Assignment 4  PLTW Computer Science CSP Core Training |

# Object-Oriented Programming

|  |  |
| --- | --- |
|  | Learning Objectives |

LO4.1 While working through the content of Lesson 1.4, the teacher will:

* Become conversant in the terminology of object-oriented programming: describe what a class is and does, use terms like instantiation and encapsulation in their proper contexts, and differentiate between an attribute and a method (both conceptually and symbolically).
* Reflect on how the understanding of objects and their manipulation impacts the utility of libraries (APIs).

LO4.2 While working through Activity 1.4.2, the teacher will:

* Articulate how objects use abstraction to simplify complex programming tasks.
* Use libraries of code (APIs) to complete assigned programming tasks.
* Use an official API reference to understand how objects, attributes, and methods function within a given library, and to understand what kind of functionality is provided by that library.

LO4.3 While working through Activity 1.4.3, the teacher will:

* Use objects to simplify complex tasks, such as reading an image from a storage medium to an array.
* Use arrays to hold and manipulate images.
* Apply an understanding of pixel arrays, the alpha channel, and opacity to the process of masking.

LO4.4 While working through Activity 1.4.4, the teacher will:

* Use various forms of documentation in understanding how third-party code functions, leveraging that understanding through the integration of objects into one’s own code to solve particular problems or to perform specific tasks.

LO4.5 While working through Activity 1.4.5, the teacher will:

* Integrate the ideas, concepts, and understandings of all previous activities in Lesson 1.4 to an assigned project, following the PLTW APB model.

|  |  |
| --- | --- |
|  | Introduction |

It’s safe to say that most software development today is based on objects. It is easy to see why: objects provide abstraction for complex ideas, protect data through encapsulation, allow for reuse through inheritance, and support differentiation and evolution through polymorphism.

If some of the concepts in the paragraph above seem foreign to you, do not despair; you’re not alone! The solution to the conundrum of object’s importance versus lack of understanding is why this unit exists. We will tackle what objects are, how to create them, and how to use them. We’ll also demonstrate how we can use code that others have written (third-party software) in our own software projects. That code, which heavily uses objects, can be understood only if we understand how objects work. This unit also shows how APIs and other forms of documentation can help us understand and use third-party software in our own projects.

### Overview of Object-Oriented Programming in the Course

Exposing students to a few object-oriented (OO) programming concepts makes many powerful libraries accessible to them. This course intentionally omits some important object-oriented programming concepts, but it does include the basics: objects belong to a class that defines properties (which are values) and methods (which are functions). Most object-oriented programming languages, including *Python*®, use a dot notation:

object.attribute is the value of the attribute for that object

object.method() calls the method on that object

Each of the lessons in Unit 1 includes some object-oriented content.

* Lesson 1.1: Scratch sprites are objects. Activity 1.1.4 encourages students to name broadcast messages using the standard object-oriented notation for calling a **method** on an **object**:

sprite.verb()

* Lesson 1.2: MIT App Inventor component types are **classes**. You can create one or more components of any type; each component is an object. The blocks for each component include blocks to get or set the value of each attribute. OO languages use a **constructor** function to create a new object of a class and to initialize the **properties** or **attributes** of the object. The App Inventor Designer view lets you create new components of a particular class and initialize the object’s properties.
* Lesson 1.3: Methods are called on some *Python* data types using the standard OO dot notation. Examples:

In []: 'This will return four pieces'.split('e')

Out[]: ['This will r', 'turn four pi', 'c', 's']

In []: my\_list = ['east', 'west']

In []: my\_list.append('north')

In []: my\_list

Out[]: ['east', 'west', 'north']

* Lesson 1.4: OO concepts are introduced formally in Activity 1.4.1 and Activity 1.4.2. The OO libraries matplotlib and PIL are introduced in the remainder of the lesson. These libraries were chosen because they are commonly used by scientists and because they allow for creative work involving graphics and images that is engaging to students.

|  |  |
| --- | --- |
|  | AP CSP Enduring Understandings (EU) and Learning Objectives (LO) |

* Programming facilitates creative expression. EU1.1 via LO1.1.1 [P2], EU1.2 via LO1.2.1 [P2] and LO1.2.3 [P2], EU1.3 via LO1.3.1 [P2], and EU1.1 via LO5.1.1 [P2] and LO5.1.3 [P6].
* Abstractions, such as objects and their methods, help programmers handle complex problems. EU2.2 via LO2.2.1[P2] and LO2.2.2[P3], and EU5.3 via LO5.3.1 [P3].
* Digitization has had positive and negative impacts on artists. To facilitate sharing and collaboration and to avoid infringing on intellectual property rights, users need to understand various kinds of copyrights, such as Creative Commons. EU7.3 via LO7.3.1 [P4].

|  |  |
| --- | --- |
|  | Part 1: Review the Student Learning Sequence for Lesson 1.4 |

Read through Activity 1.4.7, the problem for lesson 1.4. Pay particular attention to the skills one needs to master to solve the problem.

### Activity 1.4.1 Procedural Abstraction (1 day)

Students will…

* Describe the abstraction of a function with arguments.
* Understand how to use an object-oriented class to create objects and to use the objects’ attributes and methods.

### Activity 1.4.2 Objects and Methods (2 days)

Students will…

* Access a file using absolute and relative file names.
* Use escape characters in a string.
* Use the *Python* matplotlib library to investigate images using the Figure and Axes objects and their methods.

### Activity 1.4.3 Images and Arrays (2 days)

Students will…

* Manipulate the pixels of an image by iterating across pixels.
* Analyze and evaluate algorithms that create or alter images by iterating across pixels.

### Activity 1.4.4 Python Imaging Library API (2 days)

Students will…

* Use various kinds of documentation to learn about an object-oriented library.
* Use professional terminology about classes, objects, methods with arguments, and attributes.
* Use high-level abstractions from the Python Imaging Library.
* Explain the meaning of the alpha channel in RGBA images.

### Project 1.4.5 Image Algorithms (3 days)

Students will…

* Create images by combining polygons and ellipses using the Python Imaging Library.
* Apply an alpha mask to create composite images.
* Explain the exception handling of the *Python* try-except structure.

### Activity 1.4.6 Digital Property and Forensics (2 days)

Students will…

* Describe metadata.
* Describe and debate digital property rights and property rights technologies.
* Describe and debate privacy rights and privacy rights technologies.

### Problem 1.4.7 Image Artist (5 days)

Students will… Design, develop, and present an algorithm for creating or modifying images.

***Optional***: Perform the unplugged activity from Activity 1.4.1 Procedural Abstraction*.* Examine the Activity 1.4.1 example *Python* file. Instantiate an object, call the method, and access the attribute using the source code provided for class demonstration (pun intended).

|  |  |
| --- | --- |
|  | Part 2: Objects and Methods |

1. Work through Activity 1.4.2 from the beginning. Part 1 reviews filesystems, differentiating between relative and absolute pathnames. Additionally, several UNIX commands are used in iPython to demonstrate concepts. If you feel comfortable with directories and paths, skim through this section. Students often take these concepts lightly, and doing so can lead to subtle problems. You can preempt many difficulties by having students follow **four simple rules** when working in Canopy:
2. Create and name a folder on the computer or network to save all *Python* work. Be sure the name is unique (perhaps develop a convention) if computers are shared with others.
3. Save all authored *Python* files to this directory, appending the file name with the .py extension when saving.
4. Set the working directory to this folder (double-click the working directory name on the right side of the *Python* window where the editor and iPython windows meet).
5. Place any downloaded course resources, such as images and code, into this folder.
6. The activities in Lesson 1.4 have been thoroughly vetted. Errors that occur can usually be traced to failure to abide by these simple rules, or by the faulty assumption that they have been followed. If you receive a “Too many values” error, chances are you skipped directive 3!
7. Using the code and directions found in Part 2, render an image inside a window on your screen. Note that line 18 demonstrates multiple variable assignment, which is somewhat unusual to most beginning programmers. With your partner, discuss what’s happening in the code in terms of objects: instantiation, dot-accessing of methods, and the use and meaning of specific arguments to those method calls.

|  |
| --- |
| Submission Item |
| 1. Screenshot of Activity 1.4.2Part 4, steps 9a and 9b |

1. In Part 4 of Activity 1.4.2, you will generate multiple axes and images inside a single window. To be successful in this endeavor requires an understanding of arrays and how elements in a multi-dimensional array are accessed. Review those ideas with your partner. (For example, in a 4 x 4 array, how is the third element in the second row referenced?)
2. In Part 4, images are set to an axis using the method imshow(), and certain regions of an image are isolated using the method calls set\_xlim() and set\_ylim(). Someone else wrote that code for you, and you used it without much trouble.

|  |
| --- |
| Submission Item |
| 1. How is your experience an example of abstraction? How did you know how to use the methods properly? |

1. Part 4 in Activity 1.4.2 continues by showing you additional methods at your disposal and even provides a reference, called an API, for you to gain an understanding about how to use these novel methods.

|  |  |
| --- | --- |
|  | Part 3: Arrays and Images |

1. Work through Activity 1.4.3from the beginning. The first section introduces arrays and the subtleties of object-oriented programming (OOP). In Part 1, step 4, one of the first subtleties is introduced. The dot-access method is so named because a dot (period) follows an object’s name, followed by an attribute or method name. For instance, *animal.weight* would report the attribute of weight for an animal, while *animal.run()* is the run method called on an animal. The context and parentheses help you distinguish the different cases. Step 4 highlights a use of the dot-access method that does not involve an object in a strict sense. The dot access here involves calling a method on an alias (nickname) assigned to a library.

In teaching your students, you are developing experts from novices. An expert is better at showing differences between similar things than is a novice. Here, the similar thing is the dot-access method. The difference is acting on an alias versus an object. The novices you teach will likely miss such subtleties. Be on the lookout for such teaching opportunities.

1. In Part 2, nested for loops create diagonal lines. The mathematics that create the lines is less important than how changing values result in a changed shape and/or color.

|  |
| --- |
| Submission Item |
| 1. Screenshot of Activity 1.4.3Part 2, step 7 |

|  |  |
| --- | --- |
|  | Part 4: Python Imaging Library |

Up to this point, our code has used the pyplot library (aliased as plt) to access methods that create objects for our use. In Activity 1.4.4 we are introduced to a new library, the *Python* Imaging Library (PIL). Part of the intent of this activity is to show how this library can do things the pyplot library cannot, thus increasing the tools we have at our disposal. The activity teaches us how to use this new library through the inclusion of documentation available on the internet. To make things easier, a reference card is provided in the course materials. Whether you use a website or this card, the use and utility of documentation is demonstrated. Work through Activity 1.4.4. Use the Lesson 1.4 Reference Card.

|  |
| --- |
| Submission Item |
| 1. Complete the table found in step 5 of Activity 1.4.4, providing a description for methods available from the PIL library. |

The rest of Activity 1.4.4 has you download source code and images. It then steps you through the code, using pertinent OOP vocabulary and directing you to applicable documentation to enhance your understanding of objects and the use of third-party libraries.

The activity culminates in the production of an image of a young girl whose eyes are replaced by images of the earth.

|  |
| --- |
| Submission Item |
| 1. Activity 1.4.4: Screenshot of the young girl with earth eyes as directed in step 18. |

|  |  |
| --- | --- |
|  | Part 5: Image Algorithms |

Probably the most challenging aspect of Activity 1.4.5 involves understanding the concept of masking. We’ve learned that pixels have values of red, green, and blue, and that some images (.png and .gif, but not .jpg) have an additional channel that describes that pixel’s opacity: a value of 0 means transparent, and 255 means completely opaque. The opacity of a pixel plays a huge role in masking. But what is masking? If we could describe masking using a sandwich metaphor (say of a sandwich having a single slice of cheese), then:

1. The top piece of bread represents an original image to which a mask will be applied.
2. The cheese represents the mask.
3. The bottom piece of bread represents the result of applying the middle mask to the top original image.

Masking works like this: If the cheese is circular, and the top piece of bread is rectangular, the result of adding the top piece of bread (original image) to the circular cheese (mask) is a bottom piece of bread (resulting image) that is circular. In other words, the mask dictates which parts of the image come through. In *Python* we have no cheese; we have opacity. The opacity of the mask will dictate whether the pixel applied to it ends up in the final image: In our code, only those pixels whose corresponding mask pixel has an opacity of 255 will be transferred to the resulting image. Your instructor will review these ideas with you.

Work through Activity 1.4.5, which requires you to download image and code files. You will step through the code of *mask.py*, using your knowledge of objects, properties (attributes), and method calls on those objects to understand how a mask is generated and applied to an image. The caution expressed in Part 2A of this document is especially applicable here: Make sure the location of your resources (images and code) and working directory all point to the same place.

|  |
| --- |
| Submission Item |
| 1. Activity 1.4.5: Alter the working code to make a design of your own choosing as directed in step 10b and submit that code. Include a screenshot of your results. |

|  |  |
| --- | --- |
|  | Part 6: Classroom Implications |

1. Because success in this course so often depends upon the retention and application of previous knowledge and understanding, monitoring of previous learning and its application to the task at hand is an essential responsibility of the teacher. The teacher has to first identify those previous understandings (which will be applied to the current task), determine the best means of assessing mastery of those concepts, perform an assessment, and respond to the results.

How is this process best accomplished? What previous knowledge and understandings are key to success in Lesson 1.4? What type of formative assessment (be specific) could you develop that would measure mastery, and therefore, readiness? What options would you provide if a student experiences difficulties?

1. At some point, students will struggle with the material. It may be an over-simplification to say that successful students work through the difficulty, while unsuccessful students give up at the intersection with difficulty. However, it’s instructive to look at times when this is the case. How might you act to encourage a failing student? How much of a role do student expectations and confidence play in this scenario? What do you do in your classes to foster grit, that is, the willingness to persevere and move forward even in the face of great difficulty?

|  |
| --- |
| Submission Item |
| 1. Write a reflection about the things you’ve learned today. Consider highlighting new things you’ve learned, items you need to consider for implementing this in your classroom, and ideas and suggestions you’ve heard from others. Use the questions above as prompts, but don’t feel limited or constrained by just those questions. |

**Trademark Attribution** PLTW, Project Lead The Way, and the PLTW logo are registered trademarks of Project Lead The Way, Inc. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. All other brand names, product names, or trademarks belong to their respective holders.