|  |  |
| --- | --- |
|  | Concept Assignment 3  PLTW CSP Computer Science Core Training |

# Algorithms

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

LO3.1 While working through activities in Lesson 1.3, the teacher will:

* Explore different data types as used in *Python*®.
* Define variables of different data types in *Python*.
* Perform operations with different data types in *Python*.
* Typecast between different data types in *Python*.

LO3.2 While working through activities in Lesson 1.3, the teacher will:

* Learn about different types of algorithms commonly used in computer science.
* Define functions that contain multiple types of algorithms in *Python*.
* Transform data and solve basic algorithmic problems in *Python*.

LO3.3 While completing Project 1.3.10, the teacher will:

* Explore a common game theory problem known as The Prisoner’s Dilemma.
* Employ different algorithm concepts to create a strategy to play The Iterative Prisoner’s Dilemma game.
* Compete with other participants in The Iterative Prisoner’s Dilemma game and analyze the results.

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

### Data Types

The concept of data types is an important one in computer science and is an important bridge between the concepts of abstraction and algorithms. In the programming environments we’ve used so far, the data types have been mostly abstracted—the user doesn’t have to worry much about what type of data is being used. This is commonly referred to as “weakly typed.” As programming environments become less abstract, data types become more important, and the code must specify the data type that’s being used. This is referred to as “strongly typed.” *Python*, the language we’ll use the most in the next few lessons, is generally a weakly typed language, but it’s more strongly typed than MIT App Inventor or Scratch.

The basic data types that are used in *Python* include the following:

* Integer
* String
* Float
* Boolean
* Tuple
* List
* Long

In this lesson, you’ll learn about the different data types in *Python* and how they interact and are used, as well as how to switch between them (called typecasting).

### Overview of Algorithms

Students will benefit from algorithmic thinking with or without a computer. An algorithm is simply a procedure for completing a task. As a result of the course, students should be better able to think about and express procedures they follow. From the lunch line to personal hygiene, students should be better able to analyze the efficiency of procedures they participate in and should have a better eye for improving the efficiency of those procedures.

Algorithms have three building blocks:

* Sequence: Instructions are followed one at a time, one after another.
* Selection: An instruction can ask a yes-or-no question and select the next instruction based on the result.
* Iteration: A block of instructions can be repeated. The repetition can be:
  + A specific number of repetitions
  + Once for each item in a list
  + While a condition is met

Recursion is often listed as a fourth building block of algorithms, although any algorithm can be expressed with only sequence, selection, and iteration.

Two of the more common algorithmic patterns are:

* Accumulation: Calculating a running total across the items in a list. The total might be incremented only if a condition is met, or might be incremented by an amount calculated from the list item. For example, given a grocery cart of items, calculate the total price of the items that are luxuries.
* Aggregation: Collecting items in a list. An item might be appended to the list only if a condition is met, or the item appended might be the result of a calculation. For example, from a list of all students, create a list of at-risk students’ current grades.

A significant component of the AP Learning Objectives relates to the limitations of algorithms. Not all algorithms can be completed in a reasonable period. Heuristics often can provide approximate solutions to these difficult problems.

### Overview of Programming Languages in the Course

Sequencing of instructions is introduced with Lightbot in Activity 1.1.2 and with stacks of blocks in Scratch.

Conditionals and selection of instructions are introduced in Activity 1.1.3 in Scratch, repeated with App Inventor in Activity 1.2.4.

Iteration and the accumulation and aggregation patterns are introduced in the following content:

* Scratch – Activities 1.1.5 and 1.1.6
* MIT App Inventor – Activities 1.2.4 and 1.2.5
* *Python* – Activities 1.3.7 and 1.3.8. Iteration is reinforced when working with large data sets in Lesson 3.1.
* Microsoft® Excel® and *Python* – Activity 3.1.1 reviews iteration

Two algorithms for searching (linear search and binary search) are compared in Activity 1.3.8. Binary search requires that elements in a list are already sorted; several sorting algorithms are compared in Activity 1.3.8 and Activity 1.5.3.

In several activities, students are asked to reason about the comparative efficiencies of two algorithms. In Activity 2.3.2 Security by Encryption, students conduct a scientific experiment to compare different algorithms by timing their execution. This activity addresses CS Principles Learning Objectives related to complexity and decidability.

An important concept in algorithm design is parallel processing. Computers are getting faster only by using more processor cores in parallel, rather than increasing the clock speed of any one processor. Unfortunately, many developers don’t tend to think in terms of algorithms that can use more than one processor. Parallel algorithms may come more naturally to people who think about the concept early in their CS education. Parallelism is addressed in Activity 3.1.3.

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

Algorithms and algorithmic thinking practices are emphasized throughout the course. This concept assignment focuses on how they’re taught and used in Lesson 1.3 of the CSP curriculum. The following specific concepts are addressed:

* Computers execute algorithms using mathematics and logic. EU4.1 via LO4.1.1 [P2] and LO4.1.2 [P5], and EU5.5 via LO5.5.1 [P1].
* Programs are easier to create and analyze when they’re incrementally developed with functions and well-documented code. EU5.1 via LO5.1.2 [P2], EU5.3 via LO5.3.1 [P3], EU5.4 via LO5.4.1 [P4].
* Software is usually developed in teams, small or large. EU5.1 via LO5.1.3 [P6] and EU7.1 via LO7.1.2 [P4].

|  |  |
| --- | --- |
|  | Part 1: Data Types and Algorithms in Python |

Your instructor will lead the group through the following activities using the course materials. Complete the deliverables without referencing the answer key so you can work on developing the skills of the lesson. When finished, compare your solution to that in the answer key and discuss with your partner any differences between your solution and the one in the answer key.

Keep in mind that most items have multiple ways to successfully code them. As a teacher, you will need to be able to identify whether a student’s solution is acceptable even if different from the published answer key.

A lot of material is covered in the activities for Lesson 1.3. Work through the activities to get a sense of the scaffolding provided for students and to start building your skill set, but don’t spend too much time on them. Note which areas you think you’ll need to revisit as you prepare to teach the content to your students. Ask questions about areas that you find confusing so the entire group can benefit.

* Activity 1.3.2 Python Variables and Functions (completed in Prerequisite Training)
* Activity 1.3.3 Branching and Output (nothing to submit)
* Activity 1.3.4 Nested Branching and Input

|  |
| --- |
| Submission Item |
| 1. Activity 1.3.4: Submit code for steps 4 and 7b. |

* Activity 1.3.5 Strings (nothing to submit)
* Activity 1.3.6 Tuples and Lists (nothing to submit)
* Activity 1.3.7 For Loops

|  |
| --- |
| Submission Item |
| 1. Activity 1.3.7: Submit code for steps 10a, 10c, and 11a. |

* Activity 1.3.8 While Loops

|  |
| --- |
| Submission Item |
| 1. Activity 1.3.8: Submit code for step 8. |

|  |
| --- |
| Submission Item |
| 1. Complete Activity 1.3.6 Check for Understanding Quizzes 1 and 2. |

* Review a video excerpt of the Golden Balls game show. Complete Project 1.3.10 Game Theory; run the Iterative Prisoner’s Dilemma tournament and discuss results.

|  |
| --- |
| Submission Item |
| 1. Submit your *Python* code for strategy, explanation of strategy, summary of results, and explanation of results for Project 1.3.10. |

|  |  |
| --- | --- |
|  | Part 2: Considering Classroom Implications |

1. Describing a program using human language or pseudocode is helpful to beginning programmers. Comments within code are easier to include and to read in *Python* than in Scratch or App Inventor. How will you encourage students to plan their programs before beginning to code and to comment their code?
2. Stepping through code is an important pedagogical tool with which you can show variables and their values, stopping at each expression or line of code to offer explanations, take questions, and check for understanding at each step. As a core training group, describe some of the classroom formats for stepping through code, such as pythontutor.org, updating a whiteboard, using student actors. Discuss the frequency and timing in the learning sequence where you think each of these pedagogical formats for stepping through code are appropriate.
3. With experience, you’ll become familiar with common student mistakes. If you’re new to computer science, you’ll be learning content at the same time as you learn how to teach it. Answer keys provide help pertinent to an activity, and you can use Community for any given lesson as a source of teacher advice. In addition, researchers at Purdue and Michigan State created just-in-time teacher professional development materials indexed by the course activities. A portion of these materials is available to all teachers (use password pd4cs2016). To get a sense of the materials available, examine one or more of the student misconceptions and challenges catalogued at: http://open.pd4cs.org/functions-student-misconceptions-and-challenges/ and http://open.pd4cs.org/data-structures-student-misconceptions-and-challenges/
4. Balanced assessment includes formal checks for understanding as well as informal observations as you circulate in the classroom. What are some of the records for assessment that you could keep as you circulate through the classroom?
5. Discuss how you’ll encourage persistence as a personal attribute in your students.

|  |
| --- |
| Submission Item |
| 1. Write a reflection about the things you 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 in Part 2 of the assignment 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.