- Define the essential components of computer science, **algorithms** and **abstraction**
- Construct **plain-language algorithms** to solve basic tasks

- Recognize and use the basic **data types** in programs
- Interpret and react to basic **error messages** caused by programs
- Use **variables** in code and trace the different values they hold

- Understand how different **number systems** can represent the same information
- Translate **binary numbers** to decimal, and vice versa
- Interpret binary numbers as abstracted types, including **colors** and **text**

- Use **function calls** to run pre-built algorithms on specific inputs
- Identify the **argument(s)** and **returned value** of a function call
- Use **libraries** to import functions in categories like math, randomness, and graphics

- Use **function definitions** when reading and writing algorithms to implement procedures that can be repeated on different inputs
- Recognize the difference between **local** and **global scope**
- Trace **function calls** to understand how Python keeps track of **nested function calls**

- Use **logical operators** on Booleans to compute whether an expression is True or False
- Use **conditionals** when reading and writing algorithms that make choices based on data
- Recognize the different types of **errors** that can be raised when you run Python code

- Translate **Boolean expressions** to **truth tables** and **circuits**
- Translate **circuits** to **truth tables** and **Boolean expressions**
- Recognize how addition is done at the circuit level using **algorithms and abstraction**

- Use **while loops** when reading and writing algorithms to repeat actions while a certain condition is met
- Identify **start values**, **continuing conditions**, and **update actions** for **loop control variables**

- Use **for loops** when reading and writing algorithms to repeat actions a specified number of times
- Recognize which numbers will be produced by a **range** expression

- **Index** and **slice** into strings to break them up into parts
- Use for loops to loop over strings by **index**

- Translate algorithms from **control flow charts** to Python code
- Use **nesting** of statements to create complex control flow