PALs CSC-152

Section 02, Professor Kim, M-W-F

Tony Stone

November 12, 2025

Lesson 0

1 Introduction

Welcome to Lesson 0! Rather than diving into a specific computer science topic, here are some background items to keep in mind. Many times throughout this document I will ask you to utilize your 'toolbox." Consider this lesson to be your hammer, and a set of nails.

2 Python Conventions

Like any programming language, Python has a plethora of conventions which are not true syntax. This means that avoiding these conventions will not have an impact on the performance of your code. So why follow them? Sticking to a set of conventions, particularly ones set for an entire language, make code generally more **readable**, and easy to work with (more on readability later).

Perhaps the most famous piece of programming convention is the naming convention; the way you will format your variables. In Python, the standard naming convention is **snake case**. This convention uses all lower case characters, separating them by underscores, like so:

```
# snake_case
```

Maybe it is a personal quirk, but I am of the mind that snake case is hideous. Throughout these lessons, I will stick to the **camel case** naming convention, like this:

```
# camelCase
```

Camel Case is written by having the first 'word' being lowercase, with an uppercase letter distinguishing each subsequent word. All of the words are packed together in a nice dense string. Though, when the first word begins with a capital letter, it is referred to as **pascal case**.

```
# camelCase
# --is not--
# PascalCase
```

Trust and believe that pedants will enjoy calling you out for 'mixing conventions' when you decide to give a class an uppercase letter, and claim your code follows camel case. Feel free to remind them: it literally does not matter.

3 Python Datatypes

Programming is a practice of **Data** manipulation. I do not often see an explicit definition of data in many resources—perhaps its obvious, but it was embarrassingly ambiguous to me—but data means literally anything that has any sort of

value. I could list examples, but we would be here all day. Data is more or less synonymous with 'stuff.'

However we like to classify this 'stuff' or data into different kinds of data. By doing so we are able to establish different features and qualities to form a set of unique **data types**. I will not give an exhausted review of each data type, but I will happily bullet the most relevant data types that are build into Python, and when necessary we'll look deeper.

First are the **atomic data types**, which are data types that represent the smallest division of data.

- **int**: This is an integer, or any whole number.
- **float**: This is a floating point number, or a number with a decimal point.
- **bool**: A boolean is a data type valued as either true or false, left or right, high or low, cats or dogs...

collection data types are data types are composed of atomic data types. As the name implies, they are various ways one can group atomic datatypes. Perhaps the most prominent among them is the **list**, which we will explore later! There are many other build-in collection datatypes, but lets not get rutted in that just yet.

I will take a quick moment to discuss the **str** data type, or string. These are made of a sequence of characters enclosed by quote marks: "", or by apostrophes: ''. Strings are sort of atomic, in that the smallest piece of a string is merely the empty string "", as it cannot be broken up into characters like other programming languages. But strings may also be treated as collections, as we will see later. Strings also have the quality of being **immutable**, or unchangeable. When we are 'altering' strings later, we are actually just making a new string! (it just looks like we are changing the old one).

4 Python Operations

Python utilities the expected suite of arithmetic operations. When combined in an arithmetic sequence, they follow the order of good old fashioned parenthesis > exponents > multiplication/division > addition/subtraction or **PEMDAS**.

Operator	Symbol
Addition	$+$
Subtraction	$-$
Multiplication	$*$
Division	$/$
Exponentiation	$**$

Aside from the standard arithmetic operations, Python offers other kinds of operations to be applied to numbers. Both of these operations round an otherwise decimal answer, into the nearest integer.

Operator	Symbol
Integer Division	//
Remainder Division (Modulo)	%

As you may have guessed, the modulo and integer division operations always have an answer of type int.

Operations are not limited strictly to numbers! There are many ways to apply operators to manipulate the data we begin with. I'll show a few, but try experimenting!

- String concatenation:

```
astring = "hi" + ", " + "how are you?" # Answer is "hi, how are you?"
```

- String multiplication:

```
bstring = "Stone" * 3 # Answer is "StoneStoneStone"
```

5 Resource Availability

These lesson pans are by no means the full extent of the resources available! I strongly suggest you visit the Introduction to Python GitHub Repository: github.com/TonyStone23/Introduction-to-Python. This will provide you with all of the **source code** of the examples you find in the lesson! When you come across an example within these lessons, feel free to hop over to the respective Python file, copy it, and play around with it!

6 Key Terms

- **Readable**: The measure of codes ability to be perceived.
- **Data**: All that holds a value of some kind.
- **Data Type**: The categories of data that grant values certain qualities.
- **Atomic Data Types**: The most fundamental components of data.
- **Collection Data Types**: Data types composed of atomic datatypes.
- **PEMDAS**: Parenthesis > exponents > multiplication/division > addition/subtraction.
- **mutability**: The ability to be changed.
- **source Code**: The original written code used to develop something.

7 Availability

- In Lecture:
  - Wednesday: 10:00am - 11:00am
- PAL Sessions:
  - Monday: 2:00pm - 3:00pm
  - Wednesday: 9:00am - 10:00am
  - Wednesday: 2:00pm - 3:00pm

References

[1] Tony Gaddis. *Starting Out with Python*. Pearson, 5th edition, 2021.

[2] GeeksforGeeks. Python Tutorial — Learn Python Programming Language - GeeksforGeeks — geeksforgeeks.org. https://www.geeksforgeeks.org/python/python-programming-language-tutorial/. [Accessed 29-10-2025].

[3] Bradley N. Miller and David L. Ranum. Problem solving with algorithms and data structures using python. https://runestone.academy/ns/books/published/pythonds/index.html, 2014. Interactive online edition — accessed: 2024.

[4] Python Software Foundation. *Python: A dynamic, open source programming language*, 2025. Version 3.x documentation.

[5] W3Schools. W3Schools.com — w3schools.com. https://www.w3schools.com/python/default.asp. [Accessed 29-10-2025].