Variables

Assignment Statement

Form:

«variable» = «expression»

How it’s executed:

Evaluate the expression on the right-hand side (RHS) to produce a value. This value has a memory address.

Store that memory address in the variable on the left-hand side (LHS). (Create a new variable if it doesn't exist; otherwise just reuse the existing variable.)
Terminology

For this statement:

\[ x = 7 \]

We say:

“\( x \) gets 7”

“\( x \) refers to the value 7”

“\( x \) contains memory address id1”

“memory address id1 is stored in variable \( x \)”

Variable Names

Must start with a letter (or underscore).

Can include letters, digits, and underscores, but nothing else.

Case matters:

\[
\begin{align*}
\text{age} & = 11 \\
\text{aGe} & \# \text{ Error! This is not defined.}
\end{align*}
\]

Valid: \_moo\_cow, cep3, \_I\_LIKE\_TRASH

Invalid: 49ers, @home
Conventions for the format of names

There’s a Good Reason Why Words Have a Standard Capitalization Scheme

Python convention: pothole_case

CamelCase is sometimes seen, but not for function and variable names

Rarely, single-letter names are capitalized: L, X, Y

When in doubt, use lowercase_pothole

Choosing good names

Python doesn’t care about the content of the names, only their format. (It doesn’t understand English.)

For example, these are equally fine names to Python: xx3, class_average, fraggle

We choose names that will be meaningful to the humans who will read our code.

Example: if you are adding something up, total is better than x.

You will be graded on the names you pick.