Variables: What can we name them?

- Variable names must start with a letter or the underscore ("_") character.
- After that, it can be followed by any number of letters, underscores, or digits.
- Variable names are case-sensitive, so "score" is different from "Score".
- You must avoid using reserved words as variable names: these are words that have a special meaning in a programming language such as Python.
  - For example: def, str, print, etc.
  - IDLE displays reserved words in color to help you recognize them, which is useful since most people don’t know all of them.
Variables: What *should* we name them?

In addition to the hard-and-fast rules on the previous chart, there are also naming conventions that all (good) programmers obey:

• You should choose meaningful names that describe what the purpose of the variable is. This helps people reading the program (including you) understand what the code is doing.

  Use `max` rather than `m`

  Use `item` rather than `c`

• Variable names should begin with a lowercase letter.

• It is common to combine multiple words (such as `avgHeight`) into a variable name in order to be descriptive. When you do this, improve readability by using lowercase for the first “word” and uppercase for subsequent words. (This is called “camelCasing”.)
Mixed-Type Expressions

Most arithmetic operations behave as you would expect for all data types.

- Combining two `floats` results in an `float`.
- Combining two `ints` results in an `int` (provided you do division with `//`).
- Dividing two `ints` using float division is an exception: it behaves as you probably want it to. For instance, `5 / 2` gives you 2.5.

Python will figure out what the result should be and make the result the appropriate data type.
Keyboard Input

The `input()` function is used to read data from the user during program execution.

Format:

```
input (<prompt string>)
```

When it’s called:

- It displays the “prompt string”, a `str`. The intent is that it should be a message to the user that the program is waiting for the user to type in a string.
- It will wait until the user types something and hits the “Enter” or “Return” key.
- It returns whatever the user typed as a `str` as a `return value`.
A program segment using assignment statements

Execute each of these statements in sequence (on paper, not using IDLE) and show what each one does. (Beware of tricks!)

```python
print ("Start here")
firstNum = 3 + (16 - 4) / 3
print ("The first number is: ")
secondNum = 13 % 2 - 1
print ("The second number is: secondNum")
thirdNum = secondNum + 5
secondNum = secondNum + 3
print ("The numbers are: ", firstNum, secondNum, thirdNum)
```
Special assignment statements

Augmented assignment statements:

\[ i \ += \ 8 \quad \text{means the same as} \quad i \ = \ i \ + \ 8 \]
\[ i \ -= \ 8 \quad \text{means the same as} \quad i \ = \ i \ - \ 8 \]
\[ i \ *= \ 8 \quad \text{means the same as} \quad i \ = \ i \ * \ 8 \]
\[ i \ /= \ 8 \quad \text{means the same as} \quad i \ = \ i \ / \ 8 \]
\[ i \ //= \ 8 \quad \text{means the same as} \quad i \ = \ i \ // \ 8 \]
\[ i \ %= \ 8 \quad \text{means the same as} \quad i \ = \ i \ % \ 8 \]
\[ i \ **= \ 8 \quad \text{means the same as} \quad i \ = \ i \ ** \ 8 \]
Special assignment statements (cont.)

**Multiple assignment:**

\[ i = j = k = 1 \]

means the same as:

\[ k = 1 \]
\[ j = k \]
\[ i = j \]

**Simultaneous assignment:**

\[ m, n = 2, 3 \]

means the same as:

\[ m = 2 \]
\[ n = 3 \]

with the caveat that the assignments happen *at the same time.*