Data Types

- Every piece of information a Python program stores or manipulates has a "data type" that tells Python what kind of information it is.
- Python cares about the data type of information in a program because every data type is stored in a different way inside the computer.
- Common data types:
  - Integer (whole number)
  - Floating point (number with a decimal point)
  - String (sequence of letters, numbers, spaces, and/or other characters)

Literals

- A literal in a Python programs refers to a piece of information you "literally" put into the source code of the program.
- Any number (integer or floating point) that you put directly into the code of a program is a literal.
- Any string (sequence of characters surrounded by double or single quotes) that you put directly into your code is a string literal.
- Ex: `print(6)                 # 6 is a literal`
- Ex: `print("I love CS 141!")  # "I love CS 141!" is a string literal`
- Ex: `print(Hello world)       # This is a syntax error; string literals must be`
  `                          # enclosed by quotes`

Variables

- A variable is a name associated with a value in a program.
- You can use a variable in your program any place you would use a literal.
- When you use a variable (e.g., in a print statement), Python substitutes the **current** value of the variable in place of the name.
  - **Current** value is important because variable values may change throughout a program.
- Variables obtain their values through assignment statements.
  - *`name_of_variable = value_you_want_to_put_in_the_variable`*
  - After the line above, from that point forward in the program, the variable name on the left of the equals sign will have the value from the right side of the equals sign.
  - When an assignment statement is encountered, any old value of the variable is **lost**.
  - Left side of the equals sign must be a single variable.
  - Right side can be a literal, another variable, or an expression that evaluates to a value. Even if the right side has variables, they don't change as the result of an assignment statement. **Only the left side changes**.
- Ex:

```
x = 6            # assigns 6 to the variable x
y = 7            # assigns 7 to the variable y
print(x, y)      # prints 6 7
x = -2.1         # reassigns x a new value; 6 is lost
y = x            # reassigns y the value of -2.1 as well
x = "A string"   # reassigns x a new value; y is still -2.1
print(x, y)      # prints A string -2.1
z = y + 2.1      # new variable z gets the value 0
z = z + 1        # this is legal!  Right side is evaluated first,
                 # then z gets that new value.
print(z)         # prints 1
```

- Assignment statements do not cause a variable's value to be printed. If you want to see the value of a variable, you must use a print statement to display it.

Input Statements

- An input statement is used to get input from the keyboard; that is, it is used to have the user type in a number or a string and store what the user types in a variable.
- There are three types of input statements, each one corresponding to a data type.
- All three are variations of a variable assignment statement.
- When Python encounters an input statement, the program pauses, displays the prompt string, and waits for the user to type something and press enter.
- Whatever the user types is stored in the variable on the left side of the equals sign.

- For integers:
  - `variable = int(input(“Prompt”))`

- For floats:
  - `variable = float(input(“Prompt”))`

- For strings:
  - `variable = input(“Prompt”)`

- The “Prompt” part in the examples above is the prompt string. It is what is displayed to the user while the program is paused waiting for the user to type something, and is therefore usually some sort of message telling the user what to type. The string itself can be anything you want, and has no effect on what information goes into the variable.