

## An annotated *input-compute-output* program

By convention, execution starts in *main*. (See bottom of this file.)

```
def main():
    convert_to_celsius()
    convert_to_fahrenheit()
```

**Defining a function:** The *def* keyword begins the definition of a function – a chunk of code that has a name. Note the parentheses and colon. The *body* of the function is indicated by indenting it. The function definition ends when the indentation ends.

```
def convert_to_celsius():
```

```
    """
    Prompts for and inputs a Fahrenheit temperature
    and prints the equivalent Celsius temperature.
    """
```

The three-quotes-in-a-row indicates a green *doc-comment* – multi-line documentation that is helpful to others who might use this function. If you are implementing a function, you must arrange for the function to accomplish whatever the doc-comment says to do.

```
    input_string = input('What is the Fahrenheit temperature? ')
    fahrenheit = float(input_string)
    celsius = (fahrenheit - 32) * (5 / 9)
    print('That temperature is', celsius, 'degrees Celsius.')
```

The *input* function. This is a key statement that illustrates calling a function, sending it an argument, and capturing the returned value in a variable. It prints the argument on the Console as a prompt for the user, waits at the Console for the user to type a line, then returns the line that the user typed. Note that *input* always returns a *string*.

```
def convert_to_fahrenheit():
```

```
    """
    Prompts for and inputs a Celsius temperature
    and prints the equivalent Fahrenheit temperature.
    """
```

The *float* function converts the string that *input* returns to a floating-point number. It is necessary here because we want to do arithmetic on that number.

```
    celsius = float(input('What is the Celsius temperature? '))
    fahrenheit = ((9 / 5) * celsius) + 32
    print('That temperature is', fahrenheit, 'degrees Fahrenheit.')
```

The *print* function prints its arguments, all on one line, separated by spaces. We'll learn fancier printing later.

```
#-----
# If this module is running at the top level (as opposed to being
# imported by another module), then call the 'main' function.
#-----
```

The *#* sign makes the rest of the line a *comment* (ignored by the computer, important to humans). These “inline” comments are usually comments from the author of the code to other programmers who read the code.

```
if __name__ == '__main__':
    main()
```

If the program is being run via the *Run* button (or its equivalent), this statement calls the *main* function. I'll always put this at the bottom of our files, so our programs will always start in *main*, by convention.