GERMANNA
ACADEMIC CENTER
FOR EXCELLENCE

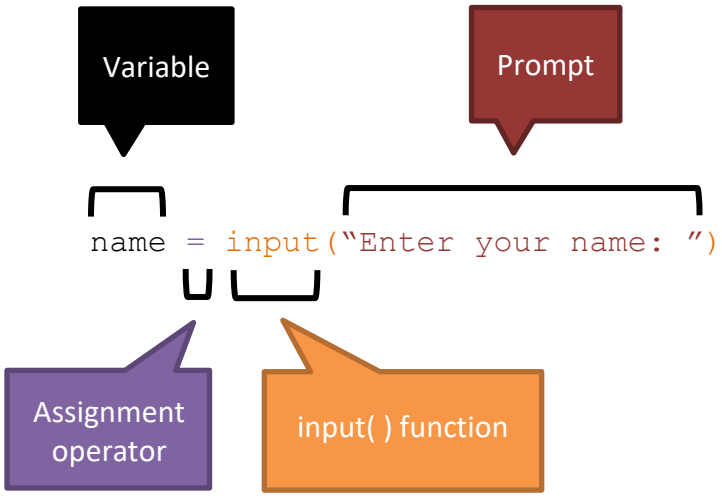
Python: Input and Output

One of the foundational concepts in programming is writing software that interacts with users by outputting information to them or allowing them to input information themselves. This handout will explain how to perform input and output operations in Python.

This handout will reference variables and data types to explain the concepts of input and output in Python. For more information about variables and data types, please refer to the Academic Center for Excellence's Python: Variables and Data Types handout.

Requesting User Input as a String

Some programs require the user to input data in order for the program to continue operating. The information that the user enters is most often used to perform a calculation or complete a task. When asking a user for input in Python, there are typically four elements that are needed: a variable, assignment operator, input function, and a prompt. Consider a Python program that prompts a user to enter their name. The following example shows the line of code that may be written to construct this program as well as a description for each element that is needed to write the line.



Variable: The variable will act as a reference name for the user's input, regardless of what the input is. The variable holds the unique content that the user assigns to it. Throughout the rest of the program, the user's input can be accessed by referencing the variable.

Assignment operator: The assignment operator assigns the content that is to the right of it — in this case, the user's input — to the variable on the left side.

input( ) function: This function pauses the program flow to allow the user to input data. The input function will make the input a string by default. A string is a sequence of characters (or a single character) that is surrounded by quotation marks and typically represents text, symbols, or numbers that will not be used for calculations.

Prompt: The prompt is a string that informs the user of what data to enter.

When this line of code is run, the result is as follows:

```
Enter your name:
```

At this point in the program, the user would enter their name. After they input their name and press the Enter key, the data that was entered is saved to the "name" variable.

Requesting User Input as a Number

Depending on the input needed, the programmer may require the user to enter data that can be interpreted as a string, such as their name. However, there are other scenarios in which the user needs to enter data that the programmer wants to be interpreted as a number, such as an age. In this case, the line of code is written differently. In addition to the elements that were present in the previous example, the int( ) function is added. The int( ) function is used to convert a non-integer to an integer. This is referred to as "casting."

When the above line of code is added to the previous example, the result is as follows:

```
Enter your name: John
Enter your age:
```

At this point in the program, the user has already entered their name and is now prompted to enter their age. Due to the int( ) function being used, the data that is entered for "age" will be converted to an integer.

Keep in mind that values can be cast to many different data types, not just integers. For example, if a program is prompting a user to enter a value that may have decimal places, such

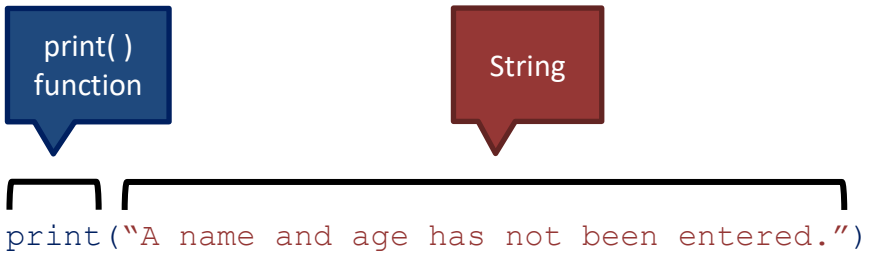
as an hourly wage, the float( ) function would be used to cast the user's input to a floating-point number.

For more information about the int( ) and float( ) functions, please refer to the Academic Center for Excellence's Python: Variables and Data Types handout.

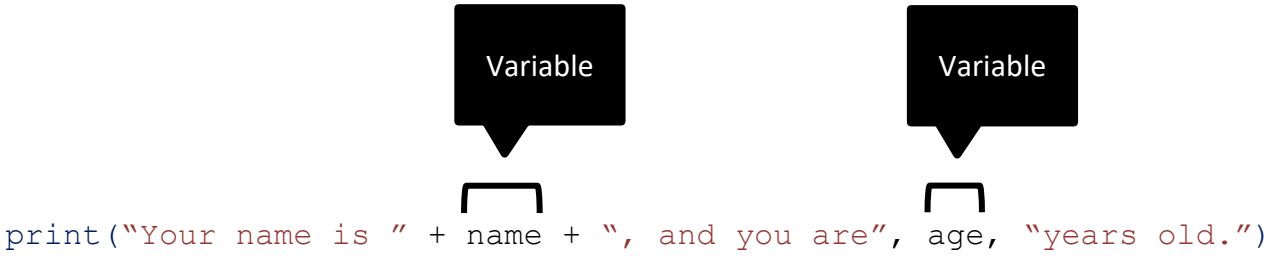
Output

In contrast to when a program asks the user for information or input, programs may need to display information or output to the user. This is most often done by using the print( ) function. When implemented, this function sends the contents within its corresponding parentheses to the screen to be displayed.

The structure of a print statement is dependent on what the programmer is attempting to display. If a string is the only object that needs to be displayed, then the print statement would solely contain that string. An example of this scenario is shown below.



However, in other cases, it may be appropriate to include a variable in the print statement along with a string. For example, when outputting the user's name and age, it would be better to display a sentence that provides context to the name and age that is being displayed rather than just displaying the values themselves. Commas and plus signs can be used to print strings and variables inside a single statement. The example shown below demonstrates this.



The plus signs are used for combining strings, which is referred to as concatenating. On the other hand, commas are used as separators; they are often used to transition from one data type to another.

The final output for all of the examples put together is shown below.

```
Enter your name: John
Enter your age: 26
Your name is John, and you are 26 years old.
```

The final line of the output — Your name is John, and you are 26 years old — demonstrates the result of the previous print statement (`print("Your name is " + name + ", and you are", age, "years old.")`). An analysis of the final line of the output is as follows:

- The phrases “Your name is”, “, and you are”, and “years old” go in between quotation marks in the print statement because they should appear in the output exactly as they are written.
- On the other hand, the words “name” and “age” are not placed in quotation marks because they should not appear in the output; they are variables that are being used to represent values.
- In the output, the name “John” takes the place of the variable “name” because the variable represents the name that was entered by the user, which was “John.”
- Likewise, “26” takes the place of the “age” variable for the same reason.

Additional Resources

For more information about Python concepts, visit the Academic Center for Excellence’s IST Course Resources webpage.

The Academic Center for Excellence (ACE) offers free on-campus and online tutoring appointments for software design. To schedule an appointment with a tutor, please call ACE at (540) 891-3017 or email us at ACE@germanna.edu.