INTERACTIVE EVALUATION

The Python Bakery

Expressions

```
# Literal values are expressions
print(5)
print(10.0)
print(True)
print("Hello")

# Math operations are expressions
print(1 + 2)
print(4 - 5)
print(8 / 4)

# Boolean operators are expressions
print(5 > 4)
print(False or not True)
```

An expression is any kind of combination of literal values, arithematic operators, comparison operators, and boolean operators.
Any value alone is an expression.
All operations are expressions.
Expressions can even be made up of other expressions.

Evaluation

- Run a program
- Evaluate an expression

When Python encounters an expression, the result of the expression must be "evaluated".
We have previously described how we "run" a program.
The word "evaluate" is similar, but used for expressions.
You "run" a program, and "evaluate" an expression.
The expression will always result in some value, which Python can then use in some way.

Substitution Model

```
199 + 1 + 300
```

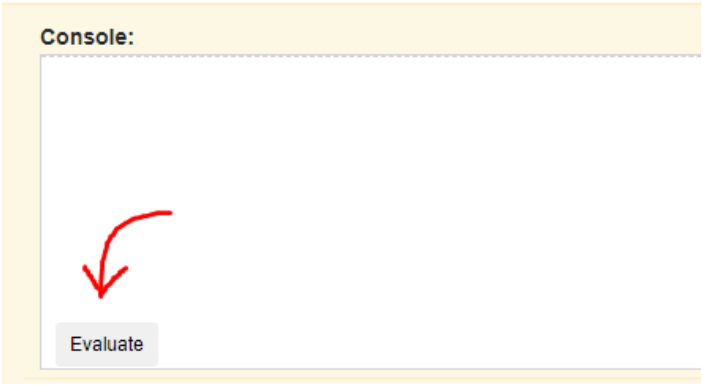
When you look at the expression 199 + 1 + 300, one way to mentally solve the problem is to mentally substitute the 199 + 1 term for 300.
Then, you can do the easier addition of 200 + 300.
Your brain mentally imagines 200 even though the value is not literally in the expression.
When evaluating an expression, Python will do the same kind of substitution.
They do not show the intermediate results like the 200, but regardless Python still did that calculation along the way.
Interestingly, by default, Python will not show the final result either.
Unless you use the print function, the result will not appear on the console when you run the code shown.
The confusing part here, however, is that Python will still compute the value.
The result is just thrown away!

Throwing Away Results

```
# This is not printed
1 + 2

# This is printed
print(1 + 9)

# This causes an error!
1 / 0

# This cannot be reached
print(4 * 2)
```

Python will evaluate every expression, from the top of the program down to the bottom.
Even if an expression is not printed, the result of the expression will be calculated.
The first 1 + 2 expression is evaluated and results in 3.
But since that 3 is not printed or used in any way, the result is ignored and left behind.
However, the 1+9 expression evaluates to 10, which is then printed on the console.
Things get interesting when we reach the third statement, which is 1/0.
That expression is invalid, but must still be evaluated.
When Python tries to do division by zero, the program will stop with an error message.
This prevents Python from reaching the second print statement or even evaluating 4 * 2.

Seeing the Results



We have previously described how the print and input commands will write text to the Console area.
We said that the console was like an interactive chat window, that you can use to communicate with the computer.
This is actually even more true than we suggested before, thanks to the Console's ability to evaluate expressions.
In BlockPy, for instance, you can click the Evaluate button to make a new box appear where you can write expressions.
Those expressions will be evaluated and the result will be shown immediately.
No print statement is required when you use the Evaluate mode.
Most programming environments have some kind of console, also known as a shell, that can let you write and evaluate expressions.
They sometimes call this a REPL, which stands for read-evaluate-print-loop.
The computer repeatedly reads input from the programmer, evaluates the expression, prints the result, and then loops to do it again.
Whenever you need a calculator, remember that the Evaluate button is right there to let you quickly do math or logic with Python.

Console Printing

```
1 + 2
print(3 + 4)
```

Since the console automatically prints, you will not need to include print statements when using the Evaluate button.
But your regular Python programs will definitely still need to print when you want to see the results of execution.
Try running the program shown.
Then, activate the Evaluate button, and finally write and run each line of the program, one at a time.
Compare the output for when you evaluate code compared to when you run code, depending on whether you used a print statement.
Another small note to remember: the evaluate button is only available AFTER a program successfully runs. If there is an error,
the button will not appear.