How Does a Computer Run a Python Program?

In order to understand what happens when you're programming, you need to have a basic understanding of how a computer executes a program.

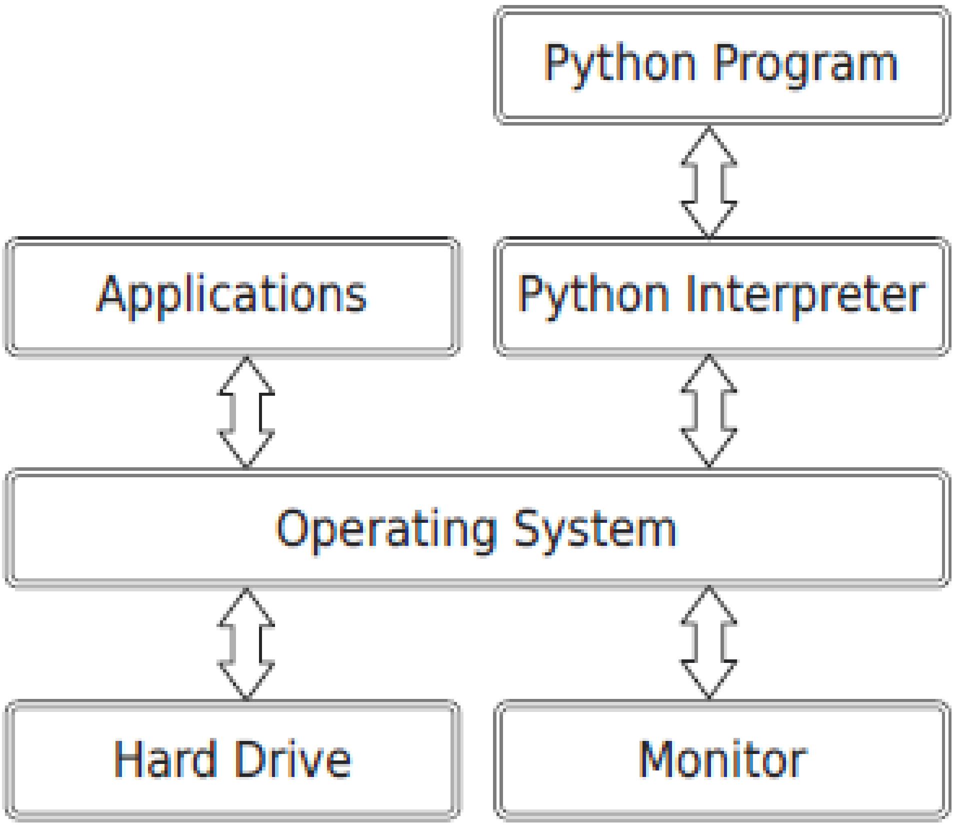
Hardware:
RAM
ROM
Processor .....
Software:
Operating System
Application
Programs

How is the program run

C++, Visual C , Fortran, programs run
directly on top of the OS.
Need compiler
Not portable – OS dependent

Python, Java, or Visual Basic, it doesn’t run directly on top of the OS. Instead, another program, called an *interpreter* or *virtual machine*, takes your program and runs it for you, translating your commands into a language the OS understands. It’s a lot easier, more secure, and more portable across operating systems Hard Drive

Python Interpreter

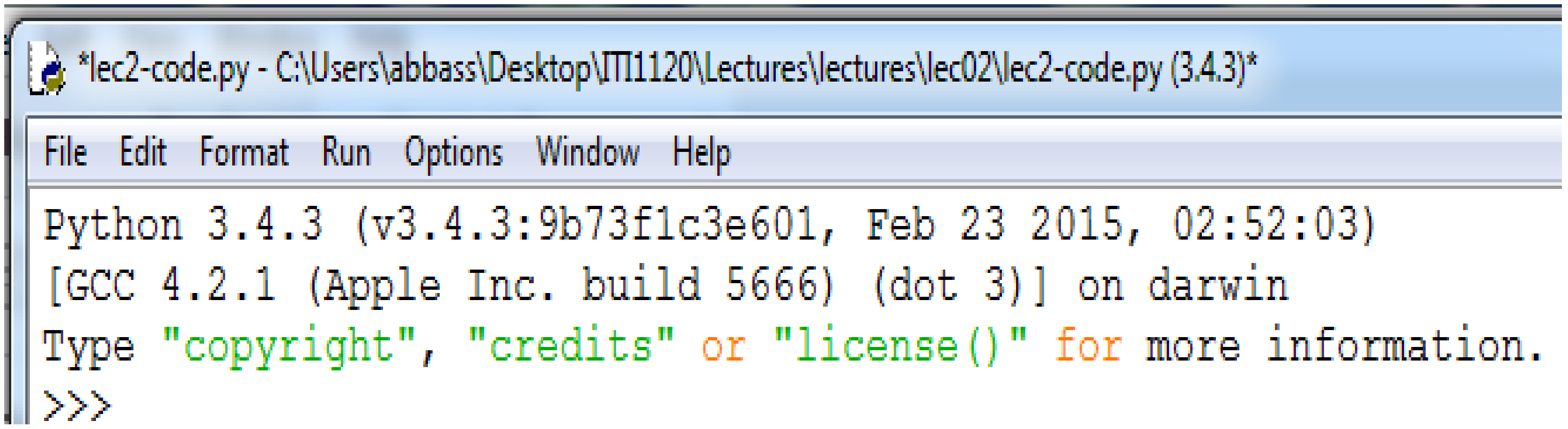


Python Interpreter

There are two ways to use the Python interpreter. One is to tell it to execute a Python program that is saved in a file with a .py extension.

Another is to interact with it in a program called a *shell*, where you type statements one at a time. The interpreter will execute each statement when you type it, do what the statement says to do, and show any output as text, all in one window.
We will explore Python in this lecture using a Python shell.

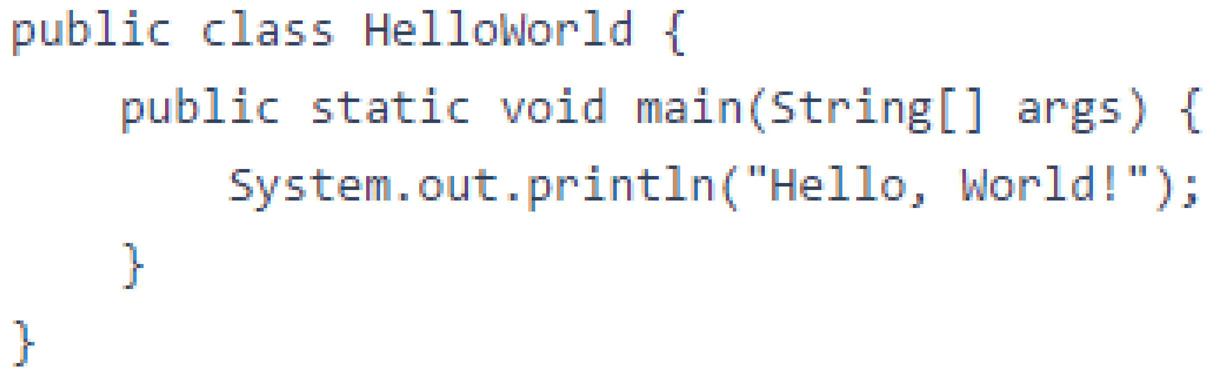
Python comes with a program called IDLE (Integrated Development and Learning Environment), which we use to write Python programs. IDLE has a Python shell that communicates with the Python interpreter and also allows you to write and run programs that are saved in a file.


lec2-code.py - C:\Users\abbass\Desktop\ITI1120\Lectures\lectures\lec02\lec2-code.py (3.4.3)
File Edit Format Run Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Hello World in Python and Java

```
>>> print ('Hello World')
Hello World
```



```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Expressions and Values: Arithmetic in Python

Mathematical expressions like 3 + 4 (“three plus four”)
and 2 - 3 / 5 (“two minus three divided by five”); each expression is built out of *values* like 2, 3, and 5 and *operators* like + and -, which combine their *operands* in different ways.

In the expression 4 / 5, the operator is “/” and the operands are 4 and 5.
Expressions don’t have to involve an operator: a number by itself is an expression.
For example, we consider 212 to be an expression as well as a value.
Like any programming language, Python can *evaluate* basic mathematical expressions. For example, the following expression adds 4 and 13:

```
>>> 4 + 13
17
```

The >>> symbol is called a *prompt*. When you opened IDLE, It is prompting you to type something.
EX: 4 + 13, and then we pressed the Return (or Enter)

Python then evaluated the expression.

When typed in the shell, Python shows the value that is produced.

Error will reported if any

```
>>> 15 - 3
12
>>> 4 * 7
28
```

The following expression divides 5 by 2:

```
>>> 5 / 2
2.5
```

The result has a decimal point. In fact, the result of division always has a decimal point even if the result is a whole number:

```
>>> 4 / 2
2.0
```

What Is a Type?

Numbers could be integers or floating-point numbers

type consists of two things:
- a set of values, and
- a set of operations that can be applied to those values.

For example, in type int, the values are …, -3, -2, -1, 0, 1, 2, 3, …
Operators can be applied to those values: +, -, *, /, //, %, and **.

The values in type float are a subset of the real numbers,
same set of operations can be applied to float values.

If an operator can be applied to more than one type of value, it is called an *overloaded operator*.

Arithmetic Operators

Symbol	Operator	Example	Result
-	Negation	-5	-5
+	Addition	11 + 3.1	14.1
-	Subtraction	5 - 19	-14
*	Multiplication	8.5 * 4	34.0
/	Division	11 / 2	5.5
//	Integer Division	11 // 2	5
%	Remainder	8.5 % 3.5	1.5
**	Exponentiation	2 ** 5	32

Operator Precedence

BEDMAS

Let’s put our knowledge of ints and floats to use in converting Fahrenheit to Celsius.

To do this, we subtract 32 from the temperature in Fahrenheit and then multiply by 5⁄9:

```
>>> 212 - 32 * 5 / 9
194.22222222222223
```

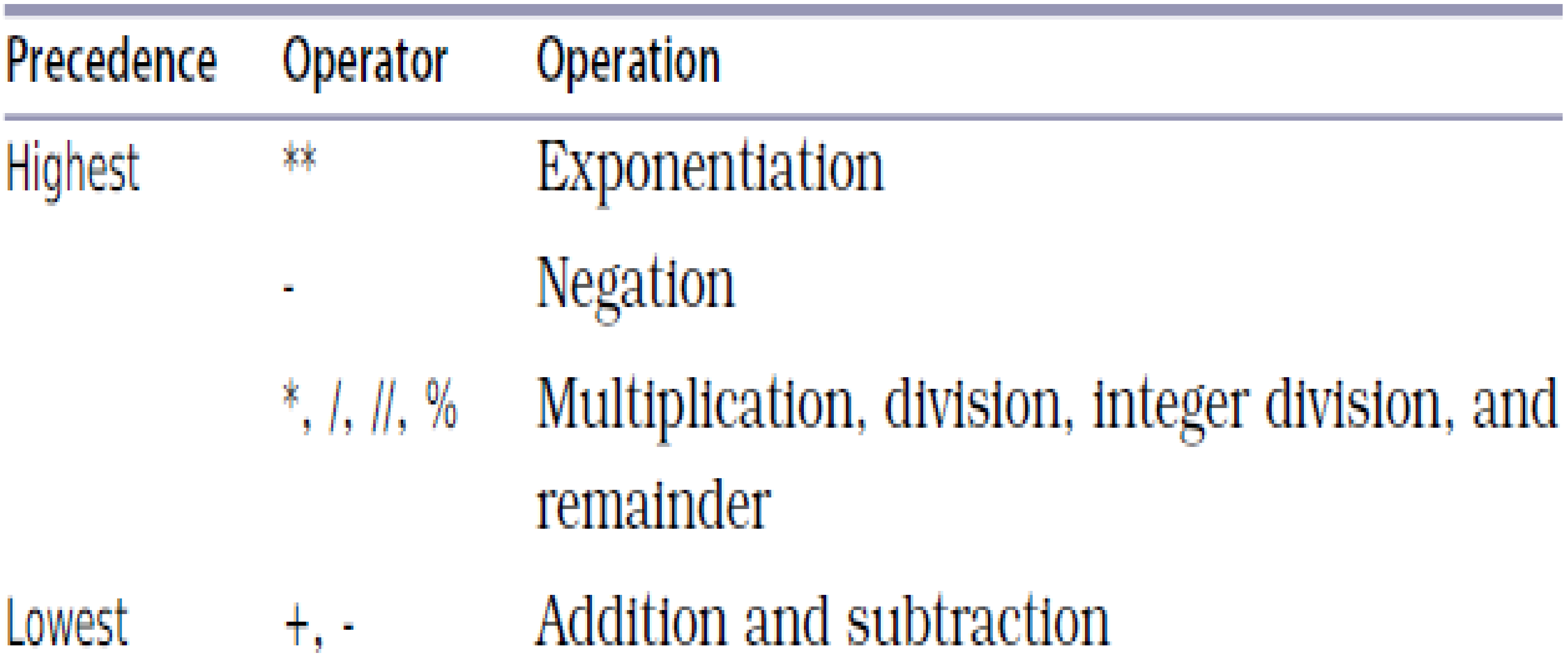
Python claims the result is 194.22222222222223 degrees Celsius, when in fact it
should be 100.

WHAT IS THE CORRECT WAY TO DO?

More Examples:

```
>>> -2 ** 4
?
>>> -(2 ** 4)
?
>>> (-2) ** 4
?
```

Arithmetic Operators Listed by Precedence from Highest to Lowest



Precedence	Operator	Operation
Highest	**	Exponentiation
	-	Negation
	*, /, //, %	Multiplication, division, integer division, and remainder
Lowest	+, -	Addition and subtraction

Is there difference between:
3 + 4 – 5 &   (3 + 4) - 5, OR
3 - 4 + 5   &   (3 - 4) + 5.

Operators on the same row have equal precedence are applied left to right, except for exponentiation, which is applied right to left.

Example, operators + and - are on the same row, parenthesize is not conditionally needed.

Good to include parenthesize for complicated formula, it helps the eye read things like 1 + 1.7 + 3.2 * 4.4 - 16 / 3.

it’s a good rule to *not* use parentheses in simple expressions such as 3.1 * 5.

Variables and Computer Memory: Remembering Values

Like mathematicians, programmers frequently name values so that they can use them later.

A name that refers to a value is called a *variable*.

In Python, variable names can use letters, digits, and the underscore symbol (but they
can't start with a digit).

You create a new variable by *assigning* it a value:

>>> degrees_celsius = 26.0

Can you do that in math: X+1 = 3

Can you do that in Python: X+1 = 3

Does that make sense in math: X=X+1

Does that make sense in Python: X=X+1

Examples

```
>>> degrees_celsius = 26.0
>>> degrees_celsius

>>> 9 / 5 * degrees_celsius + 32

>>> degrees_celsius / degrees_celsius
```

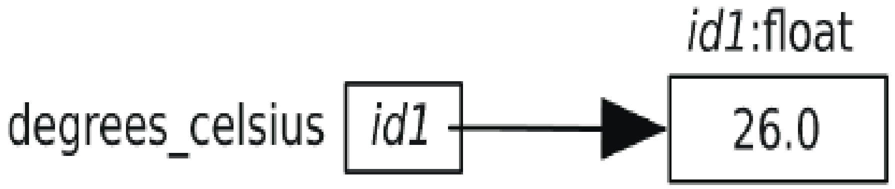
Variables are called *variables* because their values can vary as the program executes. We can assign a new value to a variable:

```
>>> degrees_celsius = 26.0
>>> 9 / 5 * degrees_celsius + 32

>>> degrees_celsius = 0.0
>>> 9 / 5 * degrees_celsius + 32
```

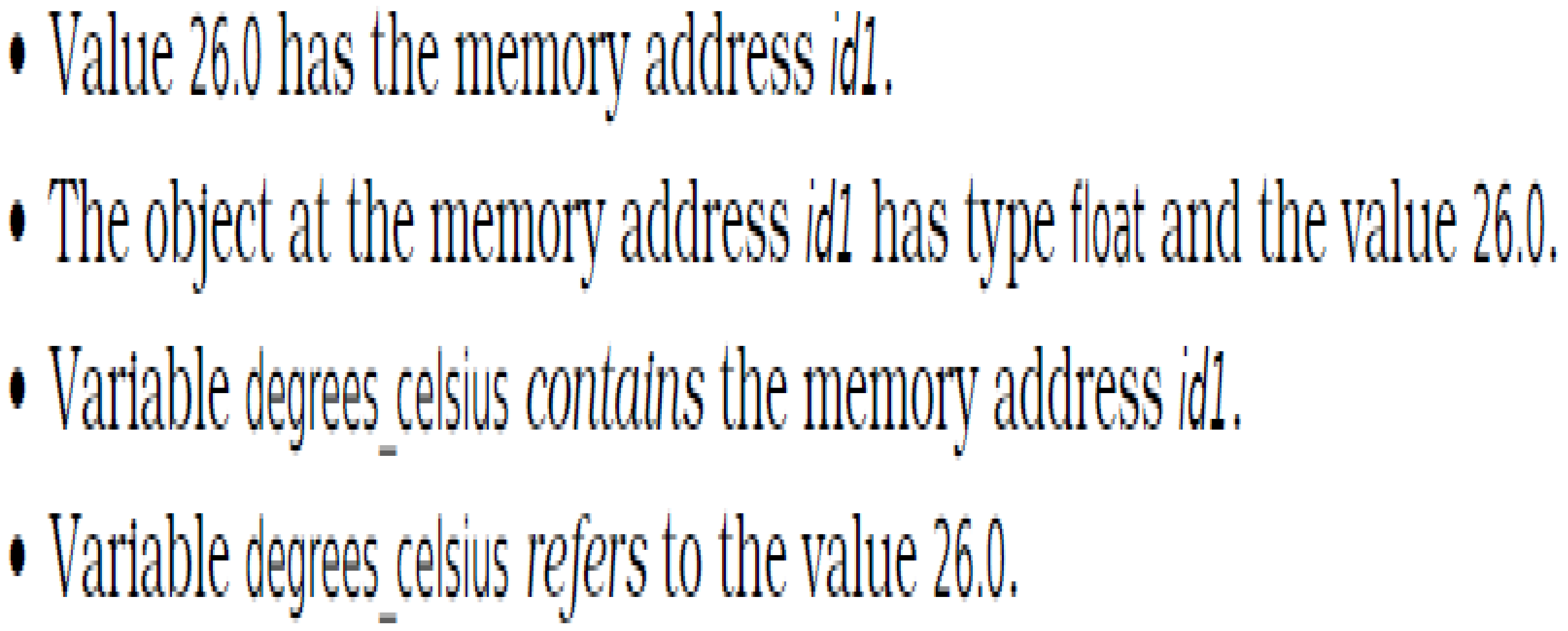
Values, Variables, and Computer Memory

In our memory model, a variable contains the memory address of the object to which it refers:



In order to make the picture easier to interpret, we will usually draw arrows from variables to their objects.

The following terminology of the assignment



- Value 26.0 has the memory address *id1*.
- The object at the memory address *id1* has type float and the value 26.0.
- Variable degrees_celsius *contains* the memory address *id1*.
- Variable degrees_celsius *refers* to the value 26.0.

Assignment Statement

Here is the general form of an assignment statement:

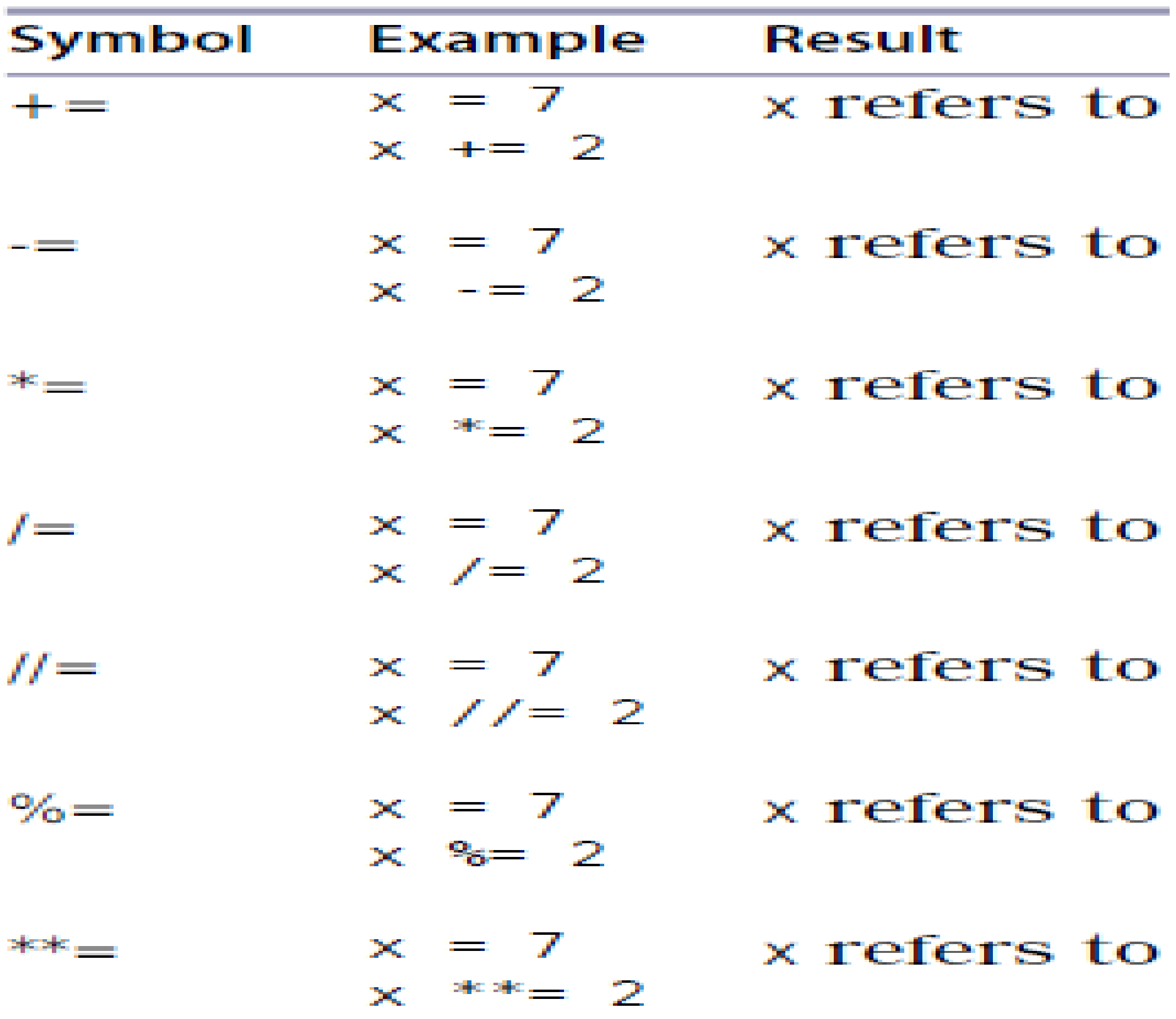
```
«variable» = «expression»
```

This is executed as follows:

1. Evaluate the expression on the right of the = sign to produce a value. This value has a memory address.
2. Store the memory address of the value in the variable on the left of the =. Create a new variable if that name doesn't already exist; otherwise, just reuse the existing variable, replacing the memory address that it contains.

Warning: = Is Not Equality in Python!

Augmented Assignment Operators



Symbol	Example	Result	

`+=`	`x = 7`
	`x += 2`
		x refers to	?

`-=`	`x = 7`
	`x -= 2`
		x refers to	?

`*=`	`x = 7`
	`x *= 2`
		x refers to	?

`/=`	`x = 7`
	`x /= 2`
		x refers to	?

`//=`	`x = 7`
	`x //= 2`
		x refers to	?

`%=`	`x = 7`
	`x %= 2`
		x refers to	?

`**=`	`x = 7`
	`x **= 2`
		x refers to	?

How Python Tells You Something Went Wrong

Two kinds of errors in Python:

syntax errors: which happen when you type something that isn't valid Python code,

semantic errors: which happen when you tell Python to do something that it just can't do, like divide a number by zero or try to use a variable that doesn't exist.

Examples:

A Single Statement That Spans Multiple Lines

Sometimes statements get pretty intricate. The recommended Python style is to limit lines to 80 characters, including spaces, tabs, and other *whitespace* characters, and that’s a common limit throughout the programming world.

Example:

Example - Problem:

we’re baking cookies.
Canada uses Celsius, but we own cookbooks that use Fahrenheit.
We are wondering how long it will take to preheat our oven.

Here are our facts:

- The room temperature is 20 degrees Celsius.
- Our oven controls use Celsius, and the oven heats up at 20 degrees per minute.
- Our cookbook uses Fahrenheit, and it says to preheat the oven to 350 degrees.

Fine the oven heating time?

Exercises

1. Which of the following expressions results in Syntax Errors?

a. 6 * -----------8

b. 8 = people

c. ((((4 ** 3))))

d. (-(-(-(-5))))

e. 4 += 7 / 2

2. Write a bullet list description of what happens when Python evaluates the statement x += x - x when x has the value 3.