The engine that translates and runs Python is called the **Python Interpreter:** There are two ways to use it: im*mediate mode* and *script mode.*

**1.** In immediate mode, you type Python expressions into the Python Interpreter window, and the interpreter immediately shows the result:

**2.** Alternatively, you can write a program in a file and use the interpreter to execute the contents of the file. Such a file is called a script. Scripts have the advantage that they can be saved to disk, printed, and so on.

The >>> is called the .

A **program** is a sequence of instructions that specifies how to perform a computation. The computation might be something mathematical, such as solving a system of equations or finding the roots of a polynomial, but it can also be a symbolic computation, such as searching and replacing text in a document.

**input** Get data from the keyboard, a file, or some other device such as a sensor.

**output** Display data on the screen or send data to a file or other device such as a motor.

**math** Perform basic mathematical operations like addition and multiplication.

**conditional execution** Check for certain conditions and execute the appropriate sequence of statements.

**repetition** Perform some action repeatedly, usually with some variation.

Programming errors are called **bugs** and the process of tracking them down and correcting them is called **debugging**.

Three kinds of errors can occur in a program: *syntax errors,* *runtime errors*, and *semantic errors*.

**Syntax errors**

Python can only execute a program if the program is syntactically correct; otherwise, the process fails and returns an error message. **Syntax** refers to the structure of a program and the rules about that structure.

**Runtime errors**

Runtime error, so called because the error does not appear until you run the program. These errors are also called exceptions because they usually indicate that something exceptional (and bad) has happened.

**Semantic errors**

If there is a semantic error in your program, it will run successfully, in the sense that the computer will not generate any error messages, but it will not do the right thing. It will do something else. Specifically, it will do what you told it to do.

**Natural languages** are the languages that people speak, such as English, Spanish, and French. They were not designed by people (although people try to impose some order on them); they evolved naturally.

**Formal languages** are languages that are designed by people for specific applications. For example, the notation that mathematicians use is a formal language that is particularly good at denoting relationships among numbers and symbols.

***Programming languages are formal languages that have been designed to express computations.***

Formal languages tend to have strict rules about syntax. For example, 3+3=6 is a syntactically correct mathematical

statement, but 3=+6$ is not. H 2 O is a syntactically correct chemical name, but 2Zz is not.

Syntax rules come in two flavors, pertaining to **tokens** and **structure**.

Tokens are the basic elements of the language, such as words, numbers, parentheses, commas, and so on. In Python, a statement like print("Happy New Year for ",2013) has 6 tokens: a function name, an open parenthesis (round bracket), a string, a comma, a number, and a close parenthesis.

The second type of syntax rule pertains to the structure of a statement— that is, the way the tokens are arranged. The statement 3=+6$ is structurally illegal because you can’t place a plus sign immediately after an equal sign. *And in our Python example*, if we omitted the comma, or if we changed the two parentheses around to say print)"Happy New Year for ",2013( our statement would still have six legal and valid tokens, but the structure is illegal.

When you read a sentence in English or a statement in a formal language, you have to figure out what the structure of the sentence is. This process is called **parsing.**

**The first program**

print("Hello, World!")

**The print function,** displays a value on the screen.

Hello, World!

The quotation marks in the program mark the beginning and end of the value; they don’t appear in the result.

**Comments**

A comment in a computer program is text that is intended only for the human reader — it is completely ignored by the interpreter.

In Python, the # token starts a comment. The rest of the line is ignored.

|  |
| --- |
| #---------------------------------------------------  # This demo program shows off how elegant Python is!  # Written by Amit Pratap Singh, January 2019.  # Anyone may freely copy or modify this program.  #---------------------------------------------------  print("Hello, World!") # Isn't this easy! |