Introduction to Python Programming - Glossary

activecode

A unique interpreter environment that allows Python to be executed from within a web browser.

algorithm

A general step by step process for solving a problem.

bug

An error in a program.

byte code

An intermediate language between source code and object code. Many modern languages first compile source code into byte code and then interpret the byte code with a program called a *virtual machine*.

codelens

An interactive environment that allows the user to control the step by step execution of a Python program

comment

Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

compile

To translate a program written in a high-level language into a low-level language all at once, in preparation for later execution.

debugging

The process of finding and removing any of the three kinds of programming errors.

exception

Another name for a runtime error.

executable

Another name for object code that is ready to be executed.

formal language

Any one of the languages that people have designed for specific purposes, such as representing mathematical ideas or computer programs; all programming languages are formal languages.

high-level language

A programming language like Python that is designed to be easy for humans to read and write.

interpret

To execute a program in a high-level language by translating it one line at a time.

low-level language

A programming language that is designed to be easy for a computer to execute; also called machine language or assembly language.

natural language

Any one of the languages that people speak that evolved naturally.

object code

The output of the compiler after it translates the program.

parse

To examine a program and analyze the syntactic structure.

portability

A property of a program that can run on more than one kind of computer.

print function

A function used in a program or script that causes the Python interpreter to display a value on its output device.

problem solving

The process of formulating a problem, finding a solution, and expressing the solution.

program

A sequence of instructions that specifies to a computer actions and computations to be performed.

programming language

A formal notation for representing solutions.

Python shell

An interactive user interface to the Python interpreter. The user of a Python shell types commands at the prompt (>>>), and presses the return key to send these commands immediately to the interpreter for processing.

runtime error

An error that does not occur until the program has started to execute but that prevents the program from continuing.

semantic error

An error in a program that makes it do something other than what the programmer intended.

semantics

The meaning of a program.

shell mode

A style of using Python where we type expressions at the command prompt, and the results are shown

immediately. Contrast with **source code**, and see the entry under **Python shell**.

source code

A program, stored in a file, in a high-level language before being compiled or interpreted.

syntax

The structure of a program.

syntax error

An error in a program that makes it impossible to parse — and therefore impossible to interpret.

token

One of the basic elements of the syntactic structure of a program, analogous to a word in a natural language.

Simple Python Data - Glossary

assignment statement

A statement that assigns a value to a name (variable). To the left of the assignment operator, `=`, is a name. To the right of the assignment token is an expression which is evaluated by the Python interpreter and then assigned to the name. The difference between the left and right hand sides of the assignment statement is often confusing to new programmers. In the following assignment:

```
n = n + 1
```

`n` plays a very different role on each side of the `=`. On the right it is a *value* and makes up part of the *expression* which will be evaluated by the Python interpreter before assigning it to the name on the left.

assignment token

`=` is Python's assignment token, which should not be confused with the mathematical comparison operator using the same symbol.

class

see **data type** below

comment

Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

data type

A set of values. The type of a value determines how it can be used in expressions. So far, the types you have seen are integers (`int`), floating-point numbers (`float`), and strings (`str`).

decrement

Decrease by 1.

evaluate

To simplify an expression by performing the operations in order to yield a single value.

expression

A combination of operators and operands (variables and values) that represents a single result value. Expressions are evaluated to give that result.

float

A Python data type which stores *floating-point* numbers. Floating-point numbers are stored internally in two parts: a *base* and an *exponent*. When printed in the standard format, they look like decimal numbers. Beware of rounding errors when you use `float`s, and remember that they are only approximate values.

increment

Both as a noun and as a verb, increment means to increase by 1.

initialization (of a variable)

To initialize a variable is to give it an initial value. Since in Python variables don't exist until they are assigned values, they are initialized when they are created. In other programming languages this is not the case, and variables can be created without being initialized, in which case they have either default or *garbage* values.

int

A Python data type that holds positive and negative **whole** numbers.

integer division

An operation that divides one integer by another and yields an integer. Integer division yields only the whole number of times that the numerator is divisible by the denominator and discards any remainder.

keyword

A reserved word that is used by the compiler to parse program; you cannot use keywords like `if`, `def`, and `while` as variable names.

modulus operator

Also called remainder operator or integer remainder operator. Gives the remainder after performing integer division.

object

Also known as a data object (or data value). The fundamental things that programs are designed to manipulate (or that programmers ask to do things for them).

operand

One of the values on which an operator operates.

operator

A special symbol that represents a simple computation like addition, multiplication, or string concatenation.

prompt string

Used during interactive input to provide the use with hints as to what type of value to enter.

reference diagram

A picture showing a variable with an arrow pointing to the value (object) that the variable refers to. See also **state snapshot**.

rules of precedence

The set of rules governing the order in which expressions involving multiple operators and operands are evaluated.

state snapshot

A graphical representation of a set of variables and the values to which they refer, taken at a particular instant during the program's execution.

statement

An instruction that the Python interpreter can execute. So far we have only seen the assignment statement, but we will soon meet the `import` statement and the `for` statement.

str

A Python data type that holds a string of characters.

type conversion function

A function that can convert a data value from one type to another.

value

A number or string (or other things to be named later) that can be stored in a variable or computed in an expression.

variable

A name that refers to a value.

variable name

A name given to a variable. Variable names in Python consist of a sequence of letters (a..z, A..Z, and _) and digits (0..9) that begins with a letter. In best programming practice, variable names should be chosen so that they describe their use in the program, making the program *self documenting*.
