

## Glossary for Programming Basics

### A

- algorithm [[Introduction to algorithms](#)]
  - A set of instructions that describes how to get the exact result you want
- argument [[Setting parameters and arguments](#)]
  - The actual value given in a function call that “fills in” the parameter variable
- array [[Understanding collections](#)]
  - A collection that groups pieces of data in a certain order and assigns the collection a name, also called an array or list in some programming languages; the position of each element is defined by an index.
- assignment operator [[Introduction to variables and data types](#)]
  - The equals sign; the value on the right is assigned to (stored in) the variable named on the left-hand side.
- attribute [[Introduction to object-oriented programming](#)]
  - Data that an object has, also called a field or property
- autocompletion [[Debugging code in an IDE](#)]
  - Also known as code completion, this is a feature of IDEs that offer to finish code for you while you’re typing.

### B

- block [[Exploring conditional code](#)]
  - A group of statements that belong together
- bug [[What is programming?](#)]
  - Something that happens unexpectedly in a program

### C

- class [[Introduction to object-oriented programming](#)]
  - A detailed description; the definition or template of what an object will be
- collection [[Understanding collections](#)]
  - A programming construct that lets you group similar items together
- comment [[Working with comments](#)]
  - A note to describe what a program does
- comment out [[Working with comments](#)]

- Using comments to temporarily ignore a section of code
- compile [[Running your code](#)]
  - Translate a program from high-level code to machine code
- concatenation [[Combining and manipulating strings](#)]
  - Combining multiple strings into a single string; in many languages, the + operator is used to concatenate strings.
- concise [[Why Python?](#)]
  - A concise programming language takes fewer lines of code to accomplish a goal.
- conditional expression [[Making decisions in code](#)]
  - Any expression that evaluates to True or False; also called a Boolean expression, named after British mathematician George Boole
- constructor function [[Creating custom classes and objects](#)]
  - A function whose purpose is to specify the properties to use when you create an object; in many program languages, you invoke the constructor by giving the name of the class it belongs to.
- crash [[What is programming?](#)]
  - When a program stops early or freezes

## D

- data type [[Introduction to variables and data types](#)]
  - A category for values; for example, a value can be a number, or it can be a string of text characters. A data type also defines how we can operate on those values.
- debug [[Introduction to debugging](#)]
  - The process of identifying and fixing bugs in a program
- dictionary (collection) [[Understanding collections](#), [Creating more complex collections](#)]
  - A collection that lets you store related information with a label for each item, also called a map, hash map, table, or associative array in some programming languages

## E

- expression [[Basic statements and expressions](#)]
  - A combination of operands and operators that break down into a single value

## F

- field [[Introduction to object-oriented programming](#)]
  - Data that an object has, also called an attribute or property

- file extension [[Writing source code](#)]
  - The characters after the last “.” in a file name; we usually use “.py” for Python, “.js” for JavaScript, etc.
- float [[Working with numbers](#)]
  - Any number with a decimal point (short for “floating point” number)
- for loop [[Iterating through collections](#)]
  - A programming construct in many languages that lets you specify iteration to a specified endpoint; usually used when you know exactly how many times the loop must iterate
- framework [[Understanding libraries and frameworks](#)]
  - Code that is used together in a specific way; frameworks define how you should accomplish a task, with some decisions already made for you.
- function [[Introduction to functions](#)]
  - A block of code packaged together with a name, also called subroutine, procedure, or method
- function body [[Creating and calling functions](#)]
  - The code block that belongs to a function; the code that the function will execute
- function call [[Creating and calling functions](#)]
  - Making a function do its action by giving its name and possibly giving it argument values for parameters, also called invoking a function

## G

- garbage collection [[Memory management across languages](#)]
  - A process by which the runtime system keeps track of which items in memory are no longer needed and deletes them
- gutter [[Running Python in an IDE](#)]
  - The left side of an IDE window, where line numbers appear

## H

- high-level language [[What is a programming language?](#)]
  - A programming language with structure, keywords, and syntax that are easier for humans to understand

## I

- IDE [[Using an IDE](#)]
  - Integrated development environment—an application that provides the tools needed to write, compile, run, and debug programs

- if-else statement [[Working with simple conditions](#)]
  - A statement that tests a condition; when the condition is True, the program will perform a specified action. When the condition is False, the program does the action following the “else.”
- if statement [[Exploring conditional code](#)]
  - A statement that tests a condition; when the condition is True, the program will do the action(s) following the “if.” When the condition is False, the program does not do the action(s).
- immutable [[Collections in other languages](#)]
  - Refers to variables or collections whose existing values cannot change after they are created
- importing [[Comparing types of external code](#)]
  - Referencing a module by its name so your program can use its code
- index [[Understanding collections](#); [Working with collections](#)]
  - A number that gives an item’s position in a list; in most programming languages, the first item in a list has an index of zero.
- index number [[Understanding collections](#); [Working with collections](#)]
  - A number that gives an item’s position in a list; in most programming languages, the first item in a list has an index of zero.
- infinite loop [[Introduction to iteration](#)]
  - A bug that occurs when a loop never stops
- integer [[Working with numbers](#)]
  - A whole number without any decimal places
- IntelliSense [[Using an IDE](#)]
  - A feature of the Visual Studio Code IDE that makes code suggestions as you type
- interpret [[Running your code](#)]
  - Translate a program from high-level code to machine code one source code line at a time
- int method [[Combining and manipulating strings](#)]
  - A method to convert a string to an integer in Python; many other languages have similar methods for type conversion.
- IO [[Introduction to input and output](#)]
  - Input/output
- iterate [[Introduction to iteration](#)]

- Repeat a procedure multiple times until it reaches a specified endpoint

## K

- keyword [[Variables across languages](#)]
  - Reserved words that mean something special in a programming language; for example, the word “for” is a keyword in many languages, including Python, Java, C, and C++.

## L

- linting [[Debugging code in an IDE](#)]
  - An IDE feature that checks code for bugs before you execute it; the name comes from the process of removing lint from clothing.
- list (collection) [[Understanding collections](#)]
  - A collection that groups pieces of data in a certain order and assigns the collection a name, also called an array or list in some programming languages; the position of each element is defined by an index.
- loop [[Introduction to iteration](#)]
  - Code that iterates (repeats until reaching a specified endpoint)
- loop, infinite [[Introduction to iteration](#)]
  - A bug that occurs when a loop never stops

## M

- machine language [[What is a programming language?](#)]
  - The language that a computer “understands”—directly using the instructions wired into the CPU
- method [[Introduction to object-oriented programming](#)]
  - Also called a behavior, a function that specifies what actions an object can perform; it is a block of code that can be called to perform some action, and it may return a value.
- module [[Comparing types of external code](#)]
  - In Python, a file that contains code, such as variables and functions that are related; for example, Python’s random module, which has several functions for generating random numbers
- multi-threading [[Introduction to multithreading](#)]
  - An approach to writing code that executes multiple threads (tasks) concurrently
- mutable [[Collections in other languages](#)]
  - Refers to variables or collections whose existing values can be changed

## O

- object-oriented programming [[Introduction to object-oriented programming](#)]
  - OOP—an approach to programming that breaks programs up into smaller parts known as objects, each of which has its own distinct focus
- operand [[Basic statements and expressions](#)]
  - The value or values that an operator works on; for example, in  $4 + 3$ , the operator is  $+$  and the operands are 4 and 3.
- operator [[Basic statements and expressions](#)]
  - A symbol for an arithmetic operation, such as  $+$ ,  $-$ ,  $*$ , and  $/$
- order of operations [[Basic statements and expressions](#)]
  - The order in which operations are performed, also called priority of operations; for example, multiplication in an expression is done before addition, just as in mathematics— $4 + 3 * 5$  is 19, not 35.

## P

- package [[Understanding libraries and frameworks](#)]
  - Also called a library, this is a group of related modules in Python to help you accomplish a task. It is like a set of tools.
- parameter [[Setting parameters and arguments](#)]
  - Variable name in a function definition, used to modify the behavior of the function code; you can think of it as a “placeholder” that will be filled in when the function is called.
- PEP 8 [[Choosing a code style](#)]
  - Python Enhancement Proposal 8—the most commonly used style guide for Python
- programming [[What is programming?](#)]
  - The process of converting ideas into instructions
- property [[Introduction to object-oriented programming](#)]
  - Data that an object has, also called a field or attribute
- pseudocode [[Writing pseudocode](#)]
  - A description of the logic you're trying to create using plain language

## R

- regular expression [[Creating regular expressions](#)]
  - An encoded description of a pattern that you want to match within a string
- relational operator [[Making decisions in code](#)]

- An operator that yields a True or False value depending on the relation between its operands; for example,  $>$  is a relational operator;  $4 > 2$  evaluates to True.
- return value [[Returning values from functions](#)]
  - The value that a function gives back to the code that called it
- runtime error [[Troubleshooting issues](#)]
  - When the computer is unable to execute part of your code; for example, when  $12 / 0$  can't be calculated

## S

- semantic error [[Troubleshooting issues](#)]
  - When the output of the program is not what you expected, also called a logic error; for example, calculating area of a rectangle as length plus width instead of length times width
- slice [[Finding patterns in strings](#)]
  - In Python, a part of a string or a list; for strings, this operation is called substring in many other programming languages.
- source code [[Writing source code](#)]
  - Instructions for the computer written in plain text
- statement [[Basic statements and expressions](#)]
  - The individual actions you want a program to take; statements can combine keywords, operations, and expressions.
- style guide [[Choosing a code style](#)]
  - A document describing a programmer's (or company's) approach to code
- syntax [[What is a programming language?](#)]
  - The rules for how a programming language expects its code to be written
- syntax error [[Troubleshooting issues](#)]
  - When a program does not follow the rules of the language; for example,  $3 + * 5$
- syntax highlighting [[Using an IDE](#)]
  - Use of color in an IDE to point out keywords and improve readability of code

## T

- test case [[Creating a test case](#)]
  - Commands or scripts that use a program in a way designed to test a specific scenario or feature of the program
- tuple [[Collections in other languages](#)]

- In Python, an immutable list

## V

- variable [[Introduction to variables and data types](#)]
  - A container for a value

## W

- while loop [[Iterating to a custom endpoint](#)]
  - A programming construct in many languages that iterates until the program arrives at a certain state
- white space [[Properly using whitespace](#)]
  - Blanks and/or blank lines; you add these to programs to make them more readable to humans.