Python programming language rules

Table of Contents

[Python programming language rules 3](#_Toc120455406)

[Lines 3](#_Toc120455407)

[Indentation 3](#_Toc120455408)

[**Character Sets** 3](#_Toc120455409)

[Tokens 4](#_Toc120455410)

# Python programming language rules

1. Each Python source file, like any other text file, is a sequence of characters.
2. Each file is made up of a sequence of lines, tokens, or statements.
3. Python is very particular about program layout, especially with regard to lines and indentation

# Lines

1. A Python program is composed of a sequence of logical lines, each made up of one or more physical lines.
2. Each physical line may end with a comment. A hash sign (#) that is not inside a string literal begins a comment. Python interpreter ignores everything after the #.
3. A line containing only whitespace, possibly with a comment, is known as a blank line, and Python totally ignores it.
4. In Python, the end of a physical line marks the end of most statements.
5. Python does not terminate statements with a delimiter, such as a semicolon (;)
6. When a statement is too long to fit on a single physical line, you can join two adjacent physical lines into a logical line by ensuring that the first physical line has no comment and ends with a backslash (\).
7. Python automatically joins adjacent physical lines into one logical line if an open parenthesis ((), bracket ([), or brace ({) has not yet been closed, and taking advantage of this mechanism, generally produces more readable code instead of explicitly inserting backslashes at physical line ends.

# Indentation

1. Python uses indentation to express the block structure of a program.
2. Unlike other languages, Python does not use braces, or other begin/end delimiters, around blocks of statements; indentation is the only way to denote such blocks.
3. Each logical line in a Python program is indented by the whitespace on its left. A block is a contiguous sequence of logical lines, all indented by the same amount;
4. A logical line with less indentation ends the block.
5. All statements in a block must have the same indentation, as must all clauses in a compound statement.
6. The first statement in a source file must have no indentation
7. Standard Python style is to use four spaces (never tabs) per indentation level.
8. NOTE: recommended to configure your text editor to expand tabs to spaces, so that all Python source code you write always contains just spaces, not tabs.

## Character Sets

1. Normally, a Python source file must be entirely made up of characters from the ASCII set (character codes between 0 and 127)
2. Python allows that specific source file to contain characters outside the ASCII set, for comments and string literals.
3. To accomplish this, start your source file with a comment whose form must be as rigid as the following:

# -\*- coding: utf-8 -\*-

# Tokens

Each line of Python code is made up of components called tokens. There are five types of tokens:

**Identifiers** - An identifier is a name used to identify a variable, function, class, module, or other object. An identifier starts with a letter (A to Z or a to z) or an underscore (\_) followed by zero or more letters, underscores, and digits (0 to 9). Case is significant in Python: lowercase and uppercase letters are distinct. Python does not allow punctuation characters such as @, $, and % within identifiers.

**Keywords** - Python has 30 keywords, which are identifiers that Python reserves for special syntactic uses. Keywords contain lowercase letters only. You cannot use keywords as regular identifiers. Some keywords begin simple statements or clauses of compound statements, while other keywords are operators.

* and
* del
* for
* is
* raise
* assert
* elif
* from
* lambda
* return
* break
* else
* global
* not
* try
* class
* except
* if
* or
* while
* continue
* exec
* import
* pass
* with (2.5)
* def
* finally
* in
* print
* yield

**Operators** - Python uses nonalphanumeric characters and character combinations as operators.

* + - addition
* - - subtraction
* \* - multiplication
* / - division
* % -
* \*\*
* //
* <<
* >>
* &
* |
* ^
* ~
* <
* <=
* >
* >=
* <>
* !=
* ==

**Delimiters -** Python uses the following symbols and symbol combinations as delimiters in expressions, lists, dictionaries, various aspects of statements, and strings,

* (
* )
* [
* ]
* {
* }
* ,
* :
* .
* `
* =
* ;
* += - augmented assignment operators
* -= - augmented assignment operators
* \*= - augmented assignment operators
* /= - augmented assignment operators
* //= - augmented assignment operators
* %= - augmented assignment operators
* &= - augmented assignment operators
* |= - augmented assignment operators
* ^= - augmented assignment operators
* >>= - augmented assignment operators
* <<= - augmented assignment operators
* \*\*= - augmented assignment operators
* ' - part of other tokens
* " - part of other tokens
* # - part of other tokens
* \ - part of other tokens

**Literals** - A literal is a number or string that appears directly in a program

* 42 # Integer literal
* 3.14 # Floating-point literal
* 1.0j # Imaginary literal
* 'hello' # String literal
* "world" # Another string literal
* """Good
* night""" # Triple-quoted string literal

**literals and delimiters** - you can create data values of some other fundamental types:

* [ 42, 3.14, 'hello' ] # List
* ( 100, 200, 300 ) # Tuple
* { 'x':42, 'y':3.14 } # Dictionary

**Statements -** You can consider a Python source file as a sequence of simple and compound statements. Unlike other languages, Python has no declarations or other top-level syntax elements, just statements.

* **Simple statements** - A simple statement is one that contains no other statements. A simple statement lies entirely within a logical line. As in other languages, you may place more than one simple statement on a single logical line, with a semicolon (;) as the separator.
  + **expression** can stand on its own as a simple statement
  + **assignment** is a simple statement that assigns values to variables
* **Compound statements** - A compound statement contains one or more other statements and controls their execution.
  + A compound statement has one or more clauses, aligned at the same indentation.
  + Each clause has a **header** starting with a keyword and ending with a colon (:), followed by a body, which is a sequence of one or more statements.
  + Block - When the body contains multiple statements. These statements should be placed on separate logical lines after the header line, indented four spaces rightward. The block lexically ends when the indentation returns to that of the clause header (or further left from there, to the indentation of some enclosing compound statement). Alternatively, the body can be a single simple statement, following the : on the same logical line as the header.

**context variable**

* A variable which can have different values depending on its context. This is similar to Thread-Local Storage in which each execution thread may have a different value for a variable. However, with context variables, there may be several contexts in one execution thread and the main usage for context variables is to keep track of variables in concurrent asynchronous tasks. See [contextvars](https://docs.python.org/3/library/contextvars.html" \l "module-contextvars" \o "contextvars: Context Variables).
* <https://docs.python.org/3/library/contextvars.html#module-contextvars>

# References:

* <https://www.oreilly.com/library/view/python-in-a/0596100469/ch04s01.html>