= 2 ; y =
2 ; z = 2 result in allocating storage to (at most) three names and one numeric object , to which all three names are bound . Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it . However at a given time a name will be bound to some object , which will have a type ; thus there is dynamic typing .

The if statement , which conditionally executes a block of code , along with else and elif (a contraction of else @-@ if) .

The for statement , which iterates over an iterable object , capturing each element to a local variable for use by the attached block .

The while statement , which executes a block of code as long as its condition is true .

The try statement , which allows exceptions raised in its attached code block to be caught and handled by except clauses ; it also ensures that clean @-@ up code in a finally block will always be run regardless of how the block exits .

The class statement , which executes a block of code and attaches its local namespace to a class , for use in object @-@ oriented programming .

The def statement , which defines a function or method .

The with statement (from Python 2 @.@ 5) , which encloses a code block within a context manager (for example , acquiring a lock before the block of code is run and releasing the lock afterwards , or opening a file and then closing it) , allowing Resource Acquisition Is Initialization (RAII) -like behavior .

The pass statement , which serves as a NOP . It is syntactically needed to create an empty code block .

The assert statement , used during debugging to check for conditions that ought to apply .

The yield statement , which returns a value from a generator function . From Python 2 @.@ 5 , yield is also an operator . This form is used to implement coroutines .

The import statement , which is used to import modules whose functions or variables can be used in the current program .

The print statement was changed to the print ( ) function in Python 3 .

Python does not support tail call optimization or first @-@ class continuations , and , according to Guido van Rossum , it never will . However , better support for coroutine @-@ like functionality is provided in 2 @.@ 5 , by extending Python 's generators . Before 2 @.@ 5 , generators were lazy iterators ; information was passed unidirectionally out of the generator . As of Python 2 @.@ 5 , it is possible to pass information back into a generator function , and as of Python 3 @.@ 3 , the information can be passed through multiple stack levels .

= = = Expressions = = =

Some Python expressions are similar to languages such as C and Java , while some are not :

Addition , subtraction , and multiplication are the same , but the behavior of division differs (see Mathematics for details) . Python also added the * * operator for exponentiation .

As of Python 3 @.@ 5 , it supports matrix multiplication directly with the @ operator , versus C and Java , which implement these as library functions . Earlier versions of Python also used methods instead of an infix operator .