Type	**Example**	**Idea**	**Operators**	**Comparison**
int	`3`	Whole number	`+ - * / // %`	`== != < <= > >=`
float	`2.7`	Number with decimal point	`+ - * /`	`== != < <= > >=`
str	`'hello'`	Sequence of characters	`+`	`== != < <= > >=`
bool	`True`	`True` or `False`	`and or not`	`== != < <= > >=`
list	`[5, 0, 3]`	Sequence of values	`+`	`== != < <= > >=`
dict	`{'a':1, 'b':2}`	Maps keys to values		`== !=`
set	`{2, 3, 5}`	Set without duplicates		`== != < <= > >=`
tuple	`(5, 0, 3)`	Immutable sequence	`+`	`== != < <= > >=`

Expression	**Example**	**Template**
Literal	`3`	*lit*
Unary operator	`-5`	*op expr*
Binary operator	`2 + 3`	*expr op expr*
Variable	`x`	*name*
Function call	`f(x, 5)`	*name*`(`*expr*`,` *expr*`)` zero or more arguments
List index	`ls[3]`	*name*`[`*int*`]`
Method call	`'surprise'.count('s')`	*expr*`.`*name* `(`*expr*`,` *expr*`)`
List comprehension	`[x**2 for x in ls if x > 10]`	`[`*expr* `for` *name* `in` *seq* `if` *bool*`]` if part is optional
List slice	`ls[2:20:3]`	*name*`[`*int*`:`*int*`:`*int*`]` start:stop:step are all optional

Statement	**Example**	**Template**	**Notes**
Assignment	`x = 5`	*name* = *expr*	

If	`if x > 80:`
	`  t = 'hot'`
	`else:`
	`  t = 'not'`
		`if` *bool*`:`
		stmt
		…
		`else:`
		stmt
		…
			else part is optional;
			`if` … `elif` … `else` is also possible

While	`while i < 10:`
	`  print(i)`
	`  i += 1`
		`while` *bool*`:`
		stmt
		…
			`break` and `continue` available inside

For	`for x in ls:`
	`  print(x)`
		`for` *name* `in` ***seq*** `:`
		stmt
		…
			`break` and `continue` available inside

Function definition	`def f(x, y):`
	`  return x + y`
		`def` *name*`(`*name*`,` *name*`):`
		stmt
		…
			zero or more arguments;
			`return` available inside

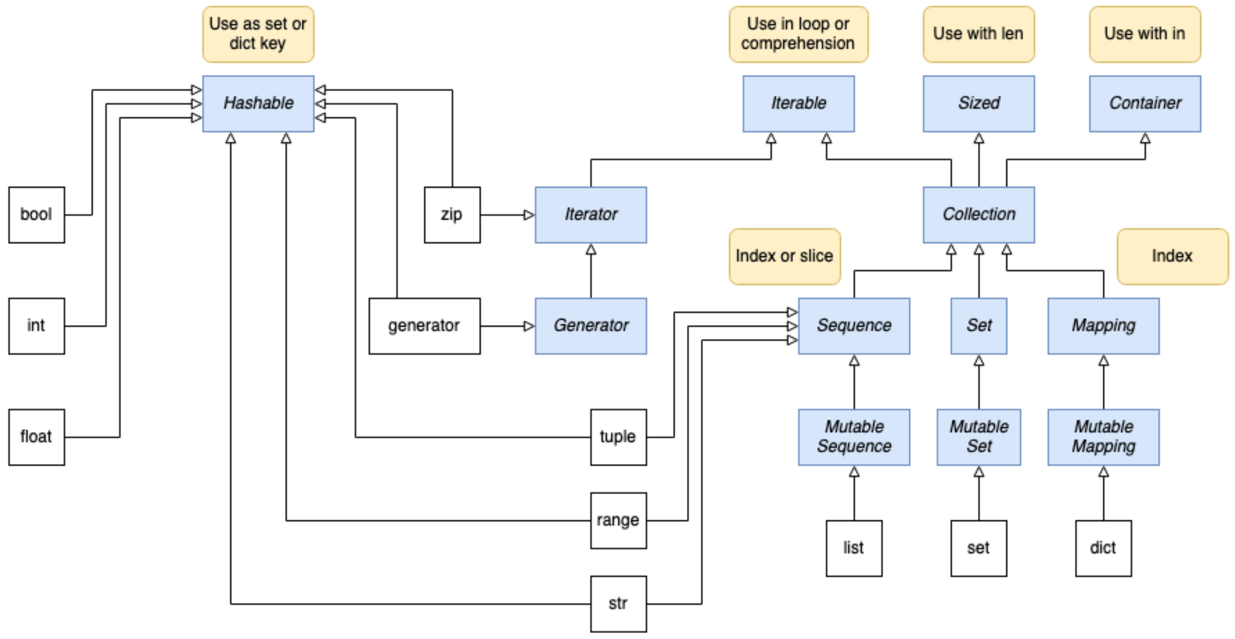
Import	`import mod as m`	`import` *module* `as` *name*	`as` part is optional

Built-In Function	**Notes**
`abs(x)`	Absolute value of `x`
`len(ls)`	Length of `ls`
`max(ls, key=f)`	Largest element in `ls`, using `key` for comparison; `key` is optional
`range(n)`	First `n` nonnegative integers; can take more arguments like a slice
`sorted(ls, key=f)`	Sorted version of `ls`, using `key` for comparison; `key` is optional
`str(x)`	Str version of `x`; analogous functions exist for the other types
`sum(ls)`	Sum of a list
`zip(ls1, ls2)`	Sequence of pairs of corresponding elements from two lists

Built-In Method	**Notes**
`ls.append(x)`	Modify `ls` by adding `x` to end
`ls.count(x)`	Number of times `x` appears in `ls`
`s.join(ls)`	String made from elements of `ls`, with `s` between them
`d.keys()`	Keys of a dictionary; `values` is similar
`s.split()`	List of words in a string

`f'x is {x}'` produces a string where the part between `{}` is evaluated

`x in c` is `True` if `x` is an element of `c` (or, if `c` is a string, a substring of `c`)


Use as set or dict key
Use in loop or comprehension
Use with len
Use with in
Hashable
Iterable
Sized
Container
bool
zip
Iterator
Collection
Index or slice
Index
int
generator
Generator
Sequence
Set
Mapping
float
tuple
Mutable Sequence
Mutable Set
Mutable Mapping
range
list
set
dict
str