Naming

The name of variables in Python can consist of lowercase letters (a to z), uppercase letters (A to Z), numbers (0 to 9), or the underscore character (_). Spaces aren't be allowed in a variable name. Additionally, a variable name must not start with a digit, and it is not recommended to start it with the underscore character except in very specific cases, for example, `if __name__ == " __m ain __"` Furthermore, it is essential to avoid using a built-in word in Python as a variable name (for example: print, range, for, from, etc.).

Type of variables

Integer	`int()`
Float	`float()`
String " " or ' ' or ''' '''	`str()`
List [..., ...]	`list()`
Dictionary {key : value, ...}	`{key:v alue} dict[key] = val ue`
Tuple (..., ...)	`tuple()`
Set {..., ...}	`set()`
Boolean True&False	
Frozenset frozenset({..., ...})	`frozen set()`

- To check the type of variable, type(variable)
- Floats can be in scientific format, like 3e8 = 3*10^8.
- To convert float to scientific format, `" %e"% float`. It will return a string
- To use mathematical constant e, it should import module math. `ma th.e`
- To make a long number visible, using underscores "_" to separate digits in the version 3.6+, like 380_000

Conversion

FLoat & Int	`float() int() round()`
List to Str	`'separ ato r'.j oi n(list)`
Str to List	`list(s tring) string.sp lit ('s epa rato r')`

- `int(float)` returns only the integer part of the float and `round( float, num)` is used to round a number to a specified number of decimal places.
- `'sep'.j oin()` cannot combine lists with full integers. [str(i) for i in list; separator by default is space

Properties & Common Functions

NUM	`int() round( value, decimal) abs()`
STRING	iterable, indexable, immutable; len(); str + str, str * positive int; `str.re pla ce(a,b) str.co unt( a) str.ti tle() str.up per() str.lo wer( ) str.st rip() str.rs trip() str.ls trip ()`
LIST	`list[s tar t:s top :step] enumer ate (l ist) max() min() sum() list.r eve rse() r evers ed( list)` list * int, list + list; `[i for i i n list for _ in range()] list.a ppend() list.i nse rt( ite m,pos) list.r emove( ) list.pop() del list[] list.i nde x(ite m) sorted (list) list.s ort()`
RANGE	`range( start, stop, step)` step could be negative; similar to lists, but immutable
DICT	iterable by key, ordered by key or value : `sorted ( dic) sorted (dic, key=di co.get) dic.it ems() dic.keys() dic.va lues() dic[key]` or `dic.ge t(key) dic[ke y]= value del di c[key] dict.p op(key)`; len()
TUPLE	len(), iterable, ordered, indexiable, immutable. Avoid containing mutable variables
SET	iterable, mutable, unordered, indexable; `set.add( ) set.re move() set.up data()set(list1) & set(list2)` sames; `set(list1) | set(li st2)` union; `set(list1) - set(list2)`
FROZENSET	`f1.uni on(f2) f1.int ers ect ion(f2)`

- If strings or lists are multiplied by a negative integer or a float, it will returns nothing but a null string/list or an error
- To duplicate a list, `list.c opy()` or `list[:]`. It should exactly avoid using `lst2 = lst1`, this creates a reference to the original list with the same ID `id()`
- `list[1:n]` stop at n-1, even if negative index
- `set()` can use to remove duplicated elements in lists and to take keys of a dictionary
- sets cannot be applied operators like + or *

Arithmetic Operators			
x + y	add	x - y	substract
x * y	multiply	x ** y	x^y
x / y	divide	x // y	integet division
x % y	modulus		

Assignment shortcuts: x op= y, for example, x += y is egal to x = x+y

Comparison Operators	
x == y	x != y
x < y	x > y
x <= y	x >= y

By **Theo666**
cheatography.com/theo666/

Published 17th September, 2023.
Last updated 25th September, 2023.