## 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__ == "__main__"` 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:value}` `dict[key] = value` |
| **Tuple** (..., ...) | `tuple()` |
| **Set** {..., ...} | `set()` |
| **Boolean** True&False | |
| **Frozenset** frozenset({..., ...}) | `frozenset()` |

- 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. `math.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 | `'separator'.join(list)` |
| Str to List | `list(string)` `string.split ('separator')` |

- `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'.join()` 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.replace(a,b)` `str.count(a)` `str.title()` `str.upper()` `str.lower()` `str.strip()` `str.rstrip()` `str.lstrip()` |
| **LIST** | `list[start:stop:step]` `enumerate (list)` `max()` `min()` `sum()` `list.reverse()` `reversed( list)` list * int, list + list; `[i for i in list for _ in range()]` `list.append()` `list.insert( item,pos)` `list.remove()` `list.pop()` `del list[]` `list.index(item)` `sorted (list)` `list.sort()` |
| **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=dico.get)` `dic.items()` `dic.keys()` `dic.values()` `dic[key]` or `dic.get(key)` `dic[key]= value` `del dic[key]` `dict.pop(key)`; len() |
| **TUPLE** | len(), iterable, ordered, indexiable, immutable. Avoid containing mutable variables |
| **SET** | iterable, mutable, unordered, indexable; `set.add()` `set.remove()` `set.update()` `set(list1) & set(list2)` sames; `set(list1) \| set(list2)` union; `set(list1) - set(list2)` |
| **FROZENSET** | `f1.union(f2)` `f1.intersection(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.copy()` 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 *

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

## 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 |