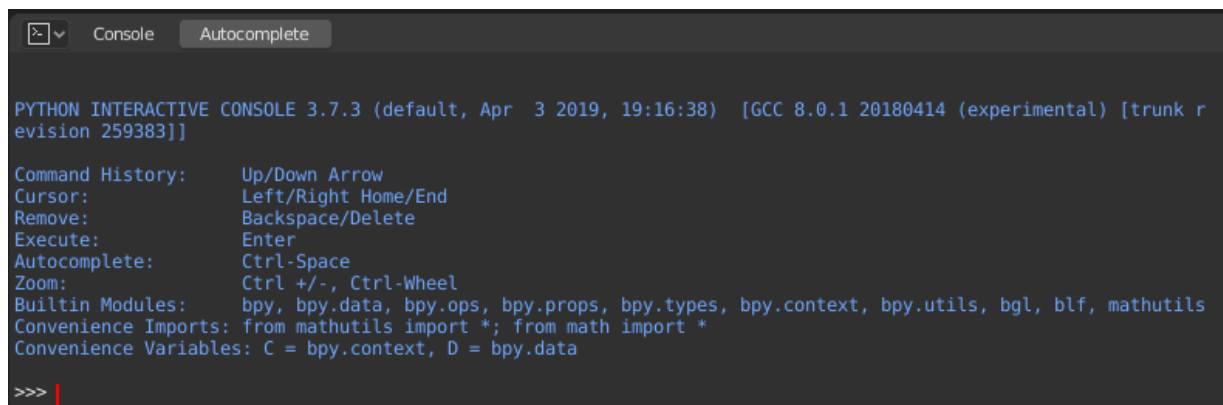


[Skip to content](#)

Python Console

The Python Console offers a quick way to test code snippets and explore Blender's API. It executes whatever you type on its `>>>` prompt and has command history and auto-complete.



```
PYTHON INTERACTIVE CONSOLE 3.7.3 (default, Apr 3 2019, 19:16:38) [GCC 8.0.1 20180414 (experimental) [trunk revision 259383]]

Command History:      Up/Down Arrow
Cursor:                Left/Right Home/End
Remove:               Backspace/Delete
Execute:              Enter
Autocomplete:         Ctrl-Space
Zoom:                 Ctrl +/-, Ctrl-Wheel
Builtin Modules:      bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports:  from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> |
```

Python Console.

Interface

Header Menus

View Menu

Zoom In / Zoom Out

Increases/decreases the font size.

Move to Previous Word **Ctrl - Left**

Moves the cursor to the beginning of the previous word. If the cursor is in the middle of a word, the cursor is moved to the beginning of the current word.

Move to Next Word **Ctrl - Right**

Moves the cursor to the end of the next word. If the cursor is in the middle of a word, the cursor is moved to the end of the current word.

Move to Line Begin **Home**

Moves the cursor to the start of the current line.

Shift - Home : Selects all text between the cursor and the start of the current line.

Move to Line End **End**

Moves the cursor to the end of the current line.

Shift - End : Selects all text between the cursor and the end of the current line.

Console Menu

Clear All

Refreshes the console, giving the view a fresh start. Note that command history is not cleared.

Clear Line **Shift - Return**

Removes everything from the prompt line.

Delete Previous Word **Ctrl - Backspace**

Deletes everything between the cursor and the beginning of the previous word (separated by periods). If the cursor is in the middle of a word, deletes everything to the beginning of the current word.

Delete Next Word **Ctrl - Delete**

Deletes everything between the cursor and the end of the next word (separated by periods). If the cursor is in the middle of a word, deletes everything to the end of the current word.

Deletes everything between the cursor and the end of the next word. If the cursor is in the middle of a word, deletes everything to the end of the current word.

Copy as Script Shift - Ctrl - C

Copies the full history buffer to the clipboard. This can be pasted into a text file to be used as a Python script.

Cut Ctrl - X

Copies the selected text into the clipboard and deletes it.

Copy Ctrl - C

Copies the selected text into the clipboard.

Paste Ctrl - V

Pastes into the command line.

Indent Tab

Inserts a tab character at the cursor.

Unindent Shift - Tab

Unindents the selection.

Backward in History Up

Changes the current command to the previous one from the command history.

Forward in History Down

Changes the current command to the next one from the command history.

Autocomplete Tab

See [Auto Completion](#).

Main View

Key Bindings

- LMB – Moves the cursor along the input line.
- Left / Right – Moves the cursor by one character.
- Ctrl - Left / Ctrl - Right – Moves the cursor by one word.
- Shift - Left / Shift - Right – Selects characters to the left/right.
- Shift - Ctrl - Left / Shift - Ctrl - Right – Selects words to the left/right.
- Ctrl - A Selects all text and text history.
- Backspace / Delete – Erase characters.
- Ctrl - Backspace / Ctrl - Delete – Erase words.
- Return – Execute command.
- Shift - Return – Add to command history without executing.

Usage

Aliases

Some variables and modules are available for convenience:

- C: Quick access to `bpy.context`.
- D: Quick access to `bpy.data`.
- `bpy`: Top level Blender Python API module.

First Look at the Console Environment

To see the list of global functions and variables, type `dir()` and press `Return` to execute it.

```
Console Autocomplete
Execute: Enter
Autocomplete: Ctrl-Space
Zoom: Ctrl +/-, Ctrl-Wheel
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> dir()
['C', 'Color', 'D', 'Euler', 'Matrix', 'Quaternion', 'Vector', '_builtins_', '_doc_', '_loader_', '_name_', '_package_', '_spec_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'bpy', 'bvhtree', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'geometry', 'help', 'hypot', 'inf', 'interpolate', 'isclose', 'isfinite', 'isinf', 'isnan', 'kdtree', 'ldexp', 'lgamma', 'log', 'log10', 'loglp', 'log2', 'modf', 'nan', 'noise', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> |
```

Auto Completion

The Console can preview the available members of a module or variable. As an example, type `bpy.` and press `Tab` :

```
Console Autocomplete
C, _package_, _spec_, acos, acosh, asin, asinh, atan, atan2, atanh, bpy, bvhtree,
'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'geometry', 'help', 'hypot', 'inf', 'interpolate', 'isclose', 'isf
inite', 'isinf', 'isnan', 'kdtree', 'ldexp', 'lgamma', 'log', 'log10', 'loglp', 'log2', 'modf', 'nan', 'noise'
, 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> bpy.
    app
    context
    data
    msgbus
    ops
    path
    props
    types
    utils
>>> bpy.|
```

The submodules are listed in green. Attributes and methods will be listed in the same way, with methods being indicated by a trailing `()`.

Examples

bpy.context

This module gives you access to the current scene, the currently selected objects, the current object mode, and so on.

Note

For the commands below to show the proper output, make sure you have selected object(s) in the 3D Viewport.

```
Console Autocomplete
mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> bpy.context.mode
'OBJECT'

>>> bpy.context.active_object
bpy.data.objects['Cube']

>>> bpy.context.selected_objects
[bpy.data.objects['Cube'], bpy.data.objects['Light'], bpy.data.objects['Camera']]

>>> |
```

Get the current 3D Viewport mode (Object, Edit, Sculpt, etc.):

```
bpy.context.mode
```

Get the active object:

```
bpy.context.object  
bpy.context.active_object
```

Change the active object's X coordinate to 1:

```
bpy.context.object.location.x = 1
```

Move the active object by 0.5 along the X axis:

```
bpy.context.object.location.x += 0.5
```

Change all three location coordinates in one go:

```
bpy.context.object.location = (1, 2, 3)
```

Change only the X and Y coordinates:

```
bpy.context.object.location.xy = (1, 2)
```

Get the selected objects:

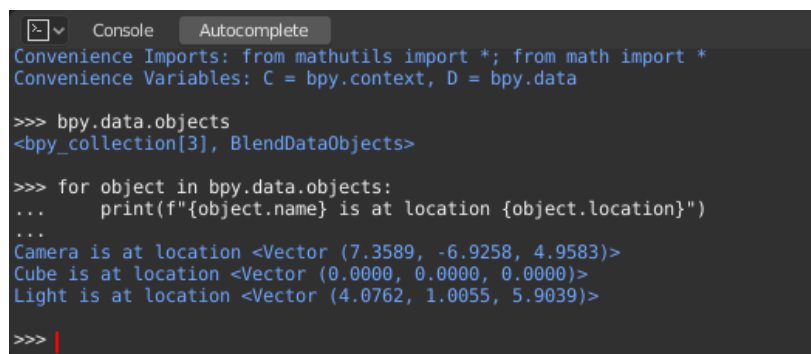
```
bpy.context.selected_objects
```

Get the selected objects excluding the active one:

```
[obj for obj in bpy.context.selected_objects if obj != bpy.context.object]
```

bpy.data

Gives you access to all the data in the blend-file, regardless of whether it's currently active or selected.



```
Console Autocomplete  
Convenience Imports: from mathutils import *; from math import *  
Convenience Variables: C = bpy.context, D = bpy.data  
  
>>> bpy.data.objects  
<bpy_collection[3], BlendDataObjects>  
  
>>> for object in bpy.data.objects:  
...     print(f"{object.name} is at location {object.location}")  
...  
Camera is at location <Vector (7.3589, -6.9258, 4.9583)>  
Cube is at location <Vector (0.0000, 0.0000, 0.0000)>  
Light is at location <Vector (4.0762, 1.0055, 5.9039)>  
  
>>> |
```

bpy.ops

“Operators” are actions that are normally triggered from a button or menu item but can also be called programmatically. See the [bpy.ops](#) API documentation for a list of all operators.

