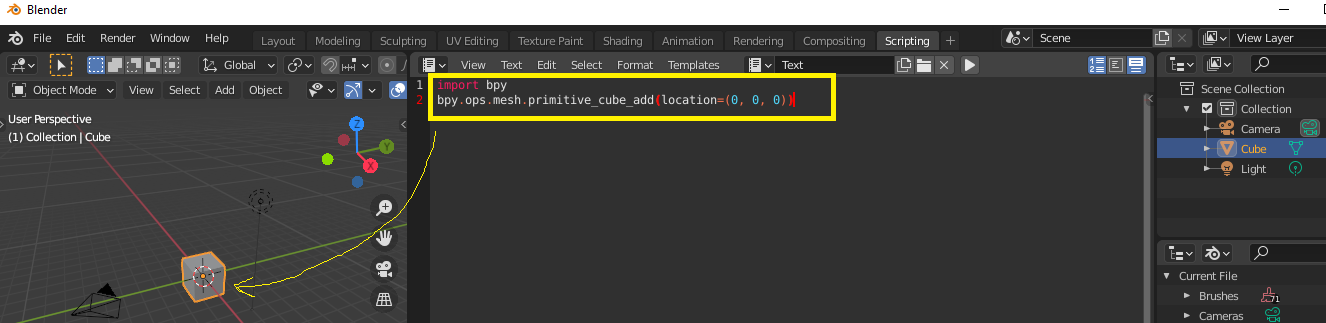
Manual

<https://docs.blender.org/manual/en/dev/>

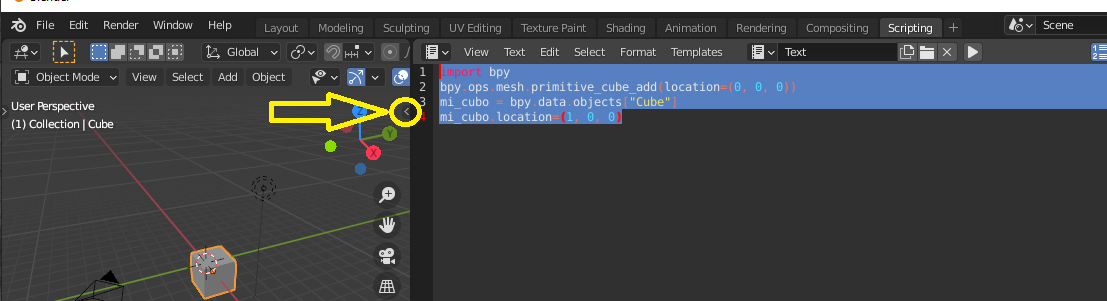
Como primer script vamos a generar un cubo para lo cual borraremos todo objeto de la  
pantalla (los objetos se pueden borrar seleccionándolos con botón izquierdo en la ventana 3D y luego presionando la tecla x) y luego copiaremos el siguiente código en el editor:

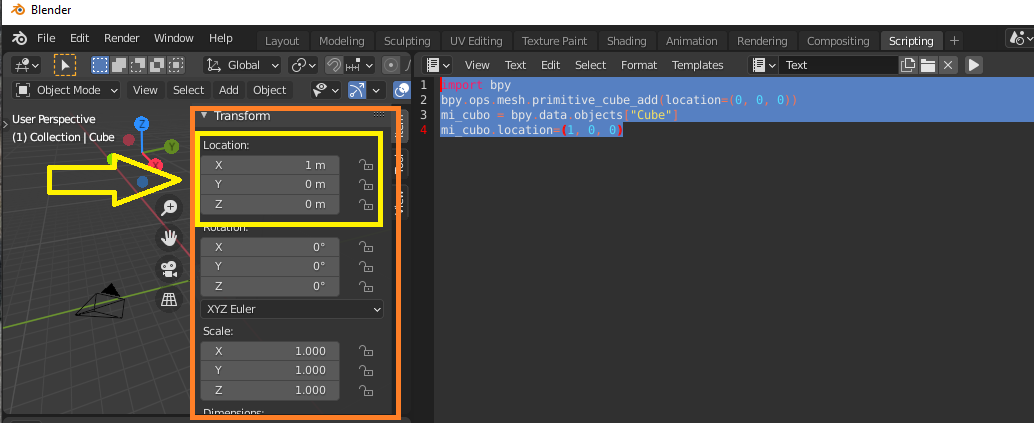
|  |
| --- |
| import bpy  bpy.ops.mesh.primitive\_cube\_add(location=(0, 0, 0)) |



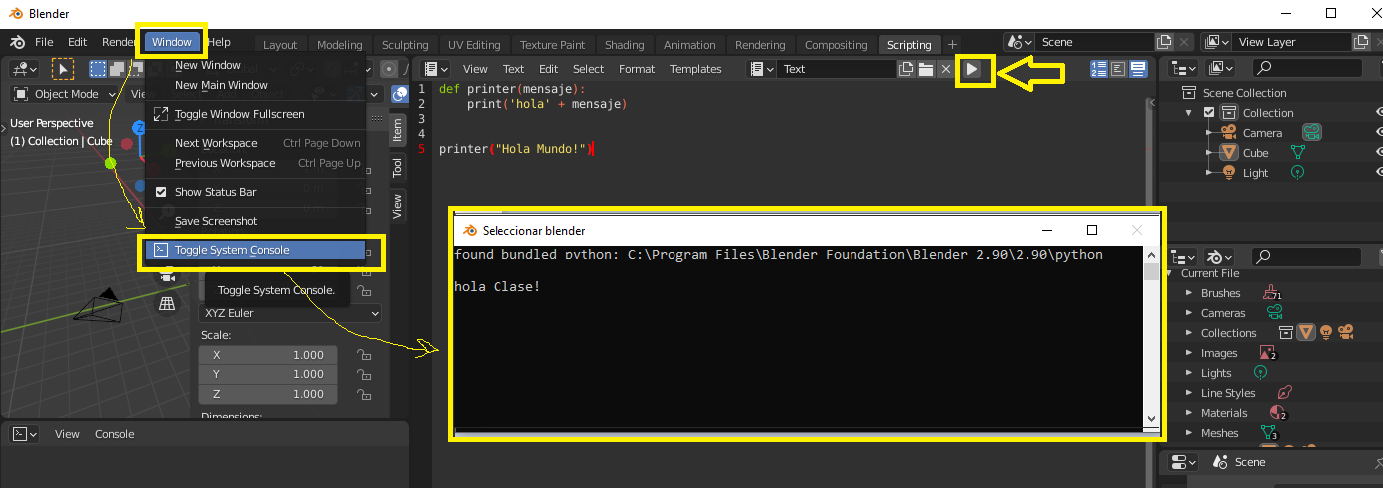
|  |
| --- |
| import bpy bpy.ops.mesh.primitive\_cube\_add(location=(0, 0, 0)) mi\_cubo = bpy.data.objects["Cube"] mi\_cubo.location=(1, 0, 0) |

De seguro en lugar del panel de información del objeto seleccionado estás viendo por  
defecto un signo de “<” lo único que tienes que hacer es presionarlo para que este se  
despliegue:



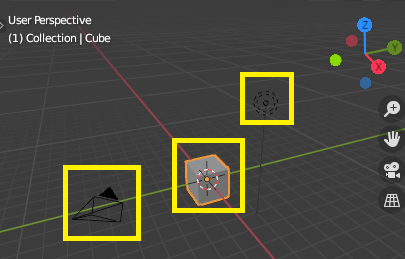


|  |
| --- |
| def printer(mensaje):  print('hola' + mensaje)      printer("Clase!") |



Programación en blender  
5.1. Accediendo a los datos.  
Python accede a los datos de Blender de la misma manera que el sistema de animación y  
la interfaz de usuario; esto implica que cualquier configuración que se pueda cambiar  
mediante un botón también se puede cambiar desde Python.  
El acceso a los datos del archivo blend actualmente cargado se realiza con el módulo  
bpy.data. Esto da acceso a los datos de la biblioteca. Por ejemplo si escribimos en la  
consola “**bpy.data.objects**” blender nos retornará la cantidad de elementos que tenemos  
en la escena, en este caso por defecto existe una cámara, una lámpara y un cubo según  
se muestra a continuación:

|  |
| --- |
| **<bpy\_collection[3], BlendDataObjects>** |



De igual forma podríamos saber la cantidad de escenas o materiales aplicados con las  
instrucciones:

|  |
| --- |
| **>>> bpy.data.scenes >>> bpy.data.materials** |

Notar que para obtener los datos desde un script por ahora podemos hacerlo en el prompt de blender pero para que funcione es necesario importar la librería bpy de la siguiente forma:

|  |
| --- |
| **import bpy print(bpy.data.objects)** |

Accediendo a los datos.  
Podemos acceder a los datos desde una lista o un diccionario, con su posición o su  
nombre.

|  |
| --- |
| **import bpy print(list(bpy.data.objects), "\n---------------------") print(bpy.data.objects['Cube'], "\n---------------------") print(bpy.data.objects[0], "\n---------------------")** |

Dando las siguientes salidas:

**[bpy.data.objects['Camera'], bpy.data.objects['Cube'], bpy.data.objects['Lamp']]  
---------------------  
<bpy\_struct, Object("Cube")>  
---------------------  
<bpy\_struct, Object("Camera")>  
---------------------**

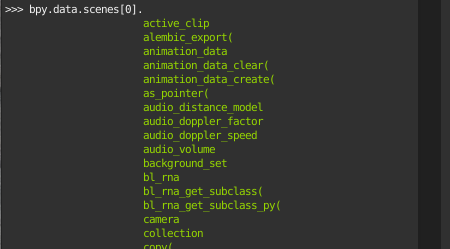
Accediendo a los atributos.  
Una vez que tiene un bloque de datos, como un material, un objeto, grupos, etc., se  
puede acceder a sus atributos de forma similar a como cambiaría una configuración  
mediante la interfaz gráfica. De hecho, la información sobre herramientas para cada botón también muestra el atributo Python que puede ayudar a encontrar qué configuraciones cambiar en un script. Si quisiera por ejemplo acceder al nombre de un elemento podría realizarlo de esta forma:

|  |
| --- |
| **import bpy**  **print(bpy.data.objects[0].name)** |

Autocompletar  
Antes de leer todo el manual es difícil saber cuáles son las palabras reservadas, métodos  
y variables que podemos utilizar en cada caso por lo que podemos utilizar la Consola de  
Blender y el botón de Autocompletar para ir aprendiendo sobre la marcha. Veamos un  
ejemplo, supongamos que queremos saber cuál es la resolución de la escena que viene  
por defecto, por lo que hemos visto en el punto anterior podemos acceder a ella de esta  
forma:

|  |
| --- |
| **bpy.data.scenes[0]** |

Si ahora escribimos esta línea en la consola seguida de un punto (.) y presionamos **la tecla de tabulación**, Blender nos retorna las opciones que podemos seleccionar a  
continuación.



Podría buscar entre las opciones y quedarme con “render” ya que la resolución de un  
render debería de estar relacionada a render (es algo que podríamos intuir, supongo). La  
búsqueda por:

|  |
| --- |
| **bpy.data.scenes[0].render.** |

Nos retorna dentro de las opciones tres que podrían ser las que estamos buscando:



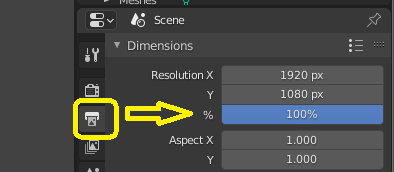
Ahora podríamos escribir:

|  |
| --- |
| **bpy.data.scenes[0].render.resolution\_percentage** |

Y nos retornaría 100



con lo cual si los valores de resolución en x e y son respectivamente  
de 1920px x 1080px nuestra pantalla de renderizado saldría de de ese tamaño. Esto lo  
podríamos chequear en la opción de render del menú de la derecha.



5.5. Accediendo a la base de datos de Blender  
Los datos que se registran en la base de datos de Blender pueden ser accedidos  
mediante:

|  |
| --- |
| **bpy.data** |

Para poder guardar, actualizar, leer o borrar datos de la base de datos debemos utilizar  
métodos específicos de Blender. Por ejemplo podemos utilizar los métodos encontrados  
con autocompletar desde la consola e indicados en verde a continuación para  
estos fines.

|  |
| --- |
| **>>> bpy.data.meshes.** |

**as\_bytes(  
bl\_rna  
data  
find(  
foreach\_get(  
foreach\_set(  
get(  
id\_data  
is\_updated  
items(  
keys(  
new(  
new\_from\_object(  
path\_from\_id(  
remove(  
rna\_type  
tag(  
update(  
values(**

PARED DE BLOQUES

|  |
| --- |
| **import bpy for i in range(0,5):  crear\_cubo = bpy.ops.mesh.primitive\_cube\_add(location=(0, 0, 0))  nombre = "Cubo" + str(i)  bpy.context.object.name = nombre  bpy.data.objects[nombre].location=(i+i, i\*2,0)** |