



## My-Addon Maker Plus

Criador : Edinaldo Cicero / Átomo Games Studio  
Data : 15 de abr. de 2022



## - - Blender MyAddons Maker - -

Aqui você aprenderá a base para se criar o seu próprio addon para o blender 2.79.

Para esse Ebook iremos seguir como uma receita de bolo para que no fim saiba fazer algo tendo em mente os conceitos lógicos de cada etapa, vamos nessa ! ^\_^.

### - Criando o Init de um Addon -

Vamos começar, crie um arquivo ( \_init\_.py ) esse será responsável por inicializar e reproduzir o seu Addon é onde será chamado e registrado os Operadores e outros elementos do seu Addon.

Veja o exemplo :

```

1 bl_info = {
2     "name"           : "Nome do Addon ",
3     "author"         : "Nome do criador do Addon",
4     "version"        : (0, 1),                # Versao do Addon
5     "blender"        : (2, 75, 0),           # Versao do Blender
6     "location"       : "View3D. Tools",      # Localização do addon
7     "description"    : "Meu primeiro Addon", # descrição do addon
8     "warning"        : "",                   # avisos sobre o addon
9     "wiki url"       : "",                  # algum link
10    "category"       : "3D View",            # categoria do addon
11 }
12
13
14 import bpy
15 from . operator_simple import *
16
17
18
19 def register():
20     bpy.utils.register_class(HelloWorldPanel)
21     bpy.utils.register_class(SimpleOperator)
22
23
24 def unregister():
25     bpy.utils.unregister_class(HelloWorldPanel)
26     bpy.utils.unregister_class(SimpleOperator)
27
28
29
30 if __name__ == "__main__":
31     register()
32
33
34

```

Como pode ver eu separei por partes com cores diferentes para poder com mais facilidade explicar cada uma das partes mostradas no script da imagem, lembrando que irei deixar o link para o acesso dos arquivos desse EBook contendo o .blend com esse script que o resultado final é um addon simples que terá o seu layout de painel visual com botões um para adicionar um novo objeto sendo ele um ( Cubo do próprio blender ) e outro logo abaixo para mostrar um print no console.

=====

Laranja : ( bl\_info ) Um dicionário comum do python contendo todas as informações sobre o Addon, deste criador a versão e avisos.

```
1 bl_info = {
2     "name" : "Nome do Addon ",
3     "author" : "Nome do criador do Addon",
4     "version" : (0, 1), # Versao do Addon
5     "blender" : (2, 75, 0), # Versao do Blender
6     "location" : "View3D. Tools", # Localizacao do addon
7     "description" : "Meu primeiro Addon", # descricao do addon
8     "warning" : "", # avisos sobre o addon
9     "wiki_url" : "", # algum link
10    "category" : "3D View", # categoria do addon
11 }
12
13
```

=====

Azul : É onde você deve importar o Bpy biblioteca responsável por elementos pertencentes ao próprio blender como elementos do layouts da sua interface gráfica e entre outras funcionalidades que pode-se encontrar na documentação da API do blender.

```
13
14 import bpy
15 from . operator_simple import *
16
17
18
```

Deves importar desse mesmo jeito que está na imagem depois do ( . ) deve-se colocar o nome do seu outro scripts que contém os Operadores, por fim o ( import \* ) para assim chamar tudo do seu script.

=====

Verde : Aqui será onde irá definir duas funções com os nomes ( register e unregister ) ambas serão responsáveis por fazer a função de registrar o seu addon o fazendo assim reproduzir e terminar quando não usando, é uma maneira que o blender faz para não ficar rodando algo que não está sendo usando no momento evitando assim sobrecarregamento e tals, continuando, ambos os nomes não podem ser mudados assim como alguns nomes de Operadores também, isso será visto mais pra frente no decorrer do Ebook.

```
19 def register():
20     bpy.utils.register_class>HelloWorldPanel)
21     bpy.utils.register_class(SimpleOperator)
22
23
24 def unregister():
25     bpy.utils.unregister_class>HelloWorldPanel)
26     bpy.utils.unregister_class(SimpleOperator)
27
```

Cada Operador ou classe existente no seu script que seja responsável por um Operador ou UiPanel do seu Addon deve ser chamado aqui no ( register ) da seguinte forma :

```
bpy.utils.register_class( aqui fica o nome da sua classe )
```

O mesmo se faz no ( unregister ) :

```
bpy.utils.unregister class( aqui fica o nome da sua classe )
```

O assim terminamos o ( \_init\_.py ) agora iremos para o script do Addon em si, vc deve esta se perguntando o porquê de não ter começado pelo script principal em se, então, fiz assim pois vi que seria mais fácil para começar e o'que você está vendo aqui já é a estrutura final para se criar um addon que poderá ser distribuído para qualquer pessoa e instalado no seu blender correspondente a versão que o Addon foi feito.

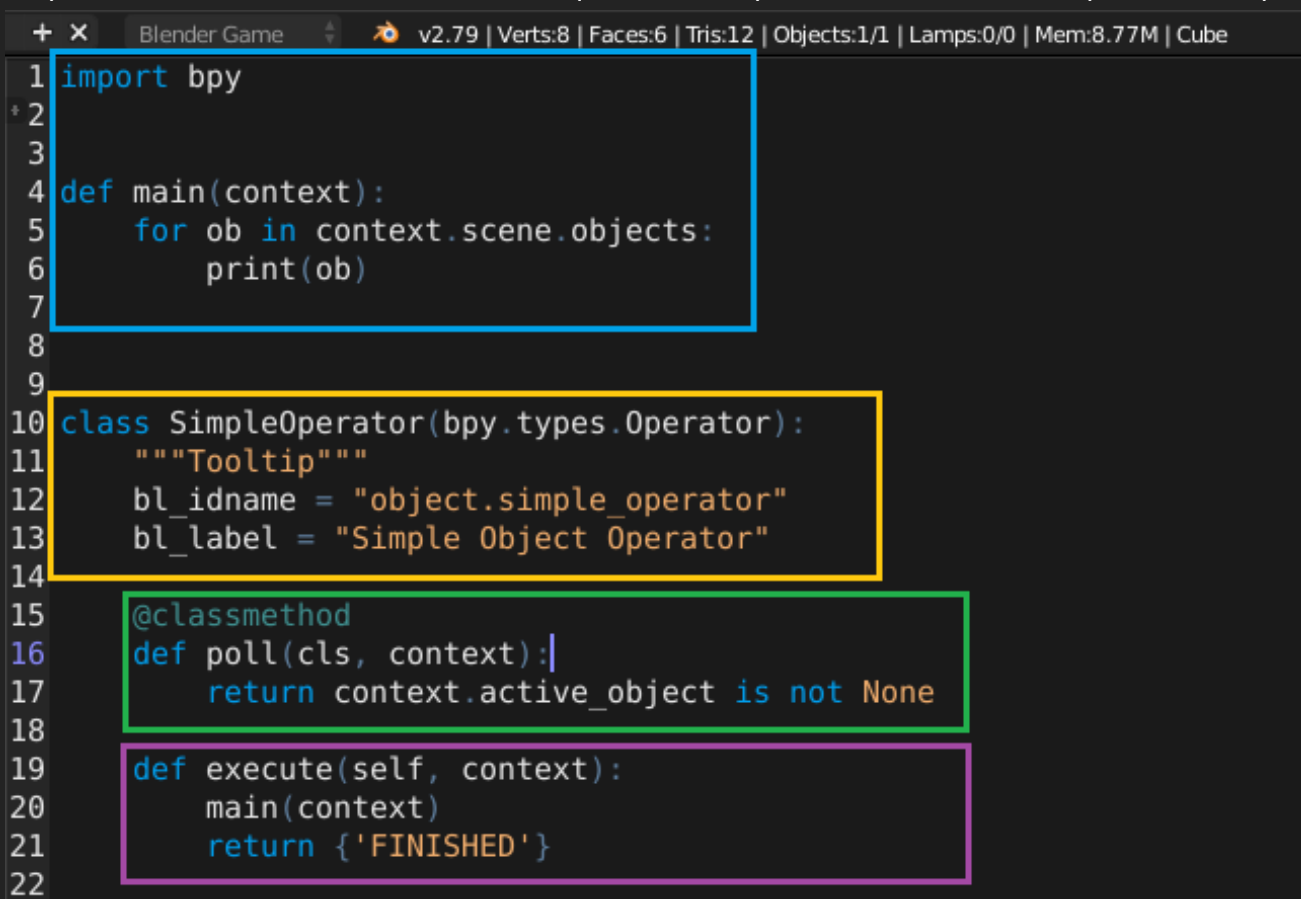
Deixando claro que essa forma de se criar um Addon está sendo para o blender 2.79b, ou seja, versões superiores do blender não há garantia de funcionalidade, certo para você poder fazer um Addon para outra nova versão do blender recomendo seguir alguns dos passo mostrados aqui no Ebook mas seguindo também a documentação do blender que estás a usar e criar um addon, certo.

## - Criando um operador simples. -

Vamos continuar, crie um novo arquivo.py com o mesmo nome que você importa no ( \_init\_.py ) nese caso nós colocamos ( operator\_simple.py ) certo e importamos no ( \_init\_.py ) assim :

```
from . operator_simple import *
```

Depois de isto estar claro vamos para o script em se , vamos começar importando o bpy pois é obrigatório e em seguida logo abaixo vamos criar um função simples, veja a imagem e siga a explicação de cada parte de acordo com a sua cor de destaque, fiz assim para ficar mais fácil de explicar cada tópico.



```
+ x Blender Game v2.79 | Verts:8 | Faces:6 | Tris:12 | Objects:1/1 | Lamps:0/0 | Mem:8.77M | Cube
1 import bpy
2
3
4 def main(context):
5     for ob in context.scene.objects:
6         print(ob)
7
8
9
10 class SimpleOperator(bpy.types.Operator):
11     """Tooltip"""
12     bl_idname = "object.simple_operator"
13     bl_label = "Simple Object Operator"
14
15     @classmethod
16     def poll(cls, context):
17         return context.active_object is not None
18
19     def execute(self, context):
20         main(context)
21         return {'FINISHED'}
22
```

Azul : Importamos o ( Bpy ) e em seguida criamos uma função simples para usarmos futuramente para testar o funcionamento do addon, essa função será responsável em apenas dar um print no console mostrando assim o nome do objeto selecionado atualmente, você pode ver que se usa o ( context.scene.objects ) para acessar os objetos da cena você pode ver mais sobre na documentação da Api do blender, não vou entrar e mais detalhes aqui pois ficaria muita informação que já existe na própria documentação certo.

Amarelo : Aqui é onde começa de fato a criação de um addon, como você pode ver é uma classe com o nome ( SimpleOperator ) que tem como herança o ( bpy.types.Operator ) responsável (como o próprio nome já diz) em operar uma ação quando essa classe for chamada em um “evento”.

Se caso haja dúvida sobre herança, recomendo dá uma revisada em python para assim vim aqui e compreender melhor, certo!

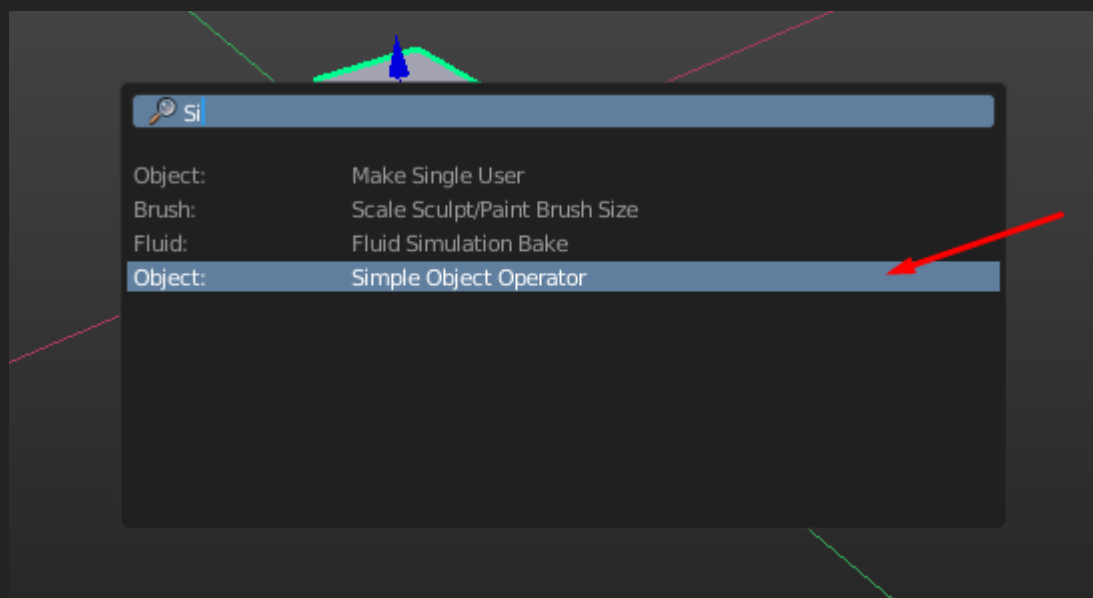
```
10 class SimpleOperator(bpy.types.Operator):  
11     """Tooltip"""  
12     bl_idname = "object.simple_operator"  
13     bl_label = "Simple Object Operator"  
14
```

Como pode ver existe um primeiro texto ( "" Tooltip "" ) serve para deixar informações sobre o Operator, já as variáveis abaixo são responsáveis por algumas ações importantes no operator.

O ( bl\_idname ) é responsável por guardar o nome do de ID do seu operator pois é por esse nome que você irá chamar o seu operator para usá-lo,

obs : as vezes se tem um erro quando é mudado totalmente o nome do ID, deixando uma dica para evitar esse erro, tente sempre deixar assim ( object.algo que vc quer ) não é tudo que se pode escrever que o blender vai aceitar, infelizmente não tenho explicação para isso mas seguindo a dica creio que dê certo.

Já o ( bl\_label ) serve para quando o seu operator não será chamado com o auxílio um botão mais sim pela barra de pesquisa do blender essa mostrada na imagem abaixo:



Será aqui que o seu operator estará disponível, mas para funcionar primeiro se precisa apertar um botão chamado ( RunScript ) que fica logo abaixo na mesma área do editor de texto do blender do lado do nome do seu arquivo.py é através dele que você poderar ir testando o seu addon no decorrer do desenvolvimento, deixando claro que quando você fizer um Run no seu script e obtiver um erro no seu addon ficará mostrando informações sobre esse erro no console dependendo do erro quando você estiver usando o blender o erro permanece a aparecer no console a não ser que você feche e abra o blender novamente ou corrija o erro e rode o script novamente.

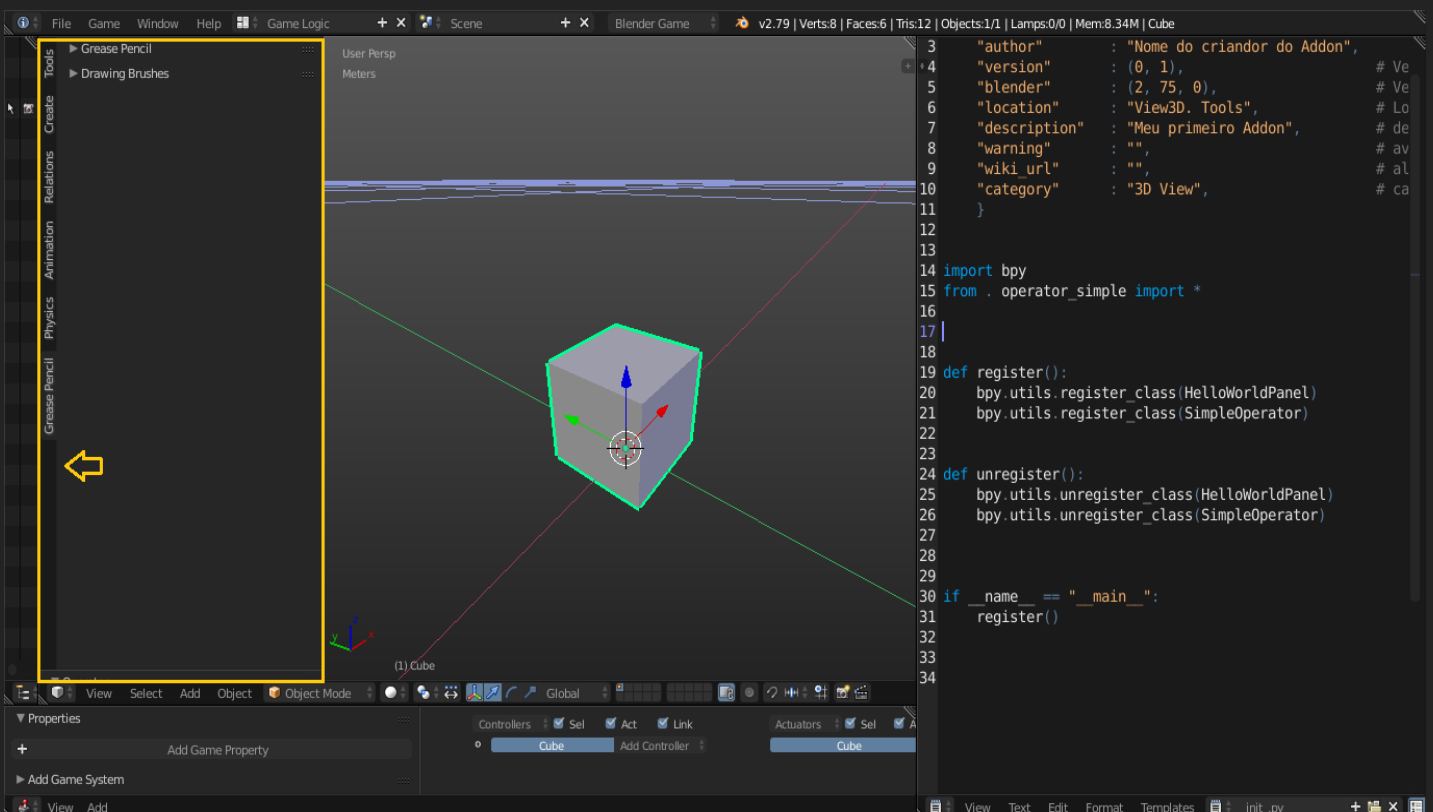
Continuando , dentro ainda dessa classe deve-se ter outras duas funções elas serão responsáveis em executar a função que criamos logo acima no inicio do script:

```
3
4 def main(context):
5     for ob in context.scene.objects:
6         print(ob)
7
8
9
10 class SimpleOperator(bpy.types.Operator):
11     """Tooltip"""
12     bl_idname = "object.simple_operator"
13     bl_label = "Simple Object Operator"
14
15     @classmethod
16     def poll(cls, context):
17         return context.active_object is not None
18
19     def execute(self, context):
20         main(context)
21         return {'FINISHED'}
```

Como podemos ver na imagem, temos as duas funções com os nomes poll( cls , context e execute( self , cont.) antes de poll se encontra um Decorator que não vou entrar em detalhes aqui simplesmente o tenha assim como a função poll, já no caso do ( execute ) como o próprio nome já diz e também conseguimos ver na imagem é onde chamaremos as nossas próprias funções de ações, nesse caso chamamos o main() que é a função responsável por mostrar o nome do objeto selecionado no console, veja também que a função deve-se ter um parâmetro obrigatório que é o ( context ) ele sério com um self de classes python falando de maneira bem rasa sobre ele, nessa mesma função ela também retorna um valor sendo ele ( return { 'FINISHED ' }) mostrando o fim do processo.

## - Criando o addon na interface do blender. -

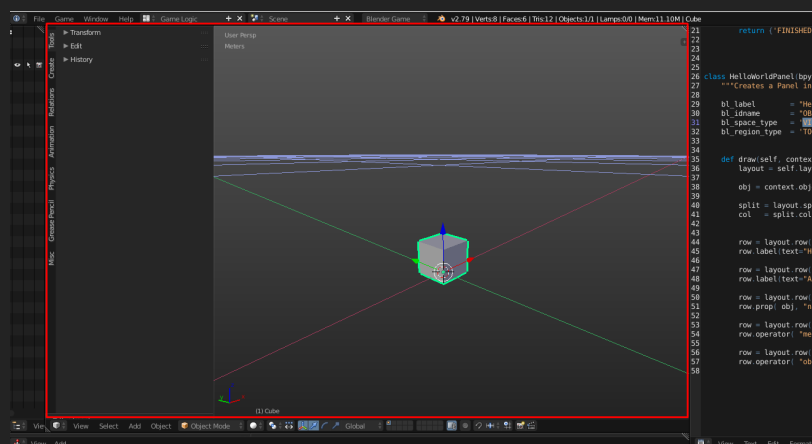
Agora vamos criar os elementos que irão aparecer na interface do blender na região Tools.



Voltando ao script, faremos agora um nova classe no mesmo arquivo do script da classe anterior tudo em um só lugar, vamo vel :

```
25
26 class HelloWorldPanel(bpy.types.Panel):
27     """Creates a Panel in the Object properties VIEW_3D-TOOLS"""
28
29     bl_label      = "Hello World Panel"
30     bl_idname     = "OBJECT_PT_hello"
31     bl_space_type = 'VIEW_3D'
32     bl_region_type = 'TOOLS'
33
34
35     def draw(self, context):
36         layout = self.layout
37
38         obj = context.object
39
40         split = layout.split()
41         col  = split.column( align=True )
42
43
44         row = layout.row()
45         row.label(text="Hello world!", icon='WORLD_DATA')
46
47         row = layout.row()
48         row.label(text="Active object is: " + obj.name)
49
50         row = layout.row()
51         row.prop( obj, "name" )
52
53         row = layout.row()
54         row.operator( "mesh.primitive_cube_add" , icon='OBJECT_DATA')
55
56         row = layout.row()
57         row.operator( "object.simple_operator" , icon='CONSOLE')
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Amarelo : Como podemos ver também é uma classe seguida de herança sendo ela (bpy.types.Panel ) que como o próprio nome diz é responsável pelos elementos dos painéis gráficos do blender que você pode acessar, logo abaixo encontram-se as duas variáveis que já citei acima na classe do operador Simples aqui elas também devem estarem e como pode ver há mais duas variáveis ambas de grande importância pois elas é que serão responsáveis por dizer aonde o addon irá ser desenhado, nesse caso o nosso addon simples será desenhado no ( bl\_space\_type = "VIEW\_3D" ) que seria no espaço correspondente ao espaço 3d veja :





Você pode acessar vários outros espaço que estão disponíveis na documentação da API do blender que mostra o nome de cada espaço disponível , como pode ver na imagem abaixo :

#### bl\_space\_type

The space where the **panel** is going to be used in

- **EMPTY** Empty.
- **VIEW\_3D** 3D View, 3D viewport.
- **TIMELINE** Timeline, Timeline and playback controls.
- **GRAPH\_EDITOR** Graph Editor, Edit drivers and keyframe interpolation.
- **DOPE SHEET\_EDITOR** Dope Sheet, Adjust timing of keyframes.
- **NLA\_EDITOR** NLA Editor, Combine and layer Actions.
- **IMAGE\_EDITOR** UV/Image Editor, View and edit images and UV Maps.
- **CLIP\_EDITOR** Movie Clip Editor, Motion tracking tools.
- **SEQUENCE\_EDITOR** Video Sequence Editor, Video editing tools.
- **NODE\_EDITOR** Node Editor, Editor for node-based shading and compositing tools.
- **TEXT\_EDITOR** Text Editor, Edit scripts and in-file documentation.
- **LOGIC\_EDITOR** Logic Editor, Game logic editing.
- **PROPERTIES** Properties, Edit properties of active object and related data-blocks.
- **OUTLINER** Outliner, Overview of scene graph and all available data-blocks.
- **USER\_PREFERENCES** User Preferences, Edit persistent configuration settings.
- **INFO** Info, Main menu bar and list of error messages (drag down to expand and display).
- **FILE\_BROWSER** File Browser, Browse for files and assets.
- **CONSOLE** Python Console, Interactive programmatic console for advanced editing and script development.

**Type:** enum in ['EMPTY', 'VIEW\_3D', 'TIMELINE', 'GRAPH\_EDITOR', 'DOPE SHEET\_EDITOR', 'NLA\_EDITOR', 'IMAGE\_EDITOR', 'CLIP\_EDITOR', 'SEQUENCE\_EDITOR', 'NODE\_EDITOR', 'TEXT\_EDITOR', 'LOGIC\_EDITOR', 'PROPERTIES', 'OUTLINER', 'USER\_PREFERENCES', 'INFO', 'FILE\_BROWSER', 'CONSOLE'], default 'EMPTY'

Link : [Aperte aqui para acessar o link da documentação da Api referente a esse tópico.](#)

Seguindo também temos a ( **bl\_region\_type** = 'TOOLS' ) responsável pela região onde o addon será desenhado também está na documentam as seguintes regiões acessíveis do blender :

#### bl\_region\_type

The region where the **panel** is going to be used in

**Type:** enum in ['WINDOW', 'HEADER', 'CHANNELS', 'TEMPORARY', 'UI', 'TOOLS', 'TOOL\_PROPS', 'PREVIEW'], default 'WINDOW'

se encontra no mesmo link deixando acima como também outras opções que você pode adicionar ao seu Addon assim como também pode encontrar um script de exemplo muito útil.

Seguindo no script, temos uma função de grande importância que não pode faltar e que a mesma é chamada ( **draw( self , context )** ) e será dentro dela onde teremos os elementos de Ui para serem desenhados para isso teremos que ter uma variável chamada ( **layout = self.layout** ) se você já sabe python, você já deve saber que pode ser qualquer nome de variável ali o que importa aqui é o valor certo.

```
34
35 def draw(self, context):
36     layout = self.layout
37
38     obj = context.object
39
40     split = layout.split()
41     col = split.column( align=True )
42
43
```

Em seguidas temos o ( **obj = context.object** ) que serve para pegamos informações do objeto atualmente ativo/selecionado no mundo 3D, seguindo temos o ( **split = layout.split()** , **col = split.column( align=True )** ) ambos importantes também para a organização dos elementos do seu Addon, esses são responsáveis por colocar os elementos em coluna ao contrário do ( **row = layout.row()** ) que é responsável pela organização em linhas, a forma de usar ambos é muito simples :



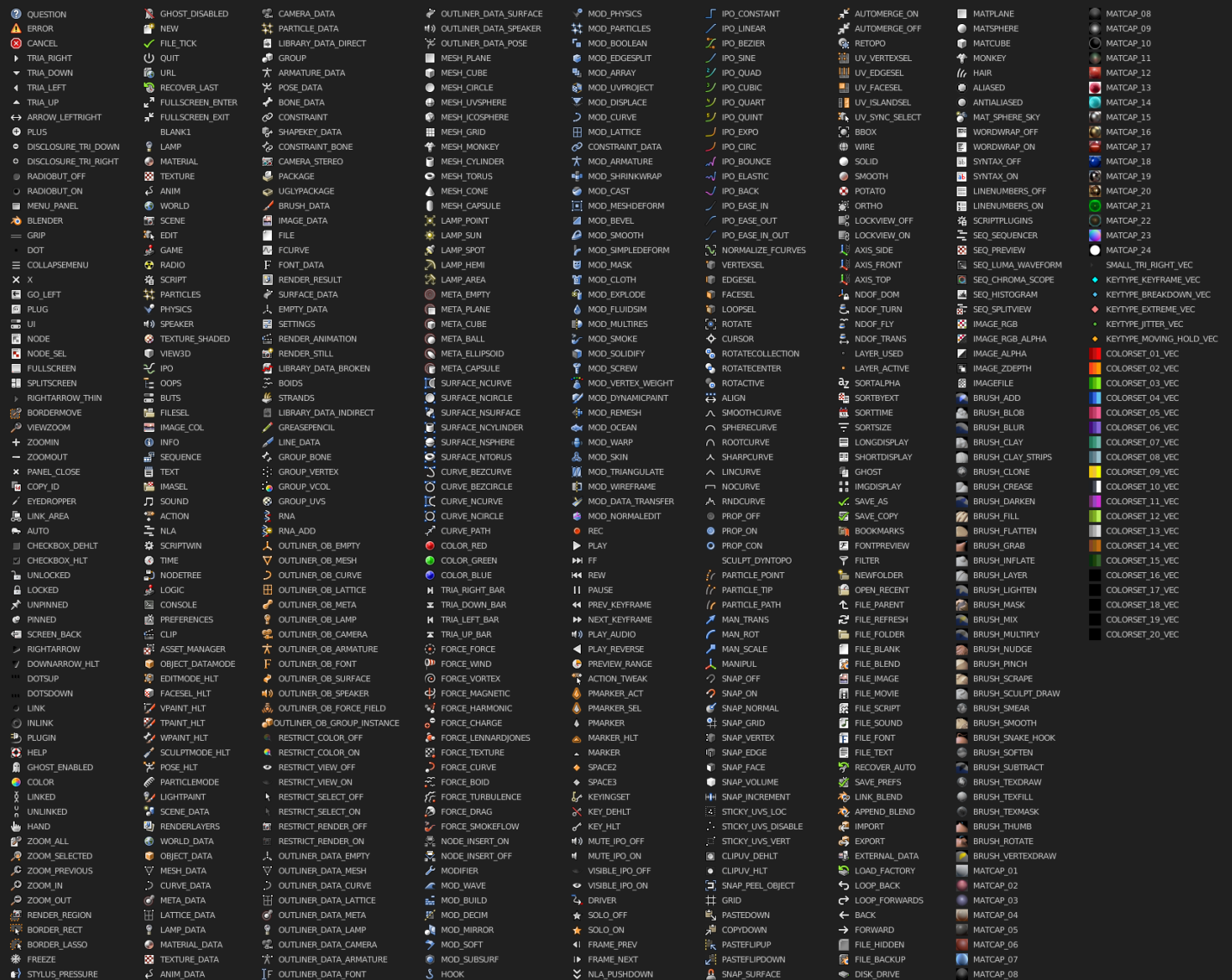
```
row = layout.row()
row.label(text="Hello world!", icon='WORLD_DATA')

row = layout.row()
row.label(text="Active object is: " + obj.name)
```

Como pode ver na imagem se faz ( row ou col ponto elemento grafico que vc quer desenhar ) e a já aproveitando aqui a imagem nela já estamos usando um dos elementos possíveis de ser usar, aqui estar o ( row.label( text="Hello world!" , icon='WORLD\_DATA' ) ) que como pode ver é seguido de alguns parâmetros sendo eles text que é onde você vai escrever o texto que queres que apareça e logo a frente o ( icon='WORLD\_DATA' ) que é onde você poderá definir um entres vários ícones internos disponíveis do blender, aqui está uma imagem contendo todos os nomes de ícones do blender 2.79 que você pode acessar no seu Addon.

Clique na imagem para acessar o link da imagem com mais qualidade ..

# Blender Icons 2.79



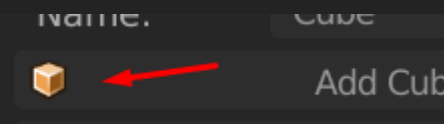
Agora vamos ver alguns dos outros elementos para se desenhar e usar, sendo eles um Input simples e um Botão funcional, então vamos lá , estamos quase terminando de criar o nosso primeiro Addon simples.

Vendo a imagem abaixo para se criar um input e um botão ambos necessariamente não se chamam Input e Button como normalmente você veria por aí em bibliotecas de UI, infelizmente não sei como explicar isso, pois, também não tenho tanta experiência na criação de Addon mesmo assim decide escrever esse Ebook para outros de devs também saberem do pouco que sei e até ampliar esse conhecimento mais além até por que isso logicamente é possível, certo !

```
49  
50 row = layout.row()  
51 row.prop( obj, "name" )  
52  
53 row = layout.row()  
54 row.operator( "mesh.primitive_cube_add" , icon='OBJECT_DATA' )  
55  
56 row = layout.row()  
57 row.operator( "object.simple_operator" , icon='CONSOLE' )  
58
```

Sendo ( `prop()` ) o input ele também receber parâmetros que aos mesmo não irei adentrar em mais detalhes mas fique tranquilo o que ele faz aqui é simplesmente verificar o objeto que está selecionado e com o ( `"name"` ) ele pega o seu nome como texto default sendo assim possível você renomear o nome do objeto que está atualmente selecionado.

Continuando temos o ( `row.operator( "mesh.primitive_cube_add" , icon='OBJECT_DATA' )` ) que é nada mais e nada menos que um Botão para se apertar, como pode ver os operator também requerem parâmetros sendo eles `operator( id_name do operador , icon = que já vimos anteriormente para que serve )` como você pode ver o primeiro operator está com assim ( `row.operator( "mesh.primitive_cube_add" )` ) esse `id_name = "mesh.primitive_cube_add"` é um operator interior do blender, ou seja, é uma classe interior do blender que é uma Class Operator essa é responsável por adicionar um Cubo nativo do blender simples assim por conta disso também adicionei um ícone chamado ( `"OBJECT_DATA"` ) que se você olhar na lista de ícones verá que se trata de um cubinho amarelo.

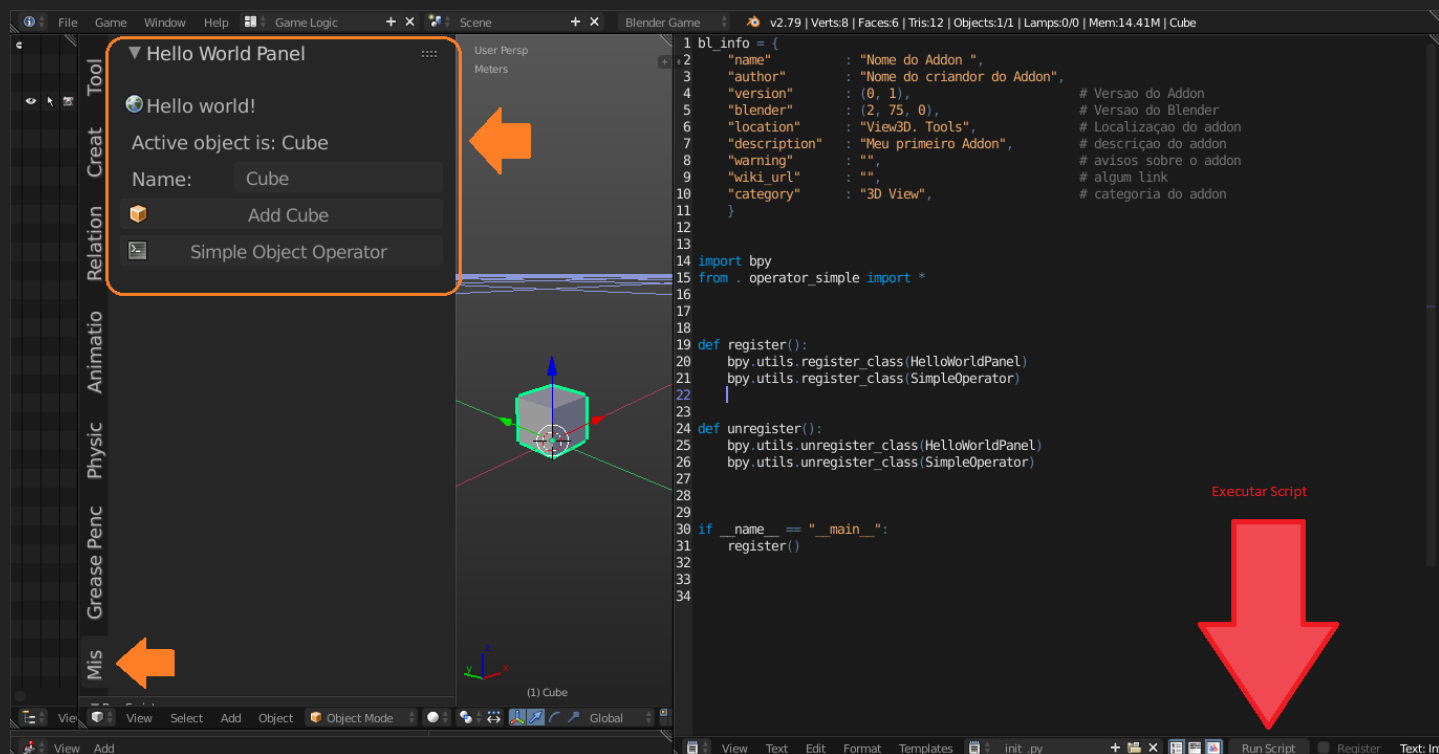


Agora já deve saber como criar um botão que executa uma ação que você quer, sim o `id_name` da sua ClassOperator deve ser colocado ali é o que faço no operator de baixo, nele eu chamo a minha classe `OperatorSimples` a partir da sua `id_name` também :

```
10 class SimpleOperator(bpy.types.Operator):  
11     """Tooltip"""  
12     bl_idname = "object.simple_operator"  
13     bl_label = "Simple Object Operator"  
14  
15     @classmethod  
16     def poll(cls, context):  
17         return context.active_object is not None  
18  
19     def execute(self, context):  
20         main(context)  
21         return {'FINISHED'}  
22
```

Como pode ver se chama ( `bl_idname = "object.simple_operator"` ) e é esse mesmo `bl_id_name` que chamo no meu botão que é o segundo operator logo abaixo, o texto do seu botão deve ser definido no `bl_label` da sua

ClasseOperator que será o nome do seu operador em questão, agora você já tem um addon simples programado o que falta agora é você voltar para o script `_init_.py` e apertar no botão ( RunScript ) como esta na imagem abaixo que se você seguiu todos os passos até aqui você terá um Addon simples parecido com esse aqui no canto esquerdo da imagem :



Legal mas agora você deve estar se perguntando como faço desses scripts um Addon que possa ser enviado para qualquer pessoa que use o blender 2.79 e vou te dizer é muito simples, de verdade você só precisa ter o winrar instalado no seu pc, crie uma pasta com o nome do seu addon e salve esses dois scripts do seu addon dentro desta pasta e em seguida compacte essa pasta no formato .zip do winrar, e voa lá, está pronto o seu próprio addon !

Espero que eu tenha conseguido passar o conhecimento de maneira adequada ao seu entendimento e que consiga fazer addons ainda mais incríveis, lembrando que o que você viu aqui não foi a formula para ser criar um addon super complexo e útil mais assim a base para se criar um simples mas que servirá como guia para addon mais avançados assim você continue a estudar lendo a documentação certo, deixando isso claro quero agradecer por você ter chegado até aqui agora tendo um pouco do conhecimento adquirido no decorrer deste E-Book feito de dev iniciante para devs iniciantes , muito obrigado e espero muito conseguido ter ajudado na sua busca por mais um novo conhecimento .

