




# **Começando a Programar com o Python no QGIS**

**André Silva**

**GEOeduc**



# **CAPÍTULO 3:**

## **Manipulação de Dados Vetoriais**

Em atividades no QGIS, trabalhar com dados tabulares e vetoriais são de extrema importância. Por exemplo ao manipular dados de um arquivo vetorial de municípios do país é importante ter informações de tabela (tabulares) ou seja informações alfanuméricas sobre os atributos ou descrições de cada município, como: Código do IBGE, informações de área, população, informações geográficas e entre outras informações que os analistas ou desenvolvedores julgarem necessário. Após a construção dos dados vetoriais e suas informações descritivas, geralmente é necessário realizar outras operações de análise, que frequentemente culminam na seleção, alteração ou personalização destes dados vetoriais. Neste presente capítulo iremos abordar as atividades citadas na descrição acima, usando o console do Python dentro do QGIS.

Objetivos do capítulo:

- Verificar informações de tabela
- Selecionar Feições
- Realizar edição de Feições
- Trabalhar com Simbologia de Camadas



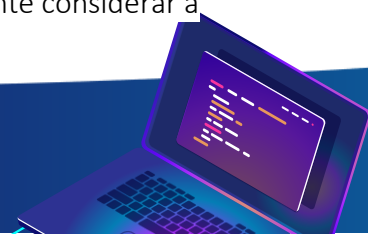
### Usando classes e funções Python no QGIS

Toda ferramenta de geoprocessamento disponível para QGIS possui um conjunto específico de parâmetros que são necessários para executar a ferramenta em um script Python. Esse grupo de parâmetros é chamado de sintaxe da ferramenta de geoprocessamento - o conjunto de regras que define como um programa Python será gravado e interpretado.

Os parâmetros têm certas propriedades importantes:

- um nome único
- Um valor de entrada ou saída
- Necessário ou opcional

Nem toda ferramenta Python tem o mesmo número de parâmetros de entrada, e a ordem dos parâmetros pode variar de ferramenta para ferramenta. É importante considerar a



sintaxe de cada ferramenta antes de usá-la em seu script. Antes de começar a escrever o código, é uma boa prática consultar a documentação de ajuda da ferramenta de geoprocessamento para considerar sua sintaxe.

Na ajuda da Web de todas as ferramentas de geoprocessamento, a seção Sintaxe indica os diferentes parâmetros necessários (e geralmente alguns opcionais). Os parâmetros necessários são necessários para executar a ferramenta de geoprocessamento e os parâmetros opcionais fornecem funcionalidade adicional ou opções de saída.

Exemplo:

```
Processing.run('qgis:dissolve', input, dissolve_all, field, output)
```

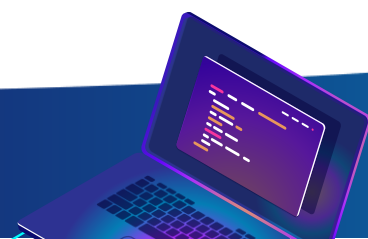
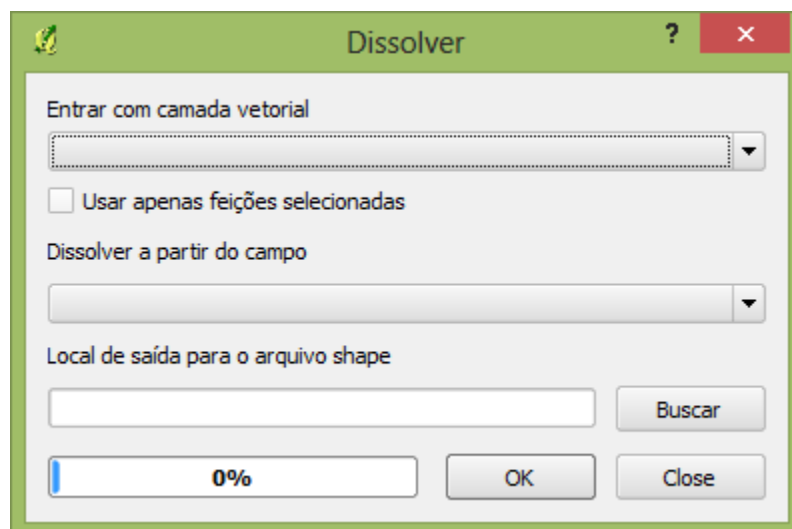
**Input:** Caminho do layer, ponto, linha ou polígono.

**Dissolve\_all:** Dissolver baseando em algum campo

**Field (opcional):** Caso seja dissolvido usando campo, informe o campo

**Output:** Local e nome da saída do arquivo final

Veja abaixo a similaridade com a interface gráfica:





## Diferentes caminhos para usar variáveis em Python

Uma das vantagens de criar um script Python é a capacidade de criar processos complexos dentro de um script que usam várias ferramentas de geoprocessamento diferentes. Dentro do script, diferentes entradas e saídas podem ser reutilizadas através de variáveis. Uma variável é simplesmente um nome que representa um valor. Para criar uma variável, você atribui um valor ao nome.

As variáveis diferenciam maiúsculas de minúsculas e são capazes de conter diferentes tipos de dados. Com esses conjuntos de dados dinâmicos, o tipo de dados é determinado quando a variável é referenciada.

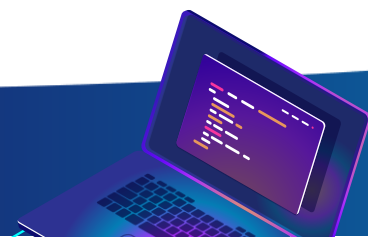
Você pode usar variáveis nas ferramentas de geoprocessamento para substituir nomes de caminhos completos dos conjuntos de dados, consumir a saída de outra ferramenta de geoprocessamento no script ou substituir um parâmetro que possa ser compartilhado entre as ferramentas de geoprocessamento.

Exemplos:

```
GPTool1(r"c:\ProjectData.gdb\points", r"c:\ProjectData.gdb\OutputPts")  
inputP = r"c:\ProjectData.gdb\points"  
outP = r"c:\ProjectData.gdb\OutputPts"  
  
GPTool1(inputP, outP)
```

Opção 2:

```
inputP = r"c:\ProjectData.gdb\points"  
outP = r"c:\ProjectData.gdb\OutputPts"  
outP2 = r"c:\ProjectData.gdb\OutputPoly"  
  
GPTool1(inputP, outP)  
GPTool2(outP, outP2)
```



Opção 3:

```
inputP = r"c:\ProjectData.gdb\points"  
outP = r"c:\ProjectData.gdb\OutputPts"  
outP2 = r"c:\ProjectData.gdb\OutputPoly"  
mPara1 = 10
```

```
GPTool1(inputP, outP, mPara1)  
GPTool2(outP, outP2, mPara1)
```



### EXERCÍCIO 1: INFORMAÇÕES TABULARES E SELEÇÃO DE FEIÇÕES

Vamos colocar a mão na massa, vamos aprender como ler informações de feições que estão em nosso campo de camadas, sem a necessidade de usarmos o caminho usual, ou seja sem fazer o uso de ícones presentes no próprio QGIS Desktop.

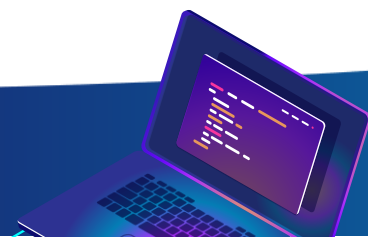


#### Passo 1 – Seleção Simples

- 1) Em um novo projeto, adicione o dado vetorial **geração energia** (Usinas e centrais elétricas do Brasil). O arquivo vetorial se encontra na pasta Pratica01.
- 2) Abra o Terminal Python no QGIS. Na guia **Complementos > Terminal Python**.
- 3) Crie uma variável chamada *layer* com o valor *iface.activeLayer()*, para que possamos ativar as camadas.

Terminal Python


```
1 Python Console  
2 Use iface to access QGIS API interface or Type help(iface) for more info  
3 Security warning: typing commands from an untrusted source can lead to data loss and/or leak  
4 >>> layer = iface.activeLayer()  
5
```



- 4) Agora vamos extrair os atributos da feição em questão, mas antes vamos fazer a leitura dele em memória. Insira um laço for, como segue abaixo:

```
for f in layer.getFeatures():
```

- 5) Para concluir a estrutura de repetição insira: `print(f)`, mas antes clique em TAB uma única vez para respeitar a indentação.

- 6) Clique em *Executar Comando*  no Terminal Python para visualizar o resultado abaixo:

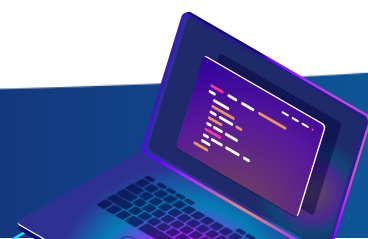
```
Terminal Python
42 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
43 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
44 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
45 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
46 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
47 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
48 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
49 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
50 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
51 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
52 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
53 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
54 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
55 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
56 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
57 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
58 <qgis._core.QgsFeature object at 0x000001E6E2FB60D8>
59 <qgis._core.QgsFeature object at 0x000001E6E2FB6B88>
```



## Passo 2 – Verificando número de feições

Vamos verificar a quantidade de feições existentes na camada que estamos trabalhando

- 1) Insira uma variável `num_feicoes` com o valor `layer.featureCount()` em seguida pressione *Enter*.
- 2) Por fim insira um comando `print(num_feicoes)`, seguido de *Enter* para visualizarmos o número de feições da camada.





### Passo 3 – Acessando atributos de uma camada

Com a linguagem Python muitas rotinas podem ser criadas dentro do QGIS, bem como em outros softwares de geotecnologias. Vamos verificar mais abaixo como é fácil também trabalhar com o acesso a informações das tabelas de uma camada.

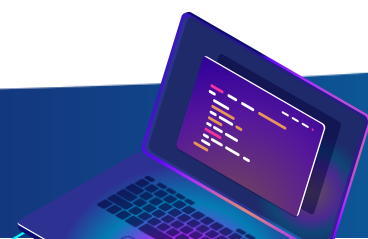
- 1) No terminal Python, abra o Editor, clicando no ícone similar ao da imagem abaixo:



- 2) Insira manualmente a camada IDH\_2018.shp
- 3) Na nova janela do Editor insira uma variável chamada *fn*, e adicione a ela o valor `"C:/<local do seus arquivos>Pratica01/IDH_2018.shp"`, que é o caminho aonde está o arquivo IDH\_2018.shp.
- 4) Em seguida crie uma variável chamada *layer* e atribua o valor `QgsVectorLayer(fn, "", 'ogr')`. **Obs:** Não use aspas duplas em nenhum destes parâmetros acima, apenas aspas simples.
- 5) Crie uma variável chamada *fc*, com a seguinte função como valor atribuído: `layer.featureCount()`.

Agora vamos criar um loop for para interar através de todos os campos da feature e acessar os dados por referência aos nomes dos campos informados.

- 6) Insira na linha abaixo o seguinte código: `for i in range(0, fc):`





- 7) Complete o loop inserindo uma variável chamada *feat* e atribua a ela a função *layer.getFeature(i)*. **Obs.:** Não esqueça da indentação.
- 8) Por fim para retornar os valores dos campos desejados, insira o código *print(feat[0], feat[1])*
- 9) Você pode também imprimir os valores dos campos baseado na posição dos nomes deles em uma lista. Em outras palavras se os campos da sua camada possuem os seguintes nomes em sequência: ID, CODIGO, MUNICIPIO, AREA, em uma lista os índices serão [0,1,2,3]. Faça um teste, execute o código finalizado no tópico anterior e altere o nome dos campos pelos seus respectivos índices. Exemplo: *print(feat[2], feat[3])*.

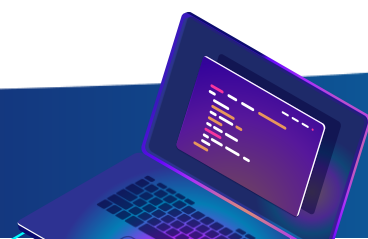


#### Passo 4 – Seleção de camadas

Agora vamos inserir um arquivo shapefile diretamente do diretório de origem para o QGIS Desktop, realizando seleções automaticamente.

Para começar a atividade abaixo, remova o layer adicionado anteriormente e limpe os códigos do editor.

- 1) insira uma variável chamada *fn*, e adicione a ela o valor "*C:/<local do seus arquivos>Pratica01/IDH\_2018.shp*", que é o caminho aonde está o arquivo *IDH\_2018.shp*.
- 2) Agora vamos carregar o arquivo dentro do QGIS, insira dentro do Editor o seguinte código: *layer = iface.addVectorLayer(fn, '', 'ogr')*

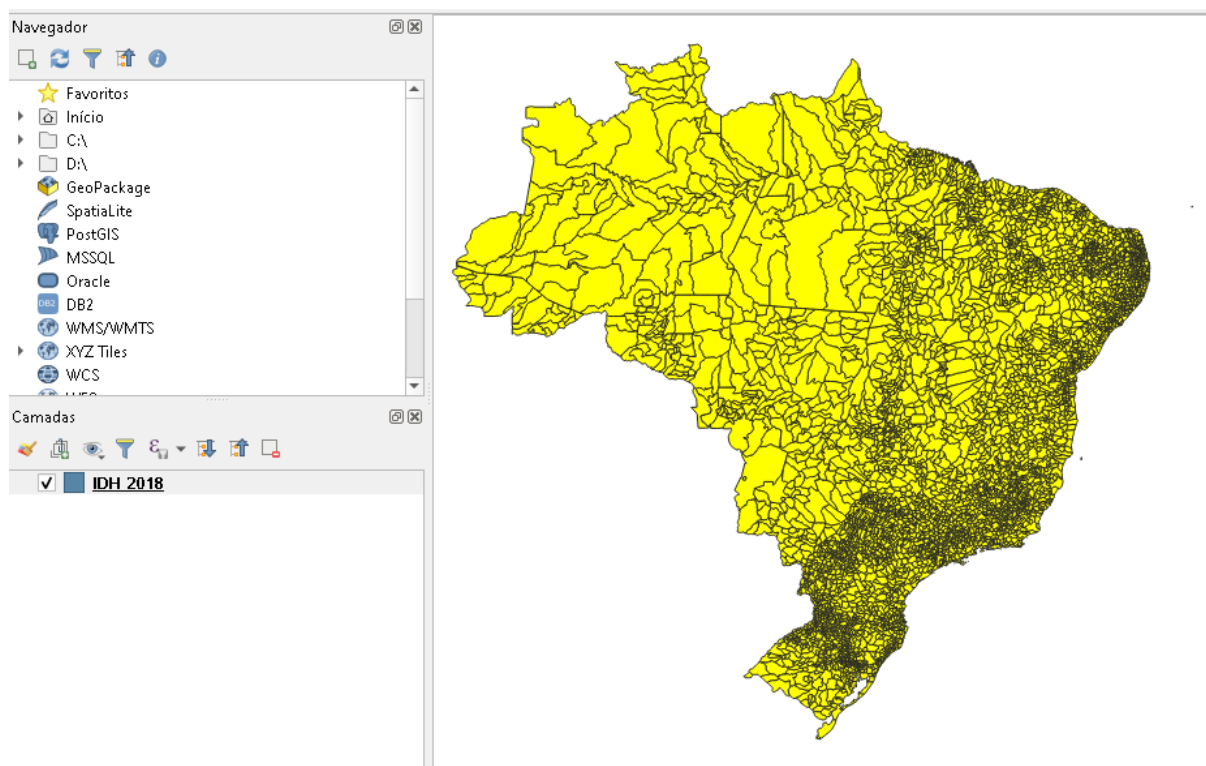


3) Por fim insira o código `layer.selectAll()`, para importar a camada com todas as features já selecionadas.

4) Execute o script clicando conforme a imagem abaixo:



5) A situação abaixo deve ocorrer:



### Passo 5 – Seleção por ID

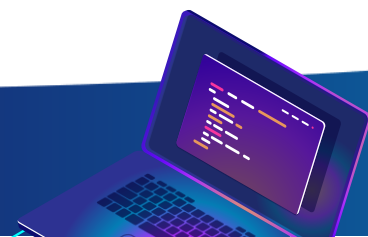
Selecionar por ID, seleciona um recurso com base no ID do recurso, em outras palavras do FID.

O FID é um identificador exclusivo para cada feature em uma camada. Geralmente, o FID



corresponde à ordem em que os recursos foram criados. Por exemplo, o primeiro recurso criado para uma camada teria um FID de 1.

- 1) Remova novamente a camada do QGIS, e comente com um # a linha aonde se encontra o código *layer.selectAll()*.
- 2) Crie uma variável chamada *selectid* e nela atribua uma lista com os seguintes valores 1, 3, 6, 8, 11.
- 3) Por fim insira o código *layer.select(selectid)*, e execute o código.
- 4) A imagem abaixo deve ser mostrada, informando quais feições foram criadas respeitando a ordem dos valores da lista.



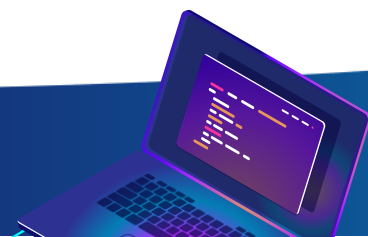


## Passo 6 – Seleção por valores de atributos

Outra maneira comum de selecionar features é referenciando valores de atributos para features específicas.

Os recursos podem ser selecionados com base nos valores de atributo com `layer.selectByExpression()`. Devemos passar uma expressão SQL definindo os atributos que queremos selecionar.

- 1) Comente as duas linhas inseridas no passo anterior.
- 2) Insira o código `layer.selectByExpression('"CODIGO_UF"=35')`.
- 3) Uma nova camada é criada e a imagem abaixo deverá ser retornada.





## Passo 6 – Criação de features

Primeiramente remova os dados presentes no canvas do QGIS Desktop. Durante estes exercícios vamos usar classes concernentes a criação de geometrias, inicialmente vamos começar criando dois pontos, depois criaremos uma linha para ligá-los e por fim, criaremos um polígono envolvente para estes pontos.

- 1) Limpe o terminal python e o console.
- 2) Crie uma variável chamada *layer* e adicione como valor a *QgsVectorLayer* adicionando as informações de criação de pontos em seus parâmetros, como a seguir:

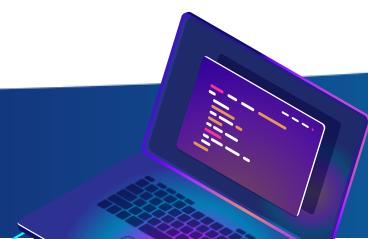
```
layer = QgsVectorLayer('Point', 'points' , "memory")  
pr = layer.dataProvider()
```

- 3) Adicione o primeiro ponto, com o seguinte trecho abaixo:

```
pt = QgsFeature()  
  
point1 = QgsPointXY(50,50)  
  
pt.setGeometry(QgsGeometry.fromPointXY(point1))  
  
pr.addFeatures([pt])
```

- 4) Atualize a extensão da camada, com o trecho abaixo:

```
layer.updateExtents()
```



5) Adicione o segundo ponto:

```
pt = QgsFeature()
point2 = QgsPointXY(100,150)
pt.setGeometry(QgsGeometry.fromPointXY(point2))
pr.addFeatures([pt])
```

6) Atualize novamente a extensão da camada, com o trecho abaixo:

```
layer.updateExtents()
```

7) Adicione o novo layer ao canvas.

```
QgsProject.instance().addMapLayers([layer])
```

8) Execute o código e verifique no canvas a criação dos dois pontos.

**Obs:** Caso apareça o pção para selecionar o sistema de coordenadas, clique em WGS84 e OK.



### Passo 7 – Criação de linha.

Iremos agora criar um feature do tipo linha, comente ou limpe os códigos anteriores

1) Crie uma variável chamada *layer* e adicione como valor a classe *QgsVectorLayer* passando como parâmetro as informações de criação de pontos, como a seguir:



```
layer = QgsVectorLayer('LineString', 'line' , "memory")
```

```
pr = layer.dataProvider()
```

- 2) Adicione o segmento de linha ligando os dois pontos, conforme o código abaixo:

```
line = QgsFeature()
```

```
line.setGeometry(QgsGeometry.fromPolylineXY([point1,point2]))
```

```
pr.addFeatures([line])
```

- 3) Atualize a extensão da camada, com o trecho abaixo:

```
layer.updateExtents()
```

- 4) Adicione a linha ao canvas:

```
QgsProject.instance().addMapLayers([layer])
```

- 5) Execute o código e verifique os resultados no QGIS Desktop.

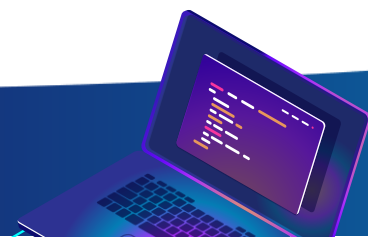
**Obs:** Caso apareça o pção para selecionar o sistema de coordenadas, clique em WGS84 e OK.



## Passo 6 – Criação de polígonos

Agora em diante iremos criar o polígono envolvente para nossas features previamente criadas, comente ou remova os códigos anteriores.

- 1) Crie uma variável chamada *layer* e adicione como valor a classe *QgsVectorLayer* adicionando as informações de criação de pontos em seus parâmetros, como a seguir:



```
layer = QgsVectorLayer('Polygon', 'poly', "memory")
```

```
pr = layer.dataProvider()
```

- 2) Crie os 4 pontos para criação do polígono, conforme o código abaixo:

```
poly = QgsFeature()
```

```
points = [point1,QgsPointXY(50,150),point2,QgsPointXY(100,50)]
```

- 3) Ligue os pontos em torno dos 4 pontos criados anteriormente:

```
poly.setGeometry(QgsGeometry.fromPolygonXY([points]))
```

```
pr.addFeatures([poly])
```

- 4) Atualize a extensão da camada, com o trecho abaixo:

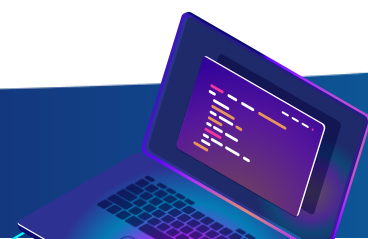
```
layer.updateExtents()
```

- 5) Adicione o polígono criado em memória para o canvas do QGIS.

- 6) Execute o código e verifique se o polígono foi criado.

```
QgsProject.instance().addMapLayers([layer])
```

**Obs:** Caso apareça o pção para selecionar o sistema de coordenadas, clique em WGS84 e OK.







Parabéns! Chegamos ao final de mais capítulo, no qual você aprendeu sobre Manipulação de dados vetoriais usando scripts no PYQGIS.

