Objetivos:

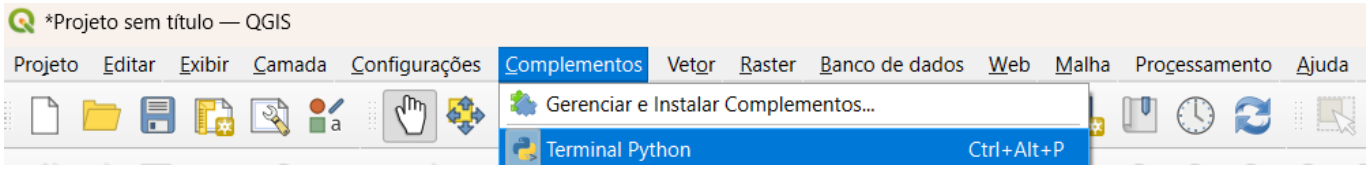
- Apresentar o ambiente de programação Python no QGIS;
- Compreender a estrutura da biblioteca de código Python do QGIS;
- Construir as principais geometrias utilizando as classes do pacote core.

Observação: Para a realização dos exercícios é necessário utilizar o ambiente de programação Python embutido no QGIS, pois algumas classes estão disponíveis exclusivamente neste ambiente.

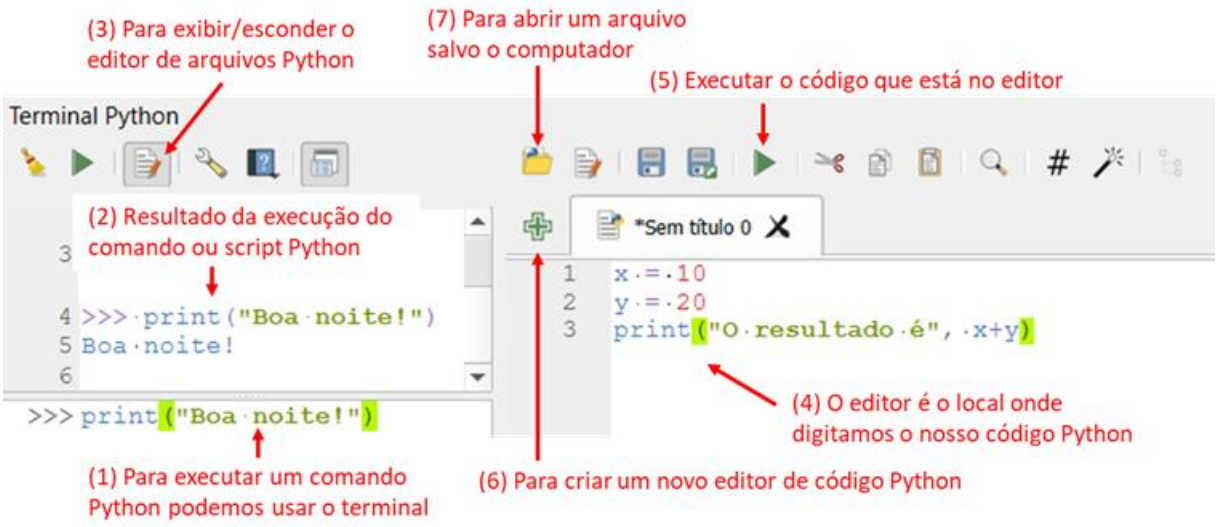
I. Ambiente de programação Python no QGIS

Siga os passos a seguir para acessar o ambiente de desenvolvimento em Python no QGIS:

a) Acesse o menu **Complementos > Terminal Python** para abrir o console interativo Python do QGIS;



b) Para escrever programas mais complexos, recomenda-se utilizar o **Editor de Scripts Python**, também disponível no menu **Complementos**. Esse editor permite criar e salvar scripts em arquivos .py, que podem ser reutilizados posteriormente.



II. Estrutura da biblioteca de código Python do QGIS

A API Python do QGIS (PyQGIS) acompanha a versão do QGIS instalada. Se você estiver utilizando o QGIS 3.40, consulte a respectiva documentação em https://qgis.org/pyqgis/3.40.

A biblioteca Python do QGIS é composta por diversos pacotes organizados sob o pacote principal `qgis`, sendo os principais:

- `core`: funcionalidades essenciais, incluindo classes de geometria;

- `gui`: elementos gráficos e interfaces do QGIS;
- `analysis`: ferramentas de análise espacial;
- `server`: suporte ao QGIS Server;
- `processing`: ferramentas do framework de geoprocessamento;
- `_3d`: suporte a visualizações tridimensionais.

Cada um desses pacotes contém **módulos**, que por sua vez contêm **classes**. Por exemplo, a classe `QgsPointXY` (https://qgis.org/pyqgis/3.40/core/QgsPointXY.html), usada para representar um ponto, está localizada em `qgis.core`.

Exemplo de uso da classe QgsPointXY

A classe `QgsPointXY` permite criar objetos que representam pontos geográficos com coordenadas X e Y.

Para criar um objeto dessa classe, utiliza-se um de seus **construtores** (método especial `__init__`). Os principais formatos são:

- QgsPointXY(x: float, y: float) – cria um ponto a partir de coordenadas;
- QgsPointXY(p: QgsPointXY) – cria um novo ponto a partir de outro;
- Outros formatos aceitam objetos `QPointF`, `QPoint` ou `QgsPoint`.

Exemplo: cria um objeto da classe `QgsPointXY`

```
jcr = QgsPointXY(-45.96642, -23.30555)  # Coordenadas de Jacareí
```

Métodos da classe QgsPointXY

Métodos são funcionalidades associadas ao objeto. Para utilizá-los, basta chamá-los a partir do objeto criado. A seguir, destacamos alguns métodos relevantes:

Método	Descrição
`asWkt()`	Retorna o ponto no formato WKT (Well-Known Text)
`distance(x, y)`	Calcula a distância para outro ponto (coordenadas x, y)
`x()` / `y()`	Retorna os valores das coordenadas
`toString()`	Retorna uma string formatada com as coordenadas
`isEmpty()`	Verifica se a geometria está vazia
`set(x, y)`	Define os valores das coordenadas

Exemplo de uso dos métodos

Código:

```
# Coordenadas de Jacareí
jcr = QgsPointXY(-45.96642, -23.30555)
# Distância até Santa Branca
```

```
print("Distância:", jcr.distance(-45.88441, -23.39755) )
print("Coordenadas:", jcr.x(), jcr.y() )
print("WKT:", jcr.asWkt() )
print("String:", jcr.toString() )
```

Resultado:

```
Distância: 0.12324625795536044
Coordenadas: -45.96642 -23.30555
WKT: POINT(-45.96641999999999939 -23.30555000000000021)
String: -45.96642, -23.30555
```

Construindo geometrias com o pacote core

A seguir, são apresentados exemplos de construção das principais geometrias espaciais utilizando o pacote `qgis.core`:

- Ponto: o construtor `QgsPointXY(x,y)` cria um objeto com as coordenadas de um ponto.

```
jcr = QgsPointXY(-45.96642, -23.30555)
print( jcr.toString() )
```

- Retângulo: o construtor `QgsRectangle(xMin, yMin, xMax, yMax)` recebe as coordenadas dos pontos inferior esquerdo e superior direito.

```
retangulo = QgsRectangle(-46.03542, -23.41555, -45.88441, -23.30555)
print("Perímetro:", retangulo.perimeter() ) # perímetro do retângulo
```

- LineString: na prática uma linha é formada por uma sequência de pontos. Temos duas formas de construir uma linha:
  a) Classe `QgsLineString` (https://qgis.org/pyqgis/3.40/core/QgsLineString.html): o construtor `QgsLineString([])` recebe uma lista onde cada elemento é um objeto do tipo `QgsPoint` (não é QgsPointXY).

```
jcr = QgsPoint(-45.96642, -23.30555)
stb = QgsPoint(-45.88441, -23.39755)
gua = QgsPoint(-46.03542, -23.41555)

linha = QgsLineString([jcr,stb,gua])
print( linha.length() ) # comprimento da linha
```

  b) Classe QgsGeometry (https://qgis.org/pyqgis/3.40/core/QgsGeometry.html): o construtor `QgsGeometry()` não recebe parâmetros. Essa classe possui métodos para construir os principais tipos de geometria. O método `fromPolylineXY([])` recebe como parâmetro uma lista, onde cada elemento dessa lista é um objeto do tipo `QgsPointXY` e retorna um objeto `QgsGeometry` contendo uma linha.

```
jcr = QgsPoint(-45.96642, -23.30555)
```

```
stb = QgsPoint(-45.88441, -23.39755)
gua = QgsPoint(-46.03542, -23.41555)

linha = QgsLineString([jcr,stb,gua])
print( linha.length() ) # extensão da linha
```

- MultiPoint: para construir uma geometria múltipla de pontos temos de usar o método `fromMultiPointXY([])`. Ele recebe uma lista de objetos `QgsPointXY` e retorna uma geometria `QgsGeometry` contendo os pontos de uma geometria múltipla.

```
jcr = QgsPointXY(-45.96642, -23.30555)
stb = QgsPointXY(-45.88441, -23.39755)
gua = QgsPointXY(-46.03542, -23.41555)

geom = QgsGeometry()
pontos = geom.fromMultiPointXY([jcr,stb,gua])
print( pontos.asWkt() )
```

- Polígono: para construir um polígono é composto por uma lista de anéis, onde o primeiro é o anel exterior e os demais, anéis interiores (buracos). O método `fromPolygonXY([])`, da classe `QgsGeometry`, recebe uma lista com os pontos que compõem o polígono. Como um polígono é formado por 1 anel exterior e vários anéis interiores (buracos). Então o parâmetro passado para o método `fromPolygonXY` precisa estar no formato [[anel exterior],[anéis interiores]]. Cada anel é formado por uma lista de objetos do tipo `QgsPointXY`.

```
jcr = QgsPointXY(-45.96642, -23.30555)
stb = QgsPointXY(-45.88441, -23.39755)
gua = QgsPointXY(-46.03542, -23.41555)

# estamos passando apenas o anel exterior
poligono = QgsGeometry().fromPolygonXY([[jcr,stb,gua]])
print( poligono.asWkt() )
```

Para mais detalhes acesse o capítulo Geometry Handling do PyQGIS Cookbook (https://docs.qgis.org/3.40/en/docs/pyqgis_developer_cookbook/geometry.html)

Exercícios

Exercício 1: Construa dois objetos representando as coordenadas geográficas das cidades de Jacareí e Santa Branca. Em seguida, imprima na tela a distância entre elas.

Coordenadas:

- Jacareí: -45.96642, -23.30555
- Santa Branca: -45.88441, -23.39755

Dicas:

- Utilize o construtor `QgsPointXY` para criar um ponto para cada cidade;
- Utilize o método `distance`, da classe `QgsPointXY`, para calcular a distância entre os pontos.

Resposta:

```
Distância: 0.12324625795536044
```

Exercício 2: Construa uma geometria do tipo `LineString` utilizando os pontos correspondentes às cidades de Jacareí, Santa Branca e Guararema. Em seguida, obtenha o centroide da geometria e imprima suas coordenadas na tela.

Coordenadas:

- Jacareí: -45.96642, -23.30555
- Santa Branca: -45.88441, -23.39755
- Guararema: -46.03542, -23.41555

Dicas:

- Utilize o construtor `QgsPointXY` para criar um ponto para cada cidade;
- Utilize o construtor `QgsGeometry` para criar uma geometria;
- Utilize o método `fromPolylineXY`, da classe `QgsGeometry`, para gerar a geometria;
- Utilize o método `centroid`, da classe `QgsGeometry`, para obter a geometria do centroide;
- Utilize o método `asPoint`, da classe `QgsGeometry`, para converter o centroide em um objeto do tipo `QgsPointXY`;
- Utilize o método `toString`, da classe `QgsPointXY`, para exibir as coordenadas em formato textual.

Resposta:

```
Centroide: -45.9444714623, -23.3819298674
```

Exercício 3: Altere o Exercício 2 para que seja exibida no console a extensão da linha criada.

Dica:

- Utilize o método `length` da classe `QgsGeometry`.

Resposta:

```
Extensão: 0.2753252509162551
```

Exercício 4: Altere o Exercício 2 para obter o REM (Retângulo Envolvente Mínimo – *Bounding Box*) da linha criada e imprima na tela as coordenadas do canto inferior esquerdo e do canto superior direito do retângulo.

Dicas:

- Utilize o método `boundingBox`, da classe `QgsGeometry`, para obter o REM como um objeto da classe `QgsRectangle`;
- Utilize o método `toString`, da classe `QgsRectangle`, para exibir as coordenadas do retângulo.

Resposta:

```
-46.0354200000000020,-23.4155499999999996 :
-45.8844100000000026,-23.3055500000000002
```

Exercício 5: Altere o Exercício 4 para que seja exibida na tela a área do REM (Retângulo Envolvente Mínimo – *Bounding Box*).

Dica:

- Utilize o método `area` da classe `QgsRectangle`.

Resposta:

```
Área: 0.01661109999999985
```

Exercício 6: Altere o Exercício 4 para que seja exibido na tela o perímetro do REM (Retângulo Envolvente Mínimo – *Bounding Box*).

Dica:

- Use o método `perimeter` da classe `QgsRectangle`.

Resposta:

```
Perímetro: 0.5220199999999977
```

Exercício 7: Crie dois retângulos: o primeiro com os pontos A e B, e o segundo com os pontos C e D. Em seguida, obtenha a interseção entre os dois retângulos e imprima o resultado na tela.

Coordenadas:

- A: (0,0)
- B: (2,2)
- C: (1,1)
- D: (3,3)

Dicas:

- Utilize o construtor `QgsPointXY` para criar os pontos;
- Utilize o construtor `QgsRectangle` para croar os retângulos;
- Utilize o método `intersect`, da classe `QgsRectangle`, para obter a geometria de interseção;
- Utilize o método `toString`, da classe `QgsRectangle`, para exibir as coordenadas.

Resposta:

```
Interseção: 1.0000000000000000,1.0000000000000000 : 2.0000000000000000,2.0000000000000000
```

Exercício 8: Crie um polígono utilizando as coordenadas dos pontos A, B, C e D. Em seguida, imprima o polígono no formato WKT.

Coordenadas:

- A: (0,0)
- B: (3,0)
- C: (3,3)
- D: (0,3)

Dicas:

- Utilize o construtor `QgsPointXY` para criar os pontos;
- Utilize o construtor `QgsGeometry` para criar a geometria;
- Utilize o método `fromPolygonXY`, da classe `QgsGeometry`, para gerar a geometria do polígono;
- Utilize o método `asWkt`, da classe `QgsGeometry`, para obter a representação textual em formato WKT.

Resposta:

```
Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))
```

Exercício 9: Adicione um anel interno ao polígono do Exercício 8 utilizando as seguintes coordenadas.

Coordenadas:

- E: (1,1)
- F: (2,1)
- G: (2,2)
- H: (1,2)

Dica:

- Adicione uma segunda lista como parâmetro do método `fromPolygonXY`.

Resposta:

```
Polygon ((0 0, 3 0, 3 3, 0 3, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))
```

Exercício 10: Altere o Exercício 9 para imprimir na tela a área do polígono resultante.

Dica:

- Utilize o método `area` da classe `QgsGeometry`.

Resposta:

```
Área: 8.0
```