1 Contributing to the plugin

You're welcome to contribute to this plugin! Feel free to add documentation for any tool, whether it's a QGIS geoprocessing tool or a custom tool like geopandas functions. Below is a guide on how to add tool documentation to your local computer and to the GitHub repository, helping to enhance the performance of the agent.

```
/*
notes:杨小兵-2024-12-17

1、十分欢迎对这个插件做出贡献，对任何工具添加文档都是可以的，不论是QGIS地理处理工具还是自定义工具例如geopandas函数。下列内容是一个如何将工具添加到本地计算机和github仓库的指南，这种文档将会增强agent的性能。
*/
```

2 Tool documentation format

A tool documentation consists of four sections: `tool_ID`, `tool_name`, `brief_description`, `parameters`, and `code_example` which are saved in a `Tom's Obvious Minimal Language (.toml)` file.

```
/*
notes:杨小兵-2024-12-17

1、工具文档由四个部分构成：tool_ID、tool_name、brief_description、parameters
2、工具文档还需要一个code_example，这个code_example被保存为一种称之为.toml的文件
*/
```

2.1 `tool_ID`

2.1.1 QGIS processing tool ID

Typically, QGIS processiong tools consist of two words one of - gdal, native, pdal,or qgis- and the name of the processing tool. For example the id of the tool for select by attribute is `qgis:selectbyattribute`. The QGIS tool_ID with their corresponding names can be generated using the code below. It is strongly recommended to run these codes in the QGIS python console.

```
for alg in QgsApplication.processingRegistry().algorithms():
    print(alg.id(), "->", alg.displayName())
```

```
/*
notes:杨小兵-2024-12-17
```

```
1、介绍了 QGIS（地理信息系统）中处理工具（Processing Tools）的命名规范，以及如何通过
Python 代码生成这些工具的 ID 和对应名称
2、QGIS 的处理工具通常由两部分组成
   2.1 前缀：表示工具所属的插件或模块，常见的前缀包括 `gdal`、`native`、`pdal`、`qgis`
等
   2.2 工具名称：描述具体功能的名称，例如 `selectbyattribute`
3. `qgis:selectbyattribute`：这是一个 QGIS 处理工具的 ID，其中：
   3.1 `qgis` 表示该工具属于 QGIS 原生命令集
   3.2 `selectbyattribute` 表示该工具的功能是基于属性选择
*/
```

2.1.2 Customize tool ID

To define a tool_ID for a customized tool, ensure that it adheres to the following rules: ***it must be simple and descriptive*** ; ***contains only lowercase letters and underscore without any spaces oe special characters***. For example a valid tool_ID would be : `thematic_map_creation`.

2.2 `brief_description`

A brief description (usually 1-2 sentence) of the tool should be provided. For QGIS processing tools, you can get the description by printing the help of any QGIS tool (see the code below). Conversely, for a customize tool a short description should be provided. For example the description of the `Select by attribute` tool is: *`"This algorithm creates a selection in a vector layer. The criteria for selected features is defined based on the values of an attribute from the input layer."`*

```
processing.algorithmHelp("tool_ID")

#Example the Select by attribute tool
processing.algorithmHelp("qgis:selectbyattribute")
```

2.3 `parameters`

In QGIS, processing tools typically require several parameters to define inputs, settings, and options necessary for the tool to run properly. These parameters specify the data, configurations, and choices that influence how the tool operates. The description of the parameters of each QGIS processing tools can be found on the QGIS algorithm provider website. Some few details of each parameter can be accessed by running the code provided above on the QGIS python console.

2.4 `code_example`

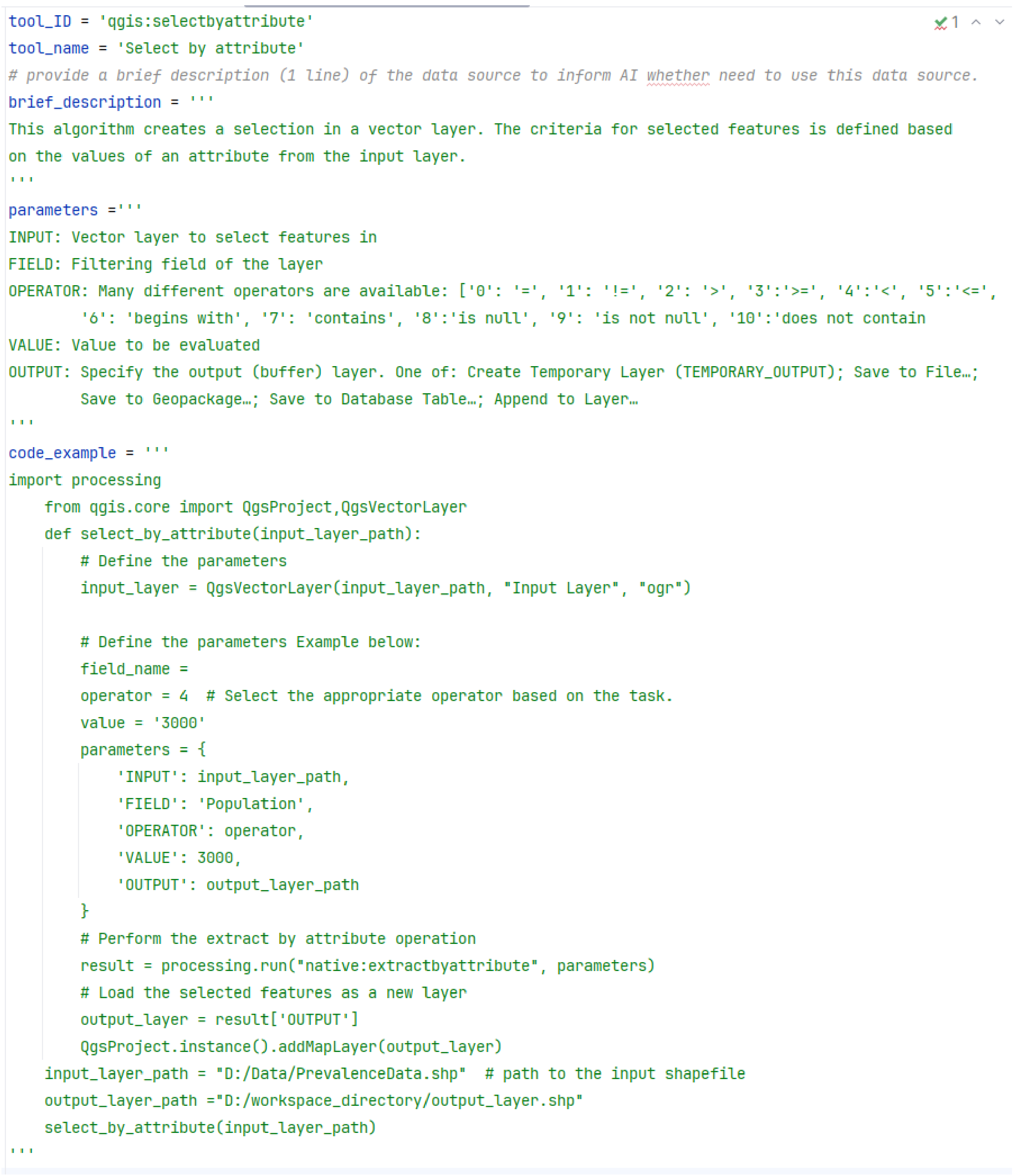
The code_example provides a practical example of how to use QGIS's processing tools with python to perform any operation. Typically, QGIS processing tools have similar format of code. The general format is described below.

```
1. Import Libraries
   └── Import the required QGIS and processing modules.
```

```
2. Load Input Layer
   └── Load the vector layer (e.g., shapefile) using `QgsVectorLayer`.

3. Define tool Parameters
   └── Set the parameters needed for the tool, such as INPUT, output settings,
etc.

4. Execute Buffer Operation
   └── Run the buffer tool using `processing.run()` with the defined parameters.

5. Load Output Layer
   └── Add the buffered layer to the QGIS project using
`QgsProject.instance().addMapLayer()`.
```

Also an example has can be seen in the image below.

2.4.1 An example of a tool documentation file



```
tool_ID = 'qgis:selectbyattribute'
tool_name = 'Select by attribute'
# provide a brief description (1 line) of the data source to inform AI whether need to use this data source.
brief_description = '''
This algorithm creates a selection in a vector layer. The criteria for selected features is defined based
on the values of an attribute from the input layer.
'''
parameters ='''
INPUT: Vector layer to select features in
FIELD: Filtering field of the layer
OPERATOR: Many different operators are available: ['0': '=', '1': '!=', '2': '>', '3':'>=', '4':'<', '5':'<=',
        '6': 'begins with', '7': 'contains', '8':'is null', '9': 'is not null', '10':'does not contain
VALUE: Value to be evaluated
OUTPUT: Specify the output (buffer) layer. One of: Create Temporary Layer (TEMPORARY_OUTPUT); Save to File…;
        Save to Geopackage…; Save to Database Table…; Append to Layer…
'''
code_example = '''
import processing
    from qgis.core import QgsProject,QgsVectorLayer
    def select_by_attribute(input_layer_path):
        # Define the parameters
        input_layer = QgsVectorLayer(input_layer_path, "Input Layer", "ogr")

        # Define the parameters Example below:
        field_name =
        operator = 4  # Select the appropriate operator based on the task.
        value = '3000'
        parameters = {
            'INPUT': input_layer_path,
            'FIELD': 'Population',
            'OPERATOR': operator,
            'VALUE': 3000,
            'OUTPUT': output_layer_path
        }
        # Perform the extract by attribute operation
        result = processing.run("native:extractbyattribute", parameters)
        # Load the selected features as a new layer
        output_layer = result['OUTPUT']
        QgsProject.instance().addMapLayer(output_layer)
    input_layer_path = "D:/Data/PrevalenceData.shp"  # path to the input shapefile
    output_layer_path ="D:/workspace_directory/output_layer.shp"
    select_by_attribute(input_layer_path)
'''
```

3 How to add tool documentation to the local machine

To add a documentation file (.toml) on your local machine, use the `Add a documentation file` button within the plugin. This allows you to select and automatically add a documentation file from any location on your local machine to the documentation folder within the plugin directory (`C:\Users\YOUR_USERENAME\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\SpatialAnalysisAgent-master\SpatialAnalysisAgent\Tools_Documentation`). **Note:** Replace "YOUR_USERNAME" with your actual username in the file path. Alternatively, you can manually add the documentation file by copying it directly to the specified plugin directory.

4 How to add tool documentation to the github repository

To add tool documentation to the github repository, use the `Contribute` button on the plugin to open the `Contribute to Spatial Analysis Agent` dialog. Then follow the steps below. **Note:** you will need to input your GitHub username and a GitHub personal access token. Learn how to get a token. Ensure the token has `repo` and `workflow` permissions. You can do this by adding them in the `Select scopes` when creating your personal token.

- Fork the plugin repository on GitHub: Click Here.
- Clone your fork to your local machine. Learn more on Cloning a repository
- Upload a TOML file using the `Contribute to Spatial Analysis Agent` dialog within the plugin. The uploaded TOML file will go to your forked repository.
- After uploading, go to GitHub and **open a pull request** from your forked to the main repository.