# REPORT PROJECT CS162

# WEEK 03

I. Tạo GitHub PRIVATE repository

- Step 1: Vào phần “Your repositories”:

A screenshot of a computer

Description automatically generated

- Step 2: Vào phần “new” tạo repository mới:



- Step 3: Đặt tên cho repository tại “Repository name”:

A screenshot of a computer program

Description automatically generated

- Step 4: Chọn “Private” và tạo repository “Create repository”:

A screenshot of a computer

Description automatically generated

II. Clone Github về máy tính:

- Step 1: Vào phần Your repositories

- Step 2: Chọn mở project muốn clone

A screenshot of a computer

Description automatically generated

- Step 3: Chọn “Set up in Desktop”

A screenshot of a computer

Description automatically generated

- Step 4: Download để clone về máy và set up

A screenshot of a computer

Description automatically generated

- Step 5: Mở GitHub chọn “Clone a repository from internet…

A screenshot of a computer

Description automatically generated

- Step 6: Copy repository’s link from internet sau đó dán vào “Clone a repository” rồi clone

A screenshot of a computer

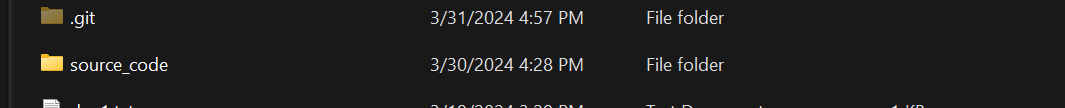
Description automatically generated

A screenshot of a computer

Description automatically generated

III. Tạo folder chứa source code:

- Tạo một folder chứa source\_code trong folder repository



IV. Create a file .gitignore:

- Step 1: Download and set up Git Bash.

- Step 2: Open the folder that is your repository. And then open Git Bash here.

A screenshot of a computer

Description automatically generated

my repository

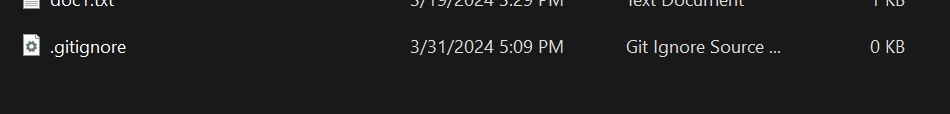
Ex: for me, it is project\_cs162.

A screenshot of a computer menu

Description automatically generated

- Step 3: write down the statement “touch .gitignore” and then press ‘Enter’, a .gitignore file is created.

A black screen with yellow text

Description automatically generated

V. Demo How to use git add, git commit, git push, git pull:

- Assume I have a file that’s name is text.txt in my repository like this:

A screenshot of a computer

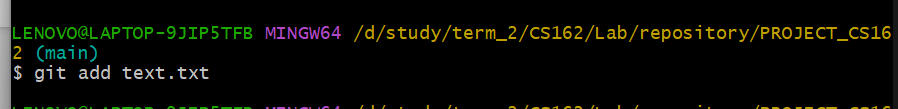
Description automatically generated

text.txt

- So if I want to operate the statement git add on this file I will:

+ Step 1: Open Git Bash in the folder which store this file.

+ Step 2: Write commands according to syntax “git add <name of your file>”, in this case that is “git add text.txt”



git add

\*Notice that: if you want to work at all thing in your repository, you use “git add .” instead of “git add <name of your file>”.

- And then, for using “git commit”, we will:

+ Step 1: Also open Git Bash in this folder.

+ Step 2: Write the statement “git commit -m “commit message”, then press “Enter”.

A computer screen with yellow text

Description automatically generated

git commit

-Next, for using “git push”, you must do:

+ Step 1: So, you still open Git Bash in your repository.

+ Step 2: Write the statement according to syntax “git push origin <branch\_name>”. (defined branch\_name is main), then press “Enter”.

A computer screen shot of a black screen

Description automatically generated

git push

- Now, in your repository on web, text.txt is stored.

A screenshot of a computer

Description automatically generated

- If text.txt is modified on github and you want to get this change for the file in your computer, you must do “git pull”.

+ Step 1: Open Git Bash in the folder which is repository.

+ Step 2: Write the statement according to syntax “git pull origin <branch\_name>”. (defined branch\_name is main), then press “Enter”.

A black screen with white text

Description automatically generated

git pull

VI. Exploring Multi-File Projects in Python:

1. Introduction:

-Creating Python projects with multiple files is essential for organizing code efficiently. This report explores methods for structuring such projects.

2. Project Structure:

-Use module-based or package-based structures.

-Modules contain related functionality, while packages are directories with modules and an \_\_init\_\_.py file.

3. Creating a Project:

-Define project requirements.

-Create a directory structure.

-Write modular code in separate files.

-Import modules using import statements.

-Handle circular dependencies.

-Test and debug modules independently.

Example:

A screenshot of a computer

Description automatically generated

'main' function

VII. Exploring Class Creation in Python:

1. Classes:

-Classes are blueprints for creating objects in Python.

-They encapsulate data for the object and define its behavior through methods.

2. Properties:

-Properties are attributes of a class instance.

-They store data related to the object's state.

3. Methods:

-Methods are functions defined within a class.

-They define the behavior of the class and can operate on its properties.

4. Creating Classes:

-Define the class using the class keyword.

-Include a constructor method (\_\_init\_\_) to initialize the object's properties.

-Define other methods to perform operations on the object's data.

5. Defining Properties:

-Use instance variables to define properties.

-Properties can be public, protected, or private using naming conventions (public\_var, \_protected\_var, \_\_private\_var).

6. Implementing Methods:

-Define methods within the class.

-Access instance variables using the self keyword.

-Methods can modify the object's state or perform computations.

Example:

A computer screen shot of a program

Description automatically generated

# WEEK 05

I. ‘vars.json’ file:

1. ‘RouteVar’ class:

a. Code:

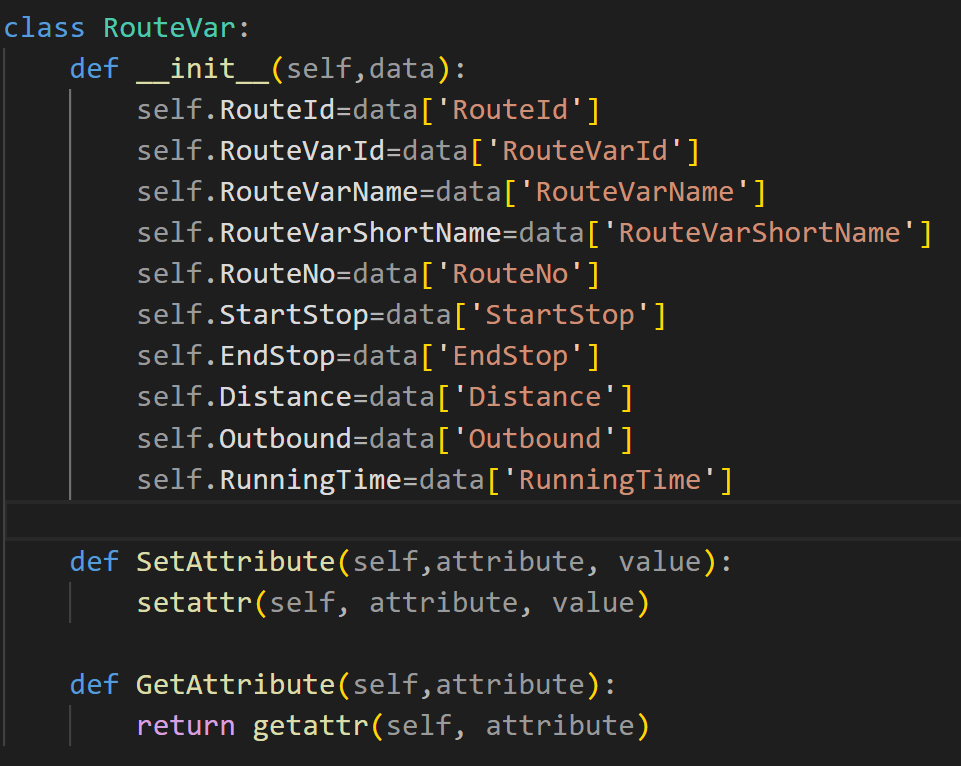


Figure 1. ‘RouteVar ‘ Class

b. Constructors:

- The ‘\_\_init\_\_(self, data)’ constructor initializes a ‘RouteVar’ object with data provided as a dictionary, assigning values to its attributes based on the dictionary keys.

c. Setter methods:

- The ‘SetAttribute(self, attribute, value)’ method dynamically assigns a value to a specified attribute using the ‘setattr’ function, enabling flexible attribute value modification while maintaining encapsulation.

e. Conclusion:

- The ‘RouteVar’ class offers a structured approach for handling route variable data, ensuring efficient attribute management and promoting encapsulation for better control and organization within the application.

2. ‘RouteVarQuery’ Class:

a. Code:

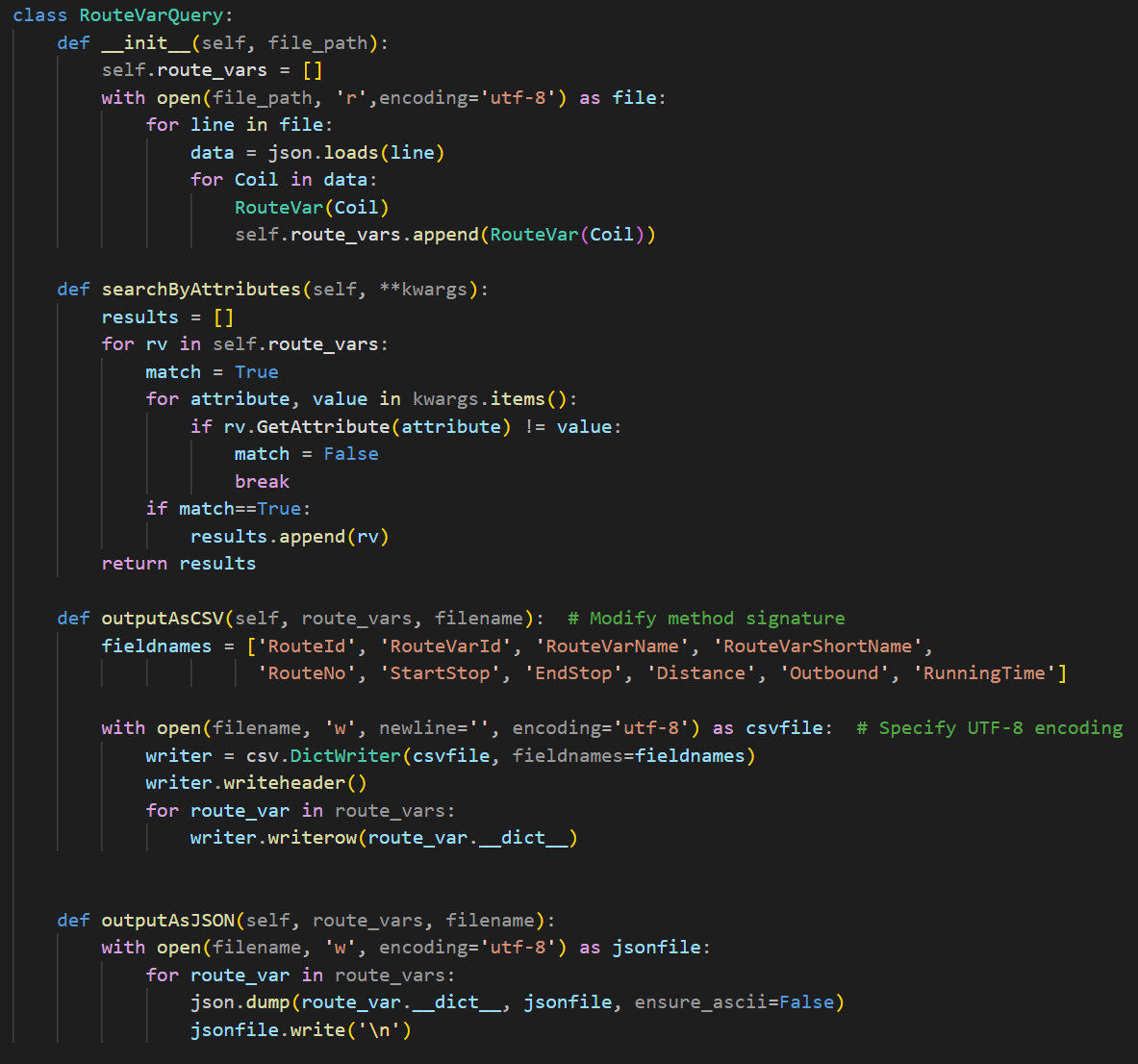


Figure 2. ‘RouteVarQuery’ class

b. Overview:

-the ‘RouteVarQuery’ class is designed to handle querying and manipulation of a collection of ‘RouteVar’ objects, as well as exporting them to CSV and JSON formats.

c. Initialization (‘\_\_init\_\_’):

-the constructor initializes a ‘RouteVarQuery’ object by reading data from ‘vars.json’ file specified by ‘file\_path’.

d. Searching (‘SearchByAttributes’):

- this method allows searching for ‘RouteVar’ objects based on specified attribute-value pairs.

-it takes keyword arguments (‘kwargs’) representing attribute-value pairs to search for.

-it iterates through each ‘RouteVar’ object in the ‘route\_vars’ list and checks if its attributes match the specified values.

-Matching objects are added to the ‘results’ list, which is returned after the iteration.

e. CSV Exporting (‘outputAsCSV’):

-This method exports ‘RouteVar’ objects to a CSV file specified by ‘filename’.

-It defines the fieldnames for the CSV file based on the attributes of a ‘RouteVar’ object.

-It opens the file in write mode with UTF-8 encoding and creates a ‘csv.DictWriter’ object with the specified fieldnames.

-It writes the header row using ‘writeheader()’ and iterates through each ‘RouteVar’ object, writing its attribute dictionary as a row in the CSV file.

f. JSON Exporting (‘outputAsJSON’):

-This method exports ‘RouteVar’ objects to a JSON file specified by ‘file\_name’.

- It iterates through each ‘RouteVar’ object in the ‘route\_vars’ list.

- For each object, it dumps its attribute dictionary to the JSON file using ‘json.dump()’, ensuring non-ASCII characters are handled properly.

-After each dump, it writes a newline character to separate JSON objects in the file.

I. ‘stops.json’ file:

1. ‘Stop’ class:

a. Code:

A screen shot of a computer program

Description automatically generated

Figure 3 'Stop' Class

b. Constructors:

- The ‘\_\_init\_\_(self, data)’ constructor initializes a ‘Stop’ object with data provided as a dictionary, assigning values to its attributes based on the dictionary keys.

c. Setter methods:

- The ‘SetAttribute(self, attribute, value)’ method dynamically assigns a value to a specified attribute using the ‘setattr’ function, enabling flexible attribute value modification while maintaining encapsulation.

e. Conclusion:

- The ‘Stop’ class offers a structured approach for handling route variable data, ensuring efficient attribute management and promoting encapsulation for better control and organization within the application.

2. ‘StopQuery’ Class:

a. Code:

A screen shot of a computer program

Description automatically generated

Figure 4 'StopQuery' Class

b. Overview:

-the ‘StopQuery’ class is designed to handle querying and manipulation of a collection of ‘Stop’ objects, as well as exporting them to CSV and JSON formats.

c. Initialization (‘\_\_init\_\_’):

-the constructor initializes a ‘StopQuery’ object by reading data from ‘stops.json’ file specified by ‘file\_path’.

d. Searching (‘SearchByAttributes’):

- this method allows searching for ‘Stop’ objects based on specified attribute-value pairs.

-it takes keyword arguments (‘kwargs’) representing attribute-value pairs to search for.

-it iterates through each ‘Stop’ object in the ‘stops’ list and checks if its attributes match the specified values.

-Matching objects are added to the ‘results’ list, which is returned after the iteration.

e. CSV Exporting (‘outputAsCSV’):

-This method exports ‘Stop’ objects to a CSV file specified by ‘filename’.

-It defines the fieldnames for the CSV file based on the attributes of a ‘Stop’ object.

-It opens the file in write mode with UTF-8 encoding and creates a ‘csv.DictWriter’ object with the specified fieldnames.

-It writes the header row using ‘writeheader()’ and iterates through each ‘Stop’ object, writing its attribute dictionary as a row in the CSV file.

f. JSON Exporting (‘outputAsJSON’):

-This method exports ‘Stop’ objects to a JSON file specified by ‘file\_name’.

- It iterates through each ‘Stop’ object in the ‘stops’ list.

- For each object, it dumps its attribute dictionary to the JSON file using ‘json.dump()’, ensuring non-ASCII characters are handled properly.

-After each dump, it writes a newline character to separate JSON objects in the file.

# WEEK 06

I. Transforming a (lat, lng) coordinate to (x,y) coordinate:

1. Code:

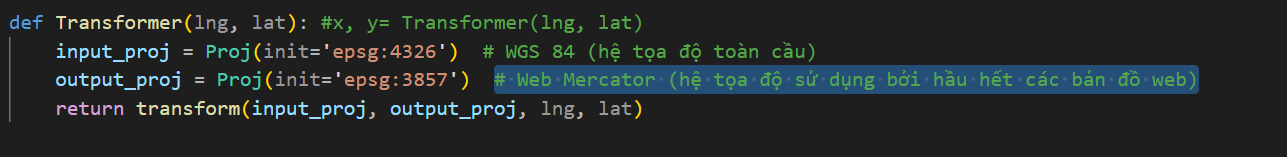


Figure 5 Converting Coordinate (lat, lng) to (x, y) Using Pyproj

2. Install ‘Pyproj’: Before using ‘Pyproj’, ensure that it is installed in your Python environment. You can install it using pip:



Figure 6 Installing statement

3. Import ‘Pyproj’: Import the necessary modules from the Pyproj library in your Python script.



Figure 7 Imported modules from the ‘Pyproj’ library

4. Specify Coordinate Reference Systems (CRS): Define the CRS for the input (lat, lng) and output (x, y) coordinate systems.

A group of text on a black background

Description automatically generated

Figure 8 Specify Coordinate Reference Systems (CRS)

5. Initialize Projections: Create projection objects for the input and output CRS using the specified EPSG codes.

A group of text on a black background

Description automatically generated

Figure 9 Initialize projections

6. Convert Coordinates: Use the transform function to convert coordinates from the input CRS to the output CRS.



Figure 10 Convert Coordinates

7. About CRS3405:

- CRS 3405 refers to a specific Coordinate Reference System. It defines a projected coordinate system used for mapping and analysis purposes, particularly in a local or regional context. Details about the projection parameters, units, and area of validity for CRS 3405 can be obtained from relevant geospatial authorities or documentation sources.

II. Exploring GeoJSON.io and Creating GeoJSON Files:

1. Overview:

-GeoJSON.io is a web-based mapping platform that allows users to visualize, edit, and share geospatial data in GeoJSON format. It provides an intuitive interface for creating and editing geographic features such as points, lines, and polygons, making it a valuable tool for geospatial data exploration and analysis.

2. Process of creating GeoJSON Files:

a. Access GeoJSON.io: Open a web browser and navigate to the GeoJSON.io website (<https://geojson.io/>).

b. Draw Point Feature:

-Click on the "Draw Point" button located in the toolbar.

-Click on the map to place the point at the desired location.

-Optionally, you can add properties to the point feature by clicking on the “Edit properties" button.

c. Draw Point Feature:

-Click on the "Draw Point" button located in the toolbar.

-Click on the map to place the point at the desired location.

-Optionally, you can add properties to the point feature by clicking on the "Edit properties" button.

d. Save GeoJSON File:

-Once you have created the desired features, click on the "Save" button located in the toolbar.

-Choose the GeoJSON format and specify a file name for the saved file.

-Click "Save" to download the GeoJSON file to your local system.

3. Process of loading GeoJSON Files on GeoJSON.io:

a. Access GeoJSON.io: Open the GeoJSON.io website in your web browser.

b. Upload GeoJSON File:

-Click on the "Open" button located in the toolbar.

-Choose the GeoJSON file you created earlier from your local system.

-The uploaded features will be displayed on the map.

c. Edit and Save Changes:

-You can further edit the features directly on the map if needed.

-After making changes, click on the "Save" button to save the modified GeoJSON file.

4. Conclusion:

-GeoJSON.io is a user-friendly platform for creating, editing, and visualizing GeoJSON data. By following the outlined steps, users can easily create GeoJSON files containing Point and LineString features and load them onto the GeoJSON.io platform for visualization and analysis. This report serves as a guide for leveraging GeoJSON.io's capabilities in working with geospatial data on the web.

III. ‘paths.json’ file:

1. ‘Path’ class

a. Code:

A screen shot of a computer program

Description automatically generated

Figure 11 ‘Path’ Class

b. Constructors:

- The ‘\_\_init\_\_(self, data)’ constructor initializes a ‘Path’ object with data provided as a dictionary, assigning values to its attributes based on the dictionary keys.

c. Setter methods:

- The ‘SetAttribute(self, attribute, value)’ method dynamically assigns a value to a specified attribute using the ‘setattr’ function, enabling flexible attribute value modification while maintaining encapsulation.

e. Conclusion:

- The ‘Path’ class offers a structured approach for handling route variable data, ensuring efficient attribute management and promoting encapsulation for better control and organization within the application.

2. ‘PathQuery’ Class

a. Code:

A screen shot of a computer program

Description automatically generated

Figure 12 'PathQuery' Class

b. Overview:

-the ‘PathQuery’ class is designed to handle querying and manipulation of a collection of ‘Path’ objects, as well as exporting them to CSV and JSON formats.

c. Initialization (‘\_\_init\_\_’):

-the constructor initializes a ‘PathQuery’ object by reading data from ‘paths.json’ file specified by ‘file\_path’.

d. Searching (‘SearchByAttributes’):

- this method allows searching for ‘Path’ objects based on specified attribute-value pairs.

-it takes keyword arguments (‘kwargs’) representing attribute-value pairs to search for.

-it iterates through each ‘Path’ object in the ‘paths’ list and checks if its attributes match the specified values.

-Matching objects are added to the ‘results’ list, which is returned after the iteration.

e. CSV Exporting (‘outputAsCSV’):

-This method exports ‘Stop’ objects to a CSV file specified by ‘filename’.

-It defines the fieldnames for the CSV file based on the attributes of a ‘Path’ object.

-It opens the file in write mode with UTF-8 encoding and creates a ‘csv.DictWriter’ object with the specified fieldnames.

-It writes the header row using ‘writeheader()’ and iterates through each ‘Path’ object, writing its attribute dictionary as a row in the CSV file.

f. JSON Exporting (‘outputAsJSON’):

-This method exports ‘Path’ objects to a JSON file specified by ‘file\_name’.

- It iterates through each ‘Path’ object in the ‘paths’ list.

- For each object, it dumps its attribute dictionary to the JSON file using ‘json.dump()’, ensuring non-ASCII characters are handled properly.

-After each dump, it writes a newline character to separate JSON objects in the file.

IV. Shapely:

1. Introduction:

- Shapely is a Python library used for geometric operations and manipulations, particularly in the context of spatial analysis and modeling. This technical report aims to explore the key functions available in Shapely and their functionalities.

2. Overview:

- Shapely provides a wide range of geometric objects and operations for working with geometries in two-dimensional Cartesian space. It allows users to create, manipulate, and analyze geometric shapes such as points, lines, polygons, and more. Shapely is built on top of the widely used GEOS (Geometry Engine - Open Source) library, making it a powerful tool for geometric computations.

3. Key Functions in Shapely:

a. Geometry Creation:

-Point: Create a point geometry with a given coordinate.

-LineString: Form a line geometry from a sequence of points.

-Polygon: Generate a polygon geometry from a sequence of points or a linear ring.

b. Geometric Operations:

-Union: Compute the union of two or more geometric shapes.

-Difference: Find the difference between two shapes.

-Intersection: Determine the common area between two shapes.

-Buffer: Generate a buffer zone around a geometry.

c. Geometric Analysis:

-Area: Calculate the area enclosed by a polygon.

-Length: Compute the length of a LineString.

-Centroid: Determine the centroid (geometric center) of a geometry.

-Distance: Find the shortest distance between two geometries.

d. Spatial Relationships:

-Crosses: Determine if two geometries cross each other.

-Equals: Check if two geometries are equal.

-Disjoint: Verify if two geometries are disjoint.

-Overlaps: Determine if two geometries overlap

4. Conclusion:

- Shapely is a powerful Python library for geometric operations and analysis, offering a comprehensive set of functions for working with geometric shapes. By understanding and leveraging the functionalities provided by Shapely, users can perform a wide range of spatial computations and solve various geometric problems efficiently. This report serves as an introduction to the key functions in Shapely and their applications in spatial analysis and modeling.

V. Exploring Rtree in Python

1. Introduction:

-Rtree is a Python library designed for spatial indexing of geometric data, enabling fast spatial queries and operations. This report delves into Rtree, elucidating its functionalities and applications within Python.

2. Understanding Rtree:

-Rtree is a spatial indexing tool, organizing geometric objects into a tree-based data structure for quick spatial searches. By utilizing techniques like R-tree, R\*-tree, and STR-tree, Rtree adeptly manages spatial datasets, making it suitable for handling extensive spatial data.

3. Key Features of Rtree:

a. Spatial Indexing:

-Rtree employs hierarchical tree structures to organize spatial objects, facilitating rapid spatial searches based on proximity.

b. Efficient Spatial Queries:

-This library supports various spatial queries, including:

-Nearest Neighbor Search: Identifying the nearest spatial object to a given point.

-Range Query: Retrieving objects within a specified spatial range.

-Intersection Query: Determining objects intersecting with a specified geometry.

-Spatial Join: Merging spatial datasets based on their spatial relationships.

c. Python Integration:

-Rtree seamlessly integrates with Python, allowing smooth collaboration with other geospatial libraries like Shapely, GeoPandas, and Fiona.

d. Versatility:

-Supporting different geometric types and multi-dimensional data, Rtree proves versatile for a wide range of spatial data analysis tasks.

4. Applications of Rtree:

a. Geographic Information Systems (GIS):

-In GIS applications, Rtree is extensively used for indexing and querying spatial datasets, spatial data management, and analysis.

b. Spatial Data Analysis:

-Rtree facilitates tasks such as clustering, hotspot detection, and spatial pattern recognition, essential for spatial data analysis.

c. Location-Based Services (LBS):

-Used in LBS, Rtree handles proximity searches, route optimization, and geofencing, enhancing location-based functionalities.

5. Conclusion:

-Rtree emerges as a powerful spatial indexing library in Python, offering fast spatial query capabilities for managing large-scale spatial datasets. Its seamless integration with Python and support for various geometric types highlight its importance in spatial data analysis, GIS applications, and location-based services. This report elucidates Rtree's features and applications, emphasizing its crucial role in spatial data processing and analysis within the Python ecosystem.

VI. LLM Libraries for Function Selection based on Queries:

1. Introduction:

-LLM (Learning to Learn from Models) libraries are designed to automate the process of selecting appropriate functions or models based on specific queries or data characteristics. This report explores various LLM libraries, their methodologies, and applications in function selection.

2. Understanding LLM Libraries:

-LLM libraries leverage machine learning techniques to learn from past experiences and data patterns, enabling intelligent function selection for given queries. These libraries typically consist of algorithms that analyze query features and historical data to recommend suitable functions or models.

3. Key LLM Libraries:

a. Scikit-Learn (sklearn):

-Scikit-Learn is a widely-used machine learning library in Python that offers various tools for data mining and analysis. While not explicitly designed for LLM, Scikit-Learn provides functionalities like model selection and hyperparameter tuning, which can be adapted for LLM tasks.

b. AutoML Libraries:

-Automated Machine Learning (AutoML) libraries like Auto-sklearn, TPOT, and H2O AutoML automate the process of model selection and hyperparameter tuning. These libraries utilize meta-learning and optimization techniques to identify the best-performing models for given queries or datasets.

c. Meta-Learning Libraries:

-Meta-learning libraries such as Learn2Learn and MAML (Model-Agnostic Meta-Learning) focus specifically on learning to learn tasks. These libraries employ meta-learning algorithms to adapt quickly to new tasks and select appropriate models based on query characteristics.

4. Methodologies and Techniques:

a. Feature Engineering:

-LLM libraries often rely on feature engineering to extract relevant information from queries or datasets. These features may include data statistics, query properties, or domain-specific attributes.

b. Meta-Learning Algorithms:

-Meta-learning algorithms learn from past experiences and data patterns to generalize across tasks. Techniques like gradient-based meta-learning, reinforcement learning, and evolutionary algorithms are commonly employed in LLM libraries.

c. Model Evaluation and Selection:

-LLM libraries use techniques like cross-validation, grid search, and Bayesian optimization to evaluate and select the best-performing models for given queries. These techniques aim to balance model accuracy, complexity, and generalization.

5. Applications of LLM Libraries:

a. Automated Machine Learning (AutoML):

-LLM libraries are extensively used in AutoML systems to automate the process of model selection, hyperparameter tuning, and pipeline optimization.

b. Dynamic System Adaptation:

-LLM techniques enable dynamic adaptation of system configurations and models based on changing query requirements or environmental conditions.

c. Recommendation Systems:

-In recommendation systems, LLM libraries assist in selecting appropriate recommendation algorithms based on user preferences, item characteristics, and contextual information.

7. Conclusion:

-LLM libraries play a vital role in automating the process of function selection based on queries or data characteristics. By leveraging machine learning and meta- learning techniques, these libraries offer efficient solutions for model selection, hyperparameter tuning, and system adaptation across various domains. Understanding the methodologies and applications of LLM libraries is crucial for leveraging their capabilities in building intelligent systems and data-driven applications.