Gardner API Utility Documentation

Willem van der Schans

Version 1.1.0
4/25/2023 6:23:00 PM

Table of Contents

README

Gardner Policy Institute API Utility

Author: Willem van der Schans

Commissioner: Gardner Policy Institute

Description: A Python utility for generating API requests from ConstructionMonitor.com, Utah Real Estate.com, Realtor.com, and the US Census APIs

VERSION INFO

1. Python=3.10
2. pandas~=1.5.2
3. requests~=2.28.1
4. beautifulsoup4~=4.11.1
5. pysimplegui~=4.60.4
6. cryptography~=38.0.1
7. pillow~=9.2.0

Note: Use the latest viable requirements for the versions above

Note: All dependencies are included in the Windows installer

Authentication Requirements

Authentication Keys are needed for utahrealestate.com and constructionmonitor.com

The program provides a safe way to store and use authentication keys

---

License

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all documented files with brief descriptions:

Class Documentation

AuthUtil.AuthUtil Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatusListedOrModified
- file_name
- append_file
- keyPath
- filePath
- k
- keyFlag
- jsonDict
- passFlagUre
- passFlagCm
- outcomeText
- popupFlag

Private Member Functions

- def __SetValues (self, values)
- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Detailed Description

Definition at line 18 of file AuthUtil.py.

Constructor & Destructor Documentation

def AuthUtil.AuthUtil.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object, which in this case means that it creates
a new window and displays it on screen.

Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

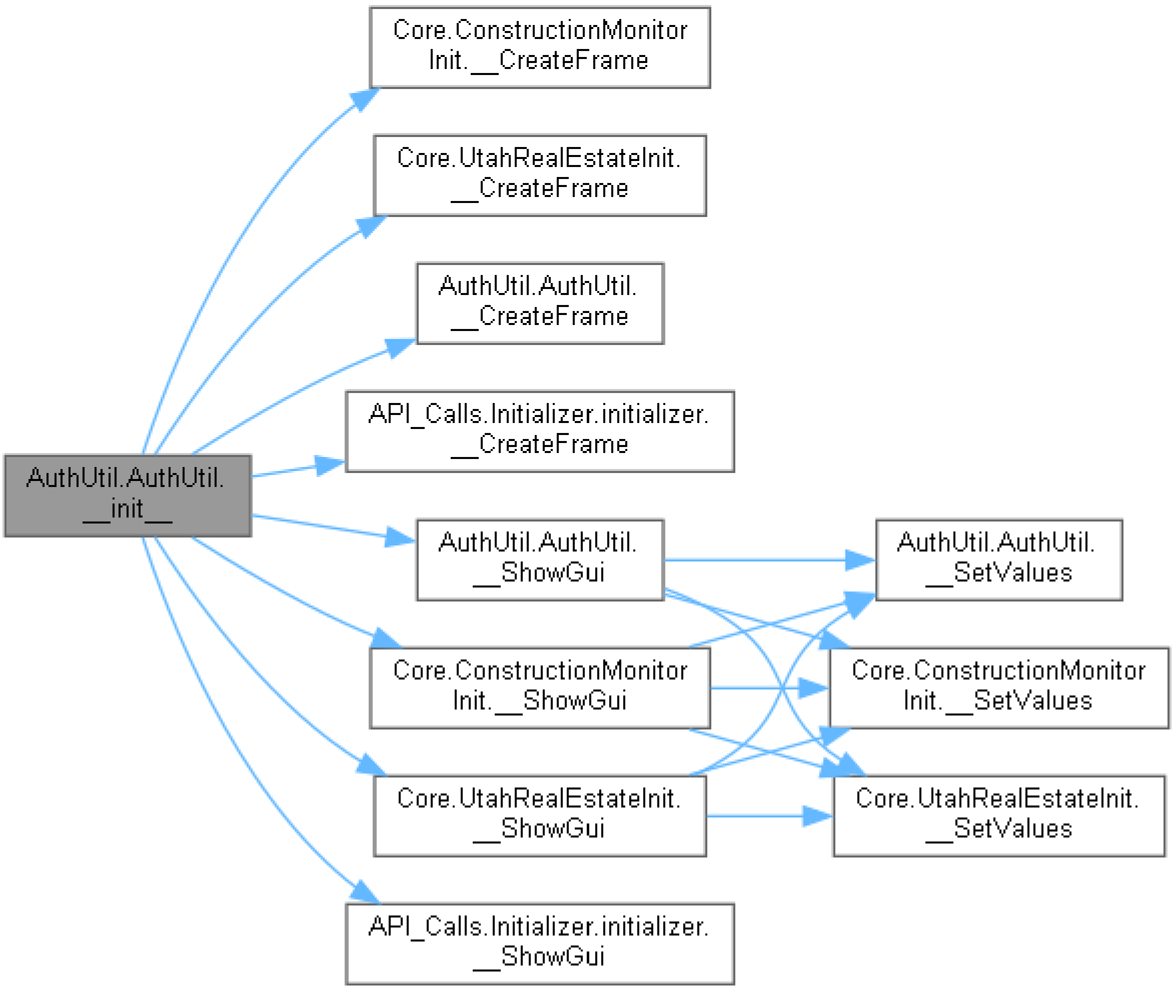
Definition at line 20 of file AuthUtil.py.

```
00020     def __init__(self):
00021
00022         """
00023     The __init__ function is called when the class is instantiated.
00024     It sets up the initial state of the object, which in this case means that
it creates a new window and displays it on screen.
```

```

    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.file_name = None
        self.append_file = None
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
        self.k = None
        self.keyFlag = True
        self.jsonDict = {}
        self.passFlagUre = False
        self.passFlagCm = False
        self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
                           "Submitting will overwrite any old values in an unrecoverable manner."

        if os.path.exists(self.filePath):
            pass
        else:
            if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
                os.mkdir(self.filePath)
            else:
                os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
                os.mkdir(self.filePath)

        if os.path.exists(self.keyPath):
            pass
        else:
            if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
                os.mkdir(self.keyPath)
            else:
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:
ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35
.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as
hidden. This is either a Permission error or Input Error. Continuing...")
            pass

```

References Core.ConstructionMonitorInit.__CreateFrame(), Core.UtahRealEstateInit.__CreateFrame(), AuthUtil.AuthUtil.__CreateFrame(), API_Calls.Initializer.initializer.__CreateFrame(), AuthUtil.AuthUtil.__ShowGui(), API_Calls.Initializer.initializer.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), Core.UtahRealEstateInit.__ShowGui(), AuthUtil.AuthUtil.append_file, Core.ConstructionMonitorInit.append_file, Core.UtahRealEstateInit.append_file, AuthUtil.AuthUtil.file_name, Core.UtahRealEstateInit.file_name, AuthUtil.AuthUtil.filePath, Core.UtahRealEstateMain.filePath, AuthUtil.AuthUtil.jsonDict, AuthUtil.AuthUtil.k, AuthUtil.AuthUtil.keyFlag, AuthUtil.AuthUtil.keyPath, Core.UtahRealEstateMain.keyPath, AuthUtil.AuthUtil.ListedOrModified, Core.UtahRealEstateInit.ListedOrModified, AuthUtil.AuthUtil.outcomeText, AuthUtil.AuthUtil.passFlagCm, AuthUtil.AuthUtil.passFlagUre, AuthUtil.AuthUtil.StandardStatus, and Core.UtahRealEstateInit.StandardStatus.

Here is the call graph for this function:


Core.ConstructionMonitorInit.__CreateFrame
Core.UtahRealEstateInit.__CreateFrame
AuthUtil.AuthUtil.__CreateFrame
API_Calls.Initializer.initializer.__CreateFrame
AuthUtil.AuthUtil.__init__
AuthUtil.AuthUtil.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__ShowGui
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__ShowGui
Core.UtahRealEstateInit.__SetValues
API_Calls.Initializer.initializer.__ShowGui

Member Function Documentation

def AuthUtil.AuthUtil.__CreateFrame (*self*) `[private]`

```
The __CreateFrame function creates the GUI layout for the Authentication Utility.
It is called by __init__ and returns a list of lists that contains all the elements
that will be displayed in the window.

Args:
self: Access the class attributes and methods

Returns:
A list of lists

Doc Author:
Trelent
```

Definition at line 235 of file AuthUtil.py.

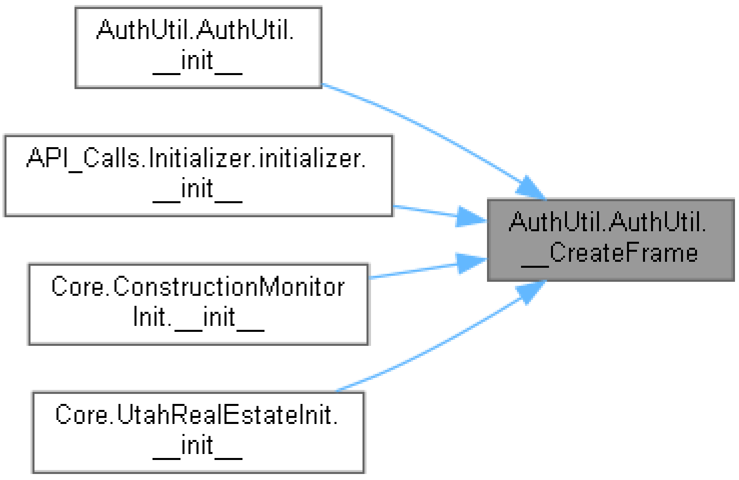
```
00235     def __CreateFrame(self):
00236         """
00237     The __CreateFrame function creates the GUI layout for the Authentication
Utility.
00238     It is called by __init__ and returns a list of lists that contains all the
elements
00239     that will be displayed in the window.
00240
00241     Args:
00242         self: Access the class attributes and methods
```

```

    Returns:
        A list of lists

    Doc Author:
        Trelent
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Push(),
                 sg.Text("Utah Real Estate API Key: ", justification="center"),
                 sg.Push()]

        line3 = [sg.Push(),
                 sg.Input(default_text="123", key="-ureAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line4 = [sg.HSeparator()]

        line5 = [sg.Push(),
                 sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
                 sg.Push()]

        line6 = [sg.Push(),
                 sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text(self.outcomeText, justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout
```

References AuthUtil.AuthUtil.outcomeText.

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__CreateFrame

def AuthUtil.AuthUtil.__SetValues (***self*, *values*) `[private]`**

```
The __SetValues function is called when the user clicks on the &quot;OK&quot; button
in the window.
It takes a dictionary of values as an argument, and then uses those values to update
the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

Args:
self: Make the function a method of the class
values: Store the values that are entered into the form

Returns:
A dictionary of the values entered by the user

Doc Author:
Willem van der Schans, Trelent AI
```

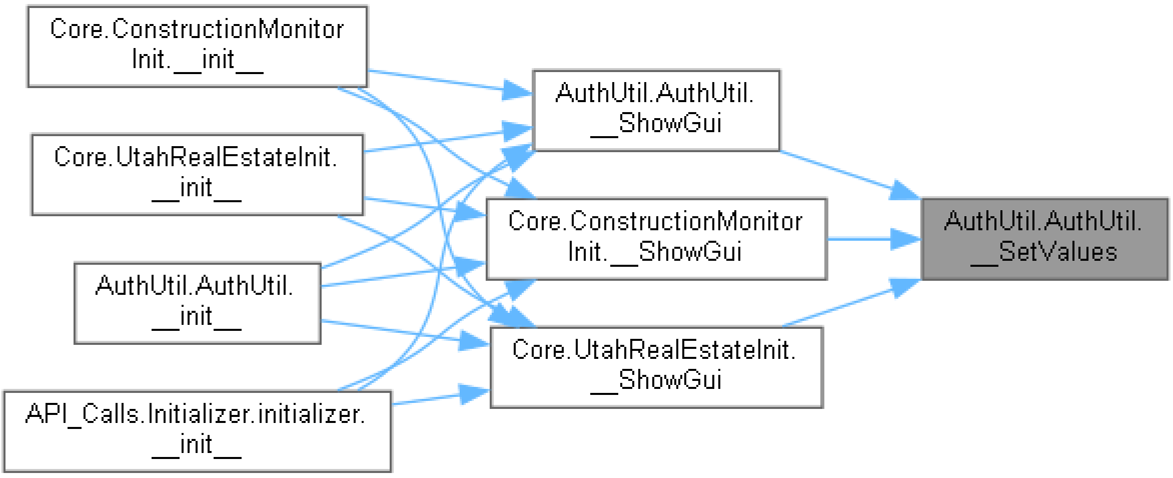
Definition at line 104 of file AuthUtil.py.

```
    def __SetValues(self, values):

        """
    The __SetValues function is called when the user clicks on the &quot;OK&quot;
button in the window.
    It takes a dictionary of values as an argument, and then uses those values
to update
    the auth.json file with new keys for both Utah Real Estate and Construction
Monitor.

    Args:
        self: Make the function a method of the class
        values: Store the values that are entered into the form

    Returns:
        A dictionary of the values entered by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None
        self.popupFlag = False

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        # Try initial decoding, if fails pass and write new keys and files
        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE functionality will be accessed")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM functionality will be accessed")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            if values["-cmAuth-"].startswith("Basic"):
                self.jsonDict.update(
                    {"cm": {"parameter": "Authorization",
                            "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
                self.passFlagCm = True
            else:
                PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction Monitor",
                             windowType="AuthError")
                self.popupFlag = True
                pass
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction Monitor",
                         windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate", windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
            PopupWrapped("Please make sure you provide a key for Construction Monitor", windowType="errorLarge")
        if self.popupFlag:
            pass
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

```

References AuthUtil.AuthUtil.filePath, Core.UtahRealEstateMain.filePath, AuthUtil.AuthUtil.jsonDict, AuthUtil.AuthUtil.k, AuthUtil.AuthUtil.passFlagCm, AuthUtil.AuthUtil.passFlagUre, and AuthUtil.AuthUtil.popupFlag.

Referenced by AuthUtil.AuthUtil.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), and Core.UtahRealEstateInit.__ShowGui().

Here is the caller graph for this function:


Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
AuthUtil.AuthUtil. __SetValues

def AuthUtil.AuthUtil.__ShowGui (***self, layout, text*****)** `[private]`

```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the window.

Args:
self: Represent the instance of the class
layout: Pass the layout of the gui to be displayed
text: Set the title of the window

Returns:
A window object
```

Definition at line 196 of file AuthUtil.py.

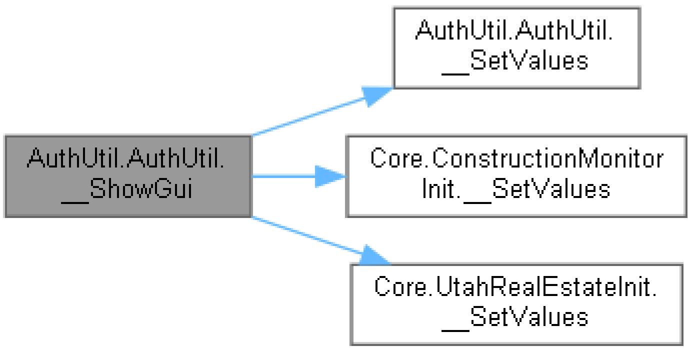
```
00196     def __ShowGui(self, layout, text):
00197 
00198         """
00199     The __ShowGui function is a helper function that displays the GUI to the user.
00200     It takes in two arguments: layout and text. The layout argument is a list
of lists,
00201     which contains all the elements that will be displayed on screen. The text
argument
00202     is simply what will be displayed at the top of the window.
00203 
00204     Args:
00205         self: Represent the instance of the class
00206         layout: Pass the layout of the gui to be displayed
00207         text: Set the title of the window
00208 
00209     Returns:
00210         A window object
00211     """
00212         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00213                            finalize=True,
00214                            icon=ImageLoader("taskbar_icon.ico"))
00215 
00216         while not self.passFlagUre or not self.passFlagCm:
00217             event, values = window.read()
00218 
00219             if event == "Submit":
00220                 try:
00221                     self.__SetValues(values)
00222                 except Exception as e:
00223                     print(e)
00224                     RESTError(993)
```

```
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

```

References AuthUtil.AuthUtil.__SetValues(), Core.ConstructionMonitorInit.__SetValues(), Core.UtahRealEstateInit.__SetValues(), AuthUtil.AuthUtil.passFlagCm, and AuthUtil.AuthUtil.passFlagUre.

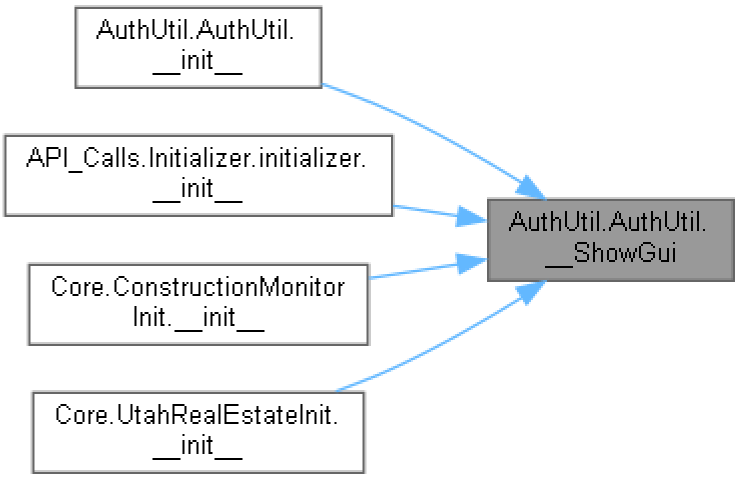
Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the call graph for this function:



AuthUtil.AuthUtil.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Here is the caller graph for this function:



AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__ShowGui

Member Data Documentation

AuthUtil.AuthUtil.append_file

Definition at line 38 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

AuthUtil.AuthUtil.file_name

Definition at line 37 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), and Core.UtahRealEstateInit.__init__().

AuthUtil.AuthUtil.filePath

Definition at line 40 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateMain.__init__(), AuthUtil.AuthUtil.__SetValues(), and Core.UtahRealEstateMain.mainFunc().

AuthUtil.AuthUtil.jsonDict

Definition at line 43 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), and AuthUtil.AuthUtil.__SetValues().

AuthUtil.AuthUtil.k

Definition at line 41 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), and AuthUtil.AuthUtil.__SetValues().

AuthUtil.AuthUtil.keyFlag

Definition at line 42 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__().

AuthUtil.AuthUtil.keyPath

Definition at line 39 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateMain.__init__(), and Core.UtahRealEstateMain.mainFunc().

AuthUtil.AuthUtil.ListedOrModified

Definition at line 36 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateInit.__init__(), and Core.UtahRealEstateInit.__SetValues().

AuthUtil.AuthUtil.outcomeText

Definition at line 46 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__CreateFrame(), and AuthUtil.AuthUtil.__init__().

AuthUtil.AuthUtil.passFlagCm

Definition at line 45 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), AuthUtil.AuthUtil.__SetValues(), and AuthUtil.AuthUtil.__ShowGui().

AuthUtil.AuthUtil.passFlagUre

Definition at line 44 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), AuthUtil.AuthUtil.__SetValues(), and AuthUtil.AuthUtil.__ShowGui().

AuthUtil.AuthUtil.popupFlag

Definition at line 124 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__SetValues().

AuthUtil.AuthUtil.StandardStatus

Definition at line 35 of file AuthUtil.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateInit.__init__(), and Core.UtahRealEstateInit.__SetValues().

The documentation for this class was generated from the following file:

- AuthUtil.py

BatchProcessing.BatchProcessorConstructionMonitor Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject)
- def FuncSelector (self)
- def ConstructionMonitorProcessor (self, valueObject)

Public Attributes

- dataframevalueObject

Private Attributes

- __numBatches__parameterDict
- __restDomain
- __headerDict
- __columnSelection
- __maxRequests
- __requestCount
- __requestCalls
- __dateTracker

Detailed Description

Definition at line 41 of file BatchProcessing.py.

Constructor & Destructor Documentation

def BatchProcessing.BatchProcessorConstructionMonitor.__init__ (*self*, *RestDomain*, *NumBatches*, *ParameterDict*, *HeaderDict*, *ColumnSelection*, *valueObject*)

```
The __init__ function is the constructor for a class. It is called when an object of
that class
is created, and it sets up the attributes of that object. In this case, we are setting
up our
object to have a dataframe attribute (which will be used to store all of our data),
as well as
attributes for each parameter in our ReST call.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to retrieve
ParameterDict: Pass in the parameters that will be used to make the api call
HeaderDict: Pass the header dictionary from the main function to this class
ColumnSelection: Determine which columns to pull from the api
valueObject: Pass in the value object that is used to determine what values are returned

Returns:
An object of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 43 of file BatchProcessing.py.

```
    def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):

        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is created, and it sets up the attributes of that object. In this case, we are setting up our
    object to have a dataframe attribute (which will be used to store all of our data), as well as
    attributes for each parameter in our ReST call.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to retrieve
        ParameterDict: Pass in the parameters that will be used to make the api call
        HeaderDict: Pass the header dictionary from the main function to this class
        ColumnSelection: Determine which columns to pull from the api
        valueObject: Pass in the value object that is used to determine what values are returned

    Returns:
        An object of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.valueObject = valueObject
        self.__maxRequests = 10000
        self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])))
        self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
        self.__dateTracker = None

```

References BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests, BatchProcessing.BatchProcessorConstructionMonitor.__numBatches, BatchProcessing.BatchProcessorUtahRealEstate.__numBatches, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls, BatchProcessing.BatchProcessorConstructionMonitor.__requestCount, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, BatchProcessing.BatchProcessorConstructionMonitor.valueObject, and BatchProcessing.BatchProcessorUtahRealEstate.valueObject.

Member Function Documentation

def BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor (*self*, *valueObject*)

```
The ConstructionMonitorProcessor function will use requests to get data from
ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called
__df (which is local). This
process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at which point __df will contain all

Args:
self: Represent the instance of the object itself
valueObject: Update the progress bar in the gui

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 94 of file BatchProcessing.py.

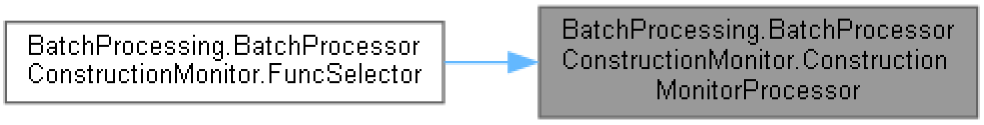
```
00094     def ConstructionMonitorProcessor(self, valueObject):
00095         """
00096     The ConstructionMonitorProcessor function will use requests to get data from
00097        ConstructionMontior.com's ReST API and store it into a pandas DataFrame
object called __df (which is local). This
00098        process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at which point __df will contain all
00099
00100     Args:
00101         self: Represent the instance of the object itself
00102         valueObject: Update the progress bar in the gui
00103
00104     Returns:
00105         A dataframe
00106
00107     Doc Author:
00108         Willem van der Schans, Trelent AI
00109     """
00110         __df = None
00111         for callNum in range(0, self.__requestCount):
00112             self.__parameterDict["from"] = 0
00113
00114             if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00115                 __batchNum = self.__requestCalls
00116                 if __df is None:
00117                     self.__dateTracker = str(date.today())
00118                 else:
00119                     self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00120             elif self.__requestCount == 1:
00121                 __batchNum = self.__numBatches
00122                 self.__dateTracker = str(date.today())
00123             else:
00124                 __batchNum = self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])) - (
00125                         self.__requestCount - 1)
00126                 self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00127
00128             self.__parameterDict['dateEnd'] = self.__dateTracker
00129
00130             for record in range(0, int(math.ceil(__batchNum))):
00131                 if record != 0:
```

```
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict} | Headers = {self.__headerDict}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)

```

References BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests, BatchProcessing.BatchProcessorConstructionMonitor.__numBatches, BatchProcessing.BatchProcessorUtahRealEstate.__numBatches, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls, BatchProcessing.BatchProcessorConstructionMonitor.__requestCount, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, and Core.UtahRealEstateMain.dataframe.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector().

Here is the caller graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (***self*)**

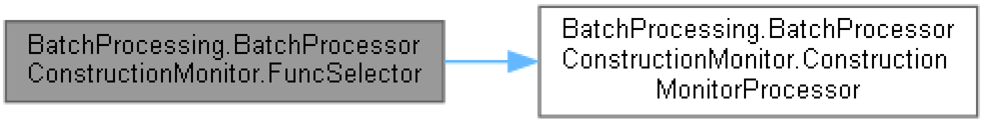
```
The FuncSelector function is a function that takes the valueObject and passes it to
the ConstructionMonitorProcessor function.
The ConstructionMonitorProcessor function then uses this valueObject to determine which
of its functions should be called.

Args:
self: Represent the instance of the class

Returns:
The result of the constructionmonitorprocessor function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 78 of file BatchProcessing.py.

```
    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject and passes
it to the ConstructionMonitorProcessor function.
    The ConstructionMonitorProcessor function then uses this valueObject to
determine which of its functions should be called.

    Args:
        self: Represent the instance of the class

    Returns:
        The result of the constructionmonitorprocessor function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.ConstructionMonitorProcessor(self.valueObject)

```

References BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProcessing.BatchProcessorConstructionMonitor.valueObject, and BatchProcessing.BatchProcessorUtahRealEstate.valueObject.

Here is the call graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

Member Data Documentation

BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection [private]

Definition at line 71 of file BatchProcessing.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker `[private]`

Definition at line 76 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorConstructionMonitor.__headerDict `[private]`

Definition at line 70 of file BatchProcessing.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests `[private]`

Definition at line 73 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorConstructionMonitor.__numBatches `[private]`

Definition at line 67 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict `[private]`

Definition at line 68 of file BatchProcessing.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls `[private]`

Definition at line 75 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorConstructionMonitor.__requestCount [private]

Definition at line 74 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorConstructionMonitor.__restDomain [private]

Definition at line 69 of file BatchProcessing.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorConstructionMonitor.dataframe

Definition at line 66 of file BatchProcessing.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorConstructionMonitor.valueObject

Definition at line 72 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector(), and BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector().

The documentation for this class was generated from the following file:

- BatchProcessing.py

BatchProcessing.BatchProcessorUtahRealEstate Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject)
- def FuncSelector (self)
- def BatchProcessingUtahRealestateCom (self, valueObject)

Public Attributes

- dataframevalueObject

Private Attributes

- __numBatches __parameterString
- __restDomain
- __headerDict

Detailed Description

Definition at line 172 of file BatchProcessing.py.

Constructor & Destructor Documentation

def BatchProcessing.BatchProcessorUtahRealEstate.__init__ (*self*, *RestDomain*, *NumBatches*, *ParameterString*, *HeaderDict*, *valueObject*)

```
The __init__ function is the constructor for a class. It is called when an object of
that class
is instantiated, and it sets up the attributes of that object. In this case, we are
setting up
the dataframe attribute to be None (which will be set later), and we are also setting
up some
other attributes which will help us make our API calls.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to pull from the api
ParameterString: Pass the parameters to the rest api
HeaderDict: Pass in the header information for the api call
valueObject: Create a dataframe from the json response

Returns:
The instance of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 174 of file BatchProcessing.py.

```
00174     def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict,
valueObject):
00175         """
00176     The __init__ function is the constructor for a class. It is called when an
object of that class
00177     is instantiated, and it sets up the attributes of that object. In this case,
we are setting up
00178     the dataframe attribute to be None (which will be set later), and we are also
setting up some
00179     other attributes which will help us make our API calls.
```

```

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to pull from the api
        ParameterString: Pass the parameters to the rest api
        HeaderDict: Pass in the header information for the api call
        valueObject: Create a dataframe from the json response

    Returns:
        The instance of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterString = ParameterString
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.valueObject = valueObject

```

References BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__numBatches, BatchProcessing.BatchProcessorUtahRealEstate.__numBatches, BatchProcessing.BatchProcessorUtahRealEstate.__parameterString, Core.UtahRealEstateMain.__parameterString, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, BatchProcessing.BatchProcessorConstructionMonitor.valueObject, and BatchProcessing.BatchProcessorUtahRealEstate.valueObject.

Member Function Documentation

def BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom (*self*, *valueObject*)

```
The BatchProcessingUtahRealestateCom function is a function that takes in the
valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary
information for this
function to work properly. The BatchProcessingUtahRealestateCom function will then use
requests to get data from
UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df
(which is local). This
process will be repeated until all the data has been collected from UtahRealestate.com's
ReST API, at which point __df will contain all

Args:
self: Represent the instance of the class
valueObject: Pass the value of a progress bar to the function

Returns:
A dataframe of the scraped data

Doc Author:
```

```
Willem van der Schans, Trelent AI
```

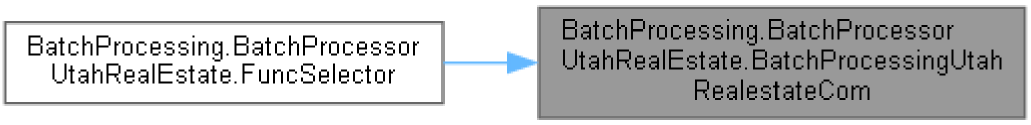
Definition at line 219 of file BatchProcessing.py.

```
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221     The BatchProcessingUtahRealestateCom function is a function that takes in
the valueObject and uses it to
00222        update the progress bar. It also takes in self, which contains all the
necessary information for this
00223        function to work properly. The BatchProcessingUtahRealestateCom function
will then use requests to get data from
00224        UtahRealestate.com's ReST API and store it into a pandas DataFrame object
called __df (which is local). This
00225        process will be repeated until all the data has been collected from
UtahRealestate.com's ReST API, at which point __df will contain all
00226
00227     Args:
00228         self: Represent the instance of the class
00229         valueObject: Pass the value of a progress bar to the function
00230
00231     Returns:
00232         A dataframe of the scraped data
00233
00234     Doc Author:
00235         Willem van der Schans, Trelent AI
00236     """
00237         __df = pd.DataFrame()
00238
00239         for batch in range(self.__numBatches):
00240
00241             if batch == 0:
00242                 response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00243                                         headers=self.__headerDict)
00244
00245                 response_temp = response.json()
00246                 __df = pd.json_normalize(response_temp, record_path=['value'])
00247
00248             else:
00249                 response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
200}",
00250                                         headers=self.__headerDict)
00251
00252                 response_temp = response.json()
00253                 response_temp = pd.json_normalize(response_temp,
record_path=['value'])
00254                 __df = pd.concat([__df, response_temp], ignore_index=True)
00255
00256             valueObject.setValue(valueObject.getValue() + 1)
00257
00258         self.dataframe = __df
00259         valueObject.setValue(-999)
```

References BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__numBatches, BatchProcessing.BatchProcessorUtahRealEstate.__numBatches, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, and Core.UtahRealEstateMain.dataframe.

Referenced by BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector().

Here is the caller graph for this function:


BatchProcessing.BatchProcessor UtahRealEstate.FuncSelector
BatchProcessing.BatchProcessor UtahRealEstate.BatchProcessingUtah RealestateCom

def BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector (*self*)

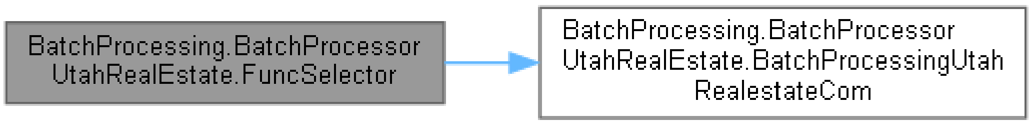
```
The FuncSelector function is a function that takes the valueObject as an argument and
then calls the appropriate
function based on what was selected in the dropdown menu.  The valueObject is passed
to each of these functions
so that they can access all of its attributes.

Args:
self: Represent the instance of the class

Returns:
The function that is selected by the user

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 202 of file BatchProcessing.py.

```
    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject as an
argument and then calls the appropriate
        function based on what was selected in the dropdown menu.  The
valueObject is passed to each of these functions
        so that they can access all of its attributes.

    Args:
        self: Represent the instance of the class

    Returns:
        The function that is selected by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

```

References BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.valueObject, and BatchProcessing.BatchProcessorUtahRealEstate.valueObject.

Here is the call graph for this function:


BatchProcessing.BatchProcessor UtahRealEstate.FuncSelector
BatchProcessing.BatchProcessor UtahRealEstate.BatchProcessingUtah RealestateCom

Member Data Documentation

BatchProcessing.BatchProcessorUtahRealEstate.__headerDict [private]

Definition at line 199 of file BatchProcessing.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorUtahRealEstate.__numBatches [private]

Definition at line 196 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), and BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor().

BatchProcessing.BatchProcessorUtahRealEstate.__parameterString [private]

Definition at line 197 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.UtahRealEstateMain.__init__(), Core.UtahRealEstateMain.__ParameterCreator(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorUtahRealEstate.__restDomain [private]

Definition at line 198 of file BatchProcessing.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorUtahRealEstate.dataframe

Definition at line 195 of file BatchProcessing.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProcessing.BatchProcessorUtahRealEstate.valueObject

Definition at line 200 of file BatchProcessing.py.

Referenced by BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector(), and BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector().

The documentation for this class was generated from the following file:

- BatchProcessing.py

BatchProgressGUI.BatchProgressGUI Class Reference

Public Member Functions

- def __init__ (self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None)
- def BatchGuiShow (self)
- def CreateProgressLayout (self)
- def createGui (self, Sourcetype)
- def ProgressUpdater (self, valueObj)
- def TimeUpdater (self, start_time)
- def ValueChecker (self, ObjectVal)

Public Attributes

dataframePrivate Attributes

- __parameterDict__restDomain
- __headerDict
- __columnSelection
- __type
- __layout
- __batches
- __window
- __batch_counter

Detailed Description

Definition at line 17 of file BatchProgressGUI.py.

Constructor & Destructor Documentation

def BatchProgressGUI.BatchProgressGUI.__init__ (***self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection*** **=** `None`**)**

```
The __init__ function is the first function that gets called when an object of this
class is created.
It initializes all the variables and sets up a layout for the GUI. It also creates a
window to display
the dataframe in.

Args:
self: Represent the instance of the class
BatchesNum: Determine the number of batches that will be created
RestDomain: Specify the domain of the rest api
ParameterDict: Pass the parameters of the request to the class
HeaderDict: Store the headers of the dataframe
Type: Determine the type of dataframe that is being created
ColumnSelection: Select the columns to be displayed in the gui

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 19 of file BatchProgressGUI.py.

```
    def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):

        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
    the dataframe in.

    Args:
        self: Represent the instance of the class
        BatchesNum: Determine the number of batches that will be created
        RestDomain: Specify the domain of the rest api
        ParameterDict: Pass the parameters of the request to the class
        HeaderDict: Store the headers of the dataframe
        Type: Determine the type of dataframe that is being created
        ColumnSelection: Select the columns to be displayed in the gui

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.__type = Type
        self.dataframe = None

        self.__layout = None
        self.__batches = BatchesNum
        self.__window = None
        self.__batch_counter = 0

```

References BatchProgressGUI.BatchProgressGUI.__batch_counter, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProgressGUI.BatchProgressGUI.__layout, PopupWrapped.PopupWrapped.__layout, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProgressGUI.BatchProgressGUI.__type, PopupWrapped.PopupWrapped.__type, BatchProgressGUI.BatchProgressGUI.__window, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, and Core.UtahRealEstateMain.dataframe.

Member Function Documentation

def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (*self*)

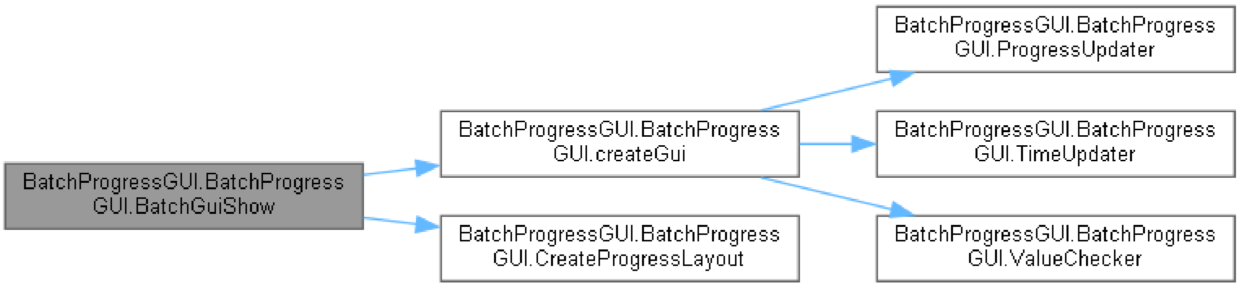
```
The BatchGuiShow function is called by the BatchGui function. It creates a progress
bar layout and then calls the createGui function to create a GUI for batch processing.

Args:
self: Represent the instance of the class

Returns:
The __type of the batchgui class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 53 of file BatchProgressGUI.py.

```
00053     def BatchGuiShow(self):
00054         """
00055     The BatchGuiShow function is called by the BatchGui function. It creates a
progress bar layout and then calls the createGui function to create a GUI for batch
processing.
00056
00057     Args:
00058         self: Represent the instance of the class
00059
00060     Returns:
00061         The __type of the batchgui class
00062
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
00066         self.CreateProgressLayout()
00067         self.createGui(self.__type)
00068
```

References BatchProgressGUI.BatchProgressGUI.__type, PopupWrapped.PopupWrapped.__type, BatchProgressGUI.BatchProgressGUI.createGui(), and BatchProgressGUI.BatchProgressGUI.CreateProgressLayout().

Here is the call graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.CreateProgressLayout
BatchProgressGUI.BatchProgressGUI.ProgressUpdater
BatchProgressGUI.BatchProgressGUI.TimeUpdater
BatchProgressGUI.BatchProgressGUI.ValueChecker

def BatchProgressGUI.BatchProgressGUI.createGui (*self*, *Sourcetype*)

```
The createGui function is the main function that creates the GUI.
It takes in a type parameter which determines what kind of batch processor to use.
The createGui function then sets up all the variables and objects needed for
the program to run, including: window, start_time, update_text, valueObj
(DataTransfer),
processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
and threading objects for TimeUpdater and ValueChecker functions. The createGui
function also starts these threads.

Args:
self: Access the object itself
Sourcetype: Determine which batch processor to use

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

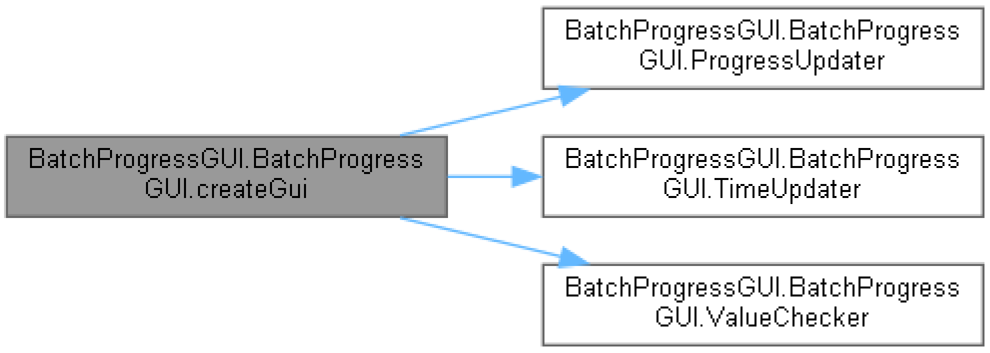
Definition at line 104 of file BatchProgressGUI.py.

```
    def createGui(self, Sourcetype):

        """
    The createGui function is the main function that creates the GUI.
    It takes in a type parameter which determines what kind of batch processor to use.
    The createGui function then sets up all the variables and objects needed for
    the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
    processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
    and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts these threads.

    Args:
        self: Access the object itself
        Sourcetype: Determine which batch processor to use

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__window = sg.Window('Progress', self.__layout, finalize=True, icon=ImageLoader("taskbar_icon.ico"))

        start_time = datetime.datetime.now().replace(microsecond=0)
        update_text = f"Batch {0} completed"
        self.__window['--progress_text--'].update(update_text)
        self.__window['--progress_bar--'].update(0)
        self.__window['--time_est--'].update("Est time needed 00:00:00")

        valueObj = DataTransfer()
        valueObj.setValue(0)

        if Sourcetype == "construction_monitor":

            processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
NumBatches=self.__batches,
ParameterDict=self.__parameterDict,
HeaderDict=self.__headerDict,
ColumnSelection=self.__columnSelection,
valueObject=valueObj)
        elif Sourcetype == "utah_real_estate":
            processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
NumBatches=self.__batches,
ParameterString=self.__parameterDict,
HeaderDict=self.__headerDict,
valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

```

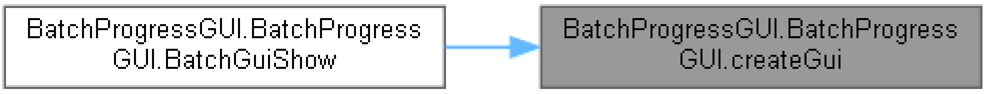
References BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProgressGUI.BatchProgressGUI.__layout, PopupWrapped.PopupWrapped.__layout, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProgressGUI.BatchProgressGUI.__window, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, BatchProgressGUI.BatchProgressGUI.ProgressUpdater(), BatchProgressGUI.BatchProgressGUI.TimeUpdater(), and BatchProgressGUI.BatchProgressGUI.ValueChecker().

Referenced by BatchProgressGUI.BatchProgressGUI.BatchGuiShow().

Here is the call graph for this function:


BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.ProgressUpdater
BatchProgressGUI.BatchProgressGUI.TimeUpdater
BatchProgressGUI.BatchProgressGUI.ValueChecker

Here is the caller graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui

def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (*self*)

```
The CreateProgressLayout function creates the layout for the progress window.
The function takes in self as a parameter and returns nothing.

Parameters:
    self (object): The object that is calling this function.

Args:
self: Access the class variables and methods

Returns:
A list of lists

Doc Author:
Willem van der Schans, Trelent AI
```

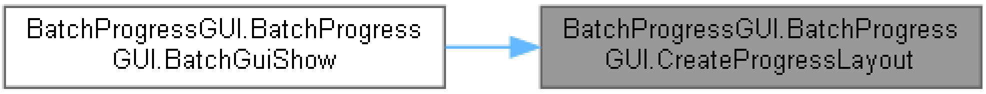
Definition at line 69 of file BatchProgressGUI.py.

```
    def CreateProgressLayout(self):

        """
    The CreateProgressLayout function creates the layout for the progress window.
        The function takes in self as a parameter and returns nothing.

        Parameters:
            self (object): The object that is calling this function.

    Args:
        self: Access the class variables and methods

    Returns:
        A list of lists

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
                   sg.Push()]

        __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
                   sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]

        __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
                           key='--progress_bar--')]


        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

```

References BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, BatchProgressGUI.BatchProgressGUI.__layout, and PopupWrapped.PopupWrapped.__layout.

Referenced by BatchProgressGUI.BatchProgressGUI.BatchGuiShow().

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.CreateProgressLayout

def BatchProgressGUI.BatchProgressGUI.ProgressUpdater (*self*, *valueObj*)

```
The ProgressUpdater function is a callback function that updates the progress bar and
text
in the GUI. It takes in one argument, which is an object containing information about
the
current batch number. The ProgressUpdater function then checks if this value has changed
from
the last time it was called (i.e., if we are on a new batch). If so, it updates both
the progress
bar and text with this new information.

Args:
self: Make the progressupdater function an instance method
valueObj: Get the current value of the batch counter

Returns:
The value of the batch counter

Doc Author:
Willem van der Schans, Trelent AI
```

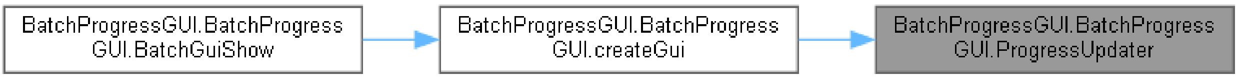
Definition at line 188 of file BatchProgressGUI.py.

```
    def ProgressUpdater(self, valueObj):
        """
    The ProgressUpdater function is a callback function that updates the progress
bar and text
    in the GUI. It takes in one argument, which is an object containing information
about the
    current batch number. The ProgressUpdater function then checks if this value
has changed from
    the last time it was called (i.e., if we are on a new batch). If so, it updates
both the progress
    bar and text with this new information.

    Args:
        self: Make the progressupdater function an instance method
        valueObj: Get the current value of the batch counter

    Returns:
        The value of the batch counter

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if valueObj.getValue() != self.__batch_counter:
            self.__batch_counter = valueObj.getValue()

            __update_text = f"Batch {self.__batch_counter}/{self.__batches}
completed"

            self.__window.write_event_value('update--progress_bar--',
self.__batch_counter)
            self.__window.write_event_value('update--progress_text--',
__update_text)
        else:
            pass

```

References BatchProgressGUI.BatchProgressGUI.__batch_counter, and BatchProgressGUI.BatchProgressGUI.__window.

Referenced by BatchProgressGUI.BatchProgressGUI.createGui().

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ProgressUpdater

def BatchProgressGUI.BatchProgressGUI.TimeUpdater (*self*, *start_time*)

```
The TimeUpdater function is a thread that updates the time elapsed and estimated time
needed to complete
the current batch. It does this by reading the start_time variable passed in, getting
the current time,
calculating how much time has passed since start_time was set and then updating a timer
string with that value.
It then calculates an estimation of how long it will take to finish all batches based
on how many batches have been completed so far.

Args:
self: Make the function a method of the class
start_time: Get the time when the function is called

Returns:
A string that is updated every 0

Doc Author:
Willem van der Schans, Trelent AI
```

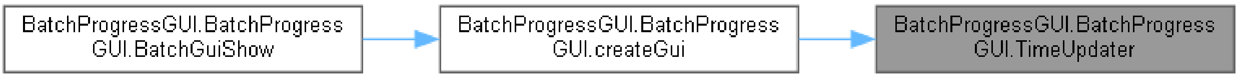
Definition at line 216 of file BatchProgressGUI.py.

```
00216     def TimeUpdater(self, start_time):
00217 
00218         """
00219     The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to complete
00220     the current batch. It does this by reading the start_time variable passed in, getting the current time,
00221     calculating how much time has passed since start_time was set and then updating a timer string with that value.
00222     It then calculates an estimation of how long it will take to finish all batches based on how many batches have been completed so far.
00223 
00224     Args:
00225         self: Make the function a method of the class
00226         start_time: Get the time when the function is called
00227 
00228     Returns:
00229         A string that is updated every 0
00230 
00231     Doc Author:
00232         Willem van der Schans, Trelent AI
00233     """
00234         while True:
00235             if self.__batch_counter < self.__batches:
00236 
00237                 __current_time = datetime.datetime.now().replace(microsecond=0)
00238 
00239                 __passed_time = __current_time - start_time
00240 
00241                 __timer_string = f"Time Elapsed {__passed_time}"
00242 
00243                 try:
00244                     self.__window.write_event_value('update--timer--', __timer_string)
00245                 except AttributeError as e:
00246                     print(
00247                         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error, this is okay if the display looks good, this exception omits fatal crashes due to an aesthetic error")
00248                     break
00249 
00250                 __passed_time = __passed_time.total_seconds()
00251 
00252                 try:
00253                     __time_est = datetime.timedelta(
00254                         seconds=(__passed_time * (self.__batches / self.__batch_counter) - __passed_time)).seconds
00255                 except:
00256                     __time_est = datetime.timedelta(
00257                              seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00258
00259                      __time_est = time.strftime('%H:%M:%S',
time.gmtime(__time_est))
00260
00261                      __end_string = f"Est time needed {__time_est}"
00262                      self.__window.write_event_value('update--time_est--',
__end_string)
00263                  else:
00264                      __end_string = f"Est time needed 00:00:00"
00265                      self.__window.write_event_value('update--time_est--',
__end_string)
00266                  time.sleep(0.25)
00267
```

References BatchProgressGUI.BatchProgressGUI.__batch_counter, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, and BatchProgressGUI.BatchProgressGUI.__window.

Referenced by BatchProgressGUI.BatchProgressGUI.createGui().

Here is the caller graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.TimeUpdater

def BatchProgressGUI.BatchProgressGUI.ValueChecker (*self*, *ObjectVal*)

```
The ValueChecker function is a thread that checks the value of an object.
It will check if the value has changed, and if it has, it will return True.
If not, then it returns False.

Args:
self: Represent the instance of the class
ObjectVal: Get the value of the object

Returns:
True if the value of the object has changed, and false if it hasn't

Doc Author:
Willem van der Schans, Trelent AI
```

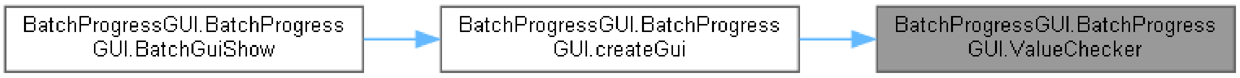
Definition at line 268 of file BatchProgressGUI.py.

```
00268      def ValueChecker(self, ObjectVal):
00269          """
00270      The ValueChecker function is a thread that checks the value of an object.
00271          It will check if the value has changed, and if it has, it will return
True.
00272          If not, then it returns False.
00273
00274      Args:
00275          self: Represent the instance of the class
00276          ObjectVal: Get the value of the object
00277
00278      Returns:
00279          True if the value of the object has changed, and false if it hasn't
00280
00281      Doc Author:
00282          Willem van der Schans, Trelent AI
00283      """
00284          while True:
00285              time.sleep(0.3)
00286              if self.__batch_counter != ObjectVal.getValue():
00287                  self.__batch_counter = ObjectVal.getValue()
00288                  return True
00289              else:
00290                  return False
```

References BatchProgressGUI.BatchProgressGUI.__batch_counter.

Referenced by BatchProgressGUI.BatchProgressGUI.createGui().

Here is the caller graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.ValueChecker

Member Data Documentation

BatchProgressGUI.BatchProgressGUI.__batch_counter `[private]`

Definition at line 51 of file BatchProgressGUI.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProgressGUI.BatchProgressGUI.ProgressUpdater(), BatchProgressGUI.BatchProgressGUI.TimeUpdater(), and BatchProgressGUI.BatchProgressGUI.ValueChecker().

BatchProgressGUI.BatchProgressGUI.__batches `[private]`

Definition at line 49 of file BatchProgressGUI.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), BatchProgressGUI.BatchProgressGUI.CreateProgressLayout(), Core.ConstructionMonitorMain.mainFunc(), Core.UtahRealEstateMain.mainFunc(), and BatchProgressGUI.BatchProgressGUI.TimeUpdater().

BatchProgressGUI.BatchProgressGUI.__columnSelection `[private]`

Definition at line 44 of file BatchProgressGUI.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

BatchProgressGUI.BatchProgressGUI.__headerDict `[private]`

Definition at line 43 of file BatchProgressGUI.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProgressGUI.BatchProgressGUI.__layout `[private]`

Definition at line 48 of file BatchProgressGUI.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), PopupWrapped.PopupWrapped.__createWindow(), BatchProgressGUI.BatchProgressGUI.__init__(), PopupWrapped.PopupWrapped.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), and BatchProgressGUI.BatchProgressGUI.CreateProgressLayout().

BatchProgressGUI.BatchProgressGUI.__parameterDict [private]

Definition at line 41 of file BatchProgressGUI.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

BatchProgressGUI.BatchProgressGUI.__restDomain [private]

Definition at line 42 of file BatchProgressGUI.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

BatchProgressGUI.BatchProgressGUI.__type [private]

Definition at line 45 of file BatchProgressGUI.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), PopupWrapped.PopupWrapped.__createWindow(), BatchProgressGUI.BatchProgressGUI.__init__(), PopupWrapped.PopupWrapped.__init__(), BatchProgressGUI.BatchProgressGUI.BatchGuiShow(), and PopupWrapped.PopupWrapped.textUpdate().

BatchProgressGUI.BatchProgressGUI.__window [private]

Definition at line 50 of file BatchProgressGUI.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), BatchProgressGUI.BatchProgressGUI.ProgressUpdater(), and BatchProgressGUI.BatchProgressGUI.TimeUpdater().

BatchProgressGUI.BatchProgressGUI.dataframe

Definition at line 46 of file BatchProgressGUI.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

The documentation for this class was generated from the following file:

- BatchProgressGUI.py

Core.CFBP Class Reference

Public Member Functions

- def __init__ (self, state_arg=None, year_arg=None)

Public Attributes

- state_argyear_arg
- uiString
- link

Private Member Functions

- def __showUi (self)
- def __dataGetter (self)

Detailed Description

Definition at line 14 of file CFBP/Core.py.

Constructor & Destructor Documentation

def Core.CFBP.__init__ (*self*, *state_arg* = None, *year_arg* = None)

```
The __init__ function is called when the class is instantiated.
Its job is to initialize the object with some default values, and do any other setup
that might be necessary.
The __init__ function can take arguments, but it doesn't have to.

Args:
self: Represent the instance of the class
state_arg: Set the state_arg attribute of the class
year_arg: Set the year of data to be retrieved

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

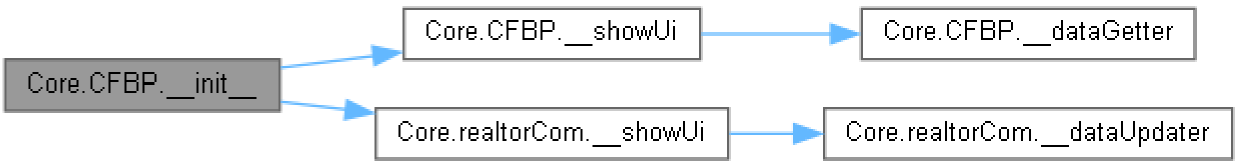
Definition at line 16 of file CFBP/Core.py.

```
    def __init__(self, state_arg=None, year_arg=None):
        """
    The __init__ function is called when the class is instantiated.
    Its job is to initialize the object with some default values, and do any other setup that might be necessary.
    The __init__ function can take arguments, but it doesn't have to.

    Args:
        self: Represent the instance of the class
        state_arg: Set the state_arg attribute of the class
        year_arg: Set the year of data to be retrieved

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
```

```
00035         self.uiString = None
00036         self.link = None
00037
00038         eventReturn = confirmDialog()
00039         if eventReturn == "Continue":
00040             startTime = datetime.datetime.now().replace(microsecond=0)
00041             self.__showUi()
00042             print(
00043                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | API Link = {self.link}")
00044             F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00045             print(
00046                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00047
00048             self.uiString = (
00049                 f"ffiec.cfpb.gov (Mortgage API) request Completed \n
{self.year_arg} data retrieved \n Data Saved at {F.getPath()}")
00050
00051             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00052         else:
00053             print(
00054                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | User Canceled Request")
00055             pass
00056
```

References Core.CFBP.__showUi(), Core.realtorCom.__showUi(), Core.CFBP.link, Core.CFBP.state_arg, Core.CFBP.uiString, Core.realtorCom.uiString, and Core.CFBP.year_arg.

Here is the call graph for this function:



Member Function Documentation

def Core.CFBP.__dataGetter (*self*) `[private]`

```
The __dataGetter function is a private function that gets the data from the CFPB API.
It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for
the API call.

Args:
self: Represent the instance of the class

Returns:
A response object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 85 of file CFBP/Core.py.

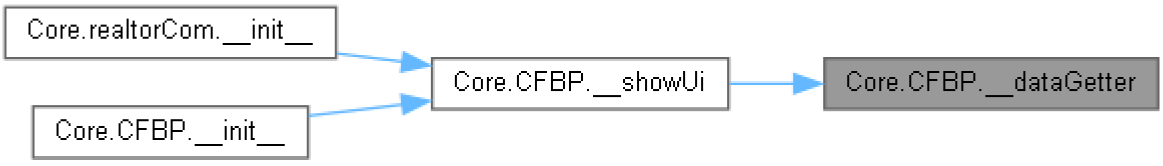
```
00085     def __dataGetter(self):
00086         """
00087     The __dataGetter function is a private function that gets the data from the
CFPB API.
00088     It takes no arguments, but uses self.state_arg and self.year_arg to create
a URL for the API call.
00089
00090     Args:
00091         self: Represent the instance of the class
00092
00093     Returns:
00094         A response object
```

```
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        arg_dict_bu = locals()

        link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link = 
"https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
+ f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

References Core.CFBP.link, Core.CFBP.state_arg, and Core.CFBP.year_arg.

Referenced by Core.CFBP.__showUi().

Here is the caller graph for this function:



def Core.CFBP.__showUi (*self*) [private]

```
The __showUi function is a function that creates a progress bar window.
The __showUi function takes class variables and returns a windowobj.


Args:
self: Represent the instance of the class

Returns:
The uiobj variable

Doc Author:
Willem van der Schans, Trelent AI
```

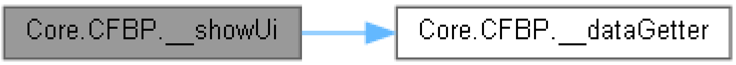
Definition at line 57 of file CFBP/Core.py.

```
    def __showUi(self):

        """
    The __showUi function is a function that creates a progress bar window.
    The __showUi function takes class variables and returns a windowobj.


    Args:
        self: Represent the instance of the class

    Returns:
        The uiobj variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Cenus Request running",
windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataGetter,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

```

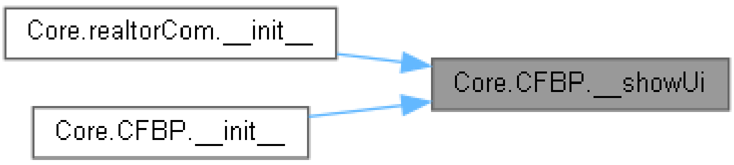
References Core.CFBP.__dataGetter().

Referenced by Core.realtorCom.__init__(), and Core.CFBP.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



Member Data Documentation

Core.CFBP.link

Definition at line 36 of file CFBP/Core.py.

Referenced by Core.CFBP.__dataGetter(), and Core.CFBP.__init__().

Core.CFBP.state_arg

Definition at line 33 of file CFBP/Core.py.

Referenced by Core.CFBP.__dataGetter(), and Core.CFBP.__init__().

Core.CFBP.uiString

Definition at line 35 of file CFBP/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), Core.realtorCom.__init__(), and Core.CFBP.__init__().

Core.CFBP.year_arg

Definition at line 34 of file CFBP/Core.py.

Referenced by Core.CFBP.__dataGetter(), and Core.CFBP.__init__().

The documentation for this class was generated from the following file:

- CFBP/Core.py

Core.ConstructionMonitorInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- sizeSourceInclude
- dateStart
- dateEnd
- rest_domain
- auth_key
- ui_flag
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

Detailed Description

Definition at line 24 of file ConstructionMonitor/Core.py.

Constructor & Destructor Documentation

def Core.ConstructionMonitorInit.__init__ (*self*)

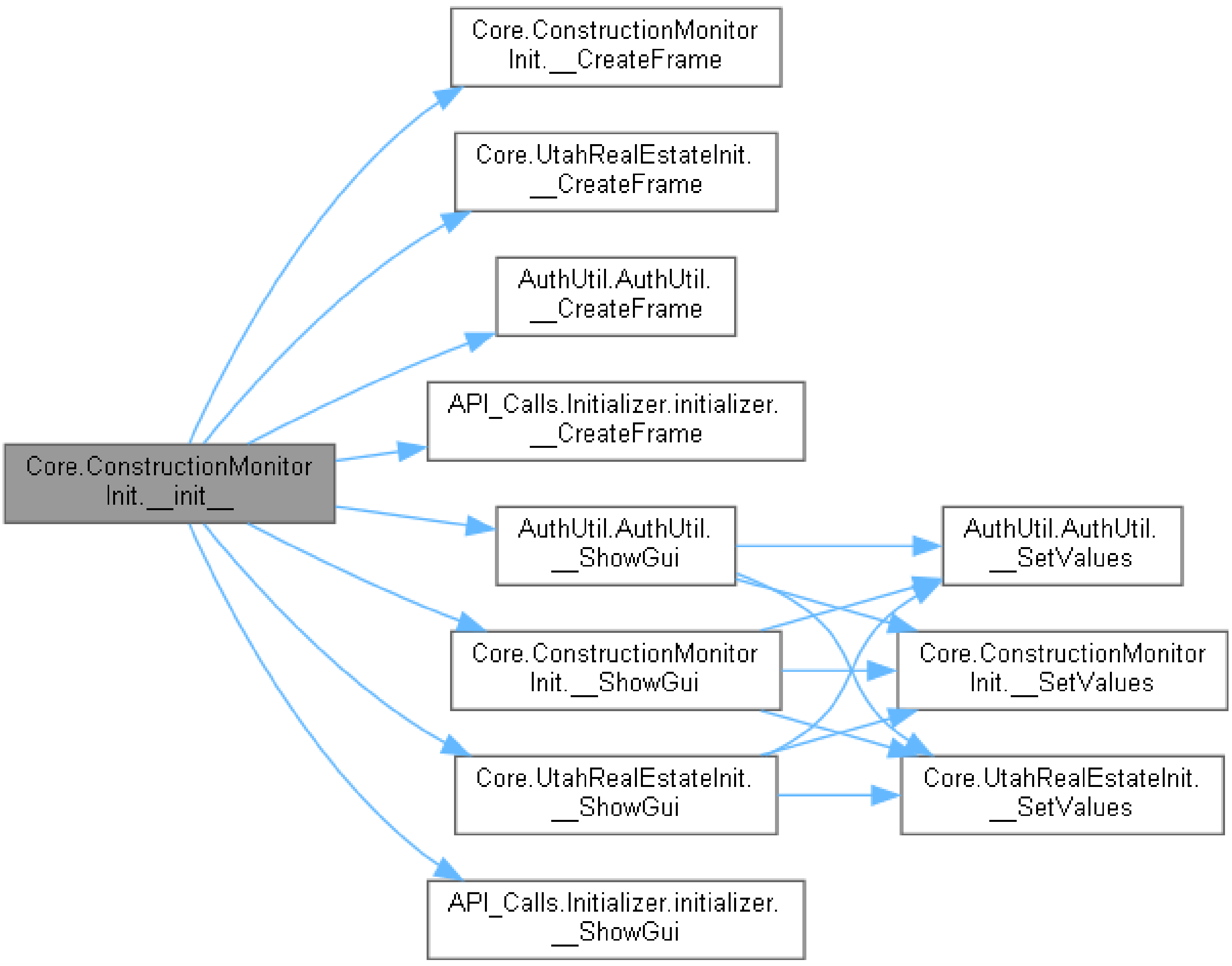
```
The __init__ function is called when the class is instantiated.
It sets up the variables that will be used by other functions in this class.


Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 26 of file ConstructionMonitor/Core.py.

```
    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the variables that will be used by other functions in this class.


    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = None
        self.SourceInclude = None
        self.dateStart = None
        self.dateEnd = None
        self.rest_domain = None
        self.auth_key = None
        self.ui_flag = None
        self.append_file = None

        passFlag = False

        while not passFlag:
            if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                    "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
                    Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                    "Security").joinpath("auth.json")):
                try:
                    f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                    key = f.readline()
                    f.close()
                    f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json"), "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
                    passFlag = True
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ConstructionMonitor/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")

```

References Core.ConstructionMonitorInit.__CreateFrame(), Core.UtahRealEstateInit.__CreateFrame(), AuthUtil.AuthUtil.__CreateFrame(), API_Calls.Initializer.initializer.__CreateFrame(), AuthUtil.AuthUtil.__ShowGui(), API_Calls.Initializer.initializer.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), Core.UtahRealEstateInit.__ShowGui(), AuthUtil.AuthUtil.append_file, Core.ConstructionMonitorInit.append_file, Core.UtahRealEstateInit.append_file, Core.ConstructionMonitorInit.auth_key, Core.ConstructionMonitorInit.dateEnd, Core.UtahRealEstateInit.dateEnd, Core.ConstructionMonitorInit.dateStart, Core.UtahRealEstateInit.dateStart, Core.ConstructionMonitorInit.rest_domain, Core.ConstructionMonitorInit.size, Core.ConstructionMonitorInit.SourceInclude, and Core.ConstructionMonitorInit.ui_flag.

Here is the call graph for this function:


Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
Core.ConstructionMonitor Init.__init__
AuthUtil.AuthUtil. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__ShowGui
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues
API_Calls.Initializer.initializer. __ShowGui

Member Function Documentation

def Core.ConstructionMonitorInit.__CreateFrame () `[static]`, `[private]`

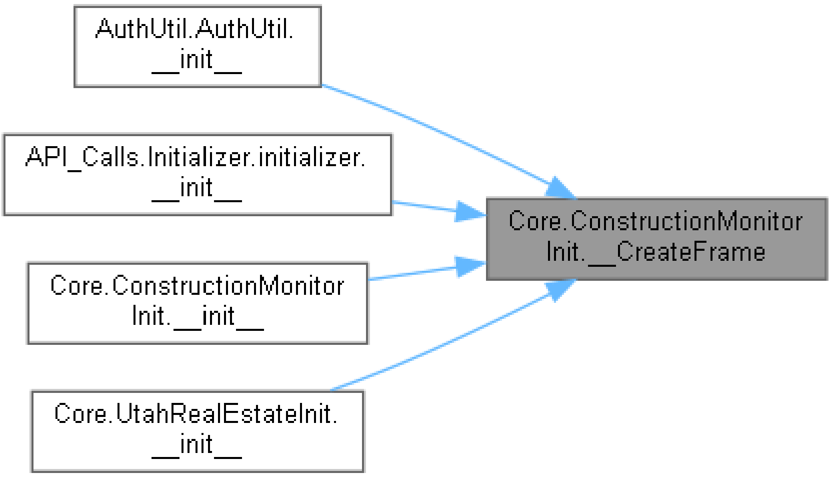
```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the GUI window.
This is done by creating each line as a list and then appending it to another list which
will contain all lines.

Args:

Returns:
The layout for the gui

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 116 of file ConstructionMonitor/Core.py.

```
00116     def __CreateFrame():
00117
00118         """
00119     The __CreateFrame function creates the GUI layout for the application.
00120         The function returns a list of lists that contains all the elements to
be displayed in the GUI window.
00121         This is done by creating each line as a list and then appending it to
another list which will contain all lines.
00122
00123     Args:

    Returns:
        The layout for the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

```

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the caller graph for this function:



def Core.ConstructionMonitorInit.__SetValues (***self*****,** ***values*****)** `[private]`

```
The __SetValues function is used to set the values of the variables that are used in
the __GetData function.
The __SetValues function takes a dictionary as an argument, and then sets each variable
based on what is passed into
the dictionary. The keys for this dictionary are defined by the user when they create
their own instance of this class.

Args:
self: Represent the instance of the class
values: Pass in the values from the ui

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 175 of file ConstructionMonitor/Core.py.

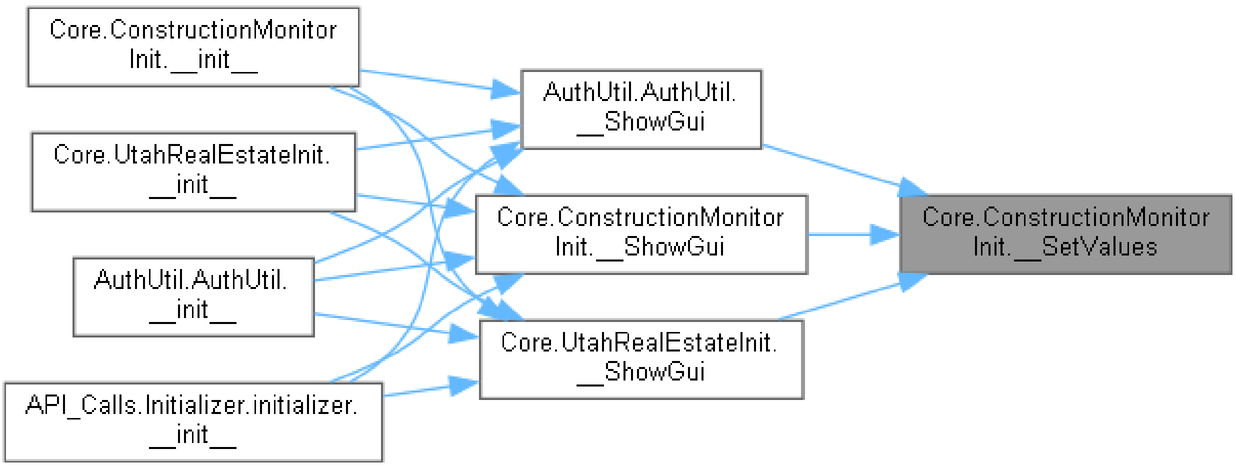
```
00175     def __SetValues(self, values):
00176
00177         """
00178     The __SetValues function is used to set the values of the variables that are
used in the __GetData function.
00179     The __SetValues function takes a dictionary as an argument, and then sets
each variable based on what is passed into
00180     the dictionary. The keys for this dictionary are defined by the user when
they create their own instance of this class.
00181
00182     Args:
00183         self: Represent the instance of the class
00184         values: Pass in the values from the ui
00185
00186     Returns:
00187         A dictionary of values
00188
00189     Doc Author:
00190         Willem van der Schans, Trelent AI
00191     """
00192         self.size = 1000
00193
00194         if values["-Cal-"] != "":
00195             self.dateStart = values["-Cal-"]
00196         else:
00197             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00198
00199         if values["-EndCal-"] != "":
00200             self.dateEnd = values["-EndCal-"]
00201         else:
00202             self.dateEnd = date.today().strftime("%Y-%m-%d")
```

```

        self.rest_domain =
"https://api.constructionmonitor.com/v2/powersearch/?"

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True

```

References AuthUtil.AuthUtil.append_file, Core.ConstructionMonitorInit.append_file, Core.UtahRealEstateInit.append_file, Core.ConstructionMonitorInit.dateEnd, Core.UtahRealEstateInit.dateEnd, Core.ConstructionMonitorInit.dateStart, Core.UtahRealEstateInit.dateStart, Core.ConstructionMonitorInit.rest_domain, Core.ConstructionMonitorInit.size, Core.ConstructionMonitorInit.SourceInclude, and Core.ConstructionMonitorInit.ui_flag.

Referenced by AuthUtil.AuthUtil.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), and Core.UtahRealEstateInit.__ShowGui().

Here is the caller graph for this function:



Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
Core.ConstructionMonitor Init.__SetValues

def Core.ConstructionMonitorInit.__ShowGui (*self*, *layout*, *text*) [private]

```
The __ShowGui function is the main function that creates and displays the GUI.
It takes in a layout, which is a list of lists containing all the elements to be displayed
on screen.
The text parameter specifies what title should appear at the top of the window.

Args:
self: Refer to the current instance of a class
layout: Determine what the gui will look like
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 77 of file ConstructionMonitor/Core.py.

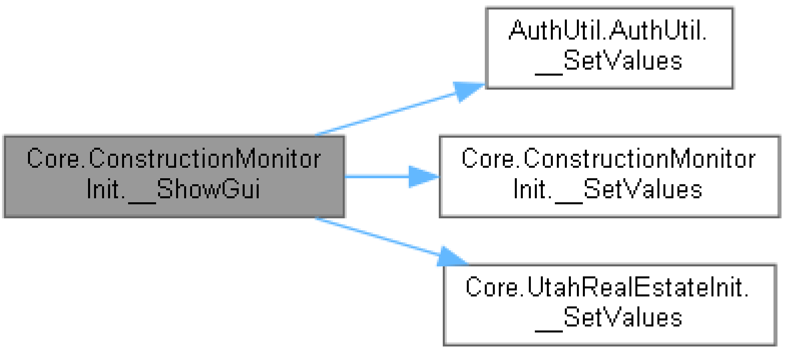
```
    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that creates and displays the
GUI.
    It takes in a layout, which is a list of lists containing all the elements
to be displayed on screen.
```

```
    The text parameter specifies what title should appear at the top of the window.

    Args:
        self: Refer to the current instance of a class
        layout: Determine what the gui will look like
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

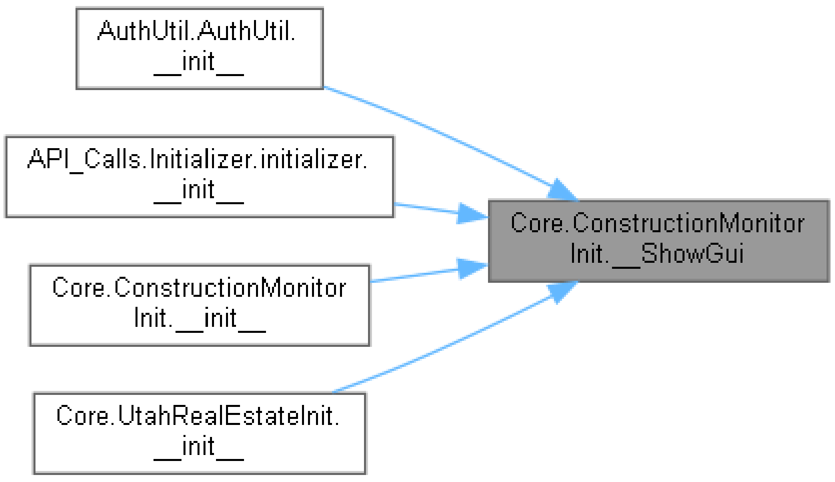
References AuthUtil.AuthUtil.__SetValues(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the call graph for this function:


Core.ConstructionMonitor Init.__ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __SetValues

Here is the caller graph for this function:



Member Data Documentation

Core.ConstructionMonitorInit.append_file

Definition at line 49 of file ConstructionMonitor/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.ConstructionMonitorInit.auth_key

Definition at line 47 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__().

Core.ConstructionMonitorInit.dateEnd

Definition at line 45 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.ConstructionMonitorInit.dateStart

Definition at line 44 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.ConstructionMonitorInit.rest_domain

Definition at line 46 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), and Core.ConstructionMonitorInit.__SetValues().

Core.ConstructionMonitorInit.size

Definition at line 42 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), and Core.ConstructionMonitorInit.__SetValues().

Core.ConstructionMonitorInit.SourceInclude

Definition at line 43 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), and Core.ConstructionMonitorInit.__SetValues().

Core.ConstructionMonitorInit.ui_flag

Definition at line 48 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), and Core.ConstructionMonitorInit.__SetValues().

The documentation for this class was generated from the following file:

- ConstructionMonitor/Core.py

Core.ConstructionMonitorMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

dataframePrivate Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __siteClass__restDomain
- __headerDict
- __columnSelection
- __appendFile
- __parameterDict
- __search_id
- __record_val
- __batches
- __ui_flag

Detailed Description

Definition at line 216 of file ConstructionMonitor/Core.py.

Constructor & Destructor Documentation

def Core.ConstructionMonitorMain.__init__ (*self*, *siteClass*)

```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to run properly.


Args:
self: Represent the instance of the class
siteClass: Identify the site that is being used

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

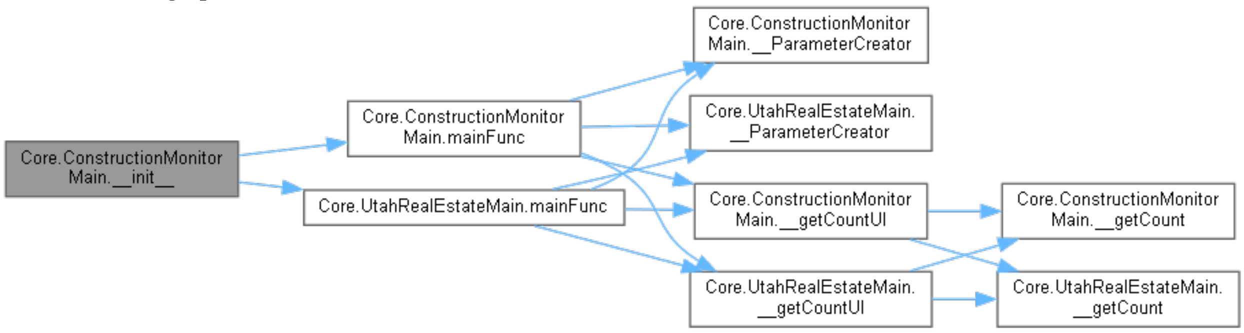
Definition at line 218 of file ConstructionMonitor/Core.py.

```
    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this
class is created.
    It sets up all the variables and functions needed for this class to run
properly.
```

```


    Args:
        self: Represent the instance of the class
        siteClass: Identify the site that is being used

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```

References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, Core.ConstructionMonitorMain.__record_val, Core.UtahRealEstateMain.__record_val, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, Core.ConstructionMonitorMain.__search_id, Core.ConstructionMonitorMain.__siteClass, Core.UtahRealEstateMain.__siteClass, Core.ConstructionMonitorMain.__ui_flag, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the call graph for this function:


Core.ConstructionMonitor Main.__init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__ParameterCreator
Core.UtahRealEstateMain. __ParameterCreator
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

Member Function Documentation

def Core.ConstructionMonitorMain.__getCount (*self*) [private]

```
The __getCount function is used to get the total number of records that are returned
from a query.
This function is called by the __init__ function and sets the self.__record_val variable
with this value.

Args:
self: Represent the instance of the class

Returns:
The total number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 371 of file ConstructionMonitor/Core.py.

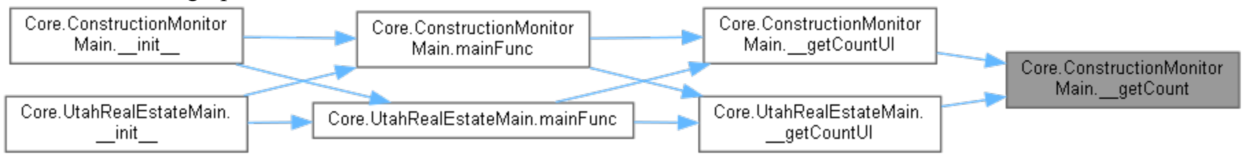
```
00371     def __getCount(self):
00372         """
00373     The __getCount function is used to get the total number of records that are
returned from a query.
00374     This function is called by the __init__ function and sets the
self.__record_val variable with this value.
00375
00376     Args:
00377         self: Represent the instance of the class
00378
00379     Returns:
00380         The total number of records in the database
00381
00382     Doc Author:
00383         Willem van der Schans, Trelent AI
00384     """
00385         __count_resp = None
00386
00387         try:
00388
00389             __temp_param_dict = copy.copy(self.__parameterDict)
```

```

                __count_resp = requests.post(url=self.__restDomain,
                                             headers=self.__headerDict,
                                             json=__temp_param_dict)

            except requests.exceptions.Timeout as e:
                print(e)
                RESTError(790)
                raise SystemExit(790)
            except requests.exceptions.TooManyRedirects as e:
                print(e)
                RESTError(791)
                raise SystemExit(791)
            except requests.exceptions.MissingSchema as e:
                print(e)
                RESTError(1101)
            except requests.exceptions.RequestException as e:
                print(e)
                RESTError(405)
                raise SystemExit(405)

            __count_resp = __count_resp.json()

            self.__record_val = __count_resp["hits"]["total"]["value"]

            del __count_resp, __temp_param_dict

```

References BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, Core.ConstructionMonitorMain.__record_val, Core.UtahRealEstateMain.__record_val, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, and Core.UtahRealEstateMain.__restDomain.

Referenced by Core.ConstructionMonitorMain.__getCountUI(), and Core.UtahRealEstateMain.__getCountUI().

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount

def Core.ConstructionMonitorMain.__getCountUI (*self*) `[private]`

```
The __getCountUI function is a wrapper for the __getCount function.
It allows the user to run __getCount in a separate thread, so that they can continue
working while it runs.
The function will display a progress bar and update with text as it progresses through
its tasks.

Args:
self: Access the class variables and methods

Returns:
The count of the number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

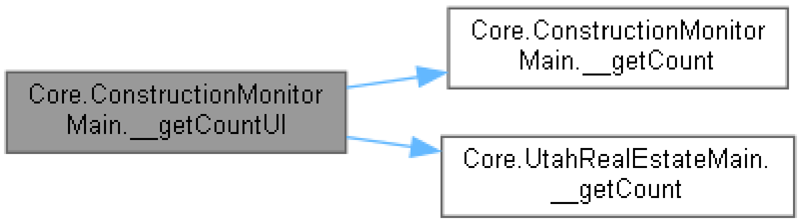
Definition at line 417 of file ConstructionMonitor/Core.py.

```
    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It allows the user to run __getCount in a separate thread, so that they can
continue working while it runs.
    The function will display a progress bar and update with text as it progresses
through its tasks.

    Args:
        self: Access the class variables and methods

    Returns:
        The count of the number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running",
windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

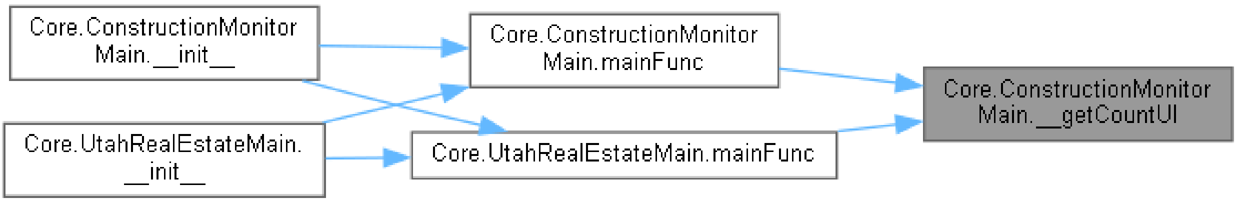
References Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), and Core.ConstructionMonitorMain.__ui_flag.

Referenced by Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the call graph for this function:


Core.ConstructionMonitor Main.__getCountUI
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI

def Core.ConstructionMonitorMain.__ParameterCreator (*self*) `[private]`

```
The __ParameterCreator function is used to create the parameter dictionary that will
be passed into the
__Request function. The function takes in a siteClass object and extracts all of its
attributes, except for
those that start with '__' or are callable. It then creates a dictionary from these
attributes and stores it as
self.__parameterDict.

Args:
self: Make the function a method of the class
```

```
Returns:
A dictionary of parameters and a list of non parameter variables

Doc Author:
Willem van der Schans, Trelent AI
```

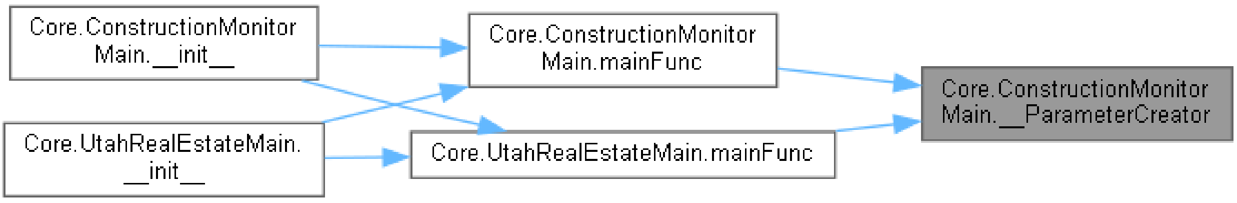
Definition at line 332 of file ConstructionMonitor/Core.py.

```
    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the parameter dictionary that will be passed into the
        __Request function. The function takes in a siteClass object and extracts all of its attributes, except for
        those that start with '__' or are callable. It then creates a dictionary from these attributes and stores it as
        self.__parameterDict.

    Args:
        self: Make the function a method of the class

    Returns:
        A dictionary of parameters and a list of non parameter variables

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                             not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

```

References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, Core.ConstructionMonitorMain.__siteClass, Core.UtahRealEstateMain.__siteClass, and Core.ConstructionMonitorMain.__ui_flag.

Referenced by Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__ParameterCreator

def Core.ConstructionMonitorMain.mainFunc (*self*)

```
The mainFunc function is the main function of this module. It will be called by the
GUI or CLI to execute
the code in this module. The mainFunc function will first create a parameter dictionary
using the __ParameterCreator
method, then it will get a count of all records that match its parameters using the
__getCountUI method, and then
it will calculate how many batches are needed to retrieve all records with those
parameters using BatchCalculator.
After that it asks if you want to continue with retrieving data from Salesforce (if
running in GUI mode). Then it shows
a progress bar for each

Args:
self: Refer to the current object

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

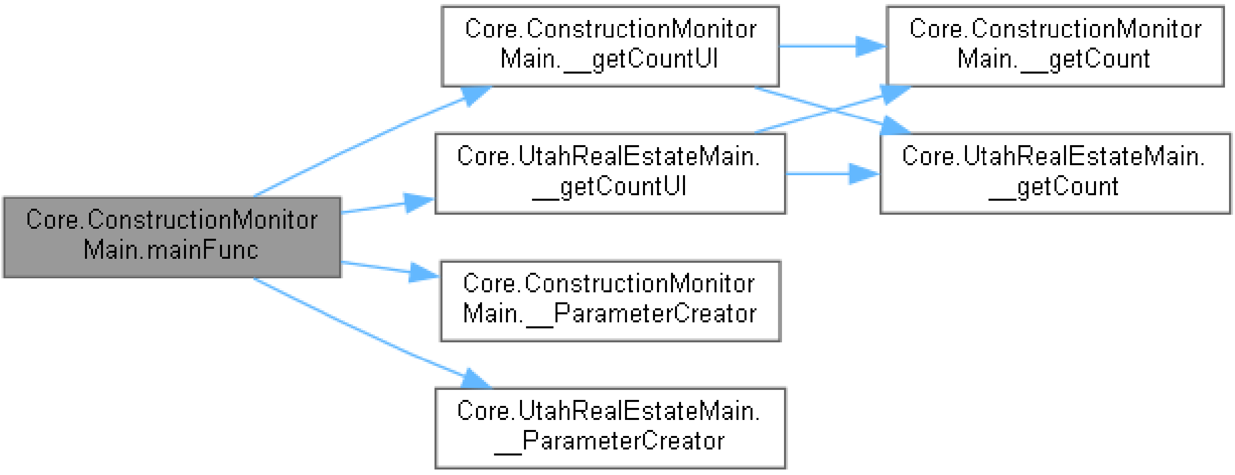
Definition at line 276 of file ConstructionMonitor/Core.py.

```
    def mainFunc(self):
        """
    The mainFunc function is the main function of this module. It will be called by the GUI or CLI to execute
    the code in this module. The mainFunc function will first create a parameter dictionary using the __ParameterCreator
    method, then it will get a count of all records that match its parameters using the __getCountUI method, and then
    it will calculate how many batches are needed to retrieve all records with those parameters using BatchCalculator.
    After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI mode). Then it shows
    a progress bar for each

    Args:
        self: Refer to the current object

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict = {self.__parameterDict}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain = {self.__restDomain}")

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterDict,
                                                  HeaderDict=self.__headerDict,
                                                  ColumnSelection=self.__columnSelection,
                                                  BatchesNum=self.__batches,
                                                  Type="construction_monitor")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
                FileSaver("cm", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

```

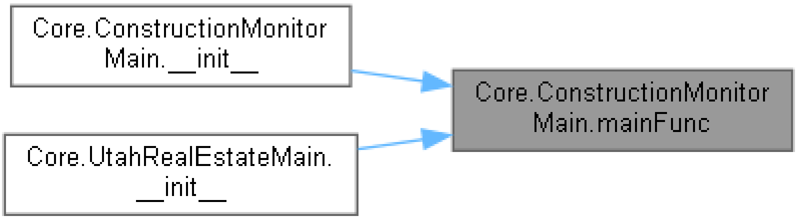
References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection, BatchProgressGUI.BatchProgressGUI.__columnSelection, Core.ConstructionMonitorMain.__columnSelection, Core.ConstructionMonitorMain.__getCountUI(), Core.UtahRealEstateMain.__getCountUI(), BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, Core.ConstructionMonitorMain.__ParameterCreator(), Core.UtahRealEstateMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict, BatchProgressGUI.BatchProgressGUI.__parameterDict, Core.ConstructionMonitorMain.__parameterDict, Core.ConstructionMonitorMain.__record_val, Core.UtahRealEstateMain.__record_val, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, and Core.UtahRealEstateMain.dataframe.

Referenced by Core.ConstructionMonitorMain.__init__(), and Core.UtahRealEstateMain.__init__().

Here is the call graph for this function:


Core.ConstructionMonitorMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Here is the caller graph for this function:


Core.ConstructionMonitorMain.__init__
Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc

Member Data Documentation

Core.ConstructionMonitorMain.__appendFile [private]

Definition at line 239 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), Core.UtahRealEstateMain.__ParameterCreator(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.ConstructionMonitorMain.__batches [private]

Definition at line 244 of file ConstructionMonitor/Core.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), BatchProgressGUI.BatchProgressGUI.CreateProgressLayout(), Core.ConstructionMonitorMain.mainFunc(), Core.UtahRealEstateMain.mainFunc(), and BatchProgressGUI.BatchProgressGUI.TimeUpdater().

Core.ConstructionMonitorMain.__columnSelection [private]

Definition at line 238 of file ConstructionMonitor/Core.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

Core.ConstructionMonitorMain.__headerDict [private]

Definition at line 237 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.ConstructionMonitorMain.__parameterDict [private]

Definition at line 241 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), Core.ConstructionMonitorMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), and Core.ConstructionMonitorMain.mainFunc().

Core.ConstructionMonitorMain.__record_val [private]

Definition at line 243 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.ConstructionMonitorMain.__restDomain [private]

Definition at line 236 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.ConstructionMonitorMain.__search_id [private]

Definition at line 242 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__init__().

Core.ConstructionMonitorMain.__siteClass [private]

Definition at line 235 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), and Core.UtahRealEstateMain.__ParameterCreator().

Core.ConstructionMonitorMain.__ui_flag [private]

Definition at line 246 of file ConstructionMonitor/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCountUI(), Core.ConstructionMonitorMain.__init__(), and Core.ConstructionMonitorMain.__ParameterCreator().

Core.ConstructionMonitorMain.dataframe

Definition at line 248 of file ConstructionMonitor/Core.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

The documentation for this class was generated from the following file:

- ConstructionMonitor/Core.py

DataTransfer.DataTransfer Class Reference

Public Member Functions

- def __init__ (self)
- def setValue (self, value)
- def getValue (self)
- def whileValue (self)

Private Attributes

__value

Detailed Description

Definition at line 4 of file DataTransfer.py.

Constructor & Destructor Documentation

def DataTransfer.DataTransfer.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets the initial value of self.__value to 0.

Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 6 of file DataTransfer.py.

```
    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets the initial value of self.__value to 0.

    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = 0

```

References DataTransfer.DataTransfer.__value.

Member Function Documentation

def DataTransfer.DataTransfer.getValue (*self*)

```
The getValue function returns the value of the private variable __value.
This is a getter function that allows access to this private variable.
```

```
Args:
self: Represent the instance of the class

Returns:
The value of the instance variable

Doc Author:
Willem van der Schans, Trelent AI
```

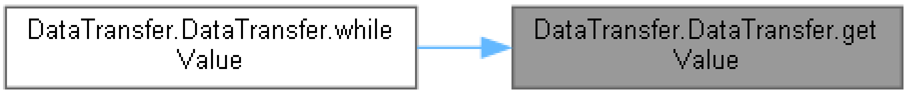
Definition at line 39 of file DataTransfer.py.

```
    def getValue(self):
        """
    The getValue function returns the value of the private variable __value.
    This is a getter function that allows access to this private variable.

    Args:
        self: Represent the instance of the class

    Returns:
        The value of the instance variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return self.__value

```

References DataTransfer.DataTransfer.__value.

Referenced by DataTransfer.DataTransfer.whileValue().

Here is the caller graph for this function:


DataTransfer.DataTransfer.while Value
DataTransfer.DataTransfer.get Value

def DataTransfer.DataTransfer.setValue (*self*, *value*)

```
The setValue function sets the value of the object.


Args:
self: Represent the instance of the class
value: Set the value of the instance variable __value

Returns:
The value that was passed to it

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 22 of file DataTransfer.py.

```
    def setValue(self, value):
        """
    The setValue function sets the value of the object.


    Args:
        self: Represent the instance of the class
        value: Set the value of the instance variable __value

    Returns:
        The value that was passed to it

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = value

```

References DataTransfer.DataTransfer.__value.

def DataTransfer.DataTransfer.whileValue (*self*)

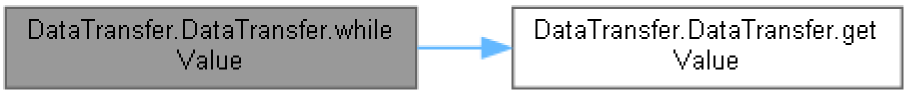
```
The whileValue function is a function that will run the getValue function until it is
told to stop.
This allows for the program to constantly be checking for new values from the sensor.

Args:
self: Refer to the current instance of the class

Returns:
The value of the input

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 55 of file DataTransfer.py.

```
    def whileValue(self):
        """
    The whileValue function is a function that will run the getValue function
until it is told to stop.
    This allows for the program to constantly be checking for new values from
the sensor.

    Args:
        self: Refer to the current instance of the class

    Returns:
        The value of the input

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            self.getValue()
```

References DataTransfer.DataTransfer.getValue().

Here is the call graph for this function:


DataTransfer.DataTransfer.whileValue
DataTransfer.DataTransfer.getValue

Member Data Documentation

DataTransfer.DataTransfer.__value `[private]`

Definition at line 20 of file DataTransfer.py.

Referenced by DataTransfer.DataTransfer.__init__(), DataTransfer.DataTransfer.getValue(), and DataTransfer.DataTransfer.setValue().

The documentation for this class was generated from the following file:

- DataTransfer.py

FileSaver.FileSaver Class Reference

Public Member Functions

- def __init__ (self, method, outputDF, AppendingPath=None)
- def getPath (self)

Public Attributes

- docPathdata
- dataAppending
- appendFlag
- fileName
- uiFlag
- primaryKey
- outputFrame

Detailed Description

Definition at line 13 of file FileSaver.py.

Constructor & Destructor Documentation

def FileSaver.FileSaver.__init__ (*self*, *method*, *outputDF*, *AppendingPath* = None)

```
The __init__ function is called when the class is instantiated.
It sets up the instance of the class, and defines all variables that will be used by
other functions in this class.
The __init__ function takes two arguments: self and method.  The first argument, self,
refers to an instance of a
class (in this case it's an instance of DataFrameSaver). The second argument, method
refers to a string value that
is passed into DataFrameSaver when it's instantiated.

Args:
self: Represent the instance of the class
method: Determine which dataframe to append the new data to
outputDF: Pass in the dataframe that will be saved to a csv file
AppendingPath: Specify the path to an existing csv file that you want to append your
dataframe to

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 15 of file FileSaver.py.

```
    def __init__(self, method, outputDF, AppendingPath=None):
        """
    The __init__ function is called when the class is instantiated.
    It sets up the instance of the class, and defines all variables that will
be used by other functions in this class.
    The __init__ function takes two arguments: self and method.  The first
argument, self, refers to an instance of a
    class (in this case it's an instance of DataFrameSaver). The second argument,
method refers to a string value that
    is passed into DataFrameSaver when it's instantiated.

    Args:
        self: Represent the instance of the class
        method: Determine which dataframe to append the new data to
        outputDF: Pass in the dataframe that will be saved to a csv file
        AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            datetime.datetime.today().strftime('%m%d%Y'))
        self.data = outputDF
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(subset=[self.primaryKey],
keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                                    windowType="savedLarge")
```

```

                     # Logging
                     print(
                         f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | {method} API request Completed | File Appended and Saved to
{self.docPath.joinpath(self.fileName)} | Exit Code 0")
                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Appending Statistics | Method: {method} | Appending file rows:
{self.dataAppending.shape[0]}, Total Rows: {(self.dataAppending.shape[0] +
self.data.shape[0])}, Duplicates Dropped {(self.dataAppending.shape[0] +
self.data.shape[0])-self.outputFrame.shape[0]}")
                 else:
                     PopupWrapped(text=f"File Saved to
{self.docPath.joinpath(self.fileName)}", windowType="savedLarge")

                     # Logging
                     print(
                         f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | {method} API request Completed | File Saved to
{self.docPath.joinpath(self.fileName)} | Exit Code 0")
             else:
                 pass

```

References FileSaver.FileSaver.appendFlag, FileSaver.FileSaver.data, FileSaver.FileSaver.dataAppending, FileSaver.FileSaver.docPath, FileSaver.FileSaver.fileName, FileSaver.FileSaver.outputFrame, FileSaver.FileSaver.primaryKey, and FileSaver.FileSaver.uiFlag.

Member Function Documentation

def FileSaver.FileSaver.getPath (*self*)

```
The getPath function returns the path to the file.
It is a string, and it joins the docPath with the fileName.

Args:
self: Represent the instance of the class

Returns:
The path to the file

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 101 of file FileSaver.py.

```
    def getPath(self):
        """
    The getPath function returns the path to the file.
        It is a string, and it joins the docPath with the fileName.

    Args:
        self: Represent the instance of the class

    Returns:
        The path to the file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return str(self.docPath.joinpath(self.fileName))
```

References FileSaver.FileSaver.docPath, and FileSaver.FileSaver.fileName.

Member Data Documentation

FileSaver.FileSaver.appendFlag

Definition at line 39 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

FileSaver.FileSaver.data

Definition at line 37 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

FileSaver.FileSaver.dataAppending

Definition at line 38 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

FileSaver.FileSaver.docPath

Definition at line 35 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__(), and FileSaver.FileSaver.getPath().

FileSaver.FileSaver.fileName

Definition at line 40 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__(), and FileSaver.FileSaver.getPath().

FileSaver.FileSaver.outputFrame

Definition at line 70 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

FileSaver.FileSaver.primaryKey

Definition at line 44 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

FileSaver.FileSaver.uiFlag

Definition at line 41 of file FileSaver.py.

Referenced by FileSaver.FileSaver.__init__().

The documentation for this class was generated from the following file:

- FileSaver.py

API_Calls.Initializer.initializer Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

classObjPrivate Member Functions

- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Detailed Description

Definition at line 22 of file Initializer.py.

Constructor & Destructor Documentation

def API_Calls.Initializer.initializer.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the logging, calls the __ShowGui function to create and display
the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

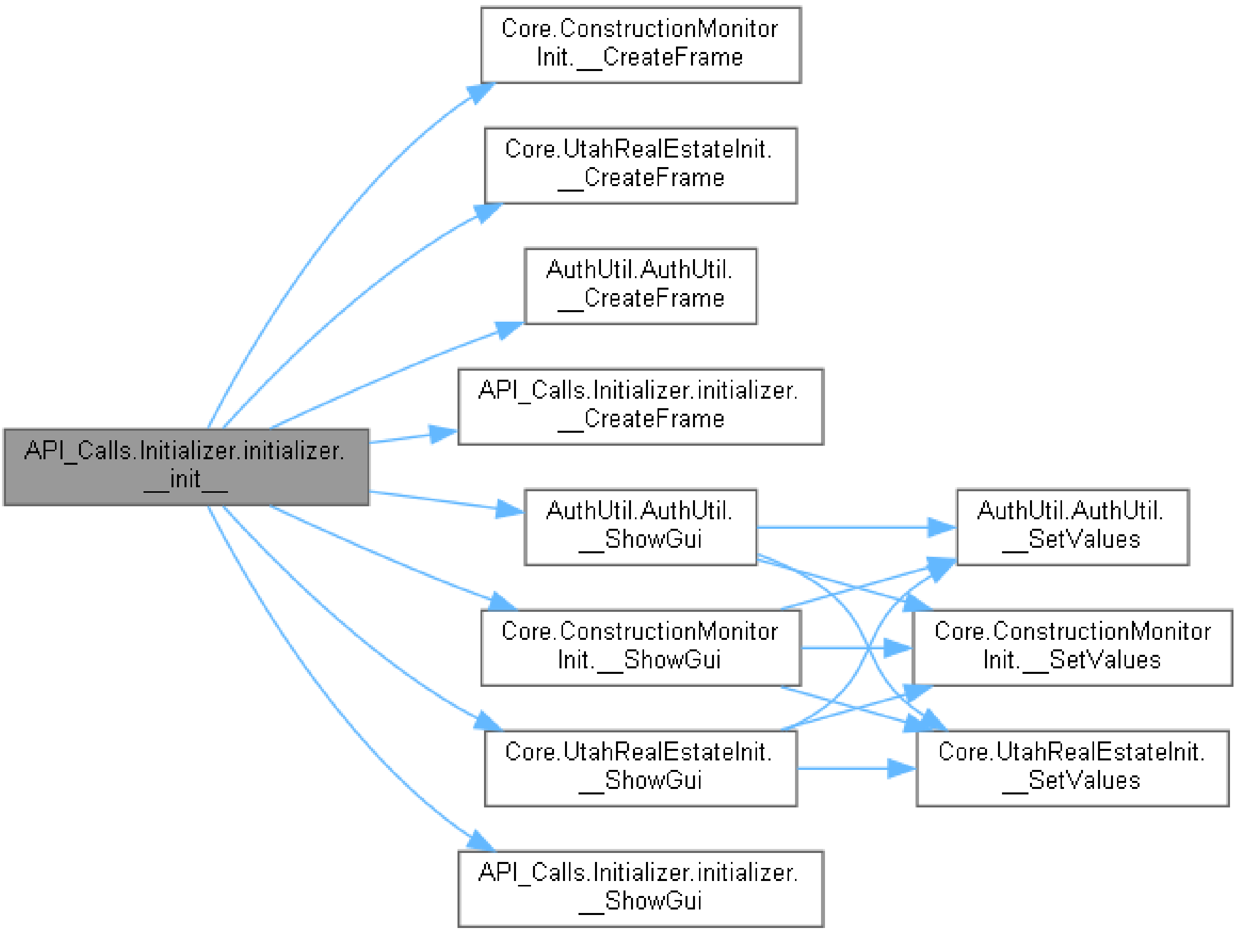
Definition at line 24 of file Initializer.py.

```
00024     def __init__(self):
00025 
00026         """
00027     The __init__ function is called when the class is instantiated.
00028     It sets up the logging, calls the __ShowGui function to create and display
00029     the GUI, and then calls __CreateFrame to create a frame for displaying
widgets.
00030 
00031 
00032     Args:
00033         self: Represent the instance of the class
00034 
00035     Returns:
00036         Nothing
00037 
00038     Doc Author:
00039         Willem van der Schans, Trelent AI
00040     """
00041         self.classObj = None
00042 
00043         logger()
00044 
00045         print("\n\n-------------Initiate Program---------------------\n\n")
00046 
00047         self.__ShowGui(self.__CreateFrame(), "Data Tool")
00048 
```

```
00049         print("\n\n-------------Closing Program---------------------\n\n")
00050
```

References Core.ConstructionMonitorInit.__CreateFrame(), Core.UtahRealEstateInit.__CreateFrame(), AuthUtil.AuthUtil.__CreateFrame(), API_Calls.Initializer.initializer.__CreateFrame(), AuthUtil.AuthUtil.__ShowGui(), API_Calls.Initializer.initializer.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), Core.UtahRealEstateInit.__ShowGui(), and API_Calls.Initializer.initializer.classObj.

Here is the call graph for this function:


API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__CreateFrame
Core.UtahRealEstateInit.__CreateFrame
AuthUtil.AuthUtil.__CreateFrame
API_Calls.Initializer.initializer.__CreateFrame
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
API_Calls.Initializer.initializer.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Member Function Documentation

def API_Calls.Initializer.initializer.__CreateFrame (*self*) [private]

```
The __CreateFrame function is a helper function that creates the layout for the main
window.
It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

Args:
self: Represent the instance of the class

Returns:
A list of lists, which is then passed to the sg

Doc Author:
Willem van der Schans, Trelent AI
```

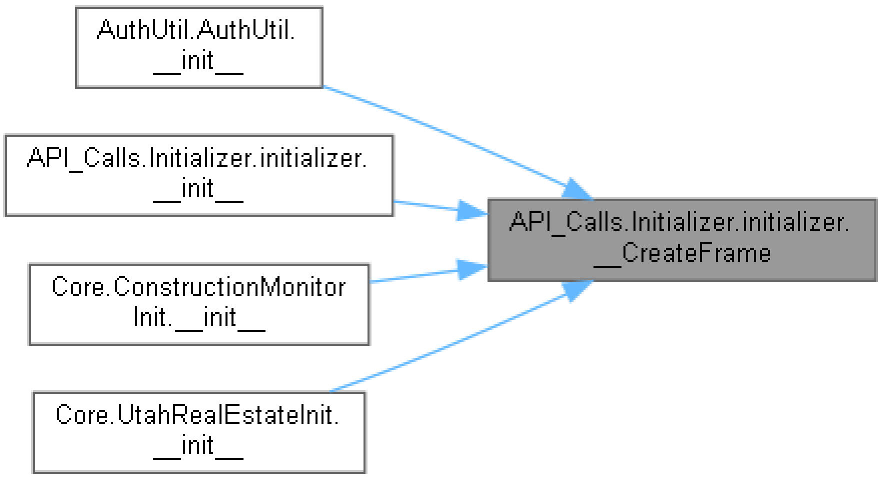
Definition at line 135 of file Initializer.py.

```
00135     def __CreateFrame(self):
00136
```

```
        """
    The __CreateFrame function is a helper function that creates the layout for the main window.
    It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

    Args:
        self: Represent the instance of the class

    Returns:
        A list of lists, which is then passed to the sg

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
                  sg.Button("CFPB Mortgage", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]

        return layout
```

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
API_Calls.Initializer.initializer.__CreateFrame

def API_Calls.Initializer.initializer.__ShowGui (***self*, *layout*, *text*) `[private]`**

```
The __ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing
a tuple with three elements:
1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;,
etc.)
2) A dictionary containing any additional parameters for that element (e.g., size,
default value, etc.)
3) An optional key name for the element (used in event handling). If no key name is
provided then one will be generated automatically by PySimpleGUIQt based on its position
in the layout list

Args:
self: Represent the instance of the class
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A window object

Doc Author:
Willem van der Schans, Trelent AI
```

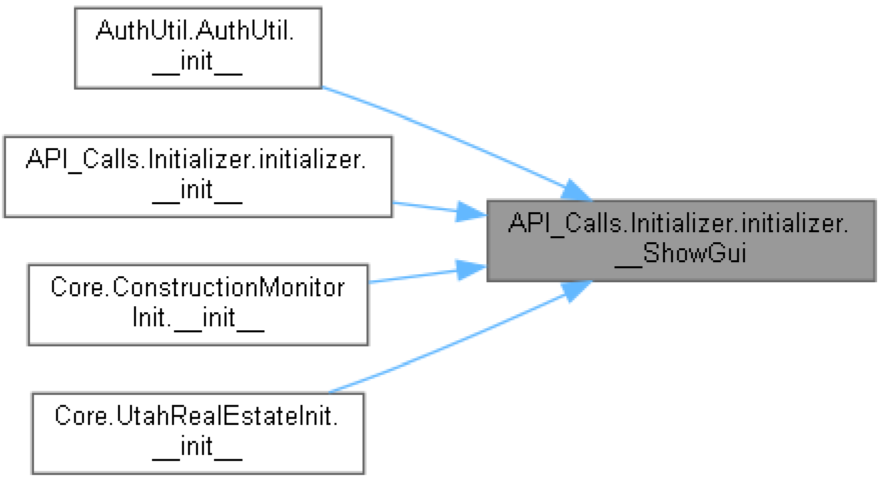
Definition at line 51 of file Initializer.py.

```
00051     def __ShowGui(self, layout, text):
00052 
00053         """
00054     The __ShowGui function is the main function that displays the GUI.
00055     It takes two arguments: layout and text. Layout is a list of lists, each
containing a tuple with three elements:
00056         1) The type of element to be displayed (e.g., &quot;Text&quot;,
&quot;InputText&quot;, etc.)
00057         2) A dictionary containing any additional parameters for that element
(e.g., size, default value, etc.)
00058         3) An optional key name for the element (used in event handling). If no
key name is provided then one will be generated automatically by PySimpleGUIQt based
on its position in the layout list
00059 
00060     Args:
00061         self: Represent the instance of the class
00062         layout: Pass the layout of the window to be created
00063         text: Set the title of the window
00064 
00065     Returns:
00066         A window object
00067 
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
00070     """
00071         versionChecker()
00072 
```

```
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Construction Monitor":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API Call-----------------")
                ConstructionMonitorMain(ConstructionMonitorInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API Call---------------------\n")
            elif event == "Utah Real Estate":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API Call-----------------")
                UtahRealEstateMain(UtahRealEstateInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API Call---------------------\n")
            elif event == "Realtor.Com":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
                realtorCom()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API Call---------------------\n")
            elif event == "CFPB Mortgage":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating ffiec.cfpb API Call-----------------")
                CFBP()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing ffiec.cfpb API Call---------------------\n")
            elif event == "Authorization Utility":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
                try:
                    os.system(f"start {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents folder",
                            windowType="errorLarge")

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Error on program exit, for logging purposes only.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
API_Calls.Initializer.initializer.__ShowGui

Member Data Documentation

API_Calls.Initializer.initializer.classObj

Definition at line 41 of file Initializer.py.

Referenced by API_Calls.Initializer.initializer.__init__().

The documentation for this class was generated from the following file:

- Initializer.py

PopupWrapped.PopupWrapped Class Reference

Public Member Functions

- def __init__ (self, text="", windowType="notice", error=None)
- def stopWindow (self)
- def textUpdate (self, sleep=0.5)
- def windowPush (self)
- def openFile (self)

Private Member Functions

- def __createLayout (self)
- def __createWindow (self)

Private Attributes

- __text__type
- __error
- __layout
- __windowObj
- __thread
- __counter
- __docpath
- __errorFlag

Detailed Description

Definition at line 14 of file PopupWrapped.py.

Constructor & Destructor Documentation

def PopupWrapped.PopupWrapped.__init__ (*self*, *text* = "", *windowType* = "notice", *error* = None)

```
The __init__ function is the first function that gets called when an object of this
class is created.
It sets up all the variables and creates a window for us to use.
Args:
self: Represent the instance of the class
text: Set the text of the window
windowType: Determine what type of window to create
error: Display the error message in the window
Returns:
Nothing
Doc Author:
Willem van der Schans, Trelent AI
```

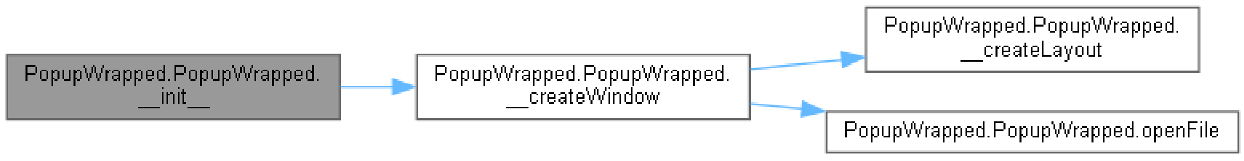
Definition at line 16 of file PopupWrapped.py.

```
00016     def __init__(self, text="", windowType="notice", error=None):
00017         """
00018     The __init__ function is the first function that gets called when an object
of this class is created.
00019     It sets up all the variables and creates a window for us to use.
00020     Args:
00021         self: Represent the instance of the class
00022         text: Set the text of the window
00023         windowType: Determine what type of window to create
00024         error: Display the error message in the window
```

```
    Returns:
        Nothing
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__text = text
        self.__type = windowType
        self.__error = error
        self.__layout = []
        self.__windowObj = None
        self.__thread = None
        self.__counter = 0
        self.__docpath = None
        self.__errorFlag = False

        try:
            if "File Appended and Saved to " in self.__text:
                self.__docpath = str(self.__text[27:])
            elif "File Saved to " in self.__text:
                self.__docpath = str(self.__text[14:])
            else:
                pass
        except Exception as e:
            if self.__type == "savedLarge":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py | Error = {e} | Error creating self.__docpath open file button not available")
                self.__errorFlag = True
            else:
                pass

        self.__createWindow()

```

References PopupWrapped.PopupWrapped.__counter, PopupWrapped.PopupWrapped.__createWindow(), PopupWrapped.PopupWrapped.__docpath, PopupWrapped.PopupWrapped.__error, PopupWrapped.PopupWrapped.__errorFlag, BatchProgressGUI.BatchProgressGUI.__layout, PopupWrapped.PopupWrapped.__layout, PopupWrapped.PopupWrapped.__text, PopupWrapped.PopupWrapped.__thread, BatchProgressGUI.BatchProgressGUI.__type, PopupWrapped.PopupWrapped.__type, and PopupWrapped.PopupWrapped.__windowObj.

Here is the call graph for this function:



PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout
PopupWrapped.PopupWrapped.openFile

Member Function Documentation

def PopupWrapped.PopupWrapped.__createLayout (*self*) [private]

```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates
a layout based on that information.
Args:
self: Refer to the current instance of a class
Returns:
A list of lists
Doc Author:
Willem van der Schans, Trelent AI
```

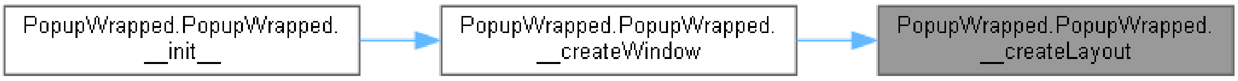
Definition at line 57 of file PopupWrapped.py.

```
    def __createLayout(self):
        """
    The __createLayout function is used to create the layout of the window.
    The function takes class variables and returns a window layout.
    It uses a series of if statements to determine what type of window it is, then creates a layout based on that information.
    Args:
        self: Refer to the current instance of a class
    Returns:
        A list of lists
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "savedLarge":
            if self.__errorFlag:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
            else:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "FatalErrorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "error":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}: {self.__error}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "AuthError":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
00116                           sg.Ok(button_text="Return", focus=True, size=(10, 1)),
sg.Push()]
00117           elif self.__type == "versionWindow":
00118               __Line1 = [sg.Push(),
00119                          sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00120                          sg.Text(f"{self.__text}", justification="center",
key="-textField-"),
00121                          sg.Push()]
00122               __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00123                          sg.Ok(button_text="Continue", focus=True, size=(10,
1)), sg.Push()]
00124           elif self.__type == "progress":
00125               __Line1 = [sg.Push(),
00126                          sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00127
00128           if self.__type == "progress":
00129               self.__layout = [__Line1, ]
00130           else:
00131               self.__layout = [__Line1, __Line2]
00132
```

References PopupWrapped.PopupWrapped.__errorFlag, BatchProgressGUI.BatchProgressGUI.__layout, PopupWrapped.PopupWrapped.__layout, PopupWrapped.PopupWrapped.__text, BatchProgressGUI.BatchProgressGUI.__type, and PopupWrapped.PopupWrapped.__type.

Referenced by PopupWrapped.PopupWrapped.__createWindow().

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout

def PopupWrapped.PopupWrapped.__createWindow (*self*) `[private]`

```
The __createWindow function is used to create the window object that will be displayed.
The function takes class variables and a window object. The function first calls
__createLayout, which creates the layout for the window based on what type of message
it is (error, notice, progress). Then it uses PySimpleGUI's Window class to create a
new window with that layout and some other parameters such as title and icon. If this
is not a progress bar or permanent message then we start a timer loop that waits until
either 100 iterations have passed or an event has been triggered (such as clicking
&quot;Ok&quot; or closing the window). Once one of these events occurs
Args:
self: Reference the instance of the class
Returns:
A window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 133 of file PopupWrapped.py.

```
00133       def __createWindow(self):
00134           """
00135       The __createWindow function is used to create the window object that will
be displayed.
00136       The function takes class variables and a window object. The function first
calls __createLayout, which creates the layout for the window based on what type of
message it is (error, notice, progress). Then it uses PySimpleGUI's Window class to
create a new window with that layout and some other parameters such as title and icon.
If this is not a progress bar or permanent message then we start a timer loop that waits
until either 100 iterations have passed or an event has been triggered (such as clicking
&quot;Ok&quot; or closing the window). Once one of these events occurs
00137       Args:
00138           self: Reference the instance of the class
00139       Returns:
00140           A window object
00141       Doc Author:
00142           Willem van der Schans, Trelent AI
00143       """
00144           self.__createLayout()
00145
```

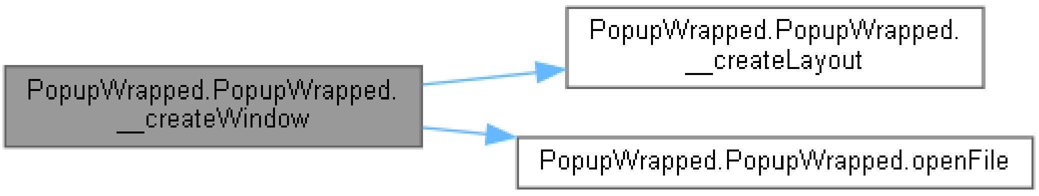
```python
        if self.__type == "progress":
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 50))
        elif self.__type == "noticeLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "savedLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=False,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "errorLarge":
            self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "FatalErrorLarge":
            self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "AuthError":
            self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "versionWindow":
            self.__windowObj = sg.Window(title="Update", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        else:
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            print("Here")
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                print(event)
                if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
                    break
                elif event == "Open Generation Tool [Web Browser]":
```

```
webbrowser.open('https://www.debugbear.com/basic-auth-header-generator', new=2,
autoraise=True)
                    pass
                elif event == "Open File":
                    threadFile = threading.Thread(target=self.openFile,
                                                  daemon=False)
                    threadFile.start()
                    time.sleep(3)
                    break
                elif event == "Download":
webbrowser.open('https://github.com/Kydoimos97/GardnerApiUtility/releases/latest',
new=2,
                                    autoraise=True)
                    pass
                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}"
)
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | PopupWrapped.py | Error = {e} | Log Folder not found please search
manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

```

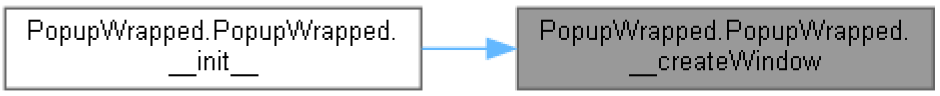
References PopupWrapped.PopupWrapped.__createLayout(), BatchProgressGUI.BatchProgressGUI.__layout, PopupWrapped.PopupWrapped.__layout, BatchProgressGUI.BatchProgressGUI.__type, PopupWrapped.PopupWrapped.__type, PopupWrapped.PopupWrapped.__windowObj, and PopupWrapped.PopupWrapped.openFile().

Referenced by PopupWrapped.PopupWrapped.__init__().

Here is the call graph for this function:


PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout
PopupWrapped.PopupWrapped.openFile

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow

def PopupWrapped.PopupWrapped.openFile (*self*)

```
The openFile function opens the file that is associated with the
document object.  It does this by calling os.system and passing it
self.__docpath as an argument.

Args:
self: Represent the instance of the object itself

Returns:
The filepath of the document

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 291 of file PopupWrapped.py.

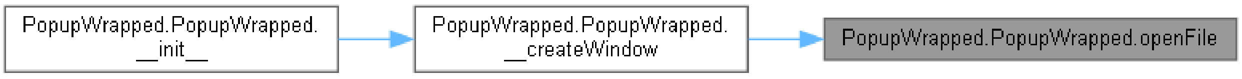
```
00291     def openFile(self):
00292         """
```

```
00293     The openFile function opens the file that is associated with the
00294         document object.  It does this by calling os.system and passing it
00295         self.__docpath as an argument.
00296
00297     Args:
00298         self: Represent the instance of the object itself
00299
00300     Returns:
00301         The filepath of the document
00302
00303     Doc Author:
00304         Willem van der Schans, Trelent AI
00305     """
00306         os.system(self.__docpath)
```

References PopupWrapped.PopupWrapped.__docpath.

Referenced by PopupWrapped.PopupWrapped.__createWindow().

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.openFile

def PopupWrapped.PopupWrapped.stopWindow (*self*)

```
The stopWindow function is used to close the window object that was created in the
startWindow function.
This is done by calling the close() method on self.__windowObj, which will cause it
to be destroyed.
Args:
self: Represent the instance of the class
Returns:
The window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 230 of file PopupWrapped.py.

```
00230     def stopWindow(self):
00231         """
00232     The stopWindow function is used to close the window object that was created
in the startWindow function.
00233     This is done by calling the close() method on self.__windowObj, which will
cause it to be destroyed.
00234     Args:
00235         self: Represent the instance of the class
00236     Returns:
00237         The window object
00238     Doc Author:
00239         Willem van der Schans, Trelent AI
00240     """
00241         self.__windowObj.close()
00242
```

References PopupWrapped.PopupWrapped.__windowObj.

def PopupWrapped.PopupWrapped.textUpdate (*self*, *sleep* = 0.5)

```
The textUpdate function is a function that updates the text in the text field.
It does this by adding dots to the end of it, and then removing them. This creates
a loading effect for when something is being processed.
Args:
self: Refer to the object itself
sleep: Control the speed of the text update
Returns:
A string that is the current text of the text field
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 243 of file PopupWrapped.py.

```
00243     def textUpdate(self, sleep=0.5):
```

```
00244           """
00245     The textUpdate function is a function that updates the text in the text field.
00246     It does this by adding dots to the end of it, and then removing them. This creates
00247     a loading effect for when something is being processed.
00248     Args:
00249         self: Refer to the object itself
00250         sleep: Control the speed of the text update
00251     Returns:
00252         A string that is the current text of the text field
00253     Doc Author:
00254         Willem van der Schans, Trelent AI
00255     """
00256         self.__counter += 1
00257         if self.__counter == 4:
00258             self.__counter = 1
00259         newString = ""
00260         if self.__type == "notice":
00261             pass
00262         elif self.__type == "error":
00263             pass
00264         elif self.__type == "progress":
00265             newString = f"{self.__text}{'.' * self.__counter}"
00266         self.__windowObj.write_event_value('update-textField-', newString)
00267 
00268         time.sleep(sleep)
00269 
```

References PopupWrapped.PopupWrapped.__counter, BatchProgressGUI.BatchProgressGUI.__type, PopupWrapped.PopupWrapped.__type, and PopupWrapped.PopupWrapped.__windowObj.

def PopupWrapped.PopupWrapped.windowPush (*self*)

```
The windowPush function is used to update the values of a window object.
The function takes in an event and values from the window object, then checks if the
event starts with 'update'.
If it does, it will take everything after 'update' as a key for updating that specific
value.
It will then update that value using its key and refresh the window.
Args:
self: Reference the object that is calling the function
Returns:
A tuple containing the event and values
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 270 of file PopupWrapped.py.

```
00270     def windowPush(self):
00271 
00272         """
00273     The windowPush function is used to update the values of a window object.
00274         The function takes in an event and values from the window object, then checks if the event starts with 'update'.
00275         If it does, it will take everything after 'update' as a key for updating that specific value.
00276         It will then update that value using its key and refresh the window.
00277     Args:
00278         self: Reference the object that is calling the function
00279     Returns:
00280         A tuple containing the event and values
00281     Doc Author:
00282         Willem van der Schans, Trelent AI
00283     """
00284         event, values = self.__windowObj.read()
00285 
00286         if event.startswith('update'):
00287             __key_to_update = event[len('update'):]
00288             self.__windowObj[__key_to_update].update(values[event])
00289             self.__windowObj.refresh()
00290 
```

References PopupWrapped.PopupWrapped.__windowObj.

Member Data Documentation

PopupWrapped.PopupWrapped.__counter [private]

Definition at line 36 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__init__(), and PopupWrapped.PopupWrapped.textUpdate().

PopupWrapped.PopupWrapped.__docpath [private]

Definition at line 37 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__init__(), and PopupWrapped.PopupWrapped.openFile().

PopupWrapped.PopupWrapped.__error [private]

Definition at line 32 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__init__().

PopupWrapped.PopupWrapped.__errorFlag [private]

Definition at line 38 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), and PopupWrapped.PopupWrapped.__init__().

PopupWrapped.PopupWrapped.__layout [private]

Definition at line 33 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), PopupWrapped.PopupWrapped.__createWindow(), BatchProgressGUI.BatchProgressGUI.__init__(), PopupWrapped.PopupWrapped.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), and BatchProgressGUI.BatchProgressGUI.CreateProgressLayout().

PopupWrapped.PopupWrapped.__text [private]

Definition at line 30 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), and PopupWrapped.PopupWrapped.__init__().

PopupWrapped.PopupWrapped.__thread [private]

Definition at line 35 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__init__().

PopupWrapped.PopupWrapped.__type [private]

Definition at line 31 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__createLayout(), PopupWrapped.PopupWrapped.__createWindow(), BatchProgressGUI.BatchProgressGUI.__init__(), PopupWrapped.PopupWrapped.__init__(), BatchProgressGUI.BatchProgressGUI.BatchGuiShow(), and PopupWrapped.PopupWrapped.textUpdate().

PopupWrapped.PopupWrapped.__windowObj [private]

Definition at line 34 of file PopupWrapped.py.

Referenced by PopupWrapped.PopupWrapped.__createWindow(), PopupWrapped.PopupWrapped.__init__(), PopupWrapped.PopupWrapped.stopWindow(), PopupWrapped.PopupWrapped.textUpdate(), and PopupWrapped.PopupWrapped.windowPush().

The documentation for this class was generated from the following file:

- PopupWrapped.py

Core.realtorCom Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- dfStatedfCounty
- dfZip
- uiString

Private Member Functions

- def __showUi (self)
- def __linkGetter (self)
- def __dataUpdater (self)

Private Attributes

- __page_html__update_date
- __last_date
- __idDict
- __linkDict

Detailed Description

Definition at line 15 of file Realtor/Core.py.

Constructor & Destructor Documentation

def Core.realtorCom.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of an object, and it's where you put code that needs to
run before anything else in your class.

Args:
self: Represent the instance of the class

Returns:
A new object

Doc Author:
Willem van der Schans, Trelent AI
```

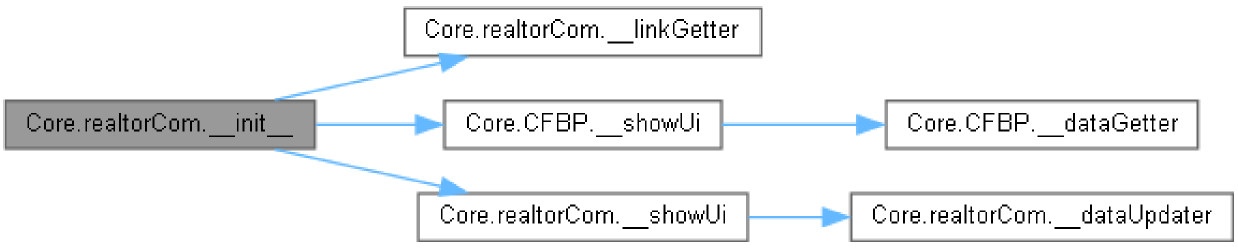
Definition at line 17 of file Realtor/Core.py.

```
00017     def __init__(self):
00018         """
00019     The __init__ function is called when the class is instantiated.
00020     It sets up the initial state of an object, and it's where you put code that
needs to run before anything else in your class.
00021
00022     Args:
00023         self: Represent the instance of the class
00024
00025     Returns:
00026         A new object
00027
00028     Doc Author:
00029         Willem van der Schans, Trelent AI
```

```
    """
        self.__page_html = None
        self.__update_date = None
        self.__last_date = None
        self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
        self.__linkDict = {}
        self.dfState = None
        self.dfCounty = None
        self.dfZip = None
        self.uiString = "Files Saved to \n"

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            page_html =
requests.get("https://www.realtor.com/research/data/").text
            self.__page_html = BeautifulSoup(page_html, "html.parser")
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__linkGetter()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Link Dictionary = {self.__idDict}")
            self.__showUi()
            PopupWrapped(text=self.uiString, windowType="noticeLarge")
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
        else:
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | User Canceled Request")
            pass

```

References Core.realtorCom.__idDict, Core.realtorCom.__last_date, Core.realtorCom.__linkDict, Core.realtorCom.__linkGetter(), Core.realtorCom.__page_html, Core.CFBP.__showUi(), Core.realtorCom.__showUi(), Core.realtorCom.__update_date, Core.realtorCom.dfCounty, Core.realtorCom.dfState, Core.realtorCom.dfZip, Core.CFBP.uiString, and Core.realtorCom.uiString.

Here is the call graph for this function:



Member Function Documentation

def Core.realtorCom.__dataUpdater (*self*) [private]

```
The __dataUpdater function is a private function that updates the dataframes for each
of the three
types of realtor data. It takes class variables and return the path to the saved file.
The function first creates an empty
dictionary called tempdf, then iterates through each key in self.__idDict (which
contains all three ids).
For each key, it reads in a csv file from the link associated with that id and saves
it to tempdf as a pandas
DataFrame object. Then, depending on which type of realtor data we are dealing with
(State/County/Zip), we save


Args:
self: Access the attributes and methods of the class
```

```
Returns:
The path of the saved file

Doc Author:
Willem van der Schans, Trelent AI
```

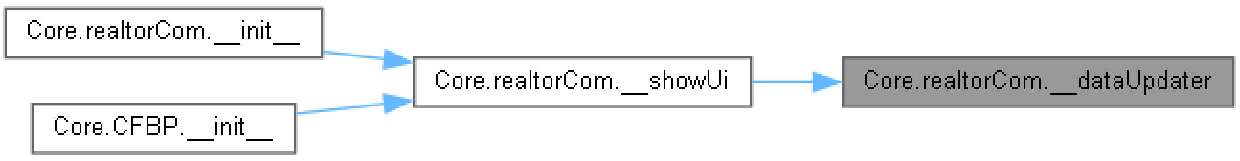
Definition at line 113 of file Realtor/Core.py.

```
00113     def __dataUpdater(self):
00114 
00115         """
00116     The __dataUpdater function is a private function that updates the dataframes
for each of the three
00117         types of realtor data. It takes class variables and return the path to
the saved file. The function first creates an empty
00118         dictionary called tempdf, then iterates through each key in self.__idDict
(which contains all three ids).
00119         For each key, it reads in a csv file from the link associated with that
id and saves it to tempdf as a pandas
00120         DataFrame object. Then, depending on which type of realtor data we are
dealing with (State/County/Zip), we save
00121 
00122 
00123     Args:
00124         self: Access the attributes and methods of the class
00125 
00126     Returns:
00127         The path of the saved file
00128 
00129     Doc Author:
00130         Willem van der Schans, Trelent AI
00131     """
00132         for key, value in self.__idDict.items():
00133             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00134 
00135             if key == "State":
00136                 self.dfState = tempdf
00137             elif key == "County":
00138                 self.dfCounty = tempdf
00139             elif key == "Zip":
00140                 self.dfZip = tempdf
00141 
00142             FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00143             self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()}
\n"
```

References Core.realtorCom.__idDict, Core.realtorCom.dfCounty, Core.realtorCom.dfState, Core.realtorCom.dfZip, Core.CFBP.uiString, and Core.realtorCom.uiString.

Referenced by Core.realtorCom.__showUi().

Here is the caller graph for this function:



def Core.realtorCom.__linkGetter (*self*) `[private]`

```
The __linkGetter function is a private function that takes the idDict dictionary and
adds
a link to each entry in the dictionary. The link is used to access historical data for
each
scope symbol.

Args:
self: Refer to the object itself

Returns:
A dictionary of all the links to the history pages
```

```
Doc Author:
Willem van der Schans, Trelent AI
```

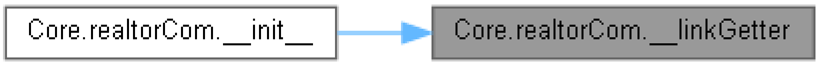
Definition at line 86 of file Realtor/Core.py.

```
00086     def __linkGetter(self):
00087 
00088         """
00089     The __linkGetter function is a private function that takes the idDict
dictionary and adds
00090     a link to each entry in the dictionary. The link is used to access historical
data for each
00091     scope symbol.
00092 
00093     Args:
00094         self: Refer to the object itself
00095 
00096     Returns:
00097         A dictionary of all the links to the history pages
00098 
00099     Doc Author:
00100         Willem van der Schans, Trelent AI
00101     """
00102         for key, value in self.__idDict.items():
00103             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00104                 try:
00105                     for nestedRow in row.find_all("a"):
00106                         if "History" in str(nestedRow.get("href")) and key in
str(nestedRow.get("href")):
00107                             self.__idDict[key] = {"id": value, "link":
nestedRow.get("href")}
00108                 except Exception as e:
00109                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
for realtor.com")
00110                     RESTError(801)
00111                     raise SystemExit(801)
00112 
```

References Core.realtorCom.__idDict, and Core.realtorCom.__page_html.

Referenced by Core.realtorCom.__init__().

Here is the caller graph for this function:



def Core.realtorCom.__showUi (*self*) [private]

```
The __showUi function is a helper function that creates and displays the progress window.
It also starts the dataUpdater thread, which will update the progress bar as it runs.


Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

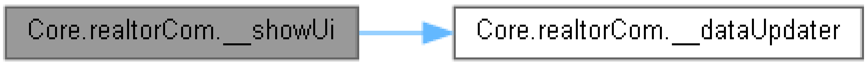
Definition at line 58 of file Realtor/Core.py.

```
00058     def __showUi(self):
00059 
00060         """
00061     The __showUi function is a helper function that creates and displays the
progress window.
00062     It also starts the dataUpdater thread, which will update the progress bar
as it runs.
00063 
00064 
00065     Args:
              self: Represent the instance of the class

          Returns:
              A popupwrapped object

          Doc Author:
              Willem van der Schans, Trelent AI
          """
              uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)

              threadGui = threading.Thread(target=self.__dataUpdater,
                                           daemon=False)
              threadGui.start()

              while threadGui.is_alive():
                  uiObj.textUpdate()
                  uiObj.windowPush()
              else:
                  uiObj.stopWindow()

```

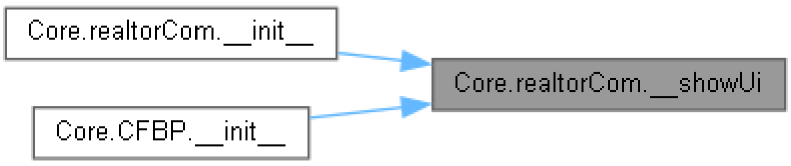
References Core.realtorCom.__dataUpdater().

Referenced by Core.realtorCom.__init__(), and Core.CFBP.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



Member Data Documentation

Core.realtorCom.__idDict [private]

Definition at line 34 of file Realtor/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), Core.realtorCom.__init__(), and Core.realtorCom.__linkGetter().

Core.realtorCom.__last_date [private]

Definition at line 33 of file Realtor/Core.py.

Referenced by Core.realtorCom.__init__().

Core.realtorCom.__linkDict [private]

Definition at line 35 of file Realtor/Core.py.

Referenced by Core.realtorCom.__init__().

Core.realtorCom.__page_html [private]

Definition at line 31 of file Realtor/Core.py.

Referenced by Core.realtorCom.__init__(), and Core.realtorCom.__linkGetter().

Core.realtorCom.__update_date [private]

Definition at line 32 of file Realtor/Core.py.

Referenced by Core.realtorCom.__init__().

Core.realtorCom.dfCounty

Definition at line 37 of file Realtor/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), and Core.realtorCom.__init__().

Core.realtorCom.dfState

Definition at line 36 of file Realtor/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), and Core.realtorCom.__init__().

Core.realtorCom.dfZip

Definition at line 38 of file Realtor/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), and Core.realtorCom.__init__().

Core.realtorCom.uiString

Definition at line 39 of file Realtor/Core.py.

Referenced by Core.realtorCom.__dataUpdater(), Core.realtorCom.__init__(), and Core.CFBP.__init__().

The documentation for this class was generated from the following file:

- Realtor/Core.py

Core.UtahRealEstateInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatusListedOrModified
- dateStart
- dateEnd
- select
- file_name
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

Detailed Description

Definition at line 24 of file UtahRealEstate/Core.py.

Constructor & Destructor Documentation

def Core.UtahRealEstateInit.__init__ (*self*)

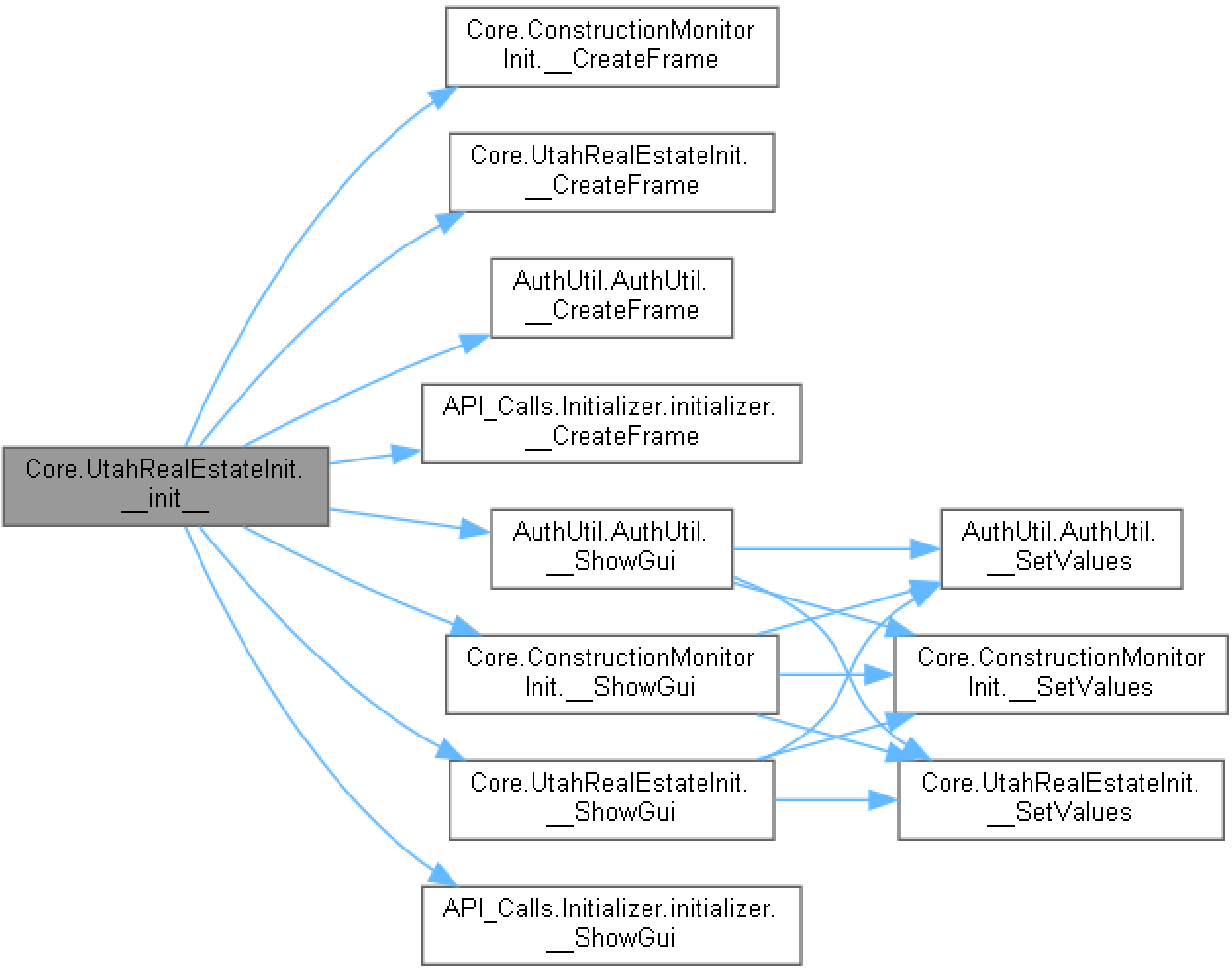
```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object.


Args:
self: Represent the instance of the class

Returns:
The __createframe function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 26 of file UtahRealEstate/Core.py.

```
    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object.


    Args:
        self: Represent the instance of the class

    Returns:
        The __createframe function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

```

References Core.ConstructionMonitorInit.__CreateFrame(), Core.UtahRealEstateInit.__CreateFrame(), AuthUtil.AuthUtil.__CreateFrame(), API_Calls.Initializer.initializer.__CreateFrame(), AuthUtil.AuthUtil.__ShowGui(), API_Calls.Initializer.initializer.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), Core.UtahRealEstateInit.__ShowGui(), AuthUtil.AuthUtil.append_file, Core.ConstructionMonitorInit.append_file, Core.UtahRealEstateInit.append_file, Core.ConstructionMonitorInit.dateEnd, Core.UtahRealEstateInit.dateEnd, Core.ConstructionMonitorInit.dateStart, Core.UtahRealEstateInit.dateStart, AuthUtil.AuthUtil.file_name, Core.UtahRealEstateInit.file_name, AuthUtil.AuthUtil.ListedOrModified, Core.UtahRealEstateInit.ListedOrModified, Core.UtahRealEstateInit.select, AuthUtil.AuthUtil.StandardStatus, and Core.UtahRealEstateInit.StandardStatus.

Here is the call graph for this function:



Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__ShowGui
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues
API_Calls.Initializer.initializer. __ShowGui

Member Function Documentation

def Core.UtahRealEstateInit.__CreateFrame () `[static]`, `[private]`

```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the window.
Each element is defined by its type and any additional parameters needed to define it.

Args:

Returns:
A list of lists, which is used to create the gui

Doc Author:
Willem van der Schans, Trelent AI
```

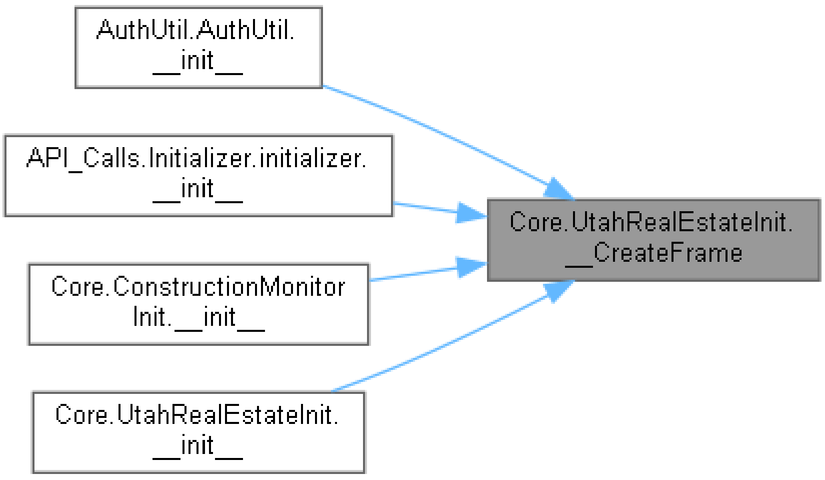
Definition at line 92 of file UtahRealEstate/Core.py.

```
00092     def __CreateFrame():
00093         """
00094     The __CreateFrame function creates the GUI layout for the application.
00095         The function returns a list of lists that contains all the elements to
be displayed in the window.
00096         Each element is defined by its type and any additional parameters needed
to define it.
00097
00098     Args:
00099
00100     Returns:
00101         A list of lists, which is used to create the gui
00102
00103     Doc Author:
00104         Willem van der Schans, Trelent AI
00105     """
00106         sg.theme('Default1')
00107
00108         line00 = [sg.HSeparator()]
00109
00110         line0 = [sg.Image(ImageLoader("logo.png")),
00111                  sg.Push(),
00112                  sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00113                  sg.Push(),
00114                  sg.Push()]
00115
00116         line1 = [sg.HSeparator()]
00117
00118         line2 = [sg.Text("MLS Status : ", size=(15, None),
justification="Right"),
00119                  sg.DropDown(default_value="Active", values=["Active",
"Closed"], key="-status-", size=(31, 1))]
00120
00121         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00122                  sg.DropDown(default_value="Listing Date", values=["Listing
Date", "Modification Date", "Close Date"],
00123                              key="-type-", size=(31, 1))]
00124
00125         line4 = [sg.Text("Start Date : ", size=(15, None),
justification="Right"),
00126                  sg.Input(default_text=(date.today() -
timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00127                           disabled=False, size=(20, 1)),
00128                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-start_date-', target="-DateStart-")]
00129
00130         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00131                  sg.Input(default_text=(date.today().strftime("%Y-%m-%d")),
key="-DateEnd-", disabled=False,
00132                           size=(20, 1)),
00133                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-end_date-', target="-DateEnd-")]
00134
```

```
        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"),
justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Text("Appending File : ", size=(15, None),
justification="Right"),
                  sg.Input(default_text="", key="-AppendingFile-",
disabled=True,
                           size=(20, 1)),
                  sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
                                target="-AppendingFile-")]

        line11 = [sg.HSeparator()]

        line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8,
line9, line10, line11,
                  line12]

        return layout

```

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the caller graph for this function:



def Core.UtahRealEstateInit.__SetValues (***self***, ***values*****)** `[private]`

```
The __SetValues function is used to set the values of the variables that are used in
the
__GetData function. The values are passed from a dictionary called 'values' which is
created
by parsing through an XML file using ElementTree. This function also sets default values
for
some of these variables if they were not specified in the XML file.

Args:
self: Represent the instance of the class
values: Pass the values from the gui to this function

Returns:
A dictionary with the following keys:

Doc Author:
Willem van der Schans, Trelent AI
```

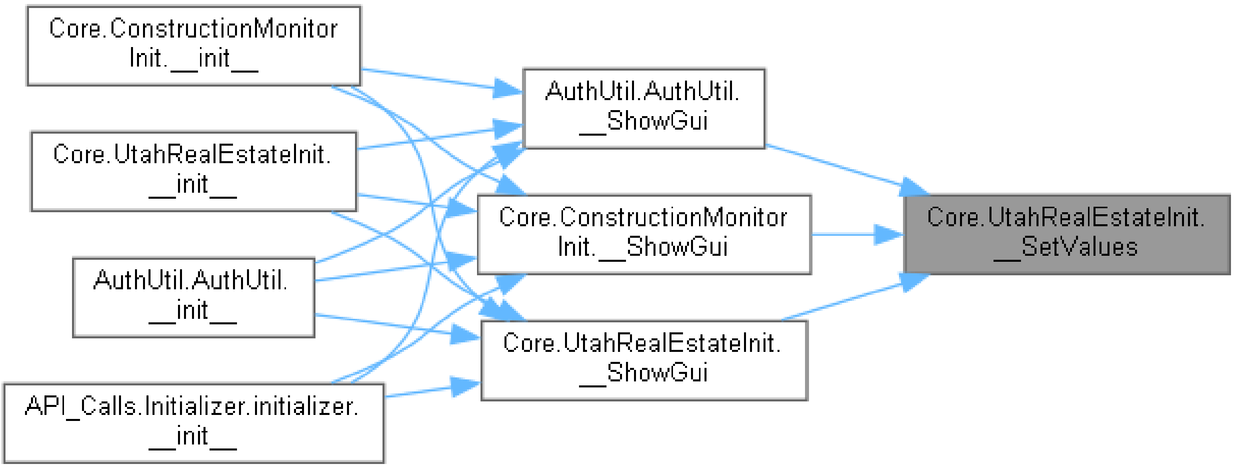
Definition at line 158 of file UtahRealEstate/Core.py.

```
    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the
       __GetData function. The values are passed from a dictionary called 'values' which is created
       by parsing through an XML file using ElementTree. This function also sets default values for
       some of these variables if they were not specified in the XML file.

    Args:
        self: Represent the instance of the class
        values: Pass the values from the gui to this function

    Returns:
        A dictionary with the following keys:

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

```

References AuthUtil.AuthUtil.append_file, Core.ConstructionMonitorInit.append_file, Core.UtahRealEstateInit.append_file, Core.ConstructionMonitorInit.dateEnd, Core.UtahRealEstateInit.dateEnd, Core.ConstructionMonitorInit.dateStart, Core.UtahRealEstateInit.dateStart, AuthUtil.AuthUtil.ListedOrModified, Core.UtahRealEstateInit.ListedOrModified, Core.UtahRealEstateInit.select, AuthUtil.AuthUtil.StandardStatus, and Core.UtahRealEstateInit.StandardStatus.

Referenced by AuthUtil.AuthUtil.__ShowGui(), Core.ConstructionMonitorInit.__ShowGui(), and Core.UtahRealEstateInit.__ShowGui().

Here is the caller graph for this function:


Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues

def Core.UtahRealEstateInit.__ShowGui (***self*****,** ***layout*****,** ***text*****)** `[private]`

```
The __ShowGui function is a helper function that creates the GUI window and displays
it to the user.
It takes in two parameters: layout, which is a list of lists containing all the elements
for each row;
and text, which is a string containing what will be displayed as the title of the window.
The __ShowGui
method then uses these parameters to create an instance of sg.Window with all its
attributes set accordingly.

Args:
self: Refer to the current class instance
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 52 of file UtahRealEstate/Core.py.

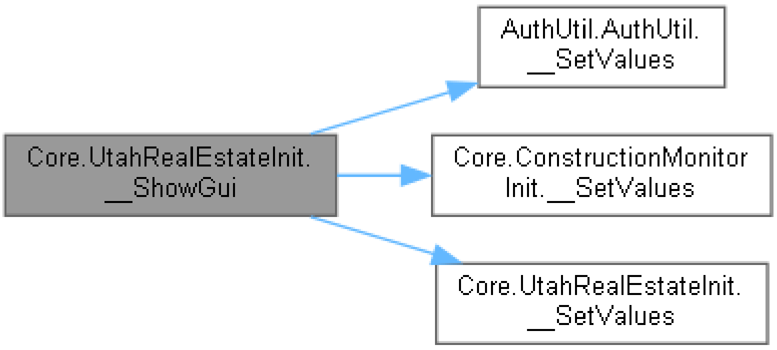
```
00052     def __ShowGui(self, layout, text):
00053 
00054         """
00055     The __ShowGui function is a helper function that creates the GUI window and
displays it to the user.
00056     It takes in two parameters: layout, which is a list of lists containing all
the elements for each row;
00057     and text, which is a string containing what will be displayed as the title
of the window. The __ShowGui
00058     method then uses these parameters to create an instance of sg.Window with
all its attributes set accordingly.
00059 
00060     Args:
00061         self: Refer to the current class instance
00062         layout: Pass the layout of the window to be created
00063         text: Set the title of the window
00064 
00065     Returns:
00066         A dictionary of values
00067 
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
00070     """
00071         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00072                            finalize=True,
00073                            icon=ImageLoader("taskbar_icon.ico"))
00074 
00075         while True:
```

```
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(993)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

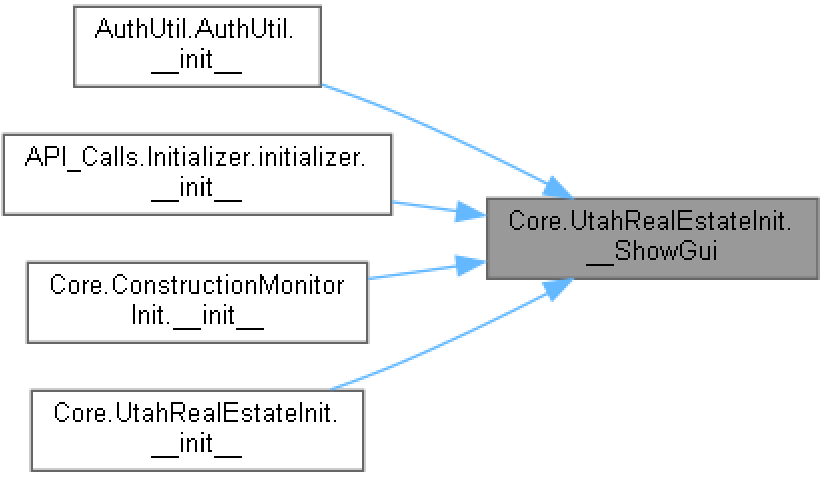
References AuthUtil.AuthUtil.__SetValues(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Referenced by AuthUtil.AuthUtil.__init__(), API_Calls.Initializer.initializer.__init__(), Core.ConstructionMonitorInit.__init__(), and Core.UtahRealEstateInit.__init__().

Here is the call graph for this function:


Core.UtahRealEstateInit.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
Core.UtahRealEstateInit.__ShowGui

Member Data Documentation

Core.UtahRealEstateInit.append_file

Definition at line 48 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.UtahRealEstateInit.dateEnd

Definition at line 45 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.UtahRealEstateInit.dateStart

Definition at line 44 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorInit.__init__(), Core.UtahRealEstateInit.__init__(), Core.ConstructionMonitorInit.__SetValues(), and Core.UtahRealEstateInit.__SetValues().

Core.UtahRealEstateInit.file_name

Definition at line 47 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), and Core.UtahRealEstateInit.__init__().

Core.UtahRealEstateInit.ListedOrModified

Definition at line 43 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateInit.__init__(), and Core.UtahRealEstateInit.__SetValues().

Core.UtahRealEstateInit.select

Definition at line 46 of file UtahRealEstate/Core.py.

Referenced by Core.UtahRealEstateInit.__init__(), and Core.UtahRealEstateInit.__SetValues().

Core.UtahRealEstateInit.StandardStatus

Definition at line 42 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateInit.__init__(), and Core.UtahRealEstateInit.__SetValues().

The documentation for this class was generated from the following file:

- UtahRealEstate/Core.py

Core.UtahRealEstateMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

- dataframekeyPath
- filePath
- key

Private Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __batches__siteClass
- __headerDict
- __parameterString
- __appendFile
- __dateStart
- __dateEnd
- __restDomain
- __record_val

Detailed Description

Definition at line 198 of file UtahRealEstate/Core.py.

Constructor & Destructor Documentation

def Core.UtahRealEstateMain.__init__ (*self*, *siteClass*)

```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to work properly.

Args:
self: Represent the instance of the class
siteClass: Determine which site to pull data from

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

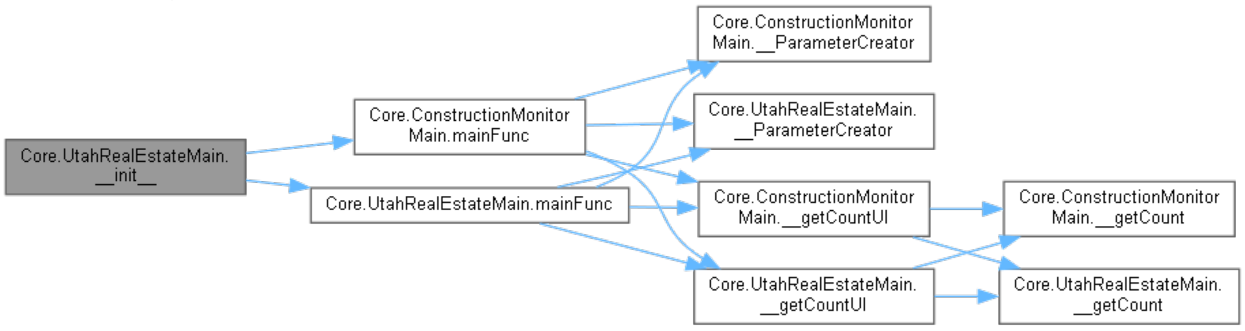
Definition at line 200 of file UtahRealEstate/Core.py.

```
00200     def __init__(self, siteClass):
00201
00202         """
00203     The __init__ function is the first function that runs when an object of this
class is created.
```

```
    It sets up all the variables and functions needed for this class to work properly.

    Args:
        self: Represent the instance of the class
        siteClass: Determine which site to pull data from

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = 'https://resoapi.utahrealestate.com/reso/odata/Property?'
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None
        self.__record_val = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```

References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, Core.UtahRealEstateMain.__dateEnd, Core.UtahRealEstateMain.__dateStart, BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__parameterString, Core.UtahRealEstateMain.__parameterString, Core.ConstructionMonitorMain.__record_val, Core.UtahRealEstateMain.__record_val, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, Core.ConstructionMonitorMain.__siteClass, Core.UtahRealEstateMain.__siteClass, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, AuthUtil.AuthUtil.filePath, Core.UtahRealEstateMain.filePath, Core.UtahRealEstateMain.key, AuthUtil.AuthUtil.keyPath, Core.UtahRealEstateMain.keyPath, Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the call graph for this function:



Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Member Function Documentation

def Core.UtahRealEstateMain.__getCount (*self*) [private]

```
The __getCount function is used to determine the number of records that will be returned
by the query.
This function is called when a user calls the count() method on a ReST object. The
__getCount function uses
the $count parameter in OData to return only an integer value representing how many
records would be returned
by the query.

Args:
self: Represent the instance of the class

Returns:
The number of records in the data set

Doc Author:
Willem van der Schans, Trelent AI
```

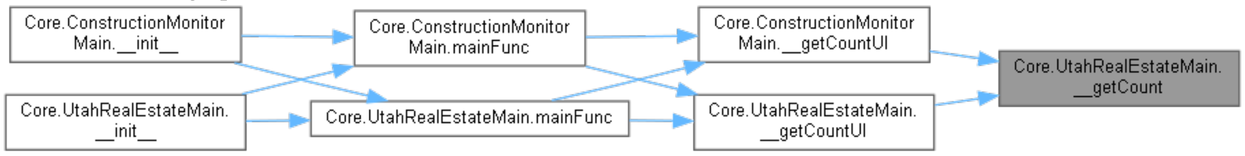
Definition at line 365 of file UtahRealEstate/Core.py.

```
    def __getCount(self):
        """
    The __getCount function is used to determine the number of records that will be returned by the query.
    This function is called when a user calls the count() method on a ReST object. The __getCount function uses
    the $count parameter in OData to return only an integer value representing how many records would be returned
    by the query.

    Args:
        self: Represent the instance of the class

    Returns:
        The number of records in the data set

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:
            __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
                                        headers=self.__headerDict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        self.__record_val = int(__count_resp.json()["@odata.count"])

```

References BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, Core.ConstructionMonitorMain.__record_val, and Core.UtahRealEstateMain.__record_val.

Referenced by Core.ConstructionMonitorMain.__getCountUI(), and Core.UtahRealEstateMain.__getCountUI().

Here is the caller graph for this function:


Core.ConstructionMonitorMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCount
Core.UtahRealEstateMain.__init__
Core.UtahRealEstateMain.mainFunc
Core.UtahRealEstateMain.__getCountUI

def Core.UtahRealEstateMain.__getCountUI (*self*) [private]

```
The __getCountUI function is a wrapper for the __getCount function.
It creates a progress window and updates it while the __getCount function runs.
The purpose of this is to keep the GUI responsive while running long processes.

Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

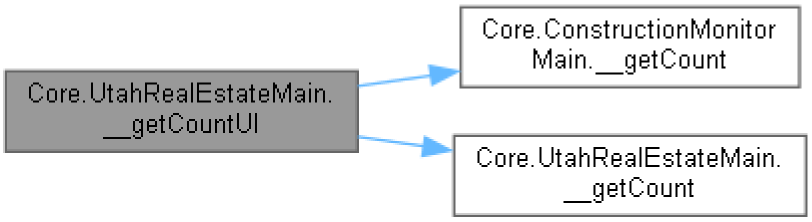
Definition at line 405 of file UtahRealEstate/Core.py.

```
    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It creates a progress window and updates it while the __getCount function runs.
    The purpose of this is to keep the GUI responsive while running long processes.

    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```

References Core.ConstructionMonitorMain.__getCount(), and Core.UtahRealEstateMain.__getCount().

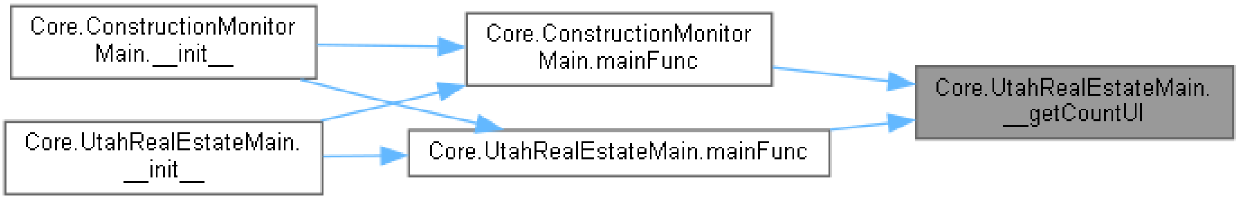
Referenced by Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the call graph for this function:



Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Here is the caller graph for this function:



Core.ConstructionMonitorMain.__init__
Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.UtahRealEstateMain.__getCountUI

def Core.UtahRealEstateMain.__ParameterCreator (*self*) `[private]`

```
The __ParameterCreator function is used to create the filter string for the ReST API
call.
The function takes in a siteClass object and extracts all of its parameters into a
dictionary.
It then creates an appropriate filter string based on those parameters.

Args:
self: Bind the object to the class

Returns:
A string to be used as the parameter in the api call

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 324 of file UtahRealEstate/Core.py.

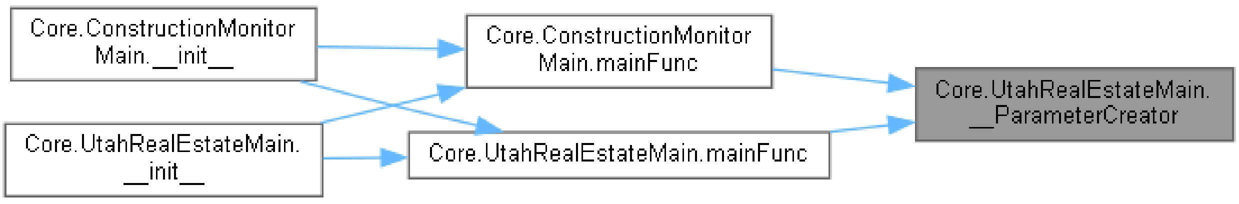
```
    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the filter string for the
ReST API call.
    The function takes in a siteClass object and extracts all of its parameters
into a dictionary.
    It then creates an appropriate filter string based on those parameters.

    Args:
        self: Bind the object to the class

    Returns:
        A string to be used as the parameter in the api call

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        filter_string = ""

        __Source_dict = {key: value for key, value in
self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}
```

```
00343
00344         self.__appendFile = __Source_dict["append_file"]
00345         __Source_dict.pop("append_file")
00346
00347         temp_dict = copy.copy(__Source_dict)
00348         for key, value in temp_dict.items():
00349             if value is None:
00350                 __Source_dict.pop(key)
00351             else:
00352                 pass
00353
00354         if __Source_dict["ListedOrModified"] == "Listing Date":
00355             filter_string =
f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingC
ontractDate%20le%20{__Source_dict['dateEnd']}"
00356         elif __Source_dict["ListedOrModified"] == "Modification Date":
00357             filter_string =
f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20a
nd%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00358         elif __Source_dict["ListedOrModified"] == "Close Date":
00359             filter_string =
f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{
__Source_dict['dateEnd']}"
00360
00361         filter_string = filter_string +
f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['Stan
dardStatus']}'"
00362
00363         self.__parameterString = filter_string
00364
```

References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProcessing.BatchProcessorUtahRealEstate.__parameterString, Core.UtahRealEstateMain.__parameterString, Core.ConstructionMonitorMain.__siteClass, and Core.UtahRealEstateMain.__siteClass.

Referenced by Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain. __ParameterCreator
Core.UtahRealEstateMain. __init__
Core.UtahRealEstateMain.mainFunc

def Core.UtahRealEstateMain.mainFunc (*self*)

```
The mainFunc function is the main function of this module. It will be called by the
GUI when a user clicks on
the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your
code for running your program, and it
should return a dataframe that contains all the data you want to display in your final
report.

Args:
self: Reference the object itself

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 247 of file UtahRealEstate/Core.py.

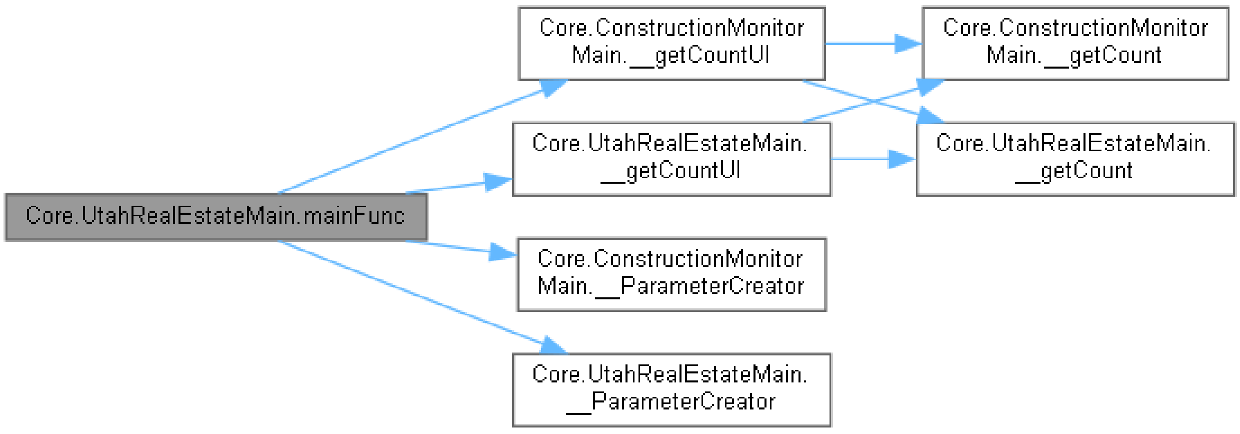
```
00247     def mainFunc(self):
00248
00249         """
```

```
    The mainFunc function is the main function of this module. It will be called by the GUI when a user clicks on
    the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your code for running your program, and it
    should return a dataframe that contains all the data you want to display in your final report.

    Args:
        self: Reference the object itself

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        passFlag = False

        while not passFlag:
            if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
                try:
                    f = open(self.keyPath, "rb")
                    key = f.readline()
                    f.close()
                    f = open(self.filePath, "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
                    self.__headerDict = {authDict["ure"]["parameter"]: authkey}
                    passFlag = True
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String = {self.__parameterString}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain = {self.__restDomain}")

        self.__getCountUI()

        if self.__record_val is None:
            self.__record_val = 0

        self.__batches = BatchCalculator(self.__record_val, None)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterString,
                                                  HeaderDict=self.__headerDict,
                                                  BatchesNum=self.__batches,
                                                  Type="utah_real_estate")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y 
%H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and 
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', 
time.gmtime((datetime.datetime.now().replace(microsecond=0) - 
startTime).total_seconds()))}")
                FileSaver("ure", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y 
%H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

```

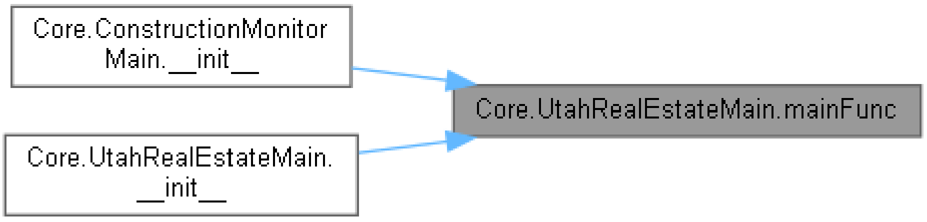
References Core.ConstructionMonitorMain.__appendFile, Core.UtahRealEstateMain.__appendFile, BatchProgressGUI.BatchProgressGUI.__batches, Core.ConstructionMonitorMain.__batches, Core.UtahRealEstateMain.__batches, Core.ConstructionMonitorMain.__getCountUI(), Core.UtahRealEstateMain.__getCountUI(), BatchProcessing.BatchProcessorConstructionMonitor.__headerDict, BatchProcessing.BatchProcessorUtahRealEstate.__headerDict, BatchProgressGUI.BatchProgressGUI.__headerDict, Core.ConstructionMonitorMain.__headerDict, Core.UtahRealEstateMain.__headerDict, Core.ConstructionMonitorMain.__ParameterCreator(), Core.UtahRealEstateMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.__parameterString, Core.UtahRealEstateMain.__parameterString, Core.ConstructionMonitorMain.__record_val, Core.UtahRealEstateMain.__record_val, BatchProcessing.BatchProcessorConstructionMonitor.__restDomain, BatchProcessing.BatchProcessorUtahRealEstate.__restDomain, BatchProgressGUI.BatchProgressGUI.__restDomain, Core.ConstructionMonitorMain.__restDomain, Core.UtahRealEstateMain.__restDomain, BatchProcessing.BatchProcessorConstructionMonitor.dataframe, BatchProcessing.BatchProcessorUtahRealEstate.dataframe, BatchProgressGUI.BatchProgressGUI.dataframe, Core.ConstructionMonitorMain.dataframe, Core.UtahRealEstateMain.dataframe, AuthUtil.AuthUtil.filePath, Core.UtahRealEstateMain.filePath, AuthUtil.AuthUtil.keyPath, and Core.UtahRealEstateMain.keyPath.

Referenced by Core.ConstructionMonitorMain.__init__(), and Core.UtahRealEstateMain.__init__().

Here is the call graph for this function:


Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCountUI
Core.UtahRealEstateMain.__getCount
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.UtahRealEstateMain.mainFunc

Member Data Documentation

Core.UtahRealEstateMain.__appendFile [private]

Definition at line 221 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), Core.UtahRealEstateMain.__ParameterCreator(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.__batches [private]

Definition at line 217 of file UtahRealEstate/Core.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProgressGUI.BatchProgressGUI.createGui(), BatchProgressGUI.BatchProgressGUI.CreateProgressLayout(), Core.ConstructionMonitorMain.mainFunc(), Core.UtahRealEstateMain.mainFunc(), and BatchProgressGUI.BatchProgressGUI.TimeUpdater().

Core.UtahRealEstateMain.__dateEnd [private]

Definition at line 223 of file UtahRealEstate/Core.py.

Referenced by Core.UtahRealEstateMain.__init__().

Core.UtahRealEstateMain.__dateStart [private]

Definition at line 222 of file UtahRealEstate/Core.py.

Referenced by Core.UtahRealEstateMain.__init__().

Core.UtahRealEstateMain.__headerDict [private]

Definition at line 219 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.__parameterString `[private]`

Definition at line 220 of file UtahRealEstate/Core.py.

Referenced by BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.UtahRealEstateMain.__init__(), Core.UtahRealEstateMain.__ParameterCreator(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.__record_val `[private]`

Definition at line 230 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), Core.UtahRealEstateMain.__getCount(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.__restDomain `[private]`

Definition at line 224 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorMain.__getCount(), BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.__siteClass `[private]`

Definition at line 218 of file UtahRealEstate/Core.py.

Referenced by Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), Core.ConstructionMonitorMain.__ParameterCreator(), and Core.UtahRealEstateMain.__ParameterCreator().

Core.UtahRealEstateMain.dataframe

Definition at line 216 of file UtahRealEstate/Core.py.

Referenced by BatchProgressGUI.BatchProgressGUI.__init__(), BatchProcessing.BatchProcessorConstructionMonitor.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.__init__(), Core.ConstructionMonitorMain.__init__(), Core.UtahRealEstateMain.__init__(), BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom(), BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor(), BatchProgressGUI.BatchProgressGUI.createGui(), Core.ConstructionMonitorMain.mainFunc(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.filePath

Definition at line 227 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateMain.__init__(), AuthUtil.AuthUtil.__SetValues(), and Core.UtahRealEstateMain.mainFunc().

Core.UtahRealEstateMain.key

Definition at line 229 of file UtahRealEstate/Core.py.

Referenced by Core.UtahRealEstateMain.__init__().

Core.UtahRealEstateMain.keyPath

Definition at line 225 of file UtahRealEstate/Core.py.

Referenced by AuthUtil.AuthUtil.__init__(), Core.UtahRealEstateMain.__init__(), and Core.UtahRealEstateMain.mainFunc().

The documentation for this class was generated from the following file:

- UtahRealEstate/Core.py

File Documentation

__init__.py

main.c

```
/* Generated by Cython 0.29.32 */

#ifndef PY_SSIZE_T_CLEAN
#define PY_SSIZE_T_CLEAN
#endif /* PY_SSIZE_T_CLEAN */
#include "Python.h"
#ifndef Py_PYTHON_H
    #error Python headers needed to compile C extensions, please install development version of Python.
#elif PY_VERSION_HEX < 0x02060000 || (0x03000000 <= PY_VERSION_HEX && PY_VERSION_HEX < 0x03030000)
    #error Cython requires Python 2.6+ or Python 3.3+.
#else
#define CYTHON_ABI "0_29_32"
#define CYTHON_HEX_VERSION 0x001D20F0
#define CYTHON_FUTURE_DIVISION 0
#include <stddef.h>
#ifndef offsetof
  #define offsetof(type, member) ( (size_t) & ((type*)0) -> member )
#endif
#if !defined(WIN32) && !defined(MS_WINDOWS)
  #ifndef __stdcall
    #define __stdcall
  #endif
  #ifndef __cdecl
    #define __cdecl
  #endif
  #ifndef __fastcall
    #define __fastcall
  #endif
#endif
#ifndef DL_IMPORT
  #define DL_IMPORT(t) t
#endif
#ifndef DL_EXPORT
  #define DL_EXPORT(t) t
#endif
#define __PYX_COMMA ,
#ifndef HAVE_LONG_LONG
  #if PY_VERSION_HEX >= 0x02070000
    #define HAVE_LONG_LONG
  #endif
#endif
#ifndef PY_LONG_LONG
  #define PY_LONG_LONG LONG_LONG
#endif
#ifndef Py_HUGE_VAL
  #define Py_HUGE_VAL HUGE_VAL
#endif
#ifdef PYPY_VERSION
  #define CYTHON_COMPILING_IN_PYPY 1
  #define CYTHON_COMPILING_IN_PYSTON 0
  #define CYTHON_COMPILING_IN_CPYTHON 0
  #define CYTHON_COMPILING_IN_NOGIL 0
  #undef CYTHON_USE_TYPE_SLOTS
  #define CYTHON_USE_TYPE_SLOTS 0
  #undef CYTHON_USE_PYTYPE_LOOKUP
  #define CYTHON_USE_PYTYPE_LOOKUP 0
  #if PY_VERSION_HEX < 0x03050000
    #undef CYTHON_USE_ASYNC_SLOTS
    #define CYTHON_USE_ASYNC_SLOTS 0
  #elif !defined(CYTHON_USE_ASYNC_SLOTS)
    #define CYTHON_USE_ASYNC_SLOTS 1
  #endif
  #undef CYTHON_USE_PYLIST_INTERNALS
  #define CYTHON_USE_PYLIST_INTERNALS 0
  #undef CYTHON_USE_UNICODE_INTERNALS
  #define CYTHON_USE_UNICODE_INTERNALS 0
  #undef CYTHON_USE_UNICODE_WRITER
  #define CYTHON_USE_UNICODE_WRITER 0
  #undef CYTHON_USE_PYLONG_INTERNALS
  #define CYTHON_USE_PYLONG_INTERNALS 0
  #undef CYTHON_AVOID_BORROWED_REFS
  #define CYTHON_AVOID_BORROWED_REFS 1
  #undef CYTHON_ASSUME_SAFE_MACROS
  #define CYTHON_ASSUME_SAFE_MACROS 0
  #undef CYTHON_UNPACK_METHODS
  #define CYTHON_UNPACK_METHODS 0
  #undef CYTHON_FAST_THREAD_STATE
  #define CYTHON_FAST_THREAD_STATE 0
  #undef CYTHON_FAST_PYCALL
  #define CYTHON_FAST_PYCALL 0
  #undef CYTHON_PEP489_MULTI_PHASE_INIT
  #define CYTHON_PEP489_MULTI_PHASE_INIT 0
  #undef CYTHON_USE_TP_FINALIZE
  #define CYTHON_USE_TP_FINALIZE 0
  #undef CYTHON_USE_DICT_VERSIONS
  #define CYTHON_USE_DICT_VERSIONS 0
  #undef CYTHON_USE_EXC_INFO_STACK
  #define CYTHON_USE_EXC_INFO_STACK 0
  #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
    #define CYTHON_UPDATE_DESCRIPTOR_DOC (PYPY_VERSION_HEX >= 0x07030900)
  #endif
#elif defined(PYSTON_VERSION)
  #define CYTHON_COMPILING_IN_PYPY 0
  #define CYTHON_COMPILING_IN_PYSTON 1
  #define CYTHON_COMPILING_IN_CPYTHON 0
  #define CYTHON_COMPILING_IN_NOGIL 0
  #ifndef CYTHON_USE_TYPE_SLOTS
    #define CYTHON_USE_TYPE_SLOTS 1
  #endif
  #undef CYTHON_USE_PYTYPE_LOOKUP
  #define CYTHON_USE_PYTYPE_LOOKUP 0
  #undef CYTHON_USE_ASYNC_SLOTS
  #define CYTHON_USE_ASYNC_SLOTS 0
  #undef CYTHON_USE_PYLIST_INTERNALS
  #define CYTHON_USE_PYLIST_INTERNALS 0
  #ifndef CYTHON_USE_UNICODE_INTERNALS
    #define CYTHON_USE_UNICODE_INTERNALS 1
  #endif
  #undef CYTHON_USE_UNICODE_WRITER
  #define CYTHON_USE_UNICODE_WRITER 0
  #undef CYTHON_USE_PYLONG_INTERNALS
  #define CYTHON_USE_PYLONG_INTERNALS 0
  #ifndef CYTHON_AVOID_BORROWED_REFS
    #define CYTHON_AVOID_BORROWED_REFS 0
  #endif
  #ifndef CYTHON_ASSUME_SAFE_MACROS
    #define CYTHON_ASSUME_SAFE_MACROS 1
  #endif
  #ifndef CYTHON_UNPACK_METHODS
    #define CYTHON_UNPACK_METHODS 1
  #endif
  #undef CYTHON_FAST_THREAD_STATE
  #define CYTHON_FAST_THREAD_STATE 0
  #undef CYTHON_FAST_PYCALL
  #define CYTHON_FAST_PYCALL 0
  #undef CYTHON_PEP489_MULTI_PHASE_INIT
  #define CYTHON_PEP489_MULTI_PHASE_INIT 0
  #undef CYTHON_USE_TP_FINALIZE
  #define CYTHON_USE_TP_FINALIZE 0
  #undef CYTHON_USE_DICT_VERSIONS
  #define CYTHON_USE_DICT_VERSIONS 0
  #undef CYTHON_USE_EXC_INFO_STACK
  #define CYTHON_USE_EXC_INFO_STACK 0
  #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
    #define CYTHON_UPDATE_DESCRIPTOR_DOC 0
  #endif
#elif defined(PY_NOGIL)
  #define CYTHON_COMPILING_IN_PYPY 0
  #define CYTHON_COMPILING_IN_PYSTON 0
  #define CYTHON_COMPILING_IN_CPYTHON 0
  #define CYTHON_COMPILING_IN_NOGIL 1
  #ifndef CYTHON_USE_TYPE_SLOTS
    #define CYTHON_USE_TYPE_SLOTS 1
  #endif
  #undef CYTHON_USE_PYTYPE_LOOKUP
  #define CYTHON_USE_PYTYPE_LOOKUP 0
  #ifndef CYTHON_USE_ASYNC_SLOTS
    #define CYTHON_USE_ASYNC_SLOTS 1
  #endif
  #undef CYTHON_USE_PYLIST_INTERNALS
  #define CYTHON_USE_PYLIST_INTERNALS 0
  #ifndef CYTHON_USE_UNICODE_INTERNALS
    #define CYTHON_USE_UNICODE_INTERNALS 1
  #endif
  #undef CYTHON_USE_UNICODE_WRITER
  #define CYTHON_USE_UNICODE_WRITER 0
  #undef CYTHON_USE_PYLONG_INTERNALS
  #define CYTHON_USE_PYLONG_INTERNALS 0
  #ifndef CYTHON_AVOID_BORROWED_REFS
    #define CYTHON_AVOID_BORROWED_REFS 0
  #endif
  #ifndef CYTHON_ASSUME_SAFE_MACROS
    #define CYTHON_ASSUME_SAFE_MACROS 1
  #endif
  #ifndef CYTHON_UNPACK_METHODS
    #define CYTHON_UNPACK_METHODS 1
  #endif
  #undef CYTHON_FAST_THREAD_STATE
  #define CYTHON_FAST_THREAD_STATE 0
  #undef CYTHON_FAST_PYCALL
  #define CYTHON_FAST_PYCALL 0
  #ifndef CYTHON_PEP489_MULTI_PHASE_INIT
    #define CYTHON_PEP489_MULTI_PHASE_INIT 1
  #endif
  #ifndef CYTHON_USE_TP_FINALIZE
    #define CYTHON_USE_TP_FINALIZE 1
  #endif
  #undef CYTHON_USE_DICT_VERSIONS
  #define CYTHON_USE_DICT_VERSIONS 0
  #undef CYTHON_USE_EXC_INFO_STACK
  #define CYTHON_USE_EXC_INFO_STACK 0
#else
  #define CYTHON_COMPILING_IN_PYPY 0
  #define CYTHON_COMPILING_IN_PYSTON 0
  #define CYTHON_COMPILING_IN_CPYTHON 1
  #define CYTHON_COMPILING_IN_NOGIL 0
  #ifndef CYTHON_USE_TYPE_SLOTS
    #define CYTHON_USE_TYPE_SLOTS 1
  #endif
  #if PY_VERSION_HEX < 0x02070000
    #undef CYTHON_USE_PYTYPE_LOOKUP
    #define CYTHON_USE_PYTYPE_LOOKUP 0
  #elif !defined(CYTHON_USE_PYTYPE_LOOKUP)
    #define CYTHON_USE_PYTYPE_LOOKUP 1
  #endif
  #if PY_MAJOR_VERSION < 3
    #undef CYTHON_USE_ASYNC_SLOTS
    #define CYTHON_USE_ASYNC_SLOTS 0
  #elif !defined(CYTHON_USE_ASYNC_SLOTS)
    #define CYTHON_USE_ASYNC_SLOTS 1
  #endif
  #if PY_VERSION_HEX < 0x02070000
    #undef CYTHON_USE_PYLONG_INTERNALS
    #define CYTHON_USE_PYLONG_INTERNALS 0
  #elif !defined(CYTHON_USE_PYLONG_INTERNALS)
    #define CYTHON_USE_PYLONG_INTERNALS 1
  #endif
  #ifndef CYTHON_USE_PYLIST_INTERNALS
    #define CYTHON_USE_PYLIST_INTERNALS 1
  #endif
  #ifndef CYTHON_USE_UNICODE_INTERNALS
    #define CYTHON_USE_UNICODE_INTERNALS 1
  #endif
  #if PY_VERSION_HEX < 0x030300F0 || PY_VERSION_HEX >= 0x030B00A2
    #undef CYTHON_USE_UNICODE_WRITER
    #define CYTHON_USE_UNICODE_WRITER 0
  #elif !defined(CYTHON_USE_UNICODE_WRITER)
    #define CYTHON_USE_UNICODE_WRITER 1
  #endif
  #ifndef CYTHON_AVOID_BORROWED_REFS
    #define CYTHON_AVOID_BORROWED_REFS 0
  #endif
  #ifndef CYTHON_ASSUME_SAFE_MACROS
    #define CYTHON_ASSUME_SAFE_MACROS 1
  #endif
  #ifndef CYTHON_UNPACK_METHODS
    #define CYTHON_UNPACK_METHODS 1
  #endif
  #if PY_VERSION_HEX >= 0x030B00A4
    #undef CYTHON_FAST_THREAD_STATE
    #define CYTHON_FAST_THREAD_STATE 0
  #elif !defined(CYTHON_FAST_THREAD_STATE)
    #define CYTHON_FAST_THREAD_STATE 1
  #endif
  #ifndef CYTHON_FAST_PYCALL
    #define CYTHON_FAST_PYCALL (PY_VERSION_HEX < 0x030A0000)
  #endif
  #ifndef CYTHON_PEP489_MULTI_PHASE_INIT
    #define CYTHON_PEP489_MULTI_PHASE_INIT (PY_VERSION_HEX >= 0x03050000)
  #endif
  #ifndef CYTHON_USE_TP_FINALIZE
    #define CYTHON_USE_TP_FINALIZE (PY_VERSION_HEX >= 0x030400a1)
  #endif
  #ifndef CYTHON_USE_DICT_VERSIONS
    #define CYTHON_USE_DICT_VERSIONS (PY_VERSION_HEX >= 0x030600B1)
  #endif
  #if PY_VERSION_HEX >= 0x030B00A4
    #undef CYTHON_USE_EXC_INFO_STACK
    #define CYTHON_USE_EXC_INFO_STACK 0
  #elif !defined(CYTHON_USE_EXC_INFO_STACK)
    #define CYTHON_USE_EXC_INFO_STACK (PY_VERSION_HEX >= 0x030700A3)
  #endif
  #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
    #define CYTHON_UPDATE_DESCRIPTOR_DOC 1
  #endif
#endif
#if !defined(CYTHON_FAST_PYCCALL)
#define CYTHON_FAST_PYCCALL  (CYTHON_FAST_PYCALL && PY_VERSION_HEX >= 0x030600B1)
#endif
#if CYTHON_USE_PYLONG_INTERNALS
  #if PY_MAJOR_VERSION < 3
    #include "longintrepr.h"
  #endif
  #undef SHIFT
  #undef BASE
  #undef MASK
  #ifdef SIZEOF_VOID_P
    enum { __pyx_check_sizeof_voidp = 1 / (int)(SIZEOF_VOID_P == sizeof(void*)) };
  #endif
#endif
#ifndef __has_attribute
  #define __has_attribute(x) 0
#endif
#ifndef __has_cpp_attribute
  #define __has_cpp_attribute(x) 0
#endif
#ifndef CYTHON_RESTRICT
  #if defined(__GNUC__)
    #define CYTHON_RESTRICT __restrict__
  #elif defined(_MSC_VER) && _MSC_VER >= 1400
    #define CYTHON_RESTRICT __restrict
  #elif defined (__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
    #define CYTHON_RESTRICT restrict
  #else
    #define CYTHON_RESTRICT
  #endif
#endif
#ifndef CYTHON_UNUSED
# if defined(__GNUC__)
#   if !(defined(__cplusplus)) || (__GNUC__ > 3 || (__GNUC__ == 3 && __GNUC_MINOR__
>= 4))
#     define CYTHON_UNUSED __attribute__ ((__unused__))
#   else
#     define CYTHON_UNUSED
#   endif
# elif defined(__ICC) || (defined(__INTEL_COMPILER) && !defined(_MSC_VER))
#   define CYTHON_UNUSED __attribute__ ((__unused__))
# else
#   define CYTHON_UNUSED
# endif
#endif
#ifndef CYTHON_MAYBE_UNUSED_VAR
```

```
#  if defined(__cplusplus)
     template<class T> void CYTHON_MAYBE_UNUSED_VAR( const T& ) { }
#  else
#    define CYTHON_MAYBE_UNUSED_VAR(x) (void)(x)
#  endif
#endif
#ifndef CYTHON_NCP_UNUSED
# if CYTHON_COMPILING_IN_CPYTHON
#  define CYTHON_NCP_UNUSED
# else
#  define CYTHON_NCP_UNUSED CYTHON_UNUSED
# endif
#endif
#define __Pyx_void_to_None(void_result) ((void)(void_result), Py_INCREF(Py_None), Py_None)
#ifdef _MSC_VER
    #ifndef _MSC_STDINT_H_
        #if _MSC_VER < 1300
           typedef unsigned char     uint8_t;
           typedef unsigned int      uint32_t;
        #else
           typedef unsigned __int8   uint8_t;
           typedef unsigned __int32  uint32_t;
        #endif
    #endif
#else
   #include <stdint.h>
#endif
#ifndef CYTHON_FALLTHROUGH
  #if defined(__cplusplus) && __cplusplus >= 201103L
    #if __has_cpp_attribute(fallthrough)
      #define CYTHON_FALLTHROUGH [[fallthrough]]
    #elif __has_cpp_attribute(clang::fallthrough)
      #define CYTHON_FALLTHROUGH [[clang::fallthrough]]
    #elif __has_cpp_attribute(gnu::fallthrough)
      #define CYTHON_FALLTHROUGH [[gnu::fallthrough]]
    #endif
  #endif
  #ifndef CYTHON_FALLTHROUGH
    #if __has_attribute(fallthrough)
      #define CYTHON_FALLTHROUGH __attribute__((fallthrough))
    #else
      #define CYTHON_FALLTHROUGH
    #endif
  #endif
  #if defined(__clang__ ) && defined(__apple_build_version__)
    #if __apple_build_version__ < 7000000
      #undef  CYTHON_FALLTHROUGH
      #define CYTHON_FALLTHROUGH
    #endif
  #endif
#endif

#ifndef CYTHON_INLINE
  #if defined(__clang__)
    #define CYTHON_INLINE __inline__ __attribute__ ((__unused__))
  #elif defined(__GNUC__)
    #define CYTHON_INLINE __inline__
  #elif defined(_MSC_VER)
    #define CYTHON_INLINE __inline
  #elif defined (__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
    #define CYTHON_INLINE inline
  #else
    #define CYTHON_INLINE
  #endif
#endif

#if CYTHON_COMPILING_IN_PYPY && PY_VERSION_HEX < 0x02070600 && !defined(Py_OptimizeFlag)
  #define Py_OptimizeFlag 0
#endif
#define __PYX_BUILD_PY_SSIZE_T "n"
#define CYTHON_FORMAT_SSIZE_T "z"
#if PY_MAJOR_VERSION < 3
  #define __Pyx_BUILTIN_MODULE_NAME "__builtin__"
  #define __Pyx_PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)\
```

```c
          PyCode_New(a+k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)
  #define __Pyx_DefaultClassType PyClass_Type
#else
  #define __Pyx_BUILTIN_MODULE_NAME "builtins"
  #define __Pyx_DefaultClassType PyType_Type
#if PY_VERSION_HEX >= 0x030B00A1
    static CYTHON_INLINE PyCodeObject* __Pyx_PyCode_New(int a, int k, int l, int s, int f,
                                                    PyObject *code, PyObject *c, PyObject* n, PyObject *v,
                                                    PyObject *fv, PyObject *cell, PyObject* fn,
                                                    PyObject *name, int fline, PyObject *lnos) {
        PyObject *kwds=NULL, *argcount=NULL, *posonlyargcount=NULL, *kwonlyargcount=NULL;
        PyObject *nlocals=NULL, *stacksize=NULL, *flags=NULL, *replace=NULL, *call_result=NULL, *empty=NULL;
        const char *fn_cstr=NULL;
        const char *name_cstr=NULL;
        PyCodeObject* co=NULL;
        PyObject *type, *value, *traceback;
        PyErr_Fetch(&type, &value, &traceback);
        if (!(kwds=PyDict_New())) goto end;
        if (!(argcount=PyLong_FromLong(a))) goto end;
        if (PyDict_SetItemString(kwds, "co_argcount", argcount) != 0) goto end;
        if (!(posonlyargcount=PyLong_FromLong(0))) goto end;
        if (PyDict_SetItemString(kwds, "co_posonlyargcount", posonlyargcount) != 0) goto end;
        if (!(kwonlyargcount=PyLong_FromLong(k))) goto end;
        if (PyDict_SetItemString(kwds, "co_kwonlyargcount", kwonlyargcount) != 0) goto end;
        if (!(nlocals=PyLong_FromLong(l))) goto end;
        if (PyDict_SetItemString(kwds, "co_nlocals", nlocals) != 0) goto end;
        if (!(stacksize=PyLong_FromLong(s))) goto end;
        if (PyDict_SetItemString(kwds, "co_stacksize", stacksize) != 0) goto end;
        if (!(flags=PyLong_FromLong(f))) goto end;
        if (PyDict_SetItemString(kwds, "co_flags", flags) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_code", code) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_consts", c) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_names", n) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_varnames", v) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_freevars", fv) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_cellvars", cell) != 0) goto end;
        if (PyDict_SetItemString(kwds, "co_linetable", lnos) != 0) goto end;
        if (!(fn_cstr=PyUnicode_AsUTF8AndSize(fn, NULL))) goto end;
        if (!(name_cstr=PyUnicode_AsUTF8AndSize(name, NULL))) goto end;
        if (!(co = PyCode_NewEmpty(fn_cstr, name_cstr, fline))) goto end;
        if (!(replace = PyObject_GetAttrString((PyObject*)co, "replace"))) goto cleanup_code_too;
        if (!(empty = PyTuple_New(0))) goto cleanup_code_too; // unfortunately __pyx_empty_tuple isn't available here
        if (!(call_result = PyObject_Call(replace, empty, kwds))) goto cleanup_code_too;
        Py_XDECREF((PyObject*)co);
        co = (PyCodeObject*)call_result;
        call_result = NULL;
        if (0) {
            cleanup_code_too:
            Py_XDECREF((PyObject*)co);
            co = NULL;
        }
        end:
        Py_XDECREF(kwds);
        Py_XDECREF(argcount);
        Py_XDECREF(posonlyargcount);
        Py_XDECREF(kwonlyargcount);
        Py_XDECREF(nlocals);
        Py_XDECREF(stacksize);
        Py_XDECREF(replace);
        Py_XDECREF(call_result);
        Py_XDECREF(empty);
        if (type) {
            PyErr_Restore(type, value, traceback);
        }
        return co;
    }
```

```
#else
  #define __Pyx_PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)\
          PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)
#endif
  #define __Pyx_DefaultClassType PyType_Type
#endif
#ifndef Py_TPFLAGS_CHECKTYPES
  #define Py_TPFLAGS_CHECKTYPES 0
#endif
#ifndef Py_TPFLAGS_HAVE_INDEX
  #define Py_TPFLAGS_HAVE_INDEX 0
#endif
#ifndef Py_TPFLAGS_HAVE_NEWBUFFER
  #define Py_TPFLAGS_HAVE_NEWBUFFER 0
#endif
#ifndef Py_TPFLAGS_HAVE_FINALIZE
  #define Py_TPFLAGS_HAVE_FINALIZE 0
#endif
#ifndef METH_STACKLESS
  #define METH_STACKLESS 0
#endif
#if PY_VERSION_HEX <= 0x030700A3 || !defined(METH_FASTCALL)
  #ifndef METH_FASTCALL
     #define METH_FASTCALL 0x80
  #endif
  typedef PyObject *(*__Pyx_PyCFunctionFast) (PyObject *self, PyObject *const *args, Py_ssize_t nargs);
  typedef PyObject *(*__Pyx_PyCFunctionFastWithKeywords) (PyObject *self, PyObject *const *args,
                                                          Py_ssize_t nargs, PyObject *kwnames);
#else
  #define __Pyx_PyCFunctionFast _PyCFunctionFast
  #define __Pyx_PyCFunctionFastWithKeywords _PyCFunctionFastWithKeywords
#endif
#if CYTHON_FAST_PYCCALL
#define __Pyx_PyFastCFunction_Check(func)\
    ((PyCFunction_Check(func) && (METH_FASTCALL == (PyCFunction_GET_FLAGS(func) & ~(METH_CLASS | METH_STATIC | METH_COEXIST | METH_KEYWORDS | METH_STACKLESS)))))
#else
#define __Pyx_PyFastCFunction_Check(func) 0
#endif
#if CYTHON_COMPILING_IN_PYPY && !defined(PyObject_Malloc)
  #define PyObject_Malloc(s)   PyMem_Malloc(s)
  #define PyObject_Free(p)     PyMem_Free(p)
  #define PyObject_Realloc(p)  PyMem_Realloc(p)
#endif
#if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX < 0x030400A1
  #define PyMem_RawMalloc(n)           PyMem_Malloc(n)
  #define PyMem_RawRealloc(p, n)       PyMem_Realloc(p, n)
  #define PyMem_RawFree(p)             PyMem_Free(p)
#endif
#if CYTHON_COMPILING_IN_PYSTON
  #define __Pyx_PyCode_HasFreeVars(co)  PyCode_HasFreeVars(co)
  #define __Pyx_PyFrame_SetLineNumber(frame, lineno) PyFrame_SetLineNumber(frame, lineno)
#else
  #define __Pyx_PyCode_HasFreeVars(co)  (PyCode_GetNumFree(co) > 0)
  #define __Pyx_PyFrame_SetLineNumber(frame, lineno)  (frame)->f_lineno = (lineno)
#endif
#if !CYTHON_FAST_THREAD_STATE || PY_VERSION_HEX < 0x02070000
  #define __Pyx_PyThreadState_Current PyThreadState_GET()
#elif PY_VERSION_HEX >= 0x03060000
  #define __Pyx_PyThreadState_Current _PyThreadState_UncheckedGet()
#elif PY_VERSION_HEX >= 0x03000000
  #define __Pyx_PyThreadState_Current PyThreadState_GET()
#else
  #define __Pyx_PyThreadState_Current _PyThreadState_Current
#endif
#if PY_VERSION_HEX < 0x030700A2 && !defined(PyThread_tss_create) && !defined(Py_tss_NEEDS_INIT)
#include "pythread.h"
#define Py_tss_NEEDS_INIT 0
typedef int Py_tss_t;
static CYTHON_INLINE int PyThread_tss_create(Py_tss_t *key) {
  *key = PyThread_create_key();
  return 0;
}
static CYTHON_INLINE Py_tss_t * PyThread_tss_alloc(void) {
  Py_tss_t *key = (Py_tss_t *)PyObject_Malloc(sizeof(Py_tss_t));
  *key = Py_tss_NEEDS_INIT;
  return key;
}
static CYTHON_INLINE void PyThread_tss_free(Py_tss_t *key) {
  PyObject_Free(key);
}
static CYTHON_INLINE int PyThread_tss_is_created(Py_tss_t *key) {
  return *key != Py_tss_NEEDS_INIT;
}
static CYTHON_INLINE void PyThread_tss_delete(Py_tss_t *key) {
  PyThread_delete_key(*key);
  *key = Py_tss_NEEDS_INIT;
}
static CYTHON_INLINE int PyThread_tss_set(Py_tss_t *key, void *value) {
  return PyThread_set_key_value(*key, value);
}
static CYTHON_INLINE void * PyThread_tss_get(Py_tss_t *key) {
  return PyThread_get_key_value(*key);
}
#endif
#if CYTHON_COMPILING_IN_CPYTHON || defined(_PyDict_NewPresized)
#define __Pyx_PyDict_NewPresized(n)  ((n <= 8) ? PyDict_New() : _PyDict_NewPresized(n))
#else
#define __Pyx_PyDict_NewPresized(n)  PyDict_New()
#endif
#if PY_MAJOR_VERSION >= 3 || CYTHON_FUTURE_DIVISION
  #define __Pyx_PyNumber_Divide(x,y)         PyNumber_TrueDivide(x,y)
  #define __Pyx_PyNumber_InPlaceDivide(x,y)  PyNumber_InPlaceTrueDivide(x,y)
#else
  #define __Pyx_PyNumber_Divide(x,y)         PyNumber_Divide(x,y)
  #define __Pyx_PyNumber_InPlaceDivide(x,y)  PyNumber_InPlaceDivide(x,y)
#endif
#if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX >= 0x030500A1 && CYTHON_USE_UNICODE_INTERNALS
#define __Pyx_PyDict_GetItemStr(dict, name)  _PyDict_GetItem_KnownHash(dict, name, ((PyASCIIObject *) name)->hash)
#else
#define __Pyx_PyDict_GetItemStr(dict, name)  PyDict_GetItem(dict, name)
#endif
#if PY_VERSION_HEX > 0x03030000 && defined(PyUnicode_KIND)
  #define CYTHON_PEP393_ENABLED 1
  #if defined(PyUnicode_IS_READY)
  #define __Pyx_PyUnicode_READY(op)       (likely(PyUnicode_IS_READY(op)) ?\
                                              0 : _PyUnicode_Ready((PyObject *)(op)))
  #else
  #define __Pyx_PyUnicode_READY(op)       (0)
  #endif
  #define __Pyx_PyUnicode_GET_LENGTH(u)    PyUnicode_GET_LENGTH(u)
  #define __Pyx_PyUnicode_READ_CHAR(u, i) PyUnicode_READ_CHAR(u, i)
  #define __Pyx_PyUnicode_MAX_CHAR_VALUE(u)    PyUnicode_MAX_CHAR_VALUE(u)
  #define __Pyx_PyUnicode_KIND(u)          PyUnicode_KIND(u)
  #define __Pyx_PyUnicode_DATA(u)          PyUnicode_DATA(u)
  #define __Pyx_PyUnicode_READ(k, d, i)    PyUnicode_READ(k, d, i)
  #define __Pyx_PyUnicode_WRITE(k, d, i, ch)  PyUnicode_WRITE(k, d, i, ch)
  #if defined(PyUnicode_IS_READY) && defined(PyUnicode_GET_SIZE)
  #if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX >= 0x03090000
  #define __Pyx_PyUnicode_IS_TRUE(u)      (0 != (likely(PyUnicode_IS_READY(u)) ? PyUnicode_GET_LENGTH(u) : ((PyCompactUnicodeObject *)(u))->wstr_length))
  #else
  #define __Pyx_PyUnicode_IS_TRUE(u)      (0 != (likely(PyUnicode_IS_READY(u)) ? PyUnicode_GET_LENGTH(u) : PyUnicode_GET_SIZE(u)))
  #endif
  #else
  #define __Pyx_PyUnicode_IS_TRUE(u)      (0 != PyUnicode_GET_LENGTH(u))
  #endif
#else
  #define CYTHON_PEP393_ENABLED 0
  #define PyUnicode_1BYTE_KIND  1
  #define PyUnicode_2BYTE_KIND  2
  #define PyUnicode_4BYTE_KIND  4
  #define __Pyx_PyUnicode_READY(op)       (0)
  #define __Pyx_PyUnicode_GET_LENGTH(u)    PyUnicode_GET_SIZE(u)
  #define __Pyx_PyUnicode_READ_CHAR(u, i) ((Py_UCS4)(PyUnicode_AS_UNICODE(u)[i]))
  #define __Pyx_PyUnicode_MAX_CHAR_VALUE(u)    ((sizeof(Py_UNICODE) == 2) ? 65535 :
1114111)
  #define __Pyx_PyUnicode_KIND(u)          (sizeof(Py_UNICODE))
  #define __Pyx_PyUnicode_DATA(u)          ((void*)PyUnicode_AS_UNICODE(u))
  #define __Pyx_PyUnicode_READ(k, d, i)    ((void)(k),
(Py_UCS4)(((Py_UNICODE*)d)[i]))
  #define __Pyx_PyUnicode_WRITE(k, d, i, ch)   (((void)(k)), ((Py_UNICODE*)d)[i] =
ch)
  #define __Pyx_PyUnicode_IS_TRUE(u)       (0 != PyUnicode_GET_SIZE(u))
#endif
#if CYTHON_COMPILING_IN_PYPY
  #define __Pyx_PyUnicode_Concat(a, b)       PyNumber_Add(a, b)
  #define __Pyx_PyUnicode_ConcatSafe(a, b)   PyNumber_Add(a, b)
#else
  #define __Pyx_PyUnicode_Concat(a, b)       PyUnicode_Concat(a, b)
  #define __Pyx_PyUnicode_ConcatSafe(a, b)   ((unlikely((a) == Py_None) ||
unlikely((b) == Py_None)) ?\
      PyNumber_Add(a, b) : __Pyx_PyUnicode_Concat(a, b))
#endif
#if CYTHON_COMPILING_IN_PYPY && !defined(PyUnicode_Contains)
  #define PyUnicode_Contains(u, s)  PySequence_Contains(u, s)
#endif
#if CYTHON_COMPILING_IN_PYPY && !defined(PyByteArray_Check)
  #define PyByteArray_Check(obj)  PyObject_TypeCheck(obj, &PyByteArray_Type)
#endif
#if CYTHON_COMPILING_IN_PYPY && !defined(PyObject_Format)
  #define PyObject_Format(obj, fmt)  PyObject_CallMethod(obj, "__format__", "O",
fmt)
#endif
#define __Pyx_PyString_FormatSafe(a, b)   ((unlikely((a) == Py_None ||
(PyString_Check(b) && !PyString_CheckExact(b)))) ? PyNumber_Remainder(a, b) :
__Pyx_PyString_Format(a, b))
#define __Pyx_PyUnicode_FormatSafe(a, b)  ((unlikely((a) == Py_None ||
(PyUnicode_Check(b)  && !PyUnicode_CheckExact(b)))) ? PyNumber_Remainder(a, b) :
PyUnicode_Format(a, b))
#if PY_MAJOR_VERSION >= 3
  #define __Pyx_PyString_Format(a, b)  PyUnicode_Format(a, b)
#else
  #define __Pyx_PyString_Format(a, b)  PyString_Format(a, b)
#endif
#if PY_MAJOR_VERSION < 3 && !defined(PyObject_ASCII)
  #define PyObject_ASCII(o)            PyObject_Repr(o)
#endif
#if PY_MAJOR_VERSION >= 3
  #define PyBaseString_Type            PyUnicode_Type
  #define PyStringObject               PyUnicodeObject
  #define PyString_Type                PyUnicode_Type
  #define PyString_Check               PyUnicode_Check
  #define PyString_CheckExact          PyUnicode_CheckExact
#ifndef PyObject_Unicode
  #define PyObject_Unicode             PyObject_Str
#endif
#endif
#if PY_MAJOR_VERSION >= 3
  #define __Pyx_PyBaseString_Check(obj) PyUnicode_Check(obj)
  #define __Pyx_PyBaseString_CheckExact(obj) PyUnicode_CheckExact(obj)
#else
  #define __Pyx_PyBaseString_Check(obj) (PyString_Check(obj) ||
PyUnicode_Check(obj))
  #define __Pyx_PyBaseString_CheckExact(obj) (PyString_CheckExact(obj) ||
PyUnicode_CheckExact(obj))
#endif
#ifndef PySet_CheckExact
  #define PySet_CheckExact(obj)        (Py_TYPE(obj) == &PySet_Type)
#endif
#if PY_VERSION_HEX >= 0x030900A4
  #define __Pyx_SET_REFCNT(obj, refcnt) Py_SET_REFCNT(obj, refcnt)
  #define __Pyx_SET_SIZE(obj, size) Py_SET_SIZE(obj, size)
#else
  #define __Pyx_SET_REFCNT(obj, refcnt) Py_REFCNT(obj) = (refcnt)
  #define __Pyx_SET_SIZE(obj, size) Py_SIZE(obj) = (size)
#endif
#if CYTHON_ASSUME_SAFE_MACROS
  #define __Pyx_PySequence_SIZE(seq)  Py_SIZE(seq)
#else
```

```
  #define __Pyx_PySequence_SIZE(seq)  PySequence_Size(seq)
#endif
#if PY_MAJOR_VERSION >= 3
  #define PyIntObject                  PyLongObject
  #define PyInt_Type                   PyLong_Type
  #define PyInt_Check(op)              PyLong_Check(op)
  #define PyInt_CheckExact(op)         PyLong_CheckExact(op)
  #define PyInt_FromString             PyLong_FromString
  #define PyInt_FromUnicode            PyLong_FromUnicode
  #define PyInt_FromLong               PyLong_FromLong
  #define PyInt_FromSize_t             PyLong_FromSize_t
  #define PyInt_FromSsize_t            PyLong_FromSsize_t
  #define PyInt_AsLong                 PyLong_AsLong
  #define PyInt_AS_LONG                PyLong_AS_LONG
  #define PyInt_AsSsize_t              PyLong_AsSsize_t
  #define PyInt_AsUnsignedLongMask     PyLong_AsUnsignedLongMask
  #define PyInt_AsUnsignedLongLongMask PyLong_AsUnsignedLongLongMask
  #define PyNumber_Int                 PyNumber_Long
#endif
#if PY_MAJOR_VERSION >= 3
  #define PyBoolObject                 PyLongObject
#endif
#if PY_MAJOR_VERSION >= 3 && CYTHON_COMPILING_IN_PYPY
  #ifndef PyUnicode_InternFromString
    #define PyUnicode_InternFromString(s) PyUnicode_FromString(s)
  #endif
#endif
#if PY_VERSION_HEX < 0x030200A4
  typedef long Py_hash_t;
  #define __Pyx_PyInt_FromHash_t PyInt_FromLong
  #define __Pyx_PyInt_AsHash_t   __Pyx_PyIndex_AsHash_t
#else
  #define __Pyx_PyInt_FromHash_t PyInt_FromSsize_t
  #define __Pyx_PyInt_AsHash_t   __Pyx_PyIndex_AsSsize_t
#endif
#if PY_MAJOR_VERSION >= 3
  #define __Pyx_PyMethod_New(func, self, klass) ((self) ? ((void)(klass),
PyMethod_New(func, self)) : __Pyx_NewRef(func))
#else
  #define __Pyx_PyMethod_New(func, self, klass) PyMethod_New(func, self, klass)
#endif
#if CYTHON_USE_ASYNC_SLOTS
  #if PY_VERSION_HEX >= 0x030500B1
    #define __Pyx_PyAsyncMethodsStruct PyAsyncMethods
    #define __Pyx_PyType_AsAsync(obj) (Py_TYPE(obj)->tp_as_async)
  #else
    #define __Pyx_PyType_AsAsync(obj) ((__Pyx_PyAsyncMethodsStruct*)
(Py_TYPE(obj)->tp_reserved))
  #endif
#else
  #define __Pyx_PyType_AsAsync(obj) NULL
#endif
#ifndef __Pyx_PyAsyncMethodsStruct
    typedef struct {
        unaryfunc am_await;
        unaryfunc am_aiter;
        unaryfunc am_anext;
    } __Pyx_PyAsyncMethodsStruct;
#endif

#if defined(_WIN32) || defined(WIN32) || defined(MS_WINDOWS)
  #if !defined(_USE_MATH_DEFINES)
    #define _USE_MATH_DEFINES
  #endif
#endif
#include <math.h>
#ifdef NAN
#define __PYX_NAN() ((float) NAN)
#else
static CYTHON_INLINE float __PYX_NAN() {
  float value;
  memset(&value, 0xFF, sizeof(value));
  return value;
}
#endif
#if defined(__CYGWIN__) && defined(_LDBL_EQ_DBL)
#define __Pyx_truncl trunc
```

```
#else
#define __Pyx_truncl truncl
#endif

#define __PYX_MARK_ERR_POS(f_index, lineno) \
    { __pyx_filename = __pyx_f[f_index]; (void)__pyx_filename; __pyx_lineno = lineno; (void)__pyx_lineno; __pyx_clineno = __LINE__; (void)__pyx_clineno; }
#define __PYX_ERR(f_index, lineno, Ln_error) \
    { __PYX_MARK_ERR_POS(f_index, lineno) goto Ln_error; }

#ifndef __PYX_EXTERN_C
  #ifdef __cplusplus
    #define __PYX_EXTERN_C extern "C"
  #else
    #define __PYX_EXTERN_C extern
  #endif
#endif

#define __PYX_HAVE__API_Calls___main_
#define __PYX_HAVE_API__API_Calls___main_
/* Early includes */
#ifdef _OPENMP
#include <omp.h>
#endif /* _OPENMP */

#if defined(PYREX_WITHOUT_ASSERTIONS) && !defined(CYTHON_WITHOUT_ASSERTIONS)
#define CYTHON_WITHOUT_ASSERTIONS
#endif

typedef struct {PyObject **p; const char *s; const Py_ssize_t n; const char* encoding;
                const char is_unicode; const char is_str; const char intern; } __Pyx_StringTabEntry;

#define __PYX_DEFAULT_STRING_ENCODING_IS_ASCII 0
#define __PYX_DEFAULT_STRING_ENCODING_IS_UTF8 0
#define __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT (PY_MAJOR_VERSION >= 3 && __PYX_DEFAULT_STRING_ENCODING_IS_UTF8)
#define __PYX_DEFAULT_STRING_ENCODING ""
#define __Pyx_PyObject_FromString __Pyx_PyBytes_FromString
#define __Pyx_PyObject_FromStringAndSize __Pyx_PyBytes_FromStringAndSize
#define __Pyx_uchar_cast(c) ((unsigned char)c)
#define __Pyx_long_cast(x) ((long)x)
#define __Pyx_fits_Py_ssize_t(v, type, is_signed)  (\
    (sizeof(type) < sizeof(Py_ssize_t))  ||\
    (sizeof(type) > sizeof(Py_ssize_t) &&\
          likely(v < (type)PY_SSIZE_T_MAX ||\
                 v == (type)PY_SSIZE_T_MAX)  &&\
          (!is_signed || likely(v > (type)PY_SSIZE_T_MIN ||\
                                v == (type)PY_SSIZE_T_MIN)))  ||\
    (sizeof(type) == sizeof(Py_ssize_t) &&\
          (is_signed || likely(v < (type)PY_SSIZE_T_MAX ||\
                               v == (type)PY_SSIZE_T_MAX)))  )
static CYTHON_INLINE int __Pyx_is_valid_index(Py_ssize_t i, Py_ssize_t limit) {
    return (size_t) i < (size_t) limit;
}
#if defined (__cplusplus) && __cplusplus >= 201103L
    #include <cstdlib>
    #define __Pyx_sst_abs(value) std::abs(value)
#elif SIZEOF_INT >= SIZEOF_SIZE_T
    #define __Pyx_sst_abs(value) abs(value)
#elif SIZEOF_LONG >= SIZEOF_SIZE_T
    #define __Pyx_sst_abs(value) labs(value)
#elif defined (_MSC_VER)
    #define __Pyx_sst_abs(value) ((Py_ssize_t)_abs64(value))
#elif defined (__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
    #define __Pyx_sst_abs(value) llabs(value)
#elif defined (__GNUC__)
    #define __Pyx_sst_abs(value) __builtin_llabs(value)
#else
    #define __Pyx_sst_abs(value) ((value<0) ? -value : value)
#endif
static CYTHON_INLINE const char* __Pyx_PyObject_AsString(PyObject*);
static CYTHON_INLINE const char* __Pyx_PyObject_AsStringAndSize(PyObject*, Py_ssize_t* length);
#define __Pyx_PyByteArray_FromString(s) PyByteArray_FromStringAndSize((const char*)s, strlen((const char*)s))
```

```c
#define __Pyx_PyByteArray_FromStringAndSize(s, l)
PyByteArray_FromStringAndSize((const char*)s, l)
#define __Pyx_PyBytes_FromString        PyBytes_FromString
#define __Pyx_PyBytes_FromStringAndSize PyBytes_FromStringAndSize
static CYTHON_INLINE PyObject* __Pyx_PyUnicode_FromString(const char*);
#if PY_MAJOR_VERSION < 3
    #define __Pyx_PyStr_FromString        __Pyx_PyBytes_FromString
    #define __Pyx_PyStr_FromStringAndSize __Pyx_PyBytes_FromStringAndSize
#else
    #define __Pyx_PyStr_FromString        __Pyx_PyUnicode_FromString
    #define __Pyx_PyStr_FromStringAndSize __Pyx_PyUnicode_FromStringAndSize
#endif
#define __Pyx_PyBytes_AsWritableString(s)     ((char*) PyBytes_AS_STRING(s))
#define __Pyx_PyBytes_AsWritableSString(s)    ((signed char*)
PyBytes_AS_STRING(s))
#define __Pyx_PyBytes_AsWritableUString(s)    ((unsigned char*)
PyBytes_AS_STRING(s))
#define __Pyx_PyBytes_AsString(s)     ((const char*) PyBytes_AS_STRING(s))
#define __Pyx_PyBytes_AsSString(s)    ((const signed char*) PyBytes_AS_STRING(s))
#define __Pyx_PyBytes_AsUString(s)    ((const unsigned char*)
PyBytes_AS_STRING(s))
#define __Pyx_PyObject_AsWritableString(s)    ((char*)
__Pyx_PyObject_AsString(s))
#define __Pyx_PyObject_AsWritableSString(s)    ((signed char*)
__Pyx_PyObject_AsString(s))
#define __Pyx_PyObject_AsWritableUString(s)    ((unsigned char*)
__Pyx_PyObject_AsString(s))
#define __Pyx_PyObject_AsSString(s)    ((const signed char*)
__Pyx_PyObject_AsString(s))
#define __Pyx_PyObject_AsUString(s)    ((const unsigned char*)
__Pyx_PyObject_AsString(s))
#define __Pyx_PyObject_FromCString(s)  __Pyx_PyObject_FromString((const char*)s)
#define __Pyx_PyBytes_FromCString(s)   __Pyx_PyBytes_FromString((const char*)s)
#define __Pyx_PyByteArray_FromCString(s)   __Pyx_PyByteArray_FromString((const
char*)s)
#define __Pyx_PyStr_FromCString(s)     __Pyx_PyStr_FromString((const char*)s)
#define __Pyx_PyUnicode_FromCString(s) __Pyx_PyUnicode_FromString((const char*)s)
static CYTHON_INLINE size_t __Pyx_Py_UNICODE_strlen(const Py_UNICODE *u) {
    const Py_UNICODE *u_end = u;
    while (*u_end++) ;
    return (size_t)(u_end - u - 1);
}
#define __Pyx_PyUnicode_FromUnicode(u)       PyUnicode_FromUnicode(u,
__Pyx_Py_UNICODE_strlen(u))
#define __Pyx_PyUnicode_FromUnicodeAndLength PyUnicode_FromUnicode
#define __Pyx_PyUnicode_AsUnicode            PyUnicode_AsUnicode
#define __Pyx_NewRef(obj) (Py_INCREF(obj), obj)
#define __Pyx_Owned_Py_None(b) __Pyx_NewRef(Py_None)
static CYTHON_INLINE PyObject * __Pyx_PyBool_FromLong(long b);
static CYTHON_INLINE int __Pyx_PyObject_IsTrue(PyObject*);
static CYTHON_INLINE int __Pyx_PyObject_IsTrueAndDecref(PyObject*);
static CYTHON_INLINE PyObject* __Pyx_PyNumber_IntOrLong(PyObject* x);
#define __Pyx_PySequence_Tuple(obj)\
    (likely(PyTuple_CheckExact(obj)) ? __Pyx_NewRef(obj) : PySequence_Tuple(obj))
static CYTHON_INLINE Py_ssize_t __Pyx_PyIndex_AsSsize_t(PyObject*);
static CYTHON_INLINE PyObject * __Pyx_PyInt_FromSize_t(size_t);
static CYTHON_INLINE Py_hash_t __Pyx_PyIndex_AsHash_t(PyObject*);
#if CYTHON_ASSUME_SAFE_MACROS
#define __pyx_PyFloat_AsDouble(x) (PyFloat_CheckExact(x) ? PyFloat_AS_DOUBLE(x) :
PyFloat_AsDouble(x))
#else
#define __pyx_PyFloat_AsDouble(x) PyFloat_AsDouble(x)
#endif
#define __pyx_PyFloat_AsFloat(x) ((float) __pyx_PyFloat_AsDouble(x))
#if PY_MAJOR_VERSION >= 3
#define __Pyx_PyNumber_Int(x) (PyLong_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Long(x))
#else
#define __Pyx_PyNumber_Int(x) (PyInt_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Int(x))
#endif
#define __Pyx_PyNumber_Float(x) (PyFloat_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Float(x))
#if PY_MAJOR_VERSION < 3 && __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
static int __Pyx_sys_getdefaultencoding_not_ascii;
static int __Pyx_init_sys_getdefaultencoding_params(void) {
    PyObject* sys;
    PyObject* default_encoding = NULL;
    PyObject* ascii_chars_u = NULL;
    PyObject* ascii_chars_b = NULL;
    const char* default_encoding_c;
    sys = PyImport_ImportModule("sys");
    if (!sys) goto bad;
    default_encoding = PyObject_CallMethod(sys, (char*) "getdefaultencoding", NULL);
    Py_DECREF(sys);
    if (!default_encoding) goto bad;
    default_encoding_c = PyBytes_AsString(default_encoding);
    if (!default_encoding_c) goto bad;
    if (strcmp(default_encoding_c, "ascii") == 0) {
        __Pyx_sys_getdefaultencoding_not_ascii = 0;
    } else {
        char ascii_chars[128];
        int c;
        for (c = 0; c < 128; c++) {
            ascii_chars[c] = c;
        }
        __Pyx_sys_getdefaultencoding_not_ascii = 1;
        ascii_chars_u = PyUnicode_DecodeASCII(ascii_chars, 128, NULL);
        if (!ascii_chars_u) goto bad;
        ascii_chars_b = PyUnicode_AsEncodedString(ascii_chars_u, default_encoding_c, NULL);
        if (!ascii_chars_b || !PyBytes_Check(ascii_chars_b) || memcmp(ascii_chars, PyBytes_AS_STRING(ascii_chars_b), 128) != 0) {
            PyErr_Format(
                PyExc_ValueError,
                "This module compiled with c_string_encoding=ascii, but default encoding '%.200s' is not a superset of ascii.",
                default_encoding_c);
            goto bad;
        }
        Py_DECREF(ascii_chars_u);
        Py_DECREF(ascii_chars_b);
    }
    Py_DECREF(default_encoding);
    return 0;
bad:
    Py_XDECREF(default_encoding);
    Py_XDECREF(ascii_chars_u);
    Py_XDECREF(ascii_chars_b);
    return -1;
}
#endif
#if __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT && PY_MAJOR_VERSION >= 3
#define __Pyx_PyUnicode_FromStringAndSize(c_str, size) PyUnicode_DecodeUTF8(c_str, size, NULL)
#else
#define __Pyx_PyUnicode_FromStringAndSize(c_str, size) PyUnicode_Decode(c_str, size, __PYX_DEFAULT_STRING_ENCODING, NULL)
#if __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
static char* __PYX_DEFAULT_STRING_ENCODING;
static int __Pyx_init_sys_getdefaultencoding_params(void) {
    PyObject* sys;
    PyObject* default_encoding = NULL;
    char* default_encoding_c;
    sys = PyImport_ImportModule("sys");
    if (!sys) goto bad;
    default_encoding = PyObject_CallMethod(sys, (char*) (const char*) "getdefaultencoding", NULL);
    Py_DECREF(sys);
    if (!default_encoding) goto bad;
    default_encoding_c = PyBytes_AsString(default_encoding);
    if (!default_encoding_c) goto bad;
    __PYX_DEFAULT_STRING_ENCODING = (char*) malloc(strlen(default_encoding_c) + 1);
    if (!__PYX_DEFAULT_STRING_ENCODING) goto bad;
    strcpy(__PYX_DEFAULT_STRING_ENCODING, default_encoding_c);
    Py_DECREF(default_encoding);
    return 0;
bad:
    Py_XDECREF(default_encoding);
    return -1;
}
#endif
```

```c
#endif


/* Test for GCC > 2.95 */
#if defined(__GNUC__)     && (__GNUC__ > 2 || (__GNUC__ == 2 && (__GNUC_MINOR__ > 95)))
  #define likely(x)   __builtin_expect(!!(x), 1)
  #define unlikely(x) __builtin_expect(!!(x), 0)
#else /* !__GNUC__ or GCC < 2.95 */
  #define likely(x)   (x)
  #define unlikely(x) (x)
#endif /* __GNUC__ */
static CYTHON_INLINE void __Pyx_pretend_to_initialize(void* ptr) { (void)ptr; }

static PyObject *__pyx_m = NULL;
static PyObject *__pyx_d;
static PyObject *__pyx_b;
static PyObject *__pyx_cython_runtime = NULL;
static PyObject *__pyx_empty_tuple;
static PyObject *__pyx_empty_bytes;
static PyObject *__pyx_empty_unicode;
static int __pyx_lineno;
static int __pyx_clineno = 0;
static const char * __pyx_cfilenm= __FILE__;
static const char *__pyx_filename;


static const char *__pyx_f[] = {
  "_main_.py",
};

/*--- Type declarations ---*/

/* --- Runtime support code (head) --- */
/* Refnanny.proto */
#ifndef CYTHON_REFNANNY
  #define CYTHON_REFNANNY 0
#endif
#if CYTHON_REFNANNY
  typedef struct {
    void (*INCREF)(void*, PyObject*, int);
    void (*DECREF)(void*, PyObject*, int);
    void (*GOTREF)(void*, PyObject*, int);
    void (*GIVEREF)(void*, PyObject*, int);
    void* (*SetupContext)(const char*, int, const char*);
    void (*FinishContext)(void**);
  } __Pyx_RefNannyAPIStruct;
  static __Pyx_RefNannyAPIStruct *__Pyx_RefNanny = NULL;
  static __Pyx_RefNannyAPIStruct *__Pyx_RefNannyImportAPI(const char *modname);
  #define __Pyx_RefNannyDeclarations void *__pyx_refnanny = NULL;
#ifdef WITH_THREAD
  #define __Pyx_RefNannySetupContext(name, acquire_gil)\
          if (acquire_gil) {\
              PyGILState_STATE __pyx_gilstate_save = PyGILState_Ensure();\
              __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__, __FILE__);\
              PyGILState_Release(__pyx_gilstate_save);\
          } else {\
              __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__, __FILE__);\
          }
#else
  #define __Pyx_RefNannySetupContext(name, acquire_gil)\
          __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__, __FILE__)
#endif
  #define __Pyx_RefNannyFinishContext()\
          __Pyx_RefNanny->FinishContext(&__pyx_refnanny)
  #define __Pyx_INCREF(r)  __Pyx_RefNanny->INCREF(__pyx_refnanny, (PyObject *)(r), __LINE__)
  #define __Pyx_DECREF(r)  __Pyx_RefNanny->DECREF(__pyx_refnanny, (PyObject *)(r), __LINE__)
  #define __Pyx_GOTREF(r)  __Pyx_RefNanny->GOTREF(__pyx_refnanny, (PyObject *)(r), __LINE__)
  #define __Pyx_GIVEREF(r) __Pyx_RefNanny->GIVEREF(__pyx_refnanny, (PyObject *)(r), __LINE__)
  #define __Pyx_XINCREF(r)  do { if((r) != NULL) {__Pyx_INCREF(r); }} while(0)
  #define __Pyx_XDECREF(r)  do { if((r) != NULL) {__Pyx_DECREF(r); }} while(0)
  #define __Pyx_XGOTREF(r)  do { if((r) != NULL) {__Pyx_GOTREF(r); }} while(0)
  #define __Pyx_XGIVEREF(r) do { if((r) != NULL) {__Pyx_GIVEREF(r);}} while(0)
#else
  #define __Pyx_RefNannyDeclarations
  #define __Pyx_RefNannySetupContext(name, acquire_gil)
  #define __Pyx_RefNannyFinishContext()
  #define __Pyx_INCREF(r) Py_INCREF(r)
  #define __Pyx_DECREF(r) Py_DECREF(r)
  #define __Pyx_GOTREF(r)
  #define __Pyx_GIVEREF(r)
  #define __Pyx_XINCREF(r) Py_XINCREF(r)
  #define __Pyx_XDECREF(r) Py_XDECREF(r)
  #define __Pyx_XGOTREF(r)
  #define __Pyx_XGIVEREF(r)
#endif
#define __Pyx_XDECREF_SET(r, v) do {\
        PyObject *tmp = (PyObject *) r;\
        r = v; __Pyx_XDECREF(tmp);\
    } while (0)
#define __Pyx_DECREF_SET(r, v) do {\
        PyObject *tmp = (PyObject *) r;\
        r = v; __Pyx_DECREF(tmp);\
    } while (0)
#define __Pyx_CLEAR(r)    do { PyObject* tmp = ((PyObject*)(r)); r = NULL; __Pyx_DECREF(tmp);} while(0)
#define __Pyx_XCLEAR(r)   do { if((r) != NULL) {PyObject* tmp = ((PyObject*)(r)); r = NULL; __Pyx_DECREF(tmp);}} while(0)

/* PyObjectGetAttrStr.proto */
#if CYTHON_USE_TYPE_SLOTS
static CYTHON_INLINE PyObject* __Pyx_PyObject_GetAttrStr(PyObject* obj, PyObject* attr_name);
#else
#define __Pyx_PyObject_GetAttrStr(o,n) PyObject_GetAttr(o,n)
#endif

/* Import.proto */
static PyObject *__Pyx_Import(PyObject *name, PyObject *from_list, int level);

/* ImportFrom.proto */
static PyObject* __Pyx_ImportFrom(PyObject* module, PyObject* name);

/* GetBuiltinName.proto */
static PyObject *__Pyx_GetBuiltinName(PyObject *name);

/* PyDictVersioning.proto */
#if CYTHON_USE_DICT_VERSIONS && CYTHON_USE_TYPE_SLOTS
#define __PYX_DICT_VERSION_INIT  ((PY_UINT64_T) -1)
#define __PYX_GET_DICT_VERSION(dict)  (((PyDictObject*)(dict))->ma_version_tag)
#define __PYX_UPDATE_DICT_CACHE(dict, value, cache_var, version_var)\
    (version_var) = __PYX_GET_DICT_VERSION(dict);\
    (cache_var) = (value);
#define __PYX_PY_DICT_LOOKUP_IF_MODIFIED(VAR, DICT, LOOKUP) {\
    static PY_UINT64_T __pyx_dict_version = 0;\
    static PyObject *__pyx_dict_cached_value = NULL;\
    if (likely(__PYX_GET_DICT_VERSION(DICT) == __pyx_dict_version)) {\
        (VAR) = __pyx_dict_cached_value;\
    } else {\
        (VAR) = __pyx_dict_cached_value = (LOOKUP);\
        __pyx_dict_version = __PYX_GET_DICT_VERSION(DICT);\
    }\
}
static CYTHON_INLINE PY_UINT64_T __Pyx_get_tp_dict_version(PyObject *obj);
static CYTHON_INLINE PY_UINT64_T __Pyx_get_object_dict_version(PyObject *obj);
static CYTHON_INLINE int __Pyx_object_dict_version_matches(PyObject* obj, PY_UINT64_T tp_dict_version, PY_UINT64_T obj_dict_version);
#else
#define __PYX_GET_DICT_VERSION(dict)  (0)
#define __PYX_UPDATE_DICT_CACHE(dict, value, cache_var, version_var)
#define __PYX_PY_DICT_LOOKUP_IF_MODIFIED(VAR, DICT, LOOKUP)  (VAR) = (LOOKUP);
#endif

/* GetModuleGlobalName.proto */
#if CYTHON_USE_DICT_VERSIONS
#define __Pyx_GetModuleGlobalName(var, name)  {\
    static PY_UINT64_T __pyx_dict_version = 0;\
```

```c
    static PyObject *__pyx_dict_cached_value = NULL;\
    (var) = (likely(__pyx_dict_version == __PYX_GET_DICT_VERSION(__pyx_d))) ?\
        (likely(__pyx_dict_cached_value) ? __Pyx_NewRef(__pyx_dict_cached_value) : __Pyx_GetBuiltinName(name)) :\
        __Pyx__GetModuleGlobalName(name, &__pyx_dict_version, &__pyx_dict_cached_value);\
}
#define __Pyx_GetModuleGlobalNameUncached(var, name)  {\
    PY_UINT64_T __pyx_dict_version;\
    PyObject *__pyx_dict_cached_value;\
    (var) = __Pyx__GetModuleGlobalName(name, &__pyx_dict_version, &__pyx_dict_cached_value);\
}
static PyObject *__Pyx__GetModuleGlobalName(PyObject *name, PY_UINT64_T *dict_version, PyObject **dict_cached_value);
#else
#define __Pyx_GetModuleGlobalName(var, name)  (var) = __Pyx__GetModuleGlobalName(name)
#define __Pyx_GetModuleGlobalNameUncached(var, name)  (var) = __Pyx__GetModuleGlobalName(name)
static CYTHON_INLINE PyObject *__Pyx__GetModuleGlobalName(PyObject *name);
#endif

/* PyFunctionFastCall.proto */
#if CYTHON_FAST_PYCALL
#define __Pyx_PyFunction_FastCall(func, args, nargs)\
    __Pyx_PyFunction_FastCallDict((func), (args), (nargs), NULL)
#if 1 || PY_VERSION_HEX < 0x030600B1
static PyObject *__Pyx_PyFunction_FastCallDict(PyObject *func, PyObject **args, Py_ssize_t nargs, PyObject *kwargs);
#else
#define __Pyx_PyFunction_FastCallDict(func, args, nargs, kwargs) _PyFunction_FastCallDict(func, args, nargs, kwargs)
#endif
#define __Pyx_BUILD_ASSERT_EXPR(cond)\
    (sizeof(char [1 - 2*!(cond)]) - 1)
#ifndef Py_MEMBER_SIZE
#define Py_MEMBER_SIZE(type, member) sizeof(((type *)0)->member)
#endif
#if CYTHON_FAST_PYCALL
  static size_t __pyx_pyframe_localsplus_offset = 0;
  #include "frameobject.h"
#if PY_VERSION_HEX >= 0x030b00a6
  #ifndef Py_BUILD_CORE
    #define Py_BUILD_CORE 1
  #endif
  #include "internal/pycore_frame.h"
#endif
  #define __Pxy_PyFrame_Initialize_Offsets()\
    ((void)__Pyx_BUILD_ASSERT_EXPR(sizeof(PyFrameObject) == offsetof(PyFrameObject, f_localsplus) + Py_MEMBER_SIZE(PyFrameObject, f_localsplus)),\
     (void)(__pyx_pyframe_localsplus_offset = ((size_t)PyFrame_Type.tp_basicsize) - Py_MEMBER_SIZE(PyFrameObject, f_localsplus)))
  #define __Pyx_PyFrame_GetLocalsplus(frame)\
    (assert(__pyx_pyframe_localsplus_offset), (PyObject **)(((char *)(frame)) + __pyx_pyframe_localsplus_offset))
#endif // CYTHON_FAST_PYCALL
#endif

/* PyObjectCall.proto */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_Call(PyObject *func, PyObject *arg, PyObject *kw);
#else
#define __Pyx_PyObject_Call(func, arg, kw) PyObject_Call(func, arg, kw)
#endif

/* PyObjectCallMethO.proto */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_CallMethO(PyObject *func, PyObject *arg);
#endif

/* PyObjectCallNoArg.proto */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_CallNoArg(PyObject *func);
#else
```

```c
#define __Pyx_PyObject_CallNoArg(func) __Pyx_PyObject_Call(func, __pyx_empty_tuple, NULL)
#endif

/* PyThreadStateGet.proto */
#if CYTHON_FAST_THREAD_STATE
#define __Pyx_PyThreadState_declare  PyThreadState *__pyx_tstate;
#define __Pyx_PyThreadState_assign  __pyx_tstate = __Pyx_PyThreadState_Current;
#define __Pyx_PyErr_Occurred()  __pyx_tstate->curexc_type
#else
#define __Pyx_PyThreadState_declare
#define __Pyx_PyThreadState_assign
#define __Pyx_PyErr_Occurred()  PyErr_Occurred()
#endif

/* PyErrFetchRestore.proto */
#if CYTHON_FAST_THREAD_STATE
#define __Pyx_PyErr_Clear() __Pyx_ErrRestore(NULL, NULL, NULL)
#define __Pyx_ErrRestoreWithState(type, value, tb) __Pyx_ErrRestoreInState(PyThreadState_GET(), type, value, tb)
#define __Pyx_ErrFetchWithState(type, value, tb) __Pyx_ErrFetchInState(PyThreadState_GET(), type, value, tb)
#define __Pyx_ErrRestore(type, value, tb)  __Pyx_ErrRestoreInState(__pyx_tstate, type, value, tb)
#define __Pyx_ErrFetch(type, value, tb)    __Pyx_ErrFetchInState(__pyx_tstate, type, value, tb)
static CYTHON_INLINE void __Pyx_ErrRestoreInState(PyThreadState *tstate, PyObject *type, PyObject *value, PyObject *tb);
static CYTHON_INLINE void __Pyx_ErrFetchInState(PyThreadState *tstate, PyObject **type, PyObject **value, PyObject **tb);
#if CYTHON_COMPILING_IN_CPYTHON
#define __Pyx_PyErr_SetNone(exc) (Py_INCREF(exc), __Pyx_ErrRestore((exc), NULL, NULL))
#else
#define __Pyx_PyErr_SetNone(exc) PyErr_SetNone(exc)
#endif
#else
#define __Pyx_PyErr_Clear() PyErr_Clear()
#define __Pyx_PyErr_SetNone(exc) PyErr_SetNone(exc)
#define __Pyx_ErrRestoreWithState(type, value, tb)  PyErr_Restore(type, value, tb)
#define __Pyx_ErrFetchWithState(type, value, tb)  PyErr_Fetch(type, value, tb)
#define __Pyx_ErrRestoreInState(tstate, type, value, tb)  PyErr_Restore(type, value, tb)
#define __Pyx_ErrFetchInState(tstate, type, value, tb)  PyErr_Fetch(type, value, tb)
#define __Pyx_ErrRestore(type, value, tb)  PyErr_Restore(type, value, tb)
#define __Pyx_ErrFetch(type, value, tb)  PyErr_Fetch(type, value, tb)
#endif

/* CLineInTraceback.proto */
#ifdef CYTHON_CLINE_IN_TRACEBACK
#define __Pyx_CLineForTraceback(tstate, c_line)  (((CYTHON_CLINE_IN_TRACEBACK)) ? c_line : 0)
#else
static int __Pyx_CLineForTraceback(PyThreadState *tstate, int c_line);
#endif

/* CodeObjectCache.proto */
typedef struct {
    PyCodeObject* code_object;
    int code_line;
} __Pyx_CodeObjectCacheEntry;
struct __Pyx_CodeObjectCache {
    int count;
    int max_count;
    __Pyx_CodeObjectCacheEntry* entries;
};
static struct __Pyx_CodeObjectCache __pyx_code_cache = {0,0,NULL};
static int __pyx_bisect_code_objects(__Pyx_CodeObjectCacheEntry* entries, int count, int code_line);
static PyCodeObject *__pyx_find_code_object(int code_line);
static void __pyx_insert_code_object(int code_line, PyCodeObject* code_object);

/* AddTraceback.proto */
static void __Pyx_AddTraceback(const char *funcname, int c_line,
                               int py_line, const char *filename);

```

```c
/* GCCDiagnostics.proto */
#if defined(__GNUC__) && (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 6))
#define __Pyx_HAS_GCC_DIAGNOSTIC
#endif

/* CIntToPy.proto */
static CYTHON_INLINE PyObject* __Pyx_PyInt_From_long(long value);

/* CIntFromPy.proto */
static CYTHON_INLINE long __Pyx_PyInt_As_long(PyObject *);

/* CIntFromPy.proto */
static CYTHON_INLINE int __Pyx_PyInt_As_int(PyObject *);

/* FastTypeChecks.proto */
#if CYTHON_COMPILING_IN_CPYTHON
#define __Pyx_TypeCheck(obj, type) __Pyx_IsSubtype(Py_TYPE(obj), (PyTypeObject *)type)
static CYTHON_INLINE int __Pyx_IsSubtype(PyTypeObject *a, PyTypeObject *b);
static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches(PyObject *err, PyObject *type);
static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches2(PyObject *err, PyObject *type1, PyObject *type2);
#else
#define __Pyx_TypeCheck(obj, type) PyObject_TypeCheck(obj, (PyTypeObject *)type)
#define __Pyx_PyErr_GivenExceptionMatches(err, type) PyErr_GivenExceptionMatches(err, type)
#define __Pyx_PyErr_GivenExceptionMatches2(err, type1, type2) (PyErr_GivenExceptionMatches(err, type1) || PyErr_GivenExceptionMatches(err, type2))
#endif
#define __Pyx_PyException_Check(obj) __Pyx_TypeCheck(obj, PyExc_Exception)

/* CheckBinaryVersion.proto */
static int __Pyx_check_binary_version(void);

/* InitStrings.proto */
static int __Pyx_InitStrings(__Pyx_StringTabEntry *t);


/* Module declarations from 'API_Calls._main_' */
#define __Pyx_MODULE_NAME "API_Calls._main_"
extern int __pyx_module_is_main_API_Calls___main_;
int __pyx_module_is_main_API_Calls___main_ = 0;

/* Implementation of 'API_Calls._main_' */
static const char __pyx_k_main[] = "__main__";
static const char __pyx_k_name[] = "__name__";
static const char __pyx_k_test[] = "__test__";
static const char __pyx_k_import[] = "__import__";
static const char __pyx_k_Initializer[] = "Initializer";
static const char __pyx_k_initializer[] = "initializer";
static const char __pyx_k_cline_in_traceback[] = "cline_in_traceback";
static PyObject *__pyx_n_s_Initializer;
static PyObject *__pyx_n_s_cline_in_traceback;
static PyObject *__pyx_n_s_import;
static PyObject *__pyx_n_s_initializer;
static PyObject *__pyx_n_s_main;
static PyObject *__pyx_n_s_name;
static PyObject *__pyx_n_s_test;
/* Late includes */

static PyMethodDef __pyx_methods[] = {
  {0, 0, 0, 0}
};

#if PY_MAJOR_VERSION >= 3
#if CYTHON_PEP489_MULTI_PHASE_INIT
static PyObject* __pyx_pymod_create(PyObject *spec, PyModuleDef *def); /*proto*/
static int __pyx_pymod_exec__main_(PyObject* module); /*proto*/
static PyModuleDef_Slot __pyx_moduledef_slots[] = {
  {Py_mod_create, (void*)__pyx_pymod_create},
  {Py_mod_exec, (void*)__pyx_pymod_exec__main_},
  {0, NULL}
};
#endif

static struct PyModuleDef __pyx_moduledef = {
    PyModuleDef_HEAD_INIT,
    "_main_",
    0, /* m_doc */
  #if CYTHON_PEP489_MULTI_PHASE_INIT
    0, /* m_size */
  #else
    -1, /* m_size */
  #endif
    __pyx_methods /* m_methods */,
  #if CYTHON_PEP489_MULTI_PHASE_INIT
    __pyx_moduledef_slots, /* m_slots */
  #else
    NULL, /* m_reload */
  #endif
    NULL, /* m_traverse */
    NULL, /* m_clear */
    NULL /* m_free */
};
#endif
#ifndef CYTHON_SMALL_CODE
#if defined(__clang__)
    #define CYTHON_SMALL_CODE
#elif defined(__GNUC__) && (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 3))
    #define CYTHON_SMALL_CODE __attribute__((cold))
#else
    #define CYTHON_SMALL_CODE
#endif
#endif

static __Pyx_StringTabEntry __pyx_string_tab[] = {
  {&__pyx_n_s_Initializer, __pyx_k_Initializer, sizeof(__pyx_k_Initializer), 0, 0, 1, 1},
  {&__pyx_n_s_cline_in_traceback, __pyx_k_cline_in_traceback, sizeof(__pyx_k_cline_in_traceback), 0, 0, 1, 1},
  {&__pyx_n_s_import, __pyx_k_import, sizeof(__pyx_k_import), 0, 0, 1, 1},
  {&__pyx_n_s_initializer, __pyx_k_initializer, sizeof(__pyx_k_initializer), 0, 0, 1, 1},
  {&__pyx_n_s_main, __pyx_k_main, sizeof(__pyx_k_main), 0, 0, 1, 1},
  {&__pyx_n_s_name, __pyx_k_name, sizeof(__pyx_k_name), 0, 0, 1, 1},
  {&__pyx_n_s_test, __pyx_k_test, sizeof(__pyx_k_test), 0, 0, 1, 1},
  {0, 0, 0, 0, 0, 0, 0}
};
static CYTHON_SMALL_CODE int __Pyx_InitCachedBuiltins(void) {
  return 0;
}

static CYTHON_SMALL_CODE int __Pyx_InitCachedConstants(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_InitCachedConstants", 0);
  __Pyx_RefNannyFinishContext();
  return 0;
}

static CYTHON_SMALL_CODE int __Pyx_InitGlobals(void) {
  if (__Pyx_InitStrings(__pyx_string_tab) < 0) __PYX_ERR(0, 1, __pyx_L1_error);
  return 0;
  __pyx_L1_error:;
  return -1;
}

static CYTHON_SMALL_CODE int __Pyx_modinit_global_init_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_variable_export_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_function_export_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_type_init_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_type_import_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_variable_import_code(void); /*proto*/
static CYTHON_SMALL_CODE int __Pyx_modinit_function_import_code(void); /*proto*/

static int __Pyx_modinit_global_init_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_global_init_code", 0);
  /*--- Global init code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_variable_export_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_variable_export_code", 0);
  /*--- Variable export code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_function_export_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_function_export_code", 0);
  /*--- Function export code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_type_init_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_type_init_code", 0);
  /*--- Type init code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_type_import_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_type_import_code", 0);
  /*--- Type import code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_variable_import_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_variable_import_code", 0);
  /*--- Variable import code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}

static int __Pyx_modinit_function_import_code(void) {
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("__Pyx_modinit_function_import_code", 0);
  /*--- Function import code ---*/
  __Pyx_RefNannyFinishContext();
  return 0;
}


#ifndef CYTHON_NO_PYINIT_EXPORT
#define __Pyx_PyMODINIT_FUNC PyMODINIT_FUNC
#elif PY_MAJOR_VERSION < 3
#ifdef __cplusplus
#define __Pyx_PyMODINIT_FUNC extern "C" void
#else
#define __Pyx_PyMODINIT_FUNC void
#endif
#else
#ifdef __cplusplus
#define __Pyx_PyMODINIT_FUNC extern "C" PyObject *
#else
#define __Pyx_PyMODINIT_FUNC PyObject *
#endif
#endif


#if PY_MAJOR_VERSION < 3
__Pyx_PyMODINIT_FUNC init_main_(void) CYTHON_SMALL_CODE; /*proto*/
__Pyx_PyMODINIT_FUNC init_main_(void)
#else
__Pyx_PyMODINIT_FUNC PyInit__main_(void) CYTHON_SMALL_CODE; /*proto*/
__Pyx_PyMODINIT_FUNC PyInit__main_(void)
#if CYTHON_PEP489_MULTI_PHASE_INIT
{
  return PyModuleDef_Init(&__pyx_moduledef);
}
static CYTHON_SMALL_CODE int __Pyx_check_single_interpreter(void) {
    #if PY_VERSION_HEX >= 0x030700A1
```

```c
    static PY_INT64_T main_interpreter_id = -1;
    PY_INT64_T current_id = PyInterpreterState_GetID(PyThreadState_Get()->interp);
    if (main_interpreter_id == -1) {
        main_interpreter_id = current_id;
        return (unlikely(current_id == -1)) ? -1 : 0;
    } else if (unlikely(main_interpreter_id != current_id))
    #else
    static PyInterpreterState *main_interpreter = NULL;
    PyInterpreterState *current_interpreter = PyThreadState_Get()->interp;
    if (!main_interpreter) {
        main_interpreter = current_interpreter;
    } else if (unlikely(main_interpreter != current_interpreter))
    #endif
    {
        PyErr_SetString(
            PyExc_ImportError,
            "Interpreter change detected - this module can only be loaded into one interpreter per process.");
        return -1;
    }
    return 0;
}
static CYTHON_SMALL_CODE int __Pyx_copy_spec_to_module(PyObject *spec, PyObject *moddict, const char* from_name, const char* to_name, int allow_none) {
    PyObject *value = PyObject_GetAttrString(spec, from_name);
    int result = 0;
    if (likely(value)) {
        if (allow_none || value != Py_None) {
            result = PyDict_SetItemString(moddict, to_name, value);
        }
        Py_DECREF(value);
    } else if (PyErr_ExceptionMatches(PyExc_AttributeError)) {
        PyErr_Clear();
    } else {
        result = -1;
    }
    return result;
}
static CYTHON_SMALL_CODE PyObject* __pyx_pymod_create(PyObject *spec, CYTHON_UNUSED PyModuleDef *def) {
    PyObject *module = NULL, *moddict, *modname;
    if (__Pyx_check_single_interpreter())
        return NULL;
    if (__pyx_m)
        return __Pyx_NewRef(__pyx_m);
    modname = PyObject_GetAttrString(spec, "name");
    if (unlikely(!modname)) goto bad;
    module = PyModule_NewObject(modname);
    Py_DECREF(modname);
    if (unlikely(!module)) goto bad;
    moddict = PyModule_GetDict(module);
    if (unlikely(!moddict)) goto bad;
    if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "loader", "__loader__", 1) < 0)) goto bad;
    if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "origin", "__file__", 1) < 0)) goto bad;
    if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "parent", "__package__", 1) < 0)) goto bad;
    if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "submodule_search_locations", "__path__", 0) < 0)) goto bad;
    return module;
bad:
    Py_XDECREF(module);
    return NULL;
}


static CYTHON_SMALL_CODE int __pyx_pymod_exec__main_(PyObject *__pyx_pyinit_module)
#endif
#endif
{
  PyObject *__pyx_t_1 = NULL;
  PyObject *__pyx_t_2 = NULL;
  int __pyx_lineno = 0;
  const char *__pyx_filename = NULL;
  int __pyx_clineno = 0;
  __Pyx_RefNannyDeclarations
  #if CYTHON_PEP489_MULTI_PHASE_INIT
  if (__pyx_m) {
    if (__pyx_m == __pyx_pyinit_module) return 0;
    PyErr_SetString(PyExc_RuntimeError, "Module '_main_' has already been imported. Re-initialisation is not supported.");
    return -1;
  }
  #elif PY_MAJOR_VERSION >= 3
  if (__pyx_m) return __Pyx_NewRef(__pyx_m);
  #endif
  #if CYTHON_REFNANNY
__Pyx_RefNanny = __Pyx_RefNannyImportAPI("refnanny");
if (!__Pyx_RefNanny) {
  PyErr_Clear();
  __Pyx_RefNanny = __Pyx_RefNannyImportAPI("Cython.Runtime.refnanny");
  if (!__Pyx_RefNanny)
      Py_FatalError("failed to import 'refnanny' module");
}
#endif
  __Pyx_RefNannySetupContext("__Pyx_PyMODINIT_FUNC PyInit__main_(void)", 0);
  if (__Pyx_check_binary_version() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #ifdef __Pxy_PyFrame_Initialize_Offsets
  __Pxy_PyFrame_Initialize_Offsets();
  #endif
  __pyx_empty_tuple = PyTuple_New(0); if (unlikely(!__pyx_empty_tuple)) __PYX_ERR(0, 1, __pyx_L1_error)
  __pyx_empty_bytes = PyBytes_FromStringAndSize("", 0); if (unlikely(!__pyx_empty_bytes)) __PYX_ERR(0, 1, __pyx_L1_error)
  __pyx_empty_unicode = PyUnicode_FromStringAndSize("", 0); if (unlikely(!__pyx_empty_unicode)) __PYX_ERR(0, 1, __pyx_L1_error)
  #ifdef __Pyx_CyFunction_USED
  if (__pyx_CyFunction_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  #ifdef __Pyx_FusedFunction_USED
  if (__pyx_FusedFunction_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  #ifdef __Pyx_Coroutine_USED
  if (__pyx_Coroutine_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  #ifdef __Pyx_Generator_USED
  if (__pyx_Generator_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  #ifdef __Pyx_AsyncGen_USED
  if (__pyx_AsyncGen_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  #ifdef __Pyx_StopAsyncIteration_USED
  if (__pyx_StopAsyncIteration_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  /*--- Library function declarations ---*/
  /*--- Threads initialization code ---*/
  #if defined(WITH_THREAD) && PY_VERSION_HEX < 0x030700F0 && defined(__PYX_FORCE_INIT_THREADS) && __PYX_FORCE_INIT_THREADS
  PyEval_InitThreads();
  #endif
  /*--- Module creation code ---*/
  #if CYTHON_PEP489_MULTI_PHASE_INIT
  __pyx_m = __pyx_pyinit_module;
  Py_INCREF(__pyx_m);
  #else
  #if PY_MAJOR_VERSION < 3
  __pyx_m = Py_InitModule4("_main_", __pyx_methods, 0, 0, PYTHON_API_VERSION); Py_XINCREF(__pyx_m);
  #else
  __pyx_m = PyModule_Create(&__pyx_moduledef);
  #endif
  if (unlikely(!__pyx_m)) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  __pyx_d = PyModule_GetDict(__pyx_m); if (unlikely(!__pyx_d)) __PYX_ERR(0, 1, __pyx_L1_error)
  Py_INCREF(__pyx_d);
  __pyx_b = PyImport_AddModule(__Pyx_BUILTIN_MODULE_NAME); if (unlikely(!__pyx_b)) __PYX_ERR(0, 1, __pyx_L1_error)
  Py_INCREF(__pyx_b);
  __pyx_cython_runtime = PyImport_AddModule((char *) "cython_runtime"); if (unlikely(!__pyx_cython_runtime)) __PYX_ERR(0, 1, __pyx_L1_error)
  Py_INCREF(__pyx_cython_runtime);
  if (PyObject_SetAttrString(__pyx_m, "__builtins__", __pyx_b) < 0) __PYX_ERR(0, 1, __pyx_L1_error);
  /*--- Initialize various global constants etc. ---*/
  if (__Pyx_InitGlobals() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #if PY_MAJOR_VERSION < 3 && (__PYX_DEFAULT_STRING_ENCODING_IS_ASCII || __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT)
  if (__Pyx_init_sys_getdefaultencoding_params() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif
  if (__pyx_module_is_main_API_Calls___main__) {
    if (PyObject_SetAttr(__pyx_m, __pyx_n_s_name, __pyx_n_s_main) < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  }
  #if PY_MAJOR_VERSION >= 3
  {
    PyObject *modules = PyImport_GetModuleDict(); if (unlikely(!modules)) __PYX_ERR(0, 1, __pyx_L1_error)
    if (!PyDict_GetItemString(modules, "API_Calls.__main__")) {
      if (unlikely(PyDict_SetItemString(modules, "API_Calls.__main__", __pyx_m) < 0)) __PYX_ERR(0, 1, __pyx_L1_error)
    }
  }
  #endif
  /*--- Builtin init code ---*/
  if (__Pyx_InitCachedBuiltins() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  /*--- Constants init code ---*/
  if (__Pyx_InitCachedConstants() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  /*--- Global type/function init code ---*/
  (void)__Pyx_modinit_global_init_code();
  (void)__Pyx_modinit_variable_export_code();
  (void)__Pyx_modinit_function_export_code();
  (void)__Pyx_modinit_type_init_code();
  (void)__Pyx_modinit_type_import_code();
  (void)__Pyx_modinit_variable_import_code();
  (void)__Pyx_modinit_function_import_code();
  /*--- Execution code ---*/
  #if defined(__Pyx_Generator_USED) || defined(__Pyx_Coroutine_USED)
  if (__Pyx_patch_abc() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  #endif

  /* "API_Calls/__main__.py":4
 *
 *
 * from Initializer import initializer             # <<<<<<<<<<<<<<
 *
 * initializer()
 */
  __pyx_t_1 = PyList_New(1); if (unlikely(!__pyx_t_1)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_1);
  __Pyx_INCREF(__pyx_n_s_initializer);
  __Pyx_GIVEREF(__pyx_n_s_initializer);
  PyList_SET_ITEM(__pyx_t_1, 0, __pyx_n_s_initializer);
  __pyx_t_2 = __Pyx_Import(__pyx_n_s_Initializer, __pyx_t_1, -1); if (unlikely(!__pyx_t_2)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_2);
  __Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;
  __pyx_t_1 = __Pyx_ImportFrom(__pyx_t_2, __pyx_n_s_initializer); if (unlikely(!__pyx_t_1)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_1);
  if (PyDict_SetItem(__pyx_d, __pyx_n_s_initializer, __pyx_t_1) < 0) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;

  /* "API_Calls/__main__.py":6
 * from Initializer import initializer
 *
 * initializer()             # <<<<<<<<<<<<<<
 */
  __Pyx_GetModuleGlobalName(__pyx_t_2, __pyx_n_s_initializer); if (unlikely(!__pyx_t_2)) __PYX_ERR(0, 6, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_2);
  __pyx_t_1 = __Pyx_PyObject_CallNoArg(__pyx_t_2); if (unlikely(!__pyx_t_1)) __PYX_ERR(0, 6, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_1);
```

```c
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
  __Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;

  /* "API_Calls/_main_.py":1
 * #   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/             # <<<<<<<<<<<<<<
 *
 *
 */
  __pyx_t_1 = __Pyx_PyDict_NewPresized(0); if (unlikely(!__pyx_t_1)) __PYX_ERR(0, 1, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_1);
  if (PyDict_SetItem(__pyx_d, __pyx_n_s_test, __pyx_t_1) < 0) __PYX_ERR(0, 1, __pyx_L1_error)
  __Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;

  /*--- Wrapped vars code ---*/

  goto __pyx_L0;
  __pyx_L1_error:;
  __Pyx_XDECREF(__pyx_t_1);
  __Pyx_XDECREF(__pyx_t_2);
  if (__pyx_m) {
    if (__pyx_d) {
      __Pyx_AddTraceback("init API_Calls._main_", __pyx_clineno, __pyx_lineno, __pyx_filename);
    }
    Py_CLEAR(__pyx_m);
  } else if (!PyErr_Occurred()) {
    PyErr_SetString(PyExc_ImportError, "init API_Calls._main_");
  }
  __pyx_L0:;
  __Pyx_RefNannyFinishContext();
  #if CYTHON_PEP489_MULTI_PHASE_INIT
  return (__pyx_m != NULL) ? 0 : -1;
  #elif PY_MAJOR_VERSION >= 3
  return __pyx_m;
  #else
  return;
  #endif
}

/* --- Runtime support code --- */
/* Refnanny */
#if CYTHON_REFNANNY
static __Pyx_RefNannyAPIStruct *__Pyx_RefNannyImportAPI(const char *modname) {
    PyObject *m = NULL, *p = NULL;
    void *r = NULL;
    m = PyImport_ImportModule(modname);
    if (!m) goto end;
    p = PyObject_GetAttrString(m, "RefNannyAPI");
    if (!p) goto end;
    r = PyLong_AsVoidPtr(p);
end:
    Py_XDECREF(p);
    Py_XDECREF(m);
    return (__Pyx_RefNannyAPIStruct *)r;
}
#endif

/* PyObjectGetAttrStr */
#if CYTHON_USE_TYPE_SLOTS
static CYTHON_INLINE PyObject* __Pyx_PyObject_GetAttrStr(PyObject* obj, PyObject* attr_name) {
    PyTypeObject* tp = Py_TYPE(obj);
    if (likely(tp->tp_getattro))
        return tp->tp_getattro(obj, attr_name);
#if PY_MAJOR_VERSION < 3
    if (likely(tp->tp_getattr))
        return tp->tp_getattr(obj, PyString_AS_STRING(attr_name));
#endif
    return PyObject_GetAttr(obj, attr_name);
}
#endif

/* Import */
static PyObject *__Pyx_Import(PyObject *name, PyObject *from_list, int level) {
    PyObject *empty_list = 0;
    PyObject *module = 0;
    PyObject *global_dict = 0;
    PyObject *empty_dict = 0;
    PyObject *list;
    #if PY_MAJOR_VERSION < 3
    PyObject *py_import;
    py_import = __Pyx_PyObject_GetAttrStr(__pyx_b, __pyx_n_s_import);
    if (!py_import)
        goto bad;
    #endif
    if (from_list)
        list = from_list;
    else {
        empty_list = PyList_New(0);
        if (!empty_list)
            goto bad;
        list = empty_list;
    }
    global_dict = PyModule_GetDict(__pyx_m);
    if (!global_dict)
        goto bad;
    empty_dict = PyDict_New();
    if (!empty_dict)
        goto bad;
    {
        #if PY_MAJOR_VERSION >= 3
        if (level == -1) {
            if ((1) && (strchr(__Pyx_MODULE_NAME, '.'))) {
                module = PyImport_ImportModuleLevelObject(
                    name, global_dict, empty_dict, list, 1);
                if (!module) {
                    if (!PyErr_ExceptionMatches(PyExc_ImportError))
                        goto bad;
                    PyErr_Clear();
                }
            }
            level = 0;
        }
        #endif
        if (!module) {
            #if PY_MAJOR_VERSION < 3
            PyObject *py_level = PyInt_FromLong(level);
            if (!py_level)
                goto bad;
            module = PyObject_CallFunctionObjArgs(py_import,
                name, global_dict, empty_dict, list, py_level, (PyObject *)NULL);
            Py_DECREF(py_level);
            #else
            module = PyImport_ImportModuleLevelObject(
                name, global_dict, empty_dict, list, level);
            #endif
        }
    }
bad:
    #if PY_MAJOR_VERSION < 3
    Py_XDECREF(py_import);
    #endif
    Py_XDECREF(empty_list);
    Py_XDECREF(empty_dict);
    return module;
}

/* ImportFrom */
static PyObject* __Pyx_ImportFrom(PyObject* module, PyObject* name) {
    PyObject* value = __Pyx_PyObject_GetAttrStr(module, name);
    if (unlikely(!value) && PyErr_ExceptionMatches(PyExc_AttributeError)) {
        PyErr_Format(PyExc_ImportError,
        #if PY_MAJOR_VERSION < 3
            "cannot import name %.230s", PyString_AS_STRING(name));
        #else
            "cannot import name %S", name);
        #endif
    }
    return value;
}

```

```
/* GetBuiltinName */
static PyObject *__Pyx_GetBuiltinName(PyObject *name) {
    PyObject* result = __Pyx_PyObject_GetAttrStr(__pyx_b, name);
    if (unlikely(!result)) {
        PyErr_Format(PyExc_NameError,
#if PY_MAJOR_VERSION >= 3
            "name '%U' is not defined", name);
#else
            "name '%.200s' is not defined", PyString_AS_STRING(name));
#endif
    }
    return result;
}

/* PyDictVersioning */
#if CYTHON_USE_DICT_VERSIONS && CYTHON_USE_TYPE_SLOTS
static CYTHON_INLINE PY_UINT64_T __Pyx_get_tp_dict_version(PyObject *obj) {
    PyObject *dict = Py_TYPE(obj)->tp_dict;
    return likely(dict) ? __PYX_GET_DICT_VERSION(dict) : 0;
}
static CYTHON_INLINE PY_UINT64_T __Pyx_get_object_dict_version(PyObject *obj) {
    PyObject **dictptr = NULL;
    Py_ssize_t offset = Py_TYPE(obj)->tp_dictoffset;
    if (offset) {
#if CYTHON_COMPILING_IN_CPYTHON
        dictptr = (likely(offset > 0)) ? (PyObject **) ((char *)obj + offset) : _PyObject_GetDictPtr(obj);
#else
        dictptr = _PyObject_GetDictPtr(obj);
#endif
    }
    return (dictptr && *dictptr) ? __PYX_GET_DICT_VERSION(*dictptr) : 0;
}
static CYTHON_INLINE int __Pyx_object_dict_version_matches(PyObject* obj, PY_UINT64_T tp_dict_version, PY_UINT64_T obj_dict_version) {
    PyObject *dict = Py_TYPE(obj)->tp_dict;
    if (unlikely(!dict) || unlikely(tp_dict_version != __PYX_GET_DICT_VERSION(dict)))
        return 0;
    return obj_dict_version == __Pyx_get_object_dict_version(obj);
}
#endif

/* GetModuleGlobalName */
#if CYTHON_USE_DICT_VERSIONS
static PyObject *__Pyx__GetModuleGlobalName(PyObject *name, PY_UINT64_T *dict_version, PyObject **dict_cached_value)
#else
static CYTHON_INLINE PyObject *__Pyx__GetModuleGlobalName(PyObject *name)
#endif
{
    PyObject *result;
#if !CYTHON_AVOID_BORROWED_REFS
#if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX >= 0x030500A1
    result = _PyDict_GetItem_KnownHash(__pyx_d, name, ((PyASCIIObject *) name)->hash);
    __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
    if (likely(result)) {
        return __Pyx_NewRef(result);
    } else if (unlikely(PyErr_Occurred())) {
        return NULL;
    }
#else
    result = PyDict_GetItem(__pyx_d, name);
    __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
    if (likely(result)) {
        return __Pyx_NewRef(result);
    }
#endif
#else
    result = PyObject_GetItem(__pyx_d, name);
    __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
    if (likely(result)) {
        return __Pyx_NewRef(result);
    }
    PyErr_Clear();
#endif
```

```
    return __Pyx_GetBuiltinName(name);
}

/* PyFunctionFastCall */
#if CYTHON_FAST_PYCALL
static PyObject* __Pyx_PyFunction_FastCallNoKw(PyCodeObject *co, PyObject **args,
Py_ssize_t na,
                                                   PyObject *globals) {
    PyFrameObject *f;
    PyThreadState *tstate = __Pyx_PyThreadState_Current;
    PyObject **fastlocals;
    Py_ssize_t i;
    PyObject *result;
    assert(globals != NULL);
    /* XXX Perhaps we should create a specialized
       PyFrame_New() that doesn't take locals, but does
       take builtins without sanity checking them.
       */
    assert(tstate != NULL);
    f = PyFrame_New(tstate, co, globals, NULL);
    if (f == NULL) {
        return NULL;
    }
    fastlocals = __Pyx_PyFrame_GetLocalsplus(f);
    for (i = 0; i < na; i++) {
        Py_INCREF(*args);
        fastlocals[i] = *args++;
    }
    result = PyEval_EvalFrameEx(f,0);
    ++tstate->recursion_depth;
    Py_DECREF(f);
    --tstate->recursion_depth;
    return result;
}
#if 1 || PY_VERSION_HEX < 0x030600B1
static PyObject *__Pyx_PyFunction_FastCallDict(PyObject *func, PyObject **args,
Py_ssize_t nargs, PyObject *kwargs) {
    PyCodeObject *co = (PyCodeObject *)PyFunction_GET_CODE(func);
    PyObject *globals = PyFunction_GET_GLOBALS(func);
    PyObject *argdefs = PyFunction_GET_DEFAULTS(func);
    PyObject *closure;
#if PY_MAJOR_VERSION >= 3
    PyObject *kwdefs;
#endif
    PyObject *kwtuple, **k;
    PyObject **d;
    Py_ssize_t nd;
    Py_ssize_t nk;
    PyObject *result;
    assert(kwargs == NULL || PyDict_Check(kwargs));
    nk = kwargs ? PyDict_Size(kwargs) : 0;
    if (Py_EnterRecursiveCall((char*)" while calling a Python object")) {
        return NULL;
    }
    if (
#if PY_MAJOR_VERSION >= 3
            co->co_kwonlyargcount == 0 &&
#endif
            likely(kwargs == NULL || nk == 0) &&
            co->co_flags == (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)) {
        if (argdefs == NULL && co->co_argcount == nargs) {
            result = __Pyx_PyFunction_FastCallNoKw(co, args, nargs, globals);
            goto done;
        }
        else if (nargs == 0 && argdefs != NULL
                 && co->co_argcount == Py_SIZE(argdefs)) {
            /* function called with no arguments, but all parameters have
               a default value: use default values as arguments .*/
            args = &PyTuple_GET_ITEM(argdefs, 0);
            result =__Pyx_PyFunction_FastCallNoKw(co, args, Py_SIZE(argdefs),
globals);
            goto done;
        }
    }
    if (kwargs != NULL) {
        Py_ssize_t pos, i;
        kwtuple = PyTuple_New(2 * nk);
        if (kwtuple == NULL) {
            result = NULL;
            goto done;
        }
        k = &PyTuple_GET_ITEM(kwtuple, 0);
        pos = i = 0;
        while (PyDict_Next(kwargs, &pos, &k[i], &k[i+1])) {
            Py_INCREF(k[i]);
            Py_INCREF(k[i+1]);
            i += 2;
        }
        nk = i / 2;
    }
    else {
        kwtuple = NULL;
        k = NULL;
    }
    closure = PyFunction_GET_CLOSURE(func);
#if PY_MAJOR_VERSION >= 3
    kwdefs = PyFunction_GET_KW_DEFAULTS(func);
#endif
    if (argdefs != NULL) {
        d = &PyTuple_GET_ITEM(argdefs, 0);
        nd = Py_SIZE(argdefs);
    }
    else {
        d = NULL;
        nd = 0;
    }
#if PY_MAJOR_VERSION >= 3
    result = PyEval_EvalCodeEx((PyObject*)co, globals, (PyObject *)NULL,
                               args, (int)nargs,
                               k, (int)nk,
                               d, (int)nd, kwdefs, closure);
#else
    result = PyEval_EvalCodeEx(co, globals, (PyObject *)NULL,
                               args, (int)nargs,
                               k, (int)nk,
                               d, (int)nd, closure);
#endif
    Py_XDECREF(kwtuple);
done:
    Py_LeaveRecursiveCall();
    return result;
}
#endif
#endif

/* PyObjectCall */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_Call(PyObject *func, PyObject *arg,
PyObject *kw) {
    PyObject *result;
    ternaryfunc call = Py_TYPE(func)->tp_call;
    if (unlikely(!call))
        return PyObject_Call(func, arg, kw);
    if (unlikely(Py_EnterRecursiveCall((char*)" while calling a Python object")))
        return NULL;
    result = (*call)(func, arg, kw);
    Py_LeaveRecursiveCall();
    if (unlikely(!result) && unlikely(!PyErr_Occurred())) {
        PyErr_SetString(
            PyExc_SystemError,
            "NULL result without error in PyObject_Call");
    }
    return result;
}
#endif

/* PyObjectCallMethO */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_CallMethO(PyObject *func, PyObject
*arg) {
    PyObject *self, *result;
    PyCFunction cfunc;
    cfunc = PyCFunction_GET_FUNCTION(func);
    self = PyCFunction_GET_SELF(func);
    if (unlikely(Py_EnterRecursiveCall((char*)" while calling a Python object")))
        return NULL;
    result = cfunc(self, arg);
    Py_LeaveRecursiveCall();
    if (unlikely(!result) && unlikely(!PyErr_Occurred())) {
        PyErr_SetString(
            PyExc_SystemError,
            "NULL result without error in PyObject_Call");
    }
    return result;
}
#endif

/* PyObjectCallNoArg */
#if CYTHON_COMPILING_IN_CPYTHON
static CYTHON_INLINE PyObject* __Pyx_PyObject_CallNoArg(PyObject *func) {
#if CYTHON_FAST_PYCALL
    if (PyFunction_Check(func)) {
        return __Pyx_PyFunction_FastCall(func, NULL, 0);
    }
#endif
#ifdef __Pyx_CyFunction_USED
    if (likely(PyCFunction_Check(func) || __Pyx_CyFunction_Check(func)))
#else
    if (likely(PyCFunction_Check(func)))
#endif
    {
        if (likely(PyCFunction_GET_FLAGS(func) & METH_NOARGS)) {
            return __Pyx_PyObject_CallMethO(func, NULL);
        }
    }
    return __Pyx_PyObject_Call(func, __pyx_empty_tuple, NULL);
}
#endif

/* PyErrFetchRestore */
#if CYTHON_FAST_THREAD_STATE
static CYTHON_INLINE void __Pyx_ErrRestoreInState(PyThreadState *tstate, PyObject *type, PyObject *value, PyObject *tb) {
    PyObject *tmp_type, *tmp_value, *tmp_tb;
    tmp_type = tstate->curexc_type;
    tmp_value = tstate->curexc_value;
    tmp_tb = tstate->curexc_traceback;
    tstate->curexc_type = type;
    tstate->curexc_value = value;
    tstate->curexc_traceback = tb;
    Py_XDECREF(tmp_type);
    Py_XDECREF(tmp_value);
    Py_XDECREF(tmp_tb);
}
static CYTHON_INLINE void __Pyx_ErrFetchInState(PyThreadState *tstate, PyObject **type, PyObject **value, PyObject **tb) {
    *type = tstate->curexc_type;
    *value = tstate->curexc_value;
    *tb = tstate->curexc_traceback;
    tstate->curexc_type = 0;
    tstate->curexc_value = 0;
    tstate->curexc_traceback = 0;
}
#endif

/* CLineInTraceback */
#ifndef CYTHON_CLINE_IN_TRACEBACK
static int __Pyx_CLineForTraceback(CYTHON_NCP_UNUSED PyThreadState *tstate, int c_line) {
    PyObject *use_cline;
    PyObject *ptype, *pvalue, *ptraceback;
#if CYTHON_COMPILING_IN_CPYTHON
    PyObject **cython_runtime_dict;
#endif
    if (unlikely(!__pyx_cython_runtime)) {
        return c_line;
    }
    __Pyx_ErrFetchInState(tstate, &ptype, &pvalue, &ptraceback);
#if CYTHON_COMPILING_IN_CPYTHON
    cython_runtime_dict = _PyObject_GetDictPtr(__pyx_cython_runtime);
    if (likely(cython_runtime_dict)) {
        __PYX_PY_DICT_LOOKUP_IF_MODIFIED(
            use_cline, *cython_runtime_dict,
            __Pyx_PyDict_GetItemStr(*cython_runtime_dict, __pyx_n_s_cline_in_traceback))
    } else
#endif
    {
      PyObject *use_cline_obj = __Pyx_PyObject_GetAttrStr(__pyx_cython_runtime, __pyx_n_s_cline_in_traceback);
      if (use_cline_obj) {
        use_cline = PyObject_Not(use_cline_obj) ? Py_False : Py_True;
        Py_DECREF(use_cline_obj);
      } else {
        PyErr_Clear();
        use_cline = NULL;
      }
    }
    if (!use_cline) {
        c_line = 0;
        (void) PyObject_SetAttr(__pyx_cython_runtime, __pyx_n_s_cline_in_traceback, Py_False);
    }
    else if (use_cline == Py_False || (use_cline != Py_True && PyObject_Not(use_cline) != 0)) {
        c_line = 0;
    }
    __Pyx_ErrRestoreInState(tstate, ptype, pvalue, ptraceback);
    return c_line;
}
#endif

/* CodeObjectCache */
static int __pyx_bisect_code_objects(__Pyx_CodeObjectCacheEntry* entries, int count, int code_line) {
    int start = 0, mid = 0, end = count - 1;
    if (end >= 0 && code_line > entries[end].code_line) {
        return count;
    }
    while (start < end) {
        mid = start + (end - start) / 2;
        if (code_line < entries[mid].code_line) {
            end = mid;
        } else if (code_line > entries[mid].code_line) {
             start = mid + 1;
        } else {
            return mid;
        }
    }
    if (code_line <= entries[mid].code_line) {
        return mid;
    } else {
        return mid + 1;
    }
}
static PyCodeObject *__pyx_find_code_object(int code_line) {
    PyCodeObject* code_object;
    int pos;
    if (unlikely(!code_line) || unlikely(!__pyx_code_cache.entries)) {
        return NULL;
    }
    pos = __pyx_bisect_code_objects(__pyx_code_cache.entries, __pyx_code_cache.count, code_line);
    if (unlikely(pos >= __pyx_code_cache.count) || unlikely(__pyx_code_cache.entries[pos].code_line != code_line)) {
        return NULL;
    }
    code_object = __pyx_code_cache.entries[pos].code_object;
    Py_INCREF(code_object);
    return code_object;
}
static void __pyx_insert_code_object(int code_line, PyCodeObject* code_object) {
    int pos, i;
    __Pyx_CodeObjectCacheEntry* entries = __pyx_code_cache.entries;
    if (unlikely(!code_line)) {
        return;
    }
    if (unlikely(!entries)) {
        entries = (__Pyx_CodeObjectCacheEntry*)PyMem_Malloc(64*sizeof(__Pyx_CodeObjectCacheEntry));
        if (likely(entries)) {
            __pyx_code_cache.entries = entries;
            __pyx_code_cache.max_count = 64;
            __pyx_code_cache.count = 1;
            entries[0].code_line = code_line;
            entries[0].code_object = code_object;
            Py_INCREF(code_object);
        }
        return;
    }
    pos = __pyx_bisect_code_objects(__pyx_code_cache.entries, __pyx_code_cache.count, code_line);
    if ((pos < __pyx_code_cache.count) && unlikely(__pyx_code_cache.entries[pos].code_line == code_line)) {
        PyCodeObject* tmp = entries[pos].code_object;
        entries[pos].code_object = code_object;
        Py_DECREF(tmp);
        return;
    }
    if (__pyx_code_cache.count == __pyx_code_cache.max_count) {
        int new_max = __pyx_code_cache.max_count + 64;
        entries = (__Pyx_CodeObjectCacheEntry*)PyMem_Realloc(
            __pyx_code_cache.entries, ((size_t)new_max) * sizeof(__Pyx_CodeObjectCacheEntry));
        if (unlikely(!entries)) {
            return;
        }
        __pyx_code_cache.entries = entries;
        __pyx_code_cache.max_count = new_max;
    }
    for (i=__pyx_code_cache.count; i>pos; i--) {
        entries[i] = entries[i-1];
    }
    entries[pos].code_line = code_line;
    entries[pos].code_object = code_object;
    __pyx_code_cache.count++;
    Py_INCREF(code_object);
}

/* AddTraceback */
#include "compile.h"
#include "frameobject.h"
#include "traceback.h"
#if PY_VERSION_HEX >= 0x030b00a6
  #ifndef Py_BUILD_CORE
    #define Py_BUILD_CORE 1
  #endif
  #include "internal/pycore_frame.h"
#endif
static PyCodeObject* __Pyx_CreateCodeObjectForTraceback(
            const char *funcname, int c_line,
            int py_line, const char *filename) {
    PyCodeObject *py_code = NULL;
    PyObject *py_funcname = NULL;
    #if PY_MAJOR_VERSION < 3
    PyObject *py_srcfile = NULL;
    py_srcfile = PyString_FromString(filename);
    if (!py_srcfile) goto bad;
    #endif
    if (c_line) {
        #if PY_MAJOR_VERSION < 3
        py_funcname = PyString_FromFormat( "%s (%s:%d)", funcname, __pyx_cfilenm, c_line);
        if (!py_funcname) goto bad;
        #else
        py_funcname = PyUnicode_FromFormat( "%s (%s:%d)", funcname, __pyx_cfilenm, c_line);
        if (!py_funcname) goto bad;
        funcname = PyUnicode_AsUTF8(py_funcname);
        if (!funcname) goto bad;
        #endif
    }
    else {
        #if PY_MAJOR_VERSION < 3
        py_funcname = PyString_FromString(funcname);
        if (!py_funcname) goto bad;
        #endif
    }
    #if PY_MAJOR_VERSION < 3
    py_code = __Pyx_PyCode_New(
        0,
        0,
        0,
        0,
        0,
        __pyx_empty_bytes, /*PyObject *code,*/
        __pyx_empty_tuple, /*PyObject *consts,*/
        __pyx_empty_tuple, /*PyObject *names,*/
        __pyx_empty_tuple, /*PyObject *varnames,*/
        __pyx_empty_tuple, /*PyObject *freevars,*/
        __pyx_empty_tuple, /*PyObject *cellvars,*/
        py_srcfile,   /*PyObject *filename,*/
        py_funcname,  /*PyObject *name,*/
        py_line,
        __pyx_empty_bytes  /*PyObject *lnotab*/
    );
    Py_DECREF(py_srcfile);
    #else
    py_code = PyCode_NewEmpty(filename, funcname, py_line);
    #endif
    Py_XDECREF(py_funcname);  // XDECREF since it's only set on Py3 if cline
    return py_code;
bad:
    Py_XDECREF(py_funcname);
    #if PY_MAJOR_VERSION < 3
    Py_XDECREF(py_srcfile);
    #endif
    return NULL;
}
static void __Pyx_AddTraceback(const char *funcname, int c_line,
                               int py_line, const char *filename) {
    PyCodeObject *py_code = 0;
    PyFrameObject *py_frame = 0;
    PyThreadState *tstate = __Pyx_PyThreadState_Current;
    PyObject *ptype, *pvalue, *ptraceback;
    if (c_line) {
        c_line = __Pyx_CLineForTraceback(tstate, c_line);
    }
    py_code = __pyx_find_code_object(c_line ? -c_line : py_line);
    if (!py_code) {
        __Pyx_ErrFetchInState(tstate, &ptype, &pvalue, &ptraceback);
        py_code = __Pyx_CreateCodeObjectForTraceback(
            funcname, c_line, py_line, filename);
        if (!py_code) {
            /* If the code object creation fails, then we should clear the
               fetched exception references and propagate the new exception */
            Py_XDECREF(ptype);
            Py_XDECREF(pvalue);
            Py_XDECREF(ptraceback);
            goto bad;
        }
        __Pyx_ErrRestoreInState(tstate, ptype, pvalue, ptraceback);
        __pyx_insert_code_object(c_line ? -c_line : py_line, py_code);
    }
    py_frame = PyFrame_New(
        tstate,            /*PyThreadState *tstate,*/
        py_code,           /*PyCodeObject *code,*/
        __pyx_d,    /*PyObject *globals,*/
        0                  /*PyObject *locals*/
    );
    if (!py_frame) goto bad;
    __Pyx_PyFrame_SetLineNumber(py_frame, py_line);
    PyTraceBack_Here(py_frame);
bad:
    Py_XDECREF(py_code);
    Py_XDECREF(py_frame);
}

/* MainFunction */
#ifdef __FreeBSD__
#include <floatingpoint.h>
#endif
```

```
#if PY_MAJOR_VERSION < 3
int main(int argc, char** argv) {
#elif defined(WIN32) || defined(MS_WINDOWS)
int wmain(int argc, wchar_t **argv) {
#else
static int __Pyx_main(int argc, wchar_t **argv) {
#endif
    /* 754 requires that FP exceptions run in "no stop" mode by default,
     * and until C vendors implement C99's ways to control FP exceptions,
     * Python requires non-stop mode.  Alas, some platforms enable FP
     * exceptions by default.  Here we disable them.
     */
#ifdef __FreeBSD__
    fp_except_t m;
    m = fpgetmask();
    fpsetmask(m & ~FP_X_OFL);
#endif
    if (argc && argv)
        Py_SetProgramName(argv[0]);
    Py_Initialize();
    if (argc && argv)
        PySys_SetArgv(argc, argv);
    {
      PyObject* m = NULL;
      __pyx_module_is_main_API_Calls___main_ = 1;
      #if PY_MAJOR_VERSION < 3
          init_main_();
      #elif CYTHON_PEP489_MULTI_PHASE_INIT
          m = PyInit__main_();
          if (!PyModule_Check(m)) {
              PyModuleDef *mdef = (PyModuleDef *) m;
              PyObject *modname = PyUnicode_FromString("__main__");
              m = NULL;
              if (modname) {
                  m = PyModule_NewObject(modname);
                  Py_DECREF(modname);
                  if (m) PyModule_ExecDef(m, mdef);
              }
          }
      #else
          m = PyInit__main_();
      #endif
      if (PyErr_Occurred()) {
          PyErr_Print();
          #if PY_MAJOR_VERSION < 3
          if (Py_FlushLine()) PyErr_Clear();
          #endif
          return 1;
      }
      Py_XDECREF(m);
    }
#if PY_VERSION_HEX < 0x03060000
    Py_Finalize();
#else
    if (Py_FinalizeEx() < 0)
        return 2;
#endif
    return 0;
}
#if PY_MAJOR_VERSION >= 3 && !defined(WIN32) && !defined(MS_WINDOWS)
#include <locale.h>
static wchar_t*
__Pyx_char2wchar(char* arg)
{
    wchar_t *res;
#ifdef HAVE_BROKEN_MBSTOWCS
    /* Some platforms have a broken implementation of
     * mbstowcs which does not count the characters that
     * would result from conversion.  Use an upper bound.
     */
    size_t argsize = strlen(arg);
#else
    size_t argsize = mbstowcs(NULL, arg, 0);
#endif
    size_t count;
    unsigned char *in;
    wchar_t *out;
```

```
#ifdef HAVE_MBRTOWC
    mbstate_t mbs;
#endif
    if (argsize != (size_t)-1) {
        res = (wchar_t *)malloc((argsize+1)*sizeof(wchar_t));
        if (!res)
            goto oom;
        count = mbstowcs(res, arg, argsize+1);
        if (count != (size_t)-1) {
            wchar_t *tmp;
            /* Only use the result if it contains no
               surrogate characters. */
            for (tmp = res; *tmp != 0 &&
                         (*tmp < 0xd800 || *tmp > 0xdfff); tmp++)
                ;
            if (*tmp == 0)
                return res;
        }
        free(res);
    }
#ifdef HAVE_MBRTOWC
    /* Overallocate; as multi-byte characters are in the argument, the
       actual output could use less memory. */
    argsize = strlen(arg) + 1;
    res = (wchar_t *)malloc(argsize*sizeof(wchar_t));
    if (!res) goto oom;
    in = (unsigned char*)arg;
    out = res;
    memset(&mbs, 0, sizeof mbs);
    while (argsize) {
        size_t converted = mbrtowc(out, (char*)in, argsize, &mbs);
        if (converted == 0)
            break;
        if (converted == (size_t)-2) {
            /* Incomplete character. This should never happen,
               since we provide everything that we have -
               unless there is a bug in the C library, or I
               misunderstood how mbrtowc works. */
            fprintf(stderr, "unexpected mbrtowc result -2\\n");
            free(res);
            return NULL;
        }
        if (converted == (size_t)-1) {
            /* Conversion error. Escape as UTF-8b, and start over
               in the initial shift state. */
            *out++ = 0xdc00 + *in++;
            argsize--;
            memset(&mbs, 0, sizeof mbs);
            continue;
        }
        if (*out >= 0xd800 && *out <= 0xdfff) {
            /* Surrogate character.  Escape the original
               byte sequence with surrogateescape. */
            argsize -= converted;
            while (converted--)
                *out++ = 0xdc00 + *in++;
            continue;
        }
        in += converted;
        argsize -= converted;
        out++;
    }
#else
    /* Cannot use C locale for escaping; manually escape as if charset
       is ASCII (i.e. escape all bytes > 128. This will still roundtrip
       correctly in the locale's charset, which must be an ASCII superset. */
    res = (wchar_t *)malloc((strlen(arg)+1)*sizeof(wchar_t));
    if (!res) goto oom;
    in = (unsigned char*)arg;
    out = res;
    while(*in)
        if(*in < 128)
            *out++ = *in++;
        else
            *out++ = 0xdc00 + *in++;
    *out = 0;
#endif
```

```
    return res;
oom:
    fprintf(stderr, "out of memory\\n");
    return NULL;
}
int
main(int argc, char **argv)
{
    if (!argc) {
        return __Pyx_main(0, NULL);
    }
    else {
        int i, res;
        wchar_t **argv_copy = (wchar_t **)malloc(sizeof(wchar_t*)*argc);
        wchar_t **argv_copy2 = (wchar_t **)malloc(sizeof(wchar_t*)*argc);
        char *oldloc = strdup(setlocale(LC_ALL, NULL));
        if (!argv_copy || !argv_copy2 || !oldloc) {
            fprintf(stderr, "out of memory\\n");
            free(argv_copy);
            free(argv_copy2);
            free(oldloc);
            return 1;
        }
        res = 0;
        setlocale(LC_ALL, "");
        for (i = 0; i < argc; i++) {
            argv_copy2[i] = argv_copy[i] = __Pyx_char2wchar(argv[i]);
            if (!argv_copy[i]) res = 1;
        }
        setlocale(LC_ALL, oldloc);
        free(oldloc);
        if (res == 0)
            res = __Pyx_main(argc, argv_copy);
        for (i = 0; i < argc; i++) {
#if PY_VERSION_HEX < 0x03050000
            free(argv_copy2[i]);
#else
            PyMem_RawFree(argv_copy2[i]);
#endif
        }
        free(argv_copy);
        free(argv_copy2);
        return res;
    }
}
#endif

/* CIntToPy */
    static CYTHON_INLINE PyObject* __Pyx_PyInt_From_long(long value) {
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wconversion"
#endif
    const long neg_one = (long) -1, const_zero = (long) 0;
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic pop
#endif
    const int is_unsigned = neg_one > const_zero;
    if (is_unsigned) {
        if (sizeof(long) < sizeof(long)) {
            return PyInt_FromLong((long) value);
        } else if (sizeof(long) <= sizeof(unsigned long)) {
            return PyLong_FromUnsignedLong((unsigned long) value);
#ifdef HAVE_LONG_LONG
        } else if (sizeof(long) <= sizeof(unsigned PY_LONG_LONG)) {
            return PyLong_FromUnsignedLongLong((unsigned PY_LONG_LONG) value);
#endif
        }
    } else {
        if (sizeof(long) <= sizeof(long)) {
            return PyInt_FromLong((long) value);
#ifdef HAVE_LONG_LONG
        } else if (sizeof(long) <= sizeof(PY_LONG_LONG)) {
            return PyLong_FromLongLong((PY_LONG_LONG) value);
#endif
        }
    }
    {
        int one = 1; int little = (int)*(unsigned char *)&one;
        unsigned char *bytes = (unsigned char *)&value;
        return _PyLong_FromByteArray(bytes, sizeof(long),
                                     little, !is_unsigned);
    }
}

/* CIntFromPyVerify */
    #define __PYX_VERIFY_RETURN_INT(target_type, func_type, func_value)\
    __PYX__VERIFY_RETURN_INT(target_type, func_type, func_value, 0)
#define __PYX_VERIFY_RETURN_INT_EXC(target_type, func_type, func_value)\
    __PYX__VERIFY_RETURN_INT(target_type, func_type, func_value, 1)
#define __PYX__VERIFY_RETURN_INT(target_type, func_type, func_value, exc)\
    {\
        func_type value = func_value;\
        if (sizeof(target_type) < sizeof(func_type)) {\
            if (unlikely(value != (func_type) (target_type) value)) {\
                func_type zero = 0;\
                if (exc && unlikely(value == (func_type)-1 && PyErr_Occurred()))\
                    return (target_type) -1;\
                if (is_unsigned && unlikely(value < zero))\
                    goto raise_neg_overflow;\
                else\
                    goto raise_overflow;\
            }\
        }\
        return (target_type) value;\
    }

/* CIntFromPy */
    static CYTHON_INLINE long __Pyx_PyInt_As_long(PyObject *x) {
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wconversion"
#endif
    const long neg_one = (long) -1, const_zero = (long) 0;
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic pop
#endif
    const int is_unsigned = neg_one > const_zero;
#if PY_MAJOR_VERSION < 3
    if (likely(PyInt_Check(x))) {
        if (sizeof(long) < sizeof(long)) {
            __PYX_VERIFY_RETURN_INT(long, long, PyInt_AS_LONG(x))
        } else {
            long val = PyInt_AS_LONG(x);
            if (is_unsigned && unlikely(val < 0)) {
                goto raise_neg_overflow;
            }
            return (long) val;
        }
    } else
#endif
    if (likely(PyLong_Check(x))) {
        if (is_unsigned) {
#if CYTHON_USE_PYLONG_INTERNALS
            const digit* digits = ((PyLongObject*)x)->ob_digit;
            switch (Py_SIZE(x)) {
                case  0: return (long) 0;
                case  1: __PYX_VERIFY_RETURN_INT(long, digit, digits[0])
                case 2:
                    if (8 * sizeof(long) > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long,
(((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) >= 2 * PyLong_SHIFT) {
                            return (long) (((((long)digits[1]) << PyLong_SHIFT) |
(long)digits[0]));
                        }
                    }
                    break;
                case 3:
                    if (8 * sizeof(long) > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long,
(((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) >= 3 * PyLong_SHIFT) {
                            return (long) (((((((long)digits[2]) << PyLong_SHIFT)
| (long)digits[1]) << PyLong_SHIFT) | (long)digits[0]));
                        }
                    }
                    break;
                case 4:
                    if (8 * sizeof(long) > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long,
(((((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) <<
PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) >= 4 * PyLong_SHIFT) {
                            return (long) (((((((((long)digits[3]) <<
PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) |
(long)digits[0]));
                        }
                    }
                    break;
            }
#endif
#if CYTHON_COMPILING_IN_CPYTHON
            if (unlikely(Py_SIZE(x) < 0)) {
                goto raise_neg_overflow;
            }
#else
            {
                int result = PyObject_RichCompareBool(x, Py_False, Py_LT);
                if (unlikely(result < 0))
                    return (long) -1;
                if (unlikely(result == 1))
                    goto raise_neg_overflow;
            }
#endif
            if (sizeof(long) <= sizeof(unsigned long)) {
                __PYX_VERIFY_RETURN_INT_EXC(long, unsigned long,
PyLong_AsUnsignedLong(x))
#ifdef HAVE_LONG_LONG
            } else if (sizeof(long) <= sizeof(unsigned PY_LONG_LONG)) {
                __PYX_VERIFY_RETURN_INT_EXC(long, unsigned PY_LONG_LONG,
PyLong_AsUnsignedLongLong(x))
#endif
            }
        } else {
#if CYTHON_USE_PYLONG_INTERNALS
            const digit* digits = ((PyLongObject*)x)->ob_digit;
            switch (Py_SIZE(x)) {
                case  0: return (long) 0;
                case -1: __PYX_VERIFY_RETURN_INT(long, sdigit, (sdigit)
(-(sdigit)digits[0]))
                case  1: __PYX_VERIFY_RETURN_INT(long,  digit, +digits[0])
                case -2:
                    if (8 * sizeof(long) - 1 > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, long, -(long)
(((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
                            return (long) (((long)-1)*(((((long)digits[1]) <<
PyLong_SHIFT) | (long)digits[0])));
                        }
                    }
                    break;
                case 2:
                    if (8 * sizeof(long) > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long,
(((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
                            return (long) ((((((long)digits[1]) << PyLong_SHIFT) |
(long)digits[0])));
                        }
                    }
                    break;
                case -3:
                    if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, long, -(long) (((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
                            return (long) (((long)-1)*(((((long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0])));
                        }
                    }
                    break;
                case 3:
                    if (8 * sizeof(long) > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long, (((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
                            return (long) (((((long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0])));
                        }
                    }
                    break;
                case -4:
                    if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, long, -(long) (((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 4 * PyLong_SHIFT) {
                            return (long) (((long)-1)*(((((((long)digits[3]) << PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0])));
                        }
                    }
                    break;
                case 4:
                    if (8 * sizeof(long) > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(long, unsigned long, (((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(long) - 1 > 4 * PyLong_SHIFT) {
                            return (long) (((((((long)digits[3]) << PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0])));
                        }
                    }
                    break;
            }
#endif
            if (sizeof(long) <= sizeof(long)) {
                __PYX_VERIFY_RETURN_INT_EXC(long, long, PyLong_AsLong(x))
#ifdef HAVE_LONG_LONG
            } else if (sizeof(long) <= sizeof(PY_LONG_LONG)) {
                __PYX_VERIFY_RETURN_INT_EXC(long, PY_LONG_LONG, PyLong_AsLongLong(x))
#endif
            }
        }
        {
#if CYTHON_COMPILING_IN_PYPY && !defined(_PyLong_AsByteArray)
            PyErr_SetString(PyExc_RuntimeError,
                            "_PyLong_AsByteArray() not available in PyPy, cannot convert large numbers");
#else
            long val;
            PyObject *v = __Pyx_PyNumber_IntOrLong(x);
 #if PY_MAJOR_VERSION < 3
            if (likely(v) && !PyLong_Check(v)) {
                PyObject *tmp = v;
                v = PyNumber_Long(tmp);
                Py_DECREF(tmp);
            }
 #endif
            if (likely(v)) {
                int one = 1; int is_little = (int)*(unsigned char *)&one;
                unsigned char *bytes = (unsigned char *)&val;
                int ret = _PyLong_AsByteArray((PyLongObject *)v,
                                              bytes, sizeof(val),
                                              is_little, !is_unsigned);
                Py_DECREF(v);
                if (likely(!ret))
                    return val;
            }
#endif
            return (long) -1;
        }
    } else {
        long val;
        PyObject *tmp = __Pyx_PyNumber_IntOrLong(x);
        if (!tmp) return (long) -1;
        val = __Pyx_PyInt_As_long(tmp);
        Py_DECREF(tmp);
        return val;
    }
raise_overflow:
    PyErr_SetString(PyExc_OverflowError,
        "value too large to convert to long");
    return (long) -1;
raise_neg_overflow:
    PyErr_SetString(PyExc_OverflowError,
        "can't convert negative value to long");
    return (long) -1;
}

/* CIntFromPy */
    static CYTHON_INLINE int __Pyx_PyInt_As_int(PyObject *x) {
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wconversion"
#endif
    const int neg_one = (int) -1, const_zero = (int) 0;
#ifdef __Pyx_HAS_GCC_DIAGNOSTIC
#pragma GCC diagnostic pop
#endif
    const int is_unsigned = neg_one > const_zero;
#if PY_MAJOR_VERSION < 3
    if (likely(PyInt_Check(x))) {
        if (sizeof(int) < sizeof(long)) {
            __PYX_VERIFY_RETURN_INT(int, long, PyInt_AS_LONG(x))
        } else {
            long val = PyInt_AS_LONG(x);
            if (is_unsigned && unlikely(val < 0)) {
                goto raise_neg_overflow;
            }
            return (int) val;
        }
    } else
#endif
    if (likely(PyLong_Check(x))) {
        if (is_unsigned) {
#if CYTHON_USE_PYLONG_INTERNALS
            const digit* digits = ((PyLongObject*)x)->ob_digit;
            switch (Py_SIZE(x)) {
                case  0: return (int) 0;
                case  1: __PYX_VERIFY_RETURN_INT(int, digit, digits[0])
                case 2:
                    if (8 * sizeof(int) > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, (((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) >= 2 * PyLong_SHIFT) {
                            return (int) (((((int)digits[1]) << PyLong_SHIFT) | (int)digits[0]));
                        }
                    }
                    break;
                case 3:
                    if (8 * sizeof(int) > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, (((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) >= 3 * PyLong_SHIFT) {
                            return (int) (((((int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0]));
                        }
                    }
                    break;
                case 4:
                    if (8 * sizeof(int) > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, (((((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) >= 4 * PyLong_SHIFT) {
                            return (int) (((((((((int)digits[3]) << PyLong_SHIFT) | (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0]));
                        }
                    }
                    break;
            }
#endif
#if CYTHON_COMPILING_IN_CPYTHON
            if (unlikely(Py_SIZE(x) < 0)) {
                goto raise_neg_overflow;
            }
#else
            {
                int result = PyObject_RichCompareBool(x, Py_False, Py_LT);
                if (unlikely(result < 0))
                    return (int) -1;
                if (unlikely(result == 1))
                    goto raise_neg_overflow;
            }
#endif
            if (sizeof(int) <= sizeof(unsigned long)) {
                __PYX_VERIFY_RETURN_INT_EXC(int, unsigned long, PyLong_AsUnsignedLong(x))
#ifdef HAVE_LONG_LONG
            } else if (sizeof(int) <= sizeof(unsigned PY_LONG_LONG)) {
                __PYX_VERIFY_RETURN_INT_EXC(int, unsigned PY_LONG_LONG, PyLong_AsUnsignedLongLong(x))
#endif
            }
        } else {
#if CYTHON_USE_PYLONG_INTERNALS
            const digit* digits = ((PyLongObject*)x)->ob_digit;
            switch (Py_SIZE(x)) {
                case  0: return (int) 0;
                case -1: __PYX_VERIFY_RETURN_INT(int, sdigit, (sdigit) (-(sdigit)digits[0]))
                case  1: __PYX_VERIFY_RETURN_INT(int,  digit, +digits[0])
                case -2:
                    if (8 * sizeof(int) - 1 > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, long, -(long) (((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
                            return (int) (((int)-1)*(((((int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
                case 2:
                    if (8 * sizeof(int) > 1 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, (((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
                            return (int) (((((int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
                case -3:
                    if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, long, -(long) ((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
                            return (int) (((int)-1)*(((((int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
                case 3:
                    if (8 * sizeof(int) > 2 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, ((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
                            return (int) (((((((int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
                case -4:
                    if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, long, -(long) (((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 4 * PyLong_SHIFT) {
                            return (int) (((int)-1)*(((((((int)digits[3]) << PyLong_SHIFT) | (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
                case 4:
                    if (8 * sizeof(int) > 3 * PyLong_SHIFT) {
                        if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
                            __PYX_VERIFY_RETURN_INT(int, unsigned long, (((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
                        } else if (8 * sizeof(int) - 1 > 4 * PyLong_SHIFT) {
                            return (int) (((((((((int)digits[3]) << PyLong_SHIFT) | (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0])));
                        }
                    }
                    break;
            }
#endif
            if (sizeof(int) <= sizeof(long)) {
                __PYX_VERIFY_RETURN_INT_EXC(int, long, PyLong_AsLong(x))
#ifdef HAVE_LONG_LONG
            } else if (sizeof(int) <= sizeof(PY_LONG_LONG)) {
                __PYX_VERIFY_RETURN_INT_EXC(int, PY_LONG_LONG, PyLong_AsLongLong(x))
#endif
            }
        }
        {
#if CYTHON_COMPILING_IN_PYPY && !defined(_PyLong_AsByteArray)
            PyErr_SetString(PyExc_RuntimeError,
                            "_PyLong_AsByteArray() not available in PyPy, cannot convert large numbers");
#else
            int val;
            PyObject *v = __Pyx_PyNumber_IntOrLong(x);
 #if PY_MAJOR_VERSION < 3
            if (likely(v) && !PyLong_Check(v)) {
                PyObject *tmp = v;
                v = PyNumber_Long(tmp);
                Py_DECREF(tmp);
            }
 #endif
            if (likely(v)) {
                int one = 1; int is_little = (int)*(unsigned char *)&one;
                unsigned char *bytes = (unsigned char *)&val;
                int ret = _PyLong_AsByteArray((PyLongObject *)v,
                                              bytes, sizeof(val),
                                              is_little, !is_unsigned);
                Py_DECREF(v);
                if (likely(!ret))
                    return val;
            }
#endif
            return (int) -1;
        }
    } else {
        int val;
        PyObject *tmp = __Pyx_PyNumber_IntOrLong(x);
        if (!tmp) return (int) -1;
        val = __Pyx_PyInt_As_int(tmp);
        Py_DECREF(tmp);
        return val;
    }
raise_overflow:
    PyErr_SetString(PyExc_OverflowError,
        "value too large to convert to int");
    return (int) -1;
raise_neg_overflow:
    PyErr_SetString(PyExc_OverflowError,
        "can't convert negative value to int");
    return (int) -1;
}

/* FastTypeChecks */
    #if CYTHON_COMPILING_IN_CPYTHON
static int __Pyx_InBases(PyTypeObject *a, PyTypeObject *b) {
    while (a) {
        a = a->tp_base;
        if (a == b)
            return 1;
    }
    return b == &PyBaseObject_Type;
}
static CYTHON_INLINE int __Pyx_IsSubtype(PyTypeObject *a, PyTypeObject *b) {
    PyObject *mro;
    if (a == b) return 1;
    mro = a->tp_mro;
    if (likely(mro)) {
        Py_ssize_t i, n;
        n = PyTuple_GET_SIZE(mro);
        for (i = 0; i < n; i++) {
            if (PyTuple_GET_ITEM(mro, i) == (PyObject *)b)
                return 1;
        }
        return 0;
    }
    return __Pyx_InBases(a, b);
}
#if PY_MAJOR_VERSION == 2
static int __Pyx_inner_PyErr_GivenExceptionMatches2(PyObject *err, PyObject*
exc_type1, PyObject* exc_type2) {
    PyObject *exception, *value, *tb;
    int res;
    __Pyx_PyThreadState_declare
    __Pyx_PyThreadState_assign
    __Pyx_ErrFetch(&exception, &value, &tb);
    res = exc_type1 ? PyObject_IsSubclass(err, exc_type1) : 0;
    if (unlikely(res == -1)) {
        PyErr_WriteUnraisable(err);
        res = 0;
    }
    if (!res) {
        res = PyObject_IsSubclass(err, exc_type2);
        if (unlikely(res == -1)) {
            PyErr_WriteUnraisable(err);
            res = 0;
        }
    }
    __Pyx_ErrRestore(exception, value, tb);
    return res;
}
#else
static CYTHON_INLINE int __Pyx_inner_PyErr_GivenExceptionMatches2(PyObject *err,
PyObject* exc_type1, PyObject *exc_type2) {
    int res = exc_type1 ? __Pyx_IsSubtype((PyTypeObject*)err, (PyTypeObject*)exc_type1) : 0;
    if (!res) {
        res = __Pyx_IsSubtype((PyTypeObject*)err, (PyTypeObject*)exc_type2);
    }
    return res;
}
#endif
static int __Pyx_PyErr_GivenExceptionMatchesTuple(PyObject *exc_type, PyObject *tuple) {
    Py_ssize_t i, n;
    assert(PyExceptionClass_Check(exc_type));
    n = PyTuple_GET_SIZE(tuple);
#if PY_MAJOR_VERSION >= 3
    for (i=0; i<n; i++) {
        if (exc_type == PyTuple_GET_ITEM(tuple, i)) return 1;
    }
#endif
    for (i=0; i<n; i++) {
        PyObject *t = PyTuple_GET_ITEM(tuple, i);
        #if PY_MAJOR_VERSION < 3
        if (likely(exc_type == t)) return 1;
        #endif
        if (likely(PyExceptionClass_Check(t))) {
            if (__Pyx_inner_PyErr_GivenExceptionMatches2(exc_type, NULL, t)) return 1;
        } else {
        }
    }
    return 0;
}
static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches(PyObject *err, PyObject* exc_type) {
    if (likely(err == exc_type)) return 1;
    if (likely(PyExceptionClass_Check(err))) {
        if (likely(PyExceptionClass_Check(exc_type))) {
            return __Pyx_inner_PyErr_GivenExceptionMatches2(err, NULL, exc_type);
        } else if (likely(PyTuple_Check(exc_type))) {
            return __Pyx_PyErr_GivenExceptionMatchesTuple(err, exc_type);
        } else {
        }
    }
    return PyErr_GivenExceptionMatches(err, exc_type);
}
static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches2(PyObject *err, PyObject *exc_type1, PyObject *exc_type2) {
    assert(PyExceptionClass_Check(exc_type1));
    assert(PyExceptionClass_Check(exc_type2));
    if (likely(err == exc_type1 || err == exc_type2)) return 1;
    if (likely(PyExceptionClass_Check(err))) {
        return __Pyx_inner_PyErr_GivenExceptionMatches2(err, exc_type1, exc_type2);
    }
    return (PyErr_GivenExceptionMatches(err, exc_type1) || PyErr_GivenExceptionMatches(err, exc_type2));
}
#endif

/* CheckBinaryVersion */
    static int __Pyx_check_binary_version(void) {
    char ctversion[5];
    int same=1, i, found_dot;
    const char* rt_from_call = Py_GetVersion();
    PyOS_snprintf(ctversion, 5, "%d.%d", PY_MAJOR_VERSION, PY_MINOR_VERSION);
    found_dot = 0;
    for (i = 0; i < 4; i++) {
        if (!ctversion[i]) {
            same = (rt_from_call[i] < '0' || rt_from_call[i] > '9');
            break;
        }
        if (rt_from_call[i] != ctversion[i]) {
            same = 0;
            break;
        }
    }
    if (!same) {
        char rtversion[5] = {'\0'};
        char message[200];
        for (i=0; i<4; ++i) {
            if (rt_from_call[i] == '.') {
                if (found_dot) break;
                found_dot = 1;
            } else if (rt_from_call[i] < '0' || rt_from_call[i] > '9') {
                break;
            }
            rtversion[i] = rt_from_call[i];
        }
        PyOS_snprintf(message, sizeof(message),
                      "compiletime version %s of module '%.100s' "
                      "does not match runtime version %s",
                      ctversion, __Pyx_MODULE_NAME, rtversion);
        return PyErr_WarnEx(NULL, message, 1);
    }
    return 0;
}

/* InitStrings */
    static int __Pyx_InitStrings(__Pyx_StringTabEntry *t) {
    while (t->p) {
        #if PY_MAJOR_VERSION < 3
        if (t->is_unicode) {
            *t->p = PyUnicode_DecodeUTF8(t->s, t->n - 1, NULL);
        } else if (t->intern) {
            *t->p = PyString_InternFromString(t->s);
        } else {
            *t->p = PyString_FromStringAndSize(t->s, t->n - 1);
        }
        #else
        if (t->is_unicode | t->is_str) {
            if (t->intern) {
                *t->p = PyUnicode_InternFromString(t->s);
            } else if (t->encoding) {
                *t->p = PyUnicode_Decode(t->s, t->n - 1, t->encoding, NULL);
            } else {
                *t->p = PyUnicode_FromStringAndSize(t->s, t->n - 1);
            }
        } else {
            *t->p = PyBytes_FromStringAndSize(t->s, t->n - 1);
        }
        #endif
        if (!*t->p)
            return -1;
        if (PyObject_Hash(*t->p) == -1)
            return -1;
        ++t;
    }
    return 0;
}

static CYTHON_INLINE PyObject* __Pyx_PyUnicode_FromString(const char* c_str) {
    return __Pyx_PyUnicode_FromStringAndSize(c_str, (Py_ssize_t)strlen(c_str));
}
static CYTHON_INLINE const char* __Pyx_PyObject_AsString(PyObject* o) {
    Py_ssize_t ignore;
    return __Pyx_PyObject_AsStringAndSize(o, &ignore);
}
#if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII || __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
#if !CYTHON_PEP393_ENABLED
static const char* __Pyx_PyUnicode_AsStringAndSize(PyObject* o, Py_ssize_t *length) {
    char* defenc_c;
    PyObject* defenc = _PyUnicode_AsDefaultEncodedString(o, NULL);
    if (!defenc) return NULL;
    defenc_c = PyBytes_AS_STRING(defenc);
#if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
    {
        char* end = defenc_c + PyBytes_GET_SIZE(defenc);
        char* c;
        for (c = defenc_c; c < end; c++) {
            if ((unsigned char) (*c) >= 128) {
                PyUnicode_AsASCIIString(o);
                return NULL;
            }
        }
    }
#endif
    *length = PyBytes_GET_SIZE(defenc);
    return defenc_c;
}
#else
static CYTHON_INLINE const char* __Pyx_PyUnicode_AsStringAndSize(PyObject* o, Py_ssize_t *length) {
    if (unlikely(__Pyx_PyUnicode_READY(o) == -1)) return NULL;
#if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
    if (likely(PyUnicode_IS_ASCII(o))) {
        *length = PyUnicode_GET_LENGTH(o);
        return PyUnicode_AsUTF8(o);
    } else {
        PyUnicode_AsASCIIString(o);
        return NULL;
    }
#else
    return PyUnicode_AsUTF8AndSize(o, length);
#endif
}
#endif
#endif
static CYTHON_INLINE const char* __Pyx_PyObject_AsStringAndSize(PyObject* o, Py_ssize_t *length) {
#if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII || __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
    if (
#if PY_MAJOR_VERSION < 3 && __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
            __Pyx_sys_getdefaultencoding_not_ascii &&
#endif
            PyUnicode_Check(o)) {
        return __Pyx_PyUnicode_AsStringAndSize(o, length);
    } else
#endif
#if (!CYTHON_COMPILING_IN_PYPY) || (defined(PyByteArray_AS_STRING) && defined(PyByteArray_GET_SIZE))
    if (PyByteArray_Check(o)) {
        *length = PyByteArray_GET_SIZE(o);
        return PyByteArray_AS_STRING(o);
    } else
#endif
    {
        char* result;
        int r = PyBytes_AsStringAndSize(o, &result, length);
        if (unlikely(r < 0)) {
            return NULL;
        } else {
            return result;
        }
    }
}
static CYTHON_INLINE int __Pyx_PyObject_IsTrue(PyObject* x) {
   int is_true = x == Py_True;
   if (is_true | (x == Py_False) | (x == Py_None)) return is_true;
   else return PyObject_IsTrue(x);
}
static CYTHON_INLINE int __Pyx_PyObject_IsTrueAndDecref(PyObject* x) {
    int retval;
    if (unlikely(!x)) return -1;
    retval = __Pyx_PyObject_IsTrue(x);
    Py_DECREF(x);
    return retval;
}
static PyObject* __Pyx_PyNumber_IntOrLongWrongResultType(PyObject* result, const char* type_name) {
#if PY_MAJOR_VERSION >= 3
    if (PyLong_Check(result)) {
        if (PyErr_WarnFormat(PyExc_DeprecationWarning, 1,
                "__int__ returned non-int (type %.200s).  "
                "The ability to return an instance of a strict subclass of int "
                "is deprecated, and may be removed in a future version of Python.",
                Py_TYPE(result)->tp_name)) {
            Py_DECREF(result);
            return NULL;
        }
        return result;
    }
#endif
    PyErr_Format(PyExc_TypeError,
                 "__%.4s__ returned non-%.4s (type %.200s)",
                 type_name, type_name, Py_TYPE(result)->tp_name);
    Py_DECREF(result);
    return NULL;
}
static CYTHON_INLINE PyObject* __Pyx_PyNumber_IntOrLong(PyObject* x) {
#if CYTHON_USE_TYPE_SLOTS
  PyNumberMethods *m;
#endif
  const char *name = NULL;
  PyObject *res = NULL;
#if PY_MAJOR_VERSION < 3
  if (likely(PyInt_Check(x) || PyLong_Check(x)))
#else
  if (likely(PyLong_Check(x)))
#endif
    return __Pyx_NewRef(x);
#if CYTHON_USE_TYPE_SLOTS
  m = Py_TYPE(x)->tp_as_number;
  #if PY_MAJOR_VERSION < 3
  if (m && m->nb_int) {
    name = "int";
    res = m->nb_int(x);
  }
  else if (m && m->nb_long) {
    name = "long";
    res = m->nb_long(x);
  }
  #else
  if (likely(m && m->nb_int)) {
    name = "int";
    res = m->nb_int(x);
  }
  #endif
#else
  if (!PyBytes_CheckExact(x) && !PyUnicode_CheckExact(x)) {
    res = PyNumber_Int(x);
  }
#endif
  if (likely(res)) {
#if PY_MAJOR_VERSION < 3
    if (unlikely(!PyInt_Check(res) && !PyLong_Check(res))) {
#else
    if (unlikely(!PyLong_CheckExact(res))) {
#endif
        return __Pyx_PyNumber_IntOrLongWrongResultType(res, name);
    }
  }
  else if (!PyErr_Occurred()) {
    PyErr_SetString(PyExc_TypeError,
                    "an integer is required");
  }
  return res;
}
static CYTHON_INLINE Py_ssize_t __Pyx_PyIndex_AsSsize_t(PyObject* b) {
  Py_ssize_t ival;
  PyObject *x;
#if PY_MAJOR_VERSION < 3
  if (likely(PyInt_CheckExact(b))) {
    if (sizeof(Py_ssize_t) >= sizeof(long))
        return PyInt_AS_LONG(b);
    else
        return PyInt_AsSsize_t(b);
  }
#endif
  if (likely(PyLong_CheckExact(b))) {
    #if CYTHON_USE_PYLONG_INTERNALS
    const digit* digits = ((PyLongObject*)b)->ob_digit;
    const Py_ssize_t size = Py_SIZE(b);
    if (likely(__Pyx_sst_abs(size) <= 1)) {
        ival = likely(size) ? digits[0] : 0;
        if (size == -1) ival = -ival;
        return ival;
    } else {
      switch (size) {
         case 2:
           if (8 * sizeof(Py_ssize_t) > 2 * PyLong_SHIFT) {
             return (Py_ssize_t) ((((size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
         case -2:
           if (8 * sizeof(Py_ssize_t) > 2 * PyLong_SHIFT) {
             return -(Py_ssize_t) ((((size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
         case 3:
           if (8 * sizeof(Py_ssize_t) > 3 * PyLong_SHIFT) {
             return (Py_ssize_t) ((((((size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
         case -3:
           if (8 * sizeof(Py_ssize_t) > 3 * PyLong_SHIFT) {
             return -(Py_ssize_t) ((((((size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
         case 4:
           if (8 * sizeof(Py_ssize_t) > 4 * PyLong_SHIFT) {
             return (Py_ssize_t) ((((((((size_t)digits[3]) << PyLong_SHIFT) | (size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
         case -4:
           if (8 * sizeof(Py_ssize_t) > 4 * PyLong_SHIFT) {
             return -(Py_ssize_t) ((((((((size_t)digits[3]) << PyLong_SHIFT) | (size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]));
           }
           break;
      }
    }
    #endif
    return PyLong_AsSsize_t(b);
  }
  x = PyNumber_Index(b);
  if (!x) return -1;
  ival = PyInt_AsSsize_t(x);
  Py_DECREF(x);
  return ival;
}
static CYTHON_INLINE Py_hash_t __Pyx_PyIndex_AsHash_t(PyObject* o) {
  if (sizeof(Py_hash_t) == sizeof(Py_ssize_t)) {
    return (Py_hash_t) __Pyx_PyIndex_AsSsize_t(o);
#if PY_MAJOR_VERSION < 3
  } else if (likely(PyInt_CheckExact(o))) {
    return PyInt_AS_LONG(o);
#endif
  } else {
    Py_ssize_t ival;
    PyObject *x;
    x = PyNumber_Index(o);
    if (!x) return -1;
    ival = PyInt_AsLong(x);
    Py_DECREF(x);
    return ival;
  }
}
static CYTHON_INLINE PyObject * __Pyx_PyBool_FromLong(long b) {
  return b ? __Pyx_NewRef(Py_True) : __Pyx_NewRef(Py_False);
}
static CYTHON_INLINE PyObject * __Pyx_PyInt_FromSize_t(size_t ival) {
    return PyInt_FromSize_t(ival);
}


#endif /* Py_PYTHON_H */
```

main.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


from Initializer import initializer

initializer()
```

AuthUtil.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import ctypes
import datetime
import json
import os
from pathlib import Path

import PySimpleGUI as sg
from cryptography.fernet import Fernet

from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class AuthUtil:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object, which in this case means that it creates a new window and displays it on screen.

    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.file_name = None
        self.append_file = None
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
        self.k = None
        self.keyFlag = True
        self.jsonDict = {}
        self.passFlagUre = False
        self.passFlagCm = False
        self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
                           "Submitting will overwrite any old values in an unrecoverable manner."

        if os.path.exists(self.filePath):
            pass
        else:
            if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
                os.mkdir(self.filePath)
            else:
                os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
                os.mkdir(self.filePath)

        if os.path.exists(self.keyPath):
            pass
        else:
            if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
                os.mkdir(self.keyPath)
            else:
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:
            ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as hidden. This is either a Permission error or Input Error. Continuing...")
            pass

    def __SetValues(self, values):
        """
    The __SetValues function is called when the user clicks on the &quot;OK&quot; button in the window.
    It takes a dictionary of values as an argument, and then uses those values to update
    the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

    Args:
        self: Make the function a method of the class
        values: Store the values that are entered into the form

    Returns:
        A dictionary of the values entered by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None
        self.popupFlag = False

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        # Try initial decoding, if fails pass and write new keys and files
        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE functionality will be accessed")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM functionality will be accessed")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            if values["-cmAuth-"].startswith("Basic"):
                self.jsonDict.update(
                    {"cm": {"parameter": "Authorization",
                            "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
                self.passFlagCm = True
            else:
                PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction Monitor",
                             windowType="AuthError")
                self.popupFlag = True
                pass
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction Monitor",
                         windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate", windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
            PopupWrapped("Please make sure you provide a key for Construction Monitor", windowType="errorLarge")
        if self.popupFlag:
            pass
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is a helper function that displays the GUI to the user.
    It takes in two arguments: layout and text. The layout argument is a list of lists,
    which contains all the elements that will be displayed on screen. The text argument
    is simply what will be displayed at the top of the window.

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the gui to be displayed
        text: Set the title of the window

    Returns:
        A window object
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while not self.passFlagUre or not self.passFlagCm:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                except Exception as e:
                    print(e)
                    RESTError(993)
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

    def __CreateFrame(self):
        """
    The __CreateFrame function creates the GUI layout for the Authentication Utility.
    It is called by __init__ and returns a list of lists that contains all the elements
    that will be displayed in the window.

    Args:
        self: Access the class attributes and methods

    Returns:
        A list of lists

    Doc Author:
        Trelent
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Push(),
                 sg.Text("Utah Real Estate API Key: ", justification="center"),
                 sg.Push()]

        line3 = [sg.Push(),
                 sg.Input(default_text="123", key="-ureAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line4 = [sg.HSeparator()]

        line5 = [sg.Push(),
                 sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
                 sg.Push()]

        line6 = [sg.Push(),
                 sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text(self.outcomeText, justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout
```

BatchProcessing.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import math
from datetime import date

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList


def BatchCalculator(TotalRecords, Argument_Dict):
    """
The BatchCalculator function takes two arguments:
    1. TotalRecords - the total number of records in the database
    2. Argument_Dict - a dictionary containing all the arguments passed to this function by the user

Args:
    TotalRecords: Determine the number of batches that will be needed to complete the query
    Argument_Dict: Pass in the arguments that will be used to query the database

Returns:
    The total number of batches that will be made

Doc Author:
    Willem van der Schans, Trelent AI
"""
    try:
        document_limit = Argument_Dict["size"]
    except Exception as e:
        # Logging
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Batch Calculator document limit overwritten to 200 from input")
        document_limit = 200

    return int(math.ceil(float(TotalRecords) / float(document_limit)))


class BatchProcessorConstructionMonitor:

    def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):

        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is created, and it sets up the attributes of that object. In this case, we are setting up our
    object to have a dataframe attribute (which will be used to store all of our data), as well as
    attributes for each parameter in our ReST call.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to retrieve
        ParameterDict: Pass in the parameters that will be used to make the api call
        HeaderDict: Pass the header dictionary from the main function to this class
        ColumnSelection: Determine which columns to pull from the api
        valueObject: Pass in the value object that is used to determine what values are returned

    Returns:
        An object of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.valueObject = valueObject
        self.__maxRequests = 10000
        self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])))
        self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
        self.__dateTracker = None

    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject and passes it to the ConstructionMonitorProcessor function.
    The ConstructionMonitorProcessor function then uses this valueObject to determine which of its functions should be called.

    Args:
        self: Represent the instance of the class

    Returns:
        The result of the constructionmonitorprocessor function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.ConstructionMonitorProcessor(self.valueObject)

    def ConstructionMonitorProcessor(self, valueObject):
        """
    The ConstructionMonitorProcessor function will use requests to get data from
       ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called __df (which is local). This
       process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST API, at which point __df will contain all

    Args:
        self: Represent the instance of the object itself
        valueObject: Update the progress bar in the gui

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __df = None
        for callNum in range(0, self.__requestCount):
            self.__parameterDict["from"] = 0

            if self.__requestCount > 1 and callNum != self.__requestCount - 1:
                __batchNum = self.__requestCalls
                if __df is None:
                    self.__dateTracker = str(date.today())
                else:
                    self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
            elif self.__requestCount == 1:
                __batchNum = self.__numBatches
                self.__dateTracker = str(date.today())
            else:
                __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])) - (
                        self.__requestCount - 1)
                self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')

            self.__parameterDict['dateEnd'] = self.__dateTracker

            for record in range(0, int(math.ceil(__batchNum))):
                if record != 0:
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict} | Headers = {self.__headerDict}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)


class BatchProcessorUtahRealEstate:

    def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject):
        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is instantiated, and it sets up the attributes of that object. In this case, we are setting up
    the dataframe attribute to be None (which will be set later), and we are also setting up some
    other attributes which will help us make our API calls.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to pull from the api
        ParameterString: Pass the parameters to the rest api
        HeaderDict: Pass in the header information for the api call
        valueObject: Create a dataframe from the json response

    Returns:
        The instance of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterString = ParameterString
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.valueObject = valueObject

    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject as an argument and then calls the appropriate
        function based on what was selected in the dropdown menu.  The valueObject is passed to each of these functions
        so that they can access all of its attributes.

    Args:
        self: Represent the instance of the class

    Returns:
        The function that is selected by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

    def BatchProcessingUtahRealestateCom(self, valueObject):
        """
    The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it to
       update the progress bar. It also takes in self, which contains all the necessary information for this
       function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to get data from
       UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is local). This
       process will be repeated until all the data has been collected from UtahRealestate.com's ReST API, at which point __df will contain all

    Args:
        self: Represent the instance of the class
        valueObject: Pass the value of a progress bar to the function

    Returns:
        A dataframe of the scraped data

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __df = pd.DataFrame()

        for batch in range(self.__numBatches):

            if batch == 0:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
                                        headers=self.__headerDict)

                response_temp = response.json()
                __df = pd.json_normalize(response_temp, record_path=['value'])

            else:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch * 200}",
                                        headers=self.__headerDict)

                response_temp = response.json()
                response_temp = pd.json_normalize(response_temp, record_path=['value'])
                __df = pd.concat([__df, response_temp], ignore_index=True)

            valueObject.setValue(valueObject.getValue() + 1)

        self.dataframe = __df
        valueObject.setValue(-999)
```

DataSupportFunctions.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


def StringToList(string):
    """
The StringToList function takes a string and converts it into a list.
    The function is used to convert the input from the user into a list of strings, which can then be iterated through.

Args:
    string: Split the string into a list

Returns:
    A list of strings

Doc Author:
    Willem van der Schans, Trelent AI
"""
    listOut = list(string.split(","))
    return listOut
```

FileSaver.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
from pathlib import Path

import pandas as pd

from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class FileSaver:

    def __init__(self, method, outputDF, AppendingPath=None):
        """
    The __init__ function is called when the class is instantiated.
    It sets up the instance of the class, and defines all variables that will be used by other functions in this class.
    The __init__ function takes two arguments: self and method.  The first argument, self, refers to an instance of a
    class (in this case it's an instance of DataFrameSaver). The second argument, method refers to a string value that
    is passed into DataFrameSaver when it's instantiated.

    Args:
        self: Represent the instance of the class
        method: Determine which dataframe to append the new data to
        outputDF: Pass in the dataframe that will be saved to a csv file
        AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            datetime.datetime.today().strftime('%m%d%Y'))
        self.data = outputDF
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                                               keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                                         windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API request Completed | File Appended and Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Appending Statistics | Method: {method} | Appending file rows: {self.dataAppending.shape[0]}, Total Rows: {(self.dataAppending.shape[0] + self.data.shape[0])}, Duplicates Dropped {(self.dataAppending.shape[0] + self.data.shape[0])-self.outputFrame.shape[0]}")
            else:
                PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}", windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API request Completed | File Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
        else:
            pass

    def getPath(self):
        """
    The getPath function returns the path to the file.
        It is a string, and it joins the docPath with the fileName.

    Args:
        self: Represent the instance of the class

    Returns:
        The path to the file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return str(self.docPath.joinpath(self.fileName))
```

versionChecker.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/
import requests

from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


def versionChecker():
    """
The versionChecker function is used to check if the current version of the program is up-to-date.
It does this by comparing the latest release on GitHub.
If they are not equal, it will pop up a window telling you that there's an update available.

Args:

Returns:
    A popup window with the current version and latest version

Doc Author:
    Willem van der Schans, Trelent AI
"""
    current_version = "1.1.0"
    response = requests.get("https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest")
    latest_version = response.json()['name']
    text_string = f"A new version is available \n" \
                  f"Running version: {current_version} \n" \
                  f"Latest version: {latest_version}"
    print(text_string)

    if current_version != latest_version:
        PopupWrapped(text_string, windowType="versionWindow")
```

ErrorPopup.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


def ErrorPopup(textString):
    """
The ErrorPopup function is used to display a popup window with an error message.
It takes one argument, textString, which is the string that will be displayed in the popup window.
The function also opens up the log folder upon program exit.

Args:
    textString: Display the error message

Returns:
    Nothing, but it does print an error message to the console

Doc Author:
    Willem van der Schans, Trelent AI
"""
    PopupWrapped(
        f"ERROR @ {textString} \n"
        f"Log folder will be opened upon program exit",
        windowType="FatalErrorLarge")
```

ErrorPrint.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime


def RESTErrorPrint(response):
    """
The RESTErrorPrint function is used to print the response from a ReST API call.
If the response is an integer, it will be printed as-is. If it's not an integer,
it will be converted to text and then printed.

Args:
    response: Print the response from a rest api call

Returns:
    The response text

Doc Author:
    Willem van der Schans, Trelent AI
"""
    if isinstance(response, int):
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response: {response}")
    else:
        response_txt = response.text
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response: {response_txt}")
```

Logger.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
import sys
from pathlib import Path


def logger():
    """
The logger function creates a log file in the user's AppData directory.
The function will create the directory if it does not exist.
The function will also delete the oldest file when 100 logs have been saved to prevent bloat.

Args:

Returns:
    A file path to the log file that was created

Doc Author:
    Willem van der Schans, Trelent AI
"""
    dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
    if os.path.exists(dir_path):
        pass
    else:
        if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
            os.mkdir(dir_path)
        else:
            os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
            os.mkdir(dir_path)

    filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
        f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
    sys.stdout = open(filePath, 'w')
    sys.stderr = sys.stdin = sys.stdout

    def sorted_ls(path):
        """
    The sorted_ls function takes a path as an argument and returns the files in that directory sorted by modification time.

    Args:
        path: Specify the directory to be sorted

    Returns:
        A list of files in a directory sorted by modification time

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
        return list(sorted(os.listdir(path), key=mtime))

    del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
    for file in del_list:
        os.remove(dir_path.joinpath(file))
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Log file {file} deleted")
```

RESTError.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime

from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
from API_Calls.Functions.ErrorFunc.ErrorPrint import RESTErrorPrint


def RESTError(response):
    """
The RESTError function is a function that checks the status codes.
If it is 200, then everything went well and nothing happens. If it isn't 200, then an error message will be printed to
the console with information about what happened (i.e., if there was an authentication error or if the resource wasn't found).
The function also raises an exception and opens an error popup for easy debugging.

Args:
    response: Print out the response from the server

Returns:
    A text string

Doc Author:
    Trelent
"""
    if isinstance(response, int):
        status_code = response
    else:
        status_code = response.status_code

    if status_code == 200:
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Api Request completed successfully"
        print(textString)
        pass
    elif status_code == 301:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Endpoint redirection; check domain name and endpoint name"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 400:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Bad Request; check input arguments"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 401:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Authentication Error: No keys found"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 402:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Authentication Error: Cannot access decryption Key in %appdata%/roaming/GardnerUtil/security"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 403:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Access Error: the resource you are trying to access is forbidden"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 404:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Resource not found: the resource you are trying to access does not exist on the server"
        ErrorPopup(textString)
        raise NameError(textString)
    elif status_code == 405:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Method is not valid, request rejected by server"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 408:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout by server"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 503:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | The resource is not ready for the get request"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 701:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Error in coercing icon to bits (Imageloader.py)"
        ErrorPopup(textString)
        raise TypeError(textString)
    elif status_code == 801:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Resource Error, HTML cannot be parsed the website's HTML source might be changed"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 790:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout within requests"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 791:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 990:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No password input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 991:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No username input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 992:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No authentication input (Basic or User/PW)"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 993:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: input values could not be coerced to arguments"
        ErrorPopup(textString)
        print(ValueError(textString))
    elif status_code == 994:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: server returned no documents"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 1000:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Catastrophic Error"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 1001:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Main Function Error Break"
        raise SystemError(textString)
    elif status_code == 1100:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User has cancelled the program execution"
        raise KeyboardInterrupt(textString)
    elif status_code == 1101:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User returned to main menu using the exit button"
        print(textString)
    else:
        RESTErrorPrint(response)
        raise Exception(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | An unknown exception occurred")
```

BatchGui.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/

import PySimpleGUI as sg

from API_Calls.Functions.Gui.ImageLoader import ImageLoader


def BatchInputGui(batches, documentCount=None):
    """
The BatchInputGui function is a simple GUI that displays the number of batches and pages
that will be requested. It also gives the user an option to cancel or continue with their request.


Args:
    batches: Determine how many batches will be run
    documentCount: Determine how many documents will be retrieved

Returns:
    The event, which is the button that was pressed

Doc Author:
    Willem van der Schans, Trelent AI
"""
    event = None
    if documentCount is None:
        __text1 = f"This request will run {batches}"
    else:
        __text1 = f"This request will run {batches} batches and will retrieve {documentCount} rows"

    __text2 = "Press Continue to start request"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Push(),
               sg.Text(__text2, justification="center"),
               sg.Push()]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=True,
                       icon=ImageLoader("taskbar_icon.ico"))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":
            break

    window.close()

    return event


def confirmDialog():
    """
The confirmDialog function is a simple confirmation dialog that asks the user if they want to continue with the request.
The function takes no arguments and returns the button event to allow for process confirmation.

```

```
Args:

Returns:
    The event that was triggered,

Doc Author:
    Willem van der Schans, Trelent AI
"""
    event = None
    __text1 = f"This request can take multiple minutes to complete"
    __text2 = "Press Continue to start the request"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Push(),
               sg.Text(__text2, justification="center"),
               sg.Push()]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=True,
                       icon=ImageLoader("taskbar_icon.ico"))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":
            break

    window.close()

    return event
```

BatchProgressGUI.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/

import datetime
import threading
import time

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.BatchProcessing import BatchProcessorConstructionMonitor, BatchProcessorUtahRealEstate
from API_Calls.Functions.Gui.DataTransfer import DataTransfer
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped

counter = 1


class BatchProgressGUI:

    def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):

        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
    the dataframe in.

    Args:
        self: Represent the instance of the class
        BatchesNum: Determine the number of batches that will be created
        RestDomain: Specify the domain of the rest api
        ParameterDict: Pass the parameters of the request to the class
        HeaderDict: Store the headers of the dataframe
        Type: Determine the type of dataframe that is being created
        ColumnSelection: Select the columns to be displayed in the gui

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.__type = Type
        self.dataframe = None

        self.__layout = None
        self.__batches = BatchesNum
        self.__window = None
        self.__batch_counter = 0

    def BatchGuiShow(self):
        """
    The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and then calls the createGui function to create a GUI for batch processing.

    Args:
        self: Represent the instance of the class

    Returns:
        The __type of the batchgui class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.CreateProgressLayout()
        self.createGui(self.__type)

    def CreateProgressLayout(self):

        """
    The CreateProgressLayout function creates the layout for the progress window.
        The function takes in self as a parameter and returns nothing.

        Parameters:
            self (object): The object that is calling this function.

    Args:
        self: Access the class variables and methods

    Returns:
        A list of lists

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--progress_text--"),
                   sg.Push()]

        __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
                   sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]

        __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h', size=(30, 20),
                           key='--progress_bar--')]


        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

    def createGui(self, Sourcetype):

        """
    The createGui function is the main function that creates the GUI.
    It takes in a type parameter which determines what kind of batch processor to use.
    The createGui function then sets up all the variables and objects needed for
    the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
    processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
    and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts these threads.

    Args:
        self: Access the object itself
        Sourcetype: Determine which batch processor to use

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__window = sg.Window('Progress', self.__layout, finalize=True, icon=ImageLoader("taskbar_icon.ico"))

        start_time = datetime.datetime.now().replace(microsecond=0)
        update_text = f"Batch {0} completed"
        self.__window['--progress_text--'].update(update_text)
        self.__window['--progress_bar--'].update(0)
        self.__window['--time_est--'].update("Est time needed 00:00:00")

        valueObj = DataTransfer()
        valueObj.setValue(0)

        if Sourcetype == "construction_monitor":

            processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
                                                                NumBatches=self.__batches,
                                                                ParameterDict=self.__parameterDict,
                                                                HeaderDict=self.__headerDict,
                                                                ColumnSelection=self.__columnSelection,
                                                                valueObject=valueObj)
        elif Sourcetype == "utah_real_estate":
            processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
                                                           NumBatches=self.__batches,
                                                           ParameterString=self.__parameterDict,
                                                           HeaderDict=self.__headerDict,
                                                           valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

    def ProgressUpdater(self, valueObj):
        """
    The ProgressUpdater function is a callback function that updates the progress bar and text
    in the GUI. It takes in one argument, which is an object containing information about the
    current batch number. The ProgressUpdater function then checks if this value has changed from
    the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
    bar and text with this new information.

    Args:
        self: Make the progressupdater function an instance method
        valueObj: Get the current value of the batch counter

    Returns:
        The value of the batch counter

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if valueObj.getValue() != self.__batch_counter:
            self.__batch_counter = valueObj.getValue()

            __update_text = f"Batch {self.__batch_counter}/{self.__batches} completed"

            self.__window.write_event_value('update--progress_bar--', self.__batch_counter)
            self.__window.write_event_value('update--progress_text--', __update_text)
        else:
            pass

    def TimeUpdater(self, start_time):

        """
    The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to complete
    the current batch. It does this by reading the start_time variable passed in, getting the current time,
    calculating how much time has passed since start_time was set and then updating a timer string with that value.
    It then calculates an estimation of how long it will take to finish all batches based on how many batches have been completed so far.

    Args:
        self: Make the function a method of the class
        start_time: Get the time when the function is called

    Returns:
        A string that is updated every 0

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            if self.__batch_counter < self.__batches:

                __current_time = datetime.datetime.now().replace(microsecond=0)

                __passed_time = __current_time - start_time

                __timer_string = f"Time Elapsed {__passed_time}"

                try:
                    self.__window.write_event_value('update--timer--', __timer_string)
                except AttributeError as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error, this is okay if the display looks good, this exception omits fatal crashes due to an aesthetic error")
                    break

                __passed_time = __passed_time.total_seconds()

                try:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * (self.__batches / self.__batch_counter) - __passed_time)).seconds
                except:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * self.__batches - __passed_time)).seconds

                __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))

                __end_string = f"Est time needed {__time_est}"
                self.__window.write_event_value('update--time_est--',
__end_string)
            else:
                __end_string = f"Est time needed 00:00:00"
                self.__window.write_event_value('update--time_est--',
__end_string)
            time.sleep(0.25)

    def ValueChecker(self, ObjectVal):
        """
    The ValueChecker function is a thread that checks the value of an object.
        It will check if the value has changed, and if it has, it will return True.
        If not, then it returns False.

    Args:
        self: Represent the instance of the class
        ObjectVal: Get the value of the object

    Returns:
        True if the value of the object has changed, and false if it hasn't

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            time.sleep(0.3)
            if self.__batch_counter != ObjectVal.getValue():
                self.__batch_counter = ObjectVal.getValue()
                return True
            else:
                return False
```

DataTransfer.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


class DataTransfer:

    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets the initial value of self.__value to 0.

    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = 0

    def setValue(self, value):
        """
    The setValue function sets the value of the object.


    Args:
        self: Represent the instance of the class
        value: Set the value of the instance variable __value

    Returns:
        The value that was passed to it

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = value

    def getValue(self):
        """
    The getValue function returns the value of the private variable __value.
    This is a getter function that allows access to this private variable.

    Args:
        self: Represent the instance of the class

    Returns:
        The value of the instance variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return self.__value

    def whileValue(self):
        """
    The whileValue function is a function that will run the getValue function until it is told to stop.
    This allows for the program to constantly be checking for new values from the sensor.

    Args:
        self: Refer to the current instance of the class

    Returns:
        The value of the input

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            self.getValue()
```

ImageLoader.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import base64
import os
from io import BytesIO
from os.path import join, normpath

from PIL import Image


def ImageLoader(file):
    """
The ImageLoader function takes in a file name and returns the image as a base64 encoded string.
This is used to send images to the API for processing.

Args:
    file: Specify the image file to be loaded

Returns:
    A base64 encoded image string

Doc Author:
    Willem van der Schans, Trelent AI
"""
    try:
        __path = normpath(join(str(os.getcwd().split("API_Calls", 1)[0]), "API_Calls"))
        __path = normpath(join(__path, "External Files"))
        __path = normpath(join(__path, "Images"))
        __path = join(__path, file).replace("\\", "/")

        image = Image.open(__path)

        __buff = BytesIO()

        image.save(__buff, format="png")

        img_str = base64.b64encode(__buff.getvalue())

        return img_str
    except Exception as e:
        # We cannot log this error like other errors due to circular imports
        raise e
```

PopupWrapped.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/
import datetime
import os
import threading
import time
import webbrowser
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.Gui.ImageLoader import ImageLoader


class PopupWrapped():

    def __init__(self, text="", windowType="notice", error=None):
        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It sets up all the variables and creates a window for us to use.
    Args:
        self: Represent the instance of the class
        text: Set the text of the window
        windowType: Determine what type of window to create
        error: Display the error message in the window
    Returns:
        Nothing
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__text = text
        self.__type = windowType
        self.__error = error
        self.__layout = []
        self.__windowObj = None
        self.__thread = None
        self.__counter = 0
        self.__docpath = None
        self.__errorFlag = False

        try:
            if "File Appended and Saved to " in self.__text:
                self.__docpath = str(self.__text[27:])
            elif "File Saved to " in self.__text:
                self.__docpath = str(self.__text[14:])
            else:
                pass
        except Exception as e:
            if self.__type == "savedLarge":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py | Error = {e} | Error creating self.__docpath open file button not available")
                self.__errorFlag = True
            else:
                pass

        self.__createWindow()

    def __createLayout(self):
        """
    The __createLayout function is used to create the layout of the window.
    The function takes class variables and returns a window layout.
    It uses a series of if statements to determine what type of window it is, then creates a layout based on that information.
    Args:
        self: Refer to the current instance of a class
    Returns:
        A list of lists
    Doc Author:
        Willem van der Schans, Trelent AI
    """
```

```python
        sg.theme('Default1')
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "savedLarge":
            if self.__errorFlag:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
            else:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "FatalErrorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "error":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}: {self.__error}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "AuthError":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
                       sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "versionWindow":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Download"),
                       sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "progress":
            __Line1 = [sg.Push(),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]

        if self.__type == "progress":
            self.__layout = [__Line1, ]
        else:
            self.__layout = [__Line1, __Line2]

    def __createWindow(self):
        """
    The __createWindow function is used to create the window object that will be displayed.
    The function takes class variables and a window object. The function first calls __createLayout, which creates the layout for the window based on what type of message it is (error, notice, progress). Then it uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits until either 100 iterations have passed or an event has been triggered (such as clicking &quot;Ok&quot; or closing the window). Once one of these events occurs
    Args:
        self: Reference the instance of the class
    Returns:
        A window object
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__createLayout()

        if self.__type == "progress":
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 50))
        elif self.__type == "noticeLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "savedLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=False,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "errorLarge":
            self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "FatalErrorLarge":
            self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "AuthError":
            self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "versionWindow":
            self.__windowObj = sg.Window(title="Update", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        else:
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            print("Here")
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                print(event)
                if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
                    break
                elif event == "Open Generation Tool [Web Browser]":
                    webbrowser.open('https://www.debugbear.com/basic-auth-header-generator', new=2, autoraise=True)
                    pass
                elif event == "Open File":
                    threadFile = threading.Thread(target=self.openFile,
                                                  daemon=False)
                    threadFile.start()
                    time.sleep(3)
                    break
                elif event == "Download":
                    webbrowser.open('https://github.com/Kydoimos97/GardnerApiUtility/releases/latest', new=2,
                                    autoraise=True)
                    pass
                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py | Error = {e} | Log Folder not found please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

    def stopWindow(self):
        """
    The stopWindow function is used to close the window object that was created in the startWindow function.
    This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
    Args:
        self: Represent the instance of the class
    Returns:
        The window object
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__windowObj.close()

    def textUpdate(self, sleep=0.5):
        """
    The textUpdate function is a function that updates the text in the text field.
    It does this by adding dots to the end of it, and then removing them. This creates
    a loading effect for when something is being processed.
    Args:
        self: Refer to the object itself
        sleep: Control the speed of the text update
    Returns:
        A string that is the current text of the text field
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__counter += 1
        if self.__counter == 4:
            self.__counter = 1
        newString = ""
        if self.__type == "notice":
            pass
        elif self.__type == "error":
            pass
        elif self.__type == "progress":
            newString = f"{self.__text}{'.' * self.__counter}"
        self.__windowObj.write_event_value('update-textField-', newString)

        time.sleep(sleep)

    def windowPush(self):

        """
    The windowPush function is used to update the values of a window object.
        The function takes in an event and values from the window object, then checks if the event starts with 'update'.
        If it does, it will take everything after 'update' as a key for updating that specific value.
        It will then update that value using its key and refresh the window.
    Args:
        self: Reference the object that is calling the function
    Returns:
        A tuple containing the event and values
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        event, values = self.__windowObj.read()

        if event.startswith('update'):
            __key_to_update = event[len('update'):]
            self.__windowObj[__key_to_update].update(values[event])
            self.__windowObj.refresh()

    def openFile(self):
        """
    The openFile function opens the file that is associated with the
        document object.  It does this by calling os.system and passing it
        self.__docpath as an argument.

    Args:
        self: Represent the instance of the object itself

    Returns:
        The filepath of the document

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        os.system(self.__docpath)
```

Initializer.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.versionChecker import versionChecker
from API_Calls.Functions.ErrorFunc.Logger import logger
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
from API_Calls.Sources.CFBP.Core import CFBP
from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
    ConstructionMonitorMain
from API_Calls.Sources.Realtor.Core import realtorCom
from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain, UtahRealEstateInit


class initializer:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the logging, calls the __ShowGui function to create and display
    the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.classObj = None

        logger()

        print("\n\n-------------Initiate Program---------------------\n\n")

        self.__ShowGui(self.__CreateFrame(), "Data Tool")

        print("\n\n-------------Closing Program---------------------\n\n")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that displays the GUI.
    It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three elements:
        1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;, etc.)
        2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
        3) An optional key name for the element (used in event handling). If no key name is provided then one will be generated automatically by PySimpleGUIQt based on its position in the layout list

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the window to be created
        text: Set the title of the window

    Returns:
        A window object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        versionChecker()

        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Construction Monitor":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API Call-----------------")
                ConstructionMonitorMain(ConstructionMonitorInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API Call---------------------\n")
            elif event == "Utah Real Estate":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API Call-----------------")
                UtahRealEstateMain(UtahRealEstateInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API Call---------------------\n")
            elif event == "Realtor.Com":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
                realtorCom()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API Call---------------------\n")
            elif event == "CFPB Mortgage":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating ffiec.cfpb API Call-----------------")
                CFBP()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing ffiec.cfpb API Call---------------------\n")
            elif event == "Authorization Utility":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
                try:
                    os.system(f"start {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents folder",
                            windowType="errorLarge")
```

```python

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Error on program exit, for logging purposes only.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    def __CreateFrame(self):

        """
    The __CreateFrame function is a helper function that creates the layout for the main window.
    It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

    Args:
        self: Represent the instance of the class

    Returns:
        A list of lists, which is then passed to the sg

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
                  sg.Button("CFPB Mortgage", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]

        return layout
```

CFBP/Core.py

```python
import datetime
import threading
import time

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import confirmDialog
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class CFBP:

    def __init__(self, state_arg=None, year_arg=None):
        """
    The __init__ function is called when the class is instantiated.
    Its job is to initialize the object with some default values, and do any other setup that might be necessary.
    The __init__ function can take arguments, but it doesn't have to.

    Args:
        self: Represent the instance of the class
        state_arg: Set the state_arg attribute of the class
        year_arg: Set the year of data to be retrieved

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
        self.uiString = None
        self.link = None

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__showUi()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | API Link = {self.link}")
            F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")

            self.uiString = (
                f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg} data retrieved \n Data Saved at {F.getPath()}")

            PopupWrapped(text=self.uiString, windowType="noticeLarge")
        else:
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled Request")
            pass

    def __showUi(self):

        """
    The __showUi function is a function that creates a progress bar window.
    The __showUi function takes class variables and returns a windowobj.


    Args:
        self: Represent the instance of the class

    Returns:
        The uiobj variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Cenus Request running", windowType="progress",
error=None)

        threadGui = threading.Thread(target=self.__dataGetter,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __dataGetter(self):
        """
    The __dataGetter function is a private function that gets the data from the CFPB
API.
    It takes no arguments, but uses self.state_arg and self.year_arg to create a URL
for the API call.

    Args:
        self: Represent the instance of the class

    Returns:
        A response object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        arg_dict_bu = locals()

        link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" +
f"states={self.state_arg}" + f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

ConstructionMonitor/Core.py

```python
import copy
import datetime
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class ConstructionMonitorInit:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the variables that will be used by other functions in this class.


    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = None
        self.SourceInclude = None
        self.dateStart = None
        self.dateEnd = None
        self.rest_domain = None
        self.auth_key = None
        self.ui_flag = None
        self.append_file = None

        passFlag = False

        while not passFlag:
            if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                    "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                    "Security").joinpath("auth.json")):
                try:
                    f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                    key = f.readline()
                    f.close()
                    f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json"), "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
                    passFlag = True
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ConstructionMonitor/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that creates and displays the GUI.
    It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
    The text parameter specifies what title should appear at the top of the window.

    Args:
        self: Refer to the current instance of a class
        layout: Determine what the gui will look like
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    @staticmethod
    def __CreateFrame():

        """
    The __CreateFrame function creates the GUI layout for the application.
        The function returns a list of lists that contains all the elements to be displayed in the GUI window.
        This is done by creating each line as a list and then appending it to another list which will contain all lines.

    Args:

    Returns:
        The layout for the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the __GetData function.
    The __SetValues function takes a dictionary as an argument, and then sets each variable based on what is passed into
    the dictionary. The keys for this dictionary are defined by the user when they create their own instance of this class.

    Args:
        self: Represent the instance of the class
        values: Pass in the values from the ui

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = 1000

        if values["-Cal-"] != "":
            self.dateStart = values["-Cal-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-EndCal-"] != "":
            self.dateEnd = values["-EndCal-"]
        else:
            self.dateEnd = date.today().strftime("%Y-%m-%d")

```

```python
        self.rest_domain = "https://api.constructionmonitor.com/v2/powersearch/?"

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True


class ConstructionMonitorMain:

    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to run properly.


    Args:
        self: Represent the instance of the class
        siteClass: Identify the site that is being used

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)
    def mainFunc(self):
        """
    The mainFunc function is the main function of this module. It will be called by the GUI or CLI to execute
    the code in this module. The mainFunc function will first create a parameter dictionary using the __ParameterCreator
    method, then it will get a count of all records that match its parameters using the __getCountUI method, and then
    it will calculate how many batches are needed to retrieve all records with those parameters using BatchCalculator.
    After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI mode). Then it shows
    a progress bar for each

    Args:
        self: Refer to the current object

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict = {self.__parameterDict}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain = {self.__restDomain}")

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterDict,
                                                  HeaderDict=self.__headerDict,
                                                  ColumnSelection=self.__columnSelection,
                                                  BatchesNum=self.__batches,
                                                  Type="construction_monitor")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
                FileSaver("cm", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the parameter dictionary that will be passed into the
```

```
        __Request function. The function takes in a siteClass object and extracts all of its attributes, except for
        those that start with '__' or are callable. It then creates a dictionary from these attributes and stores it as
        self.__parameterDict.

    Args:
        self: Make the function a method of the class

    Returns:
        A dictionary of parameters and a list of non parameter variables

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

    def __getCount(self):
        """
    The __getCount function is used to get the total number of records that are returned from a query.
    This function is called by the __init__ function and sets the self.__record_val variable with this value.

    Args:
        self: Represent the instance of the class

    Returns:
        The total number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:

            __temp_param_dict = copy.copy(self.__parameterDict)

            __count_resp = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=__temp_param_dict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        __count_resp = __count_resp.json()

        self.__record_val = __count_resp["hits"]["total"]["value"]

        del __count_resp, __temp_param_dict

    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It allows the user to run __getCount in a separate thread, so that they can continue working while it runs.
    The function will display a progress bar and update with text as it progresses through its tasks.

    Args:
        self: Access the class variables and methods

    Returns:
        The count of the number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

Realtor/Core.py

```python
import datetime
import threading
import time

import pandas as pd
import requests
from bs4 import *

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import confirmDialog
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class realtorCom:

    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of an object, and it's where you put code that needs to run before anything else in your class.

    Args:
        self: Represent the instance of the class

    Returns:
        A new object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__page_html = None
        self.__update_date = None
        self.__last_date = None
        self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
        self.__linkDict = {}
        self.dfState = None
        self.dfCounty = None
        self.dfZip = None
        self.uiString = "Files Saved to \n"

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            page_html = requests.get("https://www.realtor.com/research/data/").text
            self.__page_html = BeautifulSoup(page_html, "html.parser")
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__linkGetter()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Link Dictionary = {self.__idDict}")
            self.__showUi()
            PopupWrapped(text=self.uiString, windowType="noticeLarge")
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
        else:
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled Request")
            pass

    def __showUi(self):

        """
    The __showUi function is a helper function that creates and displays the progress window.
    It also starts the dataUpdater thread, which will update the progress bar as it runs.


    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataUpdater,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __linkGetter(self):

        """
    The __linkGetter function is a private function that takes the idDict dictionary and adds
    a link to each entry in the dictionary. The link is used to access historical data for each
    scope symbol.

    Args:
        self: Refer to the object itself

    Returns:
        A dictionary of all the links to the history pages

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            for row in self.__page_html.find_all("div", {"class": "monthly"}):
                try:
                    for nestedRow in row.find_all("a"):
                        if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
                            self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links for realtor.com")
                    RESTError(801)
                    raise SystemExit(801)

    def __dataUpdater(self):

        """
    The __dataUpdater function is a private function that updates the dataframes for each of the three
        types of realtor data. It takes class variables and return the path to the saved file. The function first creates an empty
        dictionary called tempdf, then iterates through each key in self.__idDict (which contains all three ids).
        For each key, it reads in a csv file from the link associated with that id and saves it to tempdf as a pandas
        DataFrame object. Then, depending on which type of realtor data we are dealing with (State/County/Zip), we save


    Args:
        self: Access the attributes and methods of the class

    Returns:
        The path of the saved file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)

            if key == "State":
                self.dfState = tempdf
            elif key == "County":
                self.dfCounty = tempdf
            elif key == "Zip":
                self.dfZip = tempdf

            FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
            self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

UtahRealEstate/Core.py

```python
import copy
import datetime
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class UtahRealEstateInit:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object.


    Args:
        self: Represent the instance of the class

    Returns:
        The __createframe function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is a helper function that creates the GUI window and displays it to the user.
    It takes in two parameters: layout, which is a list of lists containing all the elements for each row;
    and text, which is a string containing what will be displayed as the title of the window. The __ShowGui
    method then uses these parameters to create an instance of sg.Window with all its attributes set accordingly.

    Args:
        self: Refer to the current class instance
        layout: Pass the layout of the window to be created
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(993)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    @staticmethod
    def __CreateFrame():
        """
    The __CreateFrame function creates the GUI layout for the application.
        The function returns a list of lists that contains all the elements to be displayed in the window.
        Each element is defined by its type and any additional parameters needed to define it.

    Args:

    Returns:
        A list of lists, which is used to create the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Utah Real Estate Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-", size=(31, 1))]

        line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date", "Close Date"],
                              key="-type-", size=(31, 1))]

        line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
                          disabled=False, size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-DateStart-")]

        line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today().strftime("%Y-%m-%d")), key="-DateEnd-", disabled=False,
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-', target="-DateEnd-")]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                  sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                           size=(20, 1)),
                  sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                                target="-AppendingFile-")]

        line11 = [sg.HSeparator()]

        line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
                  line12]

        return layout

    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the
       __GetData function. The values are passed from a dictionary called 'values' which is created
       by parsing through an XML file using ElementTree. This function also sets default values for
       some of these variables if they were not specified in the XML file.

    Args:
        self: Represent the instance of the class
        values: Pass the values from the gui to this function

    Returns:
        A dictionary with the following keys:

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None


class UtahRealEstateMain:

    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to work properly.

    Args:
        self: Represent the instance of the class
        siteClass: Determine which site to pull data from

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = 'https://resoapi.utahrealestate.com/reso/odata/Property?'
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None
        self.__record_val = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

    def mainFunc(self):

        """
    The mainFunc function is the main function of this module. It will be called by the GUI when a user clicks on
    the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your code for running your program, and it
    should return a dataframe that contains all the data you want to display in your final report.

    Args:
        self: Reference the object itself

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        passFlag = False

        while not passFlag:
            if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
                try:
                    f = open(self.keyPath, "rb")
                    key = f.readline()
                    f.close()
                    f = open(self.filePath, "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
                    self.__headerDict = {authDict["ure"]["parameter"]: authkey}
```

```python
                    passFlag = True
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String = {self.__parameterString}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain = {self.__restDomain}")

        self.__getCountUI()

        if self.__record_val is None:
            self.__record_val = 0

        self.__batches = BatchCalculator(self.__record_val, None)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterString,
                                                  HeaderDict=self.__headerDict,
                                                  BatchesNum=self.__batches,
                                                  Type="utah_real_estate")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
                FileSaver("ure", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the filter string for the ReST API call.
    The function takes in a siteClass object and extracts all of its parameters into a dictionary.
    It then creates an appropriate filter string based on those parameters.

    Args:
        self: Bind the object to the class

    Returns:
        A string to be used as the parameter in the api call

    Doc Author:
        Willem van der Schans, Trelent AI
    """
```

```
        filter_string = ""

        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        if __Source_dict["ListedOrModified"] == "Listing Date":
            filter_string = f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEnd']}"
        elif __Source_dict["ListedOrModified"] == "Modification Date":
            filter_string = f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
        elif __Source_dict["ListedOrModified"] == "Close Date":
            filter_string = f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"

        filter_string = filter_string + f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['StandardStatus']}'"

        self.__parameterString = filter_string

    def __getCount(self):
        """
    The __getCount function is used to determine the number of records that will be returned by the query.
    This function is called when a user calls the count() method on a ReST object. The __getCount function uses
    the $count parameter in OData to return only an integer value representing how many records would be returned
    by the query.

    Args:
        self: Represent the instance of the class

    Returns:
        The number of records in the data set

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:
            __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
                                        headers=self.__headerDict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        self.__record_val = int(__count_resp.json()["@odata.count"])

    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It creates a progress window and updates it while the __getCount function runs.
    The purpose of this is to keep the GUI responsive while running long processes.

    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```

Index