

# **Gardner API Utility Documentation**

Willem van der Schans  
Version 1.1.0  
4/25/2023 6:23:00 PM



# Table of Contents

README .....	3
VERSION INFO .....	3
Authentication Requirements .....	3
License .....	3
Class Index .....	4
File Index .....	5
Class Documentation .....	6
AuthUtil.AuthUtil .....	6
BatchProcessing.BatchProcessorConstructionMonitor .....	17
BatchProcessing.BatchProcessorUtahRealEstate .....	24
BatchProgressGUI.BatchProgressGUI .....	30
Core.CFBP .....	42
Core.ConstructionMonitorInit .....	47
Core.ConstructionMonitorMain .....	56
DataTransfer.DataTransfer .....	67
FileSaver.FileSaver .....	70
API_Calls.Initializer.initializer .....	74
PopupWrapped.PopupWrapped .....	80
Core.realtorCom .....	90
Core.UtahRealEstateInit .....	96
Core.UtahRealEstateMain .....	104
File Documentation .....	115
__init__.py .....	115
__main__.c .....	116
__main__.py .....	164
AuthUtil.py .....	165
BatchProcessing.py .....	170
DataSupportFunctions.py .....	174
FileSaver.py .....	175
versionChecker.py .....	177
ErrorPopup.py .....	178
ErrorPrint.py .....	179
Logger.py .....	180
RESError.py .....	181
BatchGui.py .....	184
BatchProgressGUI.py .....	186
DataTransfer.py .....	191
ImageLoader.py .....	193
PopupWrapped.py .....	194
Initializer.py .....	199
Core.py .....	202
Core.py .....	204
Core.py .....	211
Core.py .....	214
Index .....	221





# 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

This software is licensed under Apache License, Version 2.0, January 2004 as found on <http://www.apache.org/licenses/>

# Class Index

## Class List

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

<a href="#"><u>AuthUtil.AuthUtil</u></a>	6
<a href="#"><u>BatchProcessing.BatchProcessorConstructionMonitor</u></a>	17
<a href="#"><u>BatchProcessing.BatchProcessorUtahRealEstate</u></a>	24
<a href="#"><u>BatchProgressGUI.BatchProgressGUI</u></a>	30
<a href="#"><u>Core.CFBP</u></a>	42
<a href="#"><u>Core.ConstructionMonitorInit</u></a>	47
<a href="#"><u>Core.ConstructionMonitorMain</u></a>	56
<a href="#"><u>DataTransfer.DataTransfer</u></a>	67
<a href="#"><u>FileSaver.FileSaver</u></a>	70
<a href="#"><u>API Calls.Initializer.initializer</u></a>	74
<a href="#"><u>PopupWrapped.PopupWrapped</u></a>	80
<a href="#"><u>Core.realtorCom</u></a>	90
<a href="#"><u>Core.UtahRealEstateInit</u></a>	96
<a href="#"><u>Core.UtahRealEstateMain</u></a>	104

# File Index

## File List

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

<a href="#"><u>init .py</u></a>	115
<a href="#"><u>main .c</u></a>	116
<a href="#"><u>main .py</u></a>	164
<a href="#"><u>AuthUtil.py</u></a>	165
<a href="#"><u>BatchProcessing.py</u></a>	170
<a href="#"><u>DataSupportFunctions.py</u></a>	174
<a href="#"><u>FileSaver.py</u></a>	175
<a href="#"><u>versionChecker.py</u></a>	177
<a href="#"><u>ErrorPopup.py</u></a>	178
<a href="#"><u>ErrorPrint.py</u></a>	179
<a href="#"><u>Logger.py</u></a>	180
<a href="#"><u>RESError.py</u></a>	181
<a href="#"><u>BatchGui.py</u></a>	184
<a href="#"><u>BatchProgressGUL.py</u></a>	186
<a href="#"><u>DataTransfer.py</u></a>	191
<a href="#"><u>ImageLoader.py</u></a>	193
<a href="#"><u>PopupWrapped.py</u></a>	194
<a href="#"><u>Initializer.py</u></a>	199
<a href="#"><u>CFBP/Core.py</u></a>	202
<a href="#"><u>ConstructionMonitor/Core.py</u></a>	204
<a href="#"><u>Realtor/Core.py</u></a>	211
<a href="#"><u>UtahRealEstate/Core.py</u></a>	214



# 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.
```

```

00025
00026     Args:
00027         self: Represent the instance of the class
00028
00029     Returns:
00030         None
00031
00032     Doc Author:
00033         Willem van der Schans, Trelent AI
00034     """
00035         self.StandardStatus = None
00036         self.ListedOrModified = None
00037         self.file_name = None
00038         self.append_file = None
00039         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00040         self.filePath =
Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath("Secu
rity")
00041         self.k = None
00042         self.keyFlag = True
00043         self.jsonDict = {}
00044         self.passFlagUre = False
00045         self.passFlagCm = False
00046         self.outcomeText = "Please input the plain text keys in the input boxes
above \n " \
00047                               "Submitting will overwrite any old values in an
unrecoverable manner."
00048
00049         if os.path.exists(self.filePath):
00050             pass
00051         else:
00052             if
os.path.exists(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData")
):
00053                 os.mkdir(self.filePath)
00054             else:
00055                 os.mkdir(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData"))
00056                 os.mkdir(self.filePath)
00057
00058         if os.path.exists(self.keyPath):
00059             pass
00060         else:
00061             if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00062                 os.mkdir(self.keyPath)
00063             else:
00064                 os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00065                 os.mkdir(self.keyPath)
00066
00067         if
os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00068             try:
00069                 f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00070                 self.k = f.readline()
00071                 f.close()
00072             except Exception as e:
00073                 print(e)
00074                 RESTError(402)
00075                 raise SystemExit(402)
00076         else:
00077             self.k = Fernet.generate_key()
00078             f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00079             f.write(self.k)
00080             f.close()
00081
00082         try:
00083             os.remove(self.filePath.joinpath("auth.json"))
00084         except Exception as e:
00085             # Logging
00086             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...")

```

```

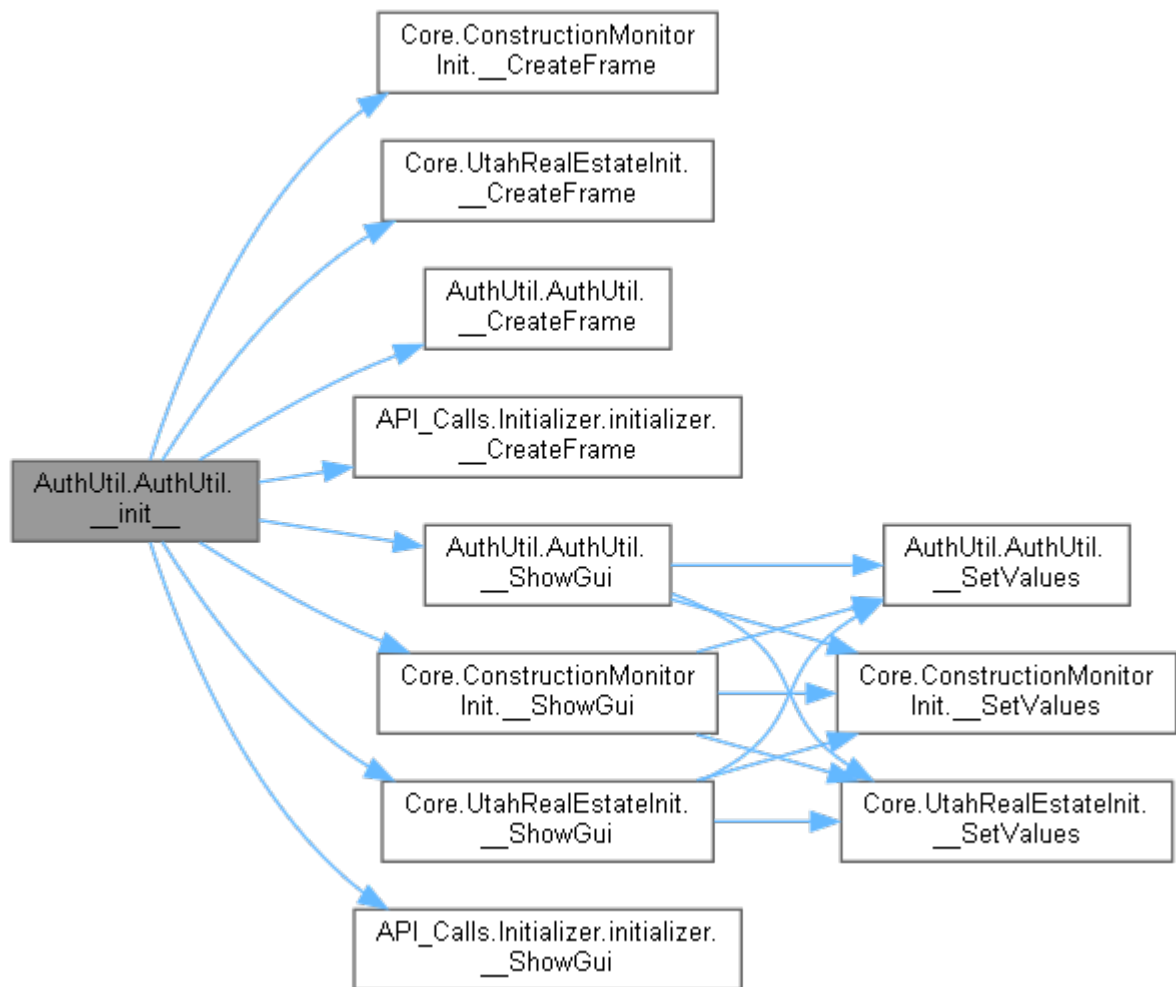
00088         pass
00089
00090         f = open(self.filePath.joinpath("auth.json"), "wb")
00091         f.close()
00092         self.keyFlag = False
00093
00094         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00095
00096         try:
00097             ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35
00098             .av3ra3rvavcr3w"), 2)
00099             except Exception as e:
00100                 # Logging
00101                 print(
00102                     f"{datetime.datetime.today().strftime('%m-%d-%Y
00103                     %H:%M:%S.%f')}[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as
00104                     hidden. This is either a Permission error or Input Error. Continuing...")
00105                 pass
00106

```

References

[Core.ConstructionMonitorInit. CreateFrame\(\)](#),  
[Core.UtahRealEstateInit. CreateFrame\(\)](#),  
[API Calls.Initializer.initializer. CreateFrame\(\)](#),  
[API Calls.Initializer.initializer. ShowGui\(\)](#),  
[Core.UtahRealEstateInit. ShowGui\(\)](#),  
[Core.ConstructionMonitorInit.append\\_file](#),  
[Core.UtahRealEstateInit.append\\_file](#),  
[AuthUtil.AuthUtil.file\\_name](#),  
[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:



## 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
00238         Utility.
00239         It is called by __init__ and returns a list of lists that contains all the
00240         elements
00241         that will be displayed in the window.
00242         Args:
00243             self: Access the class attributes and methods
  
```

```

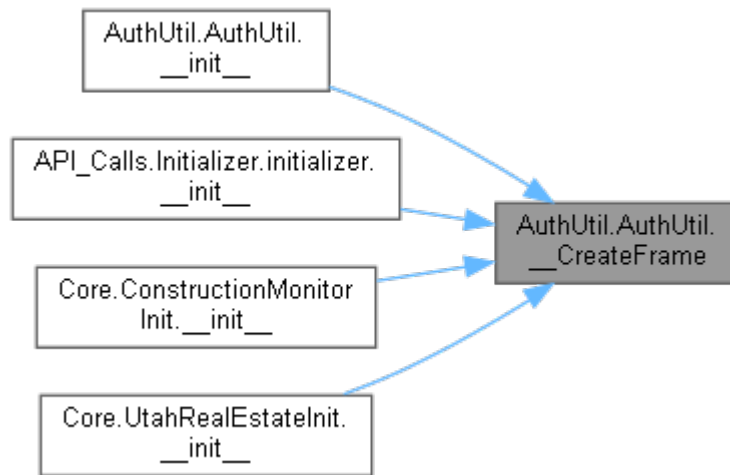
00243
00244     Returns:
00245         A list of lists
00246
00247     Doc Author:
00248         Trelent
00249     """
00250     sg.theme('Default1')
00251
00252     line00 = [sg.HSeparator()]
00253
00254     line0 = [sg.Image(ImageLoader("logo.png")),
00255              sg.Push(),
00256              sg.Text("Authentication Utility", font=("Helvetica", 12,
00257 "bold"), justification="center"),
00258              sg.Push(),
00259              sg.Push()]
00260
00261     line1 = [sg.HSeparator()]
00262
00263     line2 = [sg.Push(),
00264              sg.Text("Utah Real Estate API Key: ", justification="center"),
00265              sg.Push()]
00266
00267     line3 = [sg.Push(),
00268              sg.Input(default_text="123", key="-ureAuth-",
00269 disabled=False,
00270                      size=(40, 1)),
00271              sg.Push()]
00272
00273     line4 = [sg.HSeparator()]
00274
00275     line5 = [sg.Push(),
00276              sg.Text("Construction Monitor HTTP BASIC Key: ",
00277 justification="center"),
00278              sg.Push()]
00279
00280     line6 = [sg.Push(),
00281              sg.Input(default_text="Basic 123", key="-cmAuth-",
00282 disabled=False,
00283                      size=(40, 1)),
00284              sg.Push()]
00285
00286     line7 = [sg.HSeparator()]
00287
00288     line8 = [sg.Push(),
00289              sg.Text(self.outcomeText, justification="center"),
00290              sg.Push()]
00291
00292     line9 = [sg.HSeparator()]
00293
00294     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00295
00296     layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
00297 line8, line9, line10]
00298
00299     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:



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

The `__SetValues` function is called when the user clicks on the "OK" 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](#).

```

00104     def __SetValues(self, values):
00105
00106         """
00107         The __SetValues function is called when the user clicks on the "OK"
button in the window.
00108         It takes a dictionary of values as an argument, and then uses those values
to update
00109         the auth.json file with new keys for both Utah Real Estate and Construction
Monitor.
00110
00111         Args:
00112             self: Make the function a method of the class
00113             values: Store the values that are entered into the form
00114
00115         Returns:
00116             A dictionary of the values entered by the user
00117
00118         Doc Author:
00119             Willem van der Schans, Trelent AI
00120         """
00121         ureCurrent = None
00122         cmCurrent = None
00123         keyFile = None
00124         self.popupFlag = False
00125
00126         fernet = Fernet(self.k)
00127
00128         try:
00129             f = open(self.filePath.joinpath("auth.json"), "r")
00130             keyFile = json.load(f)
00131             fileFlag = True
00132         except:
00133             fileFlag = False
00134
  
```

```

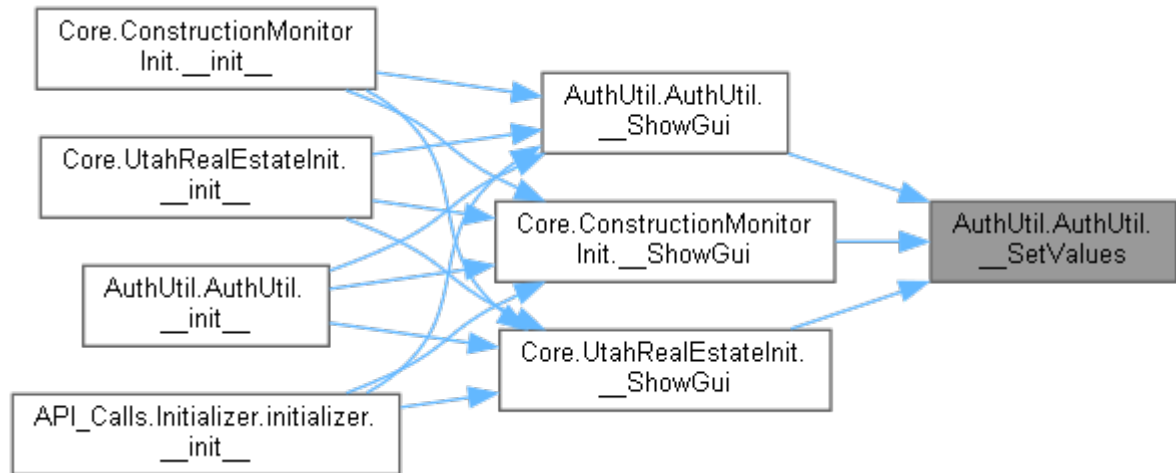
00135         # Try initial decoding, if fails pass and write new keys and files
00136         if fileFlag:
00137             try:
00138                 ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00139             except Exception as e:
00140                 # Logging
00141                 print(
00142                     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")
00143                 ureCurrent = None
00144
00145             try:
00146                 cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00147             except Exception as e:
00148                 # Logging
00149                 print(
00150                     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")
00151                 cmCurrent = None
00152
00153         if values["-ureAuth-"] != "":
00154             self.jsonDict.update(
00155                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00156             self.passFlagUre = True
00157         elif ureCurrent is not None:
00158             self.jsonDict.update(
00159                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00160             self.passFlagUre = True
00161         else:
00162             pass
00163
00164         if values["-cmAuth-"] != "":
00165             if values["-cmAuth-"].startswith("Basic"):
00166                 self.jsonDict.update(
00167                     {"cm": {"parameter": "Authorization",
00168                         "auth":
fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00169                 self.passFlagCm = True
00170             else:
00171                 PopupWrapped("Please make sure you provide a HTTP Basic Auth key
for construction Monitor",
00172                                     windowType="AuthError")
00173                 self.popupFlag = True
00174                 pass
00175         elif ureCurrent is not None:
00176             self.jsonDict.update(
00177                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00178             self.passFlagUre = True
00179         else:
00180             pass
00181
00182         if not self.passFlagUre and not self.passFlagCm:
00183             PopupWrapped("Please make sure you provide keys for both Utah Real
estate and Construction Monitor",
00184                                     windowType="errorLarge")
00185         if self.passFlagCm and not self.passFlagUre:
00186             PopupWrapped("Please make sure you provide a key for Utah Real
estate", windowType="errorLarge")
00187         if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
00188             PopupWrapped("Please make sure you provide a key for Construction
Monitor", windowType="errorLarge")
00189         if self.popupFlag:
00190             pass
00191         else:
00192             jsonOut = json.dumps(self.jsonDict, indent=4)
00193             f = open(self.filePath.joinpath("auth.json"), "w")
00194             f.write(jsonOut)
00195

```

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:



```
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
00201         of lists,
00202         which contains all the elements that will be displayed on screen. The text
00203         argument
00204         is simply what will be displayed at the top of the window.
00205
00206         Args:
00207         self: Represent the instance of the class
00208         layout: Pass the layout of the gui to be displayed
00209         text: Set the title of the window
00210
00211         Returns:
00212         A window object
00213         """
00214         window = sg.Window(text, layout, grab_anywhere=False,
00215                             return_keyboard_events=True,
00216                             finalize=True,
00217                             icon=ImageLoader("taskbar_icon.ico"))
00218
00219         while not self.passFlagUre or not self.passFlagCm:
00220             event, values = window.read()
00221
00222             if event == "Submit":
00223                 try:
00224                     self.__SetValues(values)
00225                 except Exception as e:
00226                     print(e)
00227                     RESTError(993)

```



```

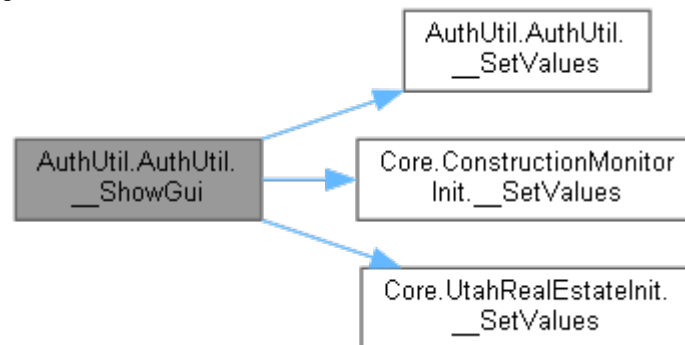
00225         finally:
00226             pass
00227         elif event == sg.WIN_CLOSED or event == "Quit":
00228             break
00229         else:
00230             pass
00231         pass
00232     window.close()
00233
00234

```

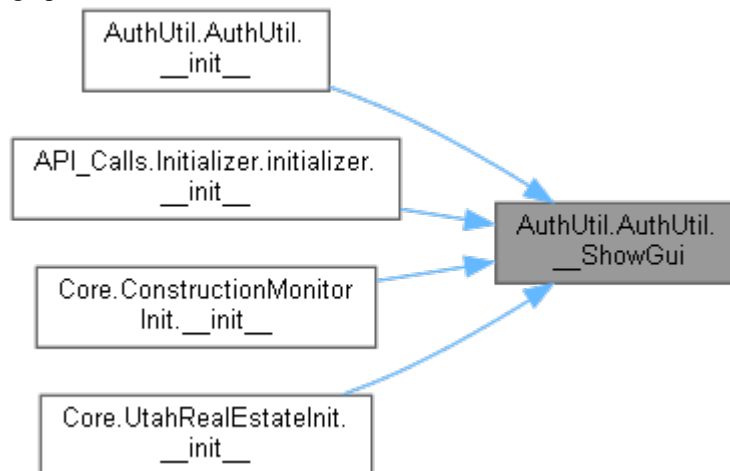
References [AuthUtil.AuthUtil.\\_\\_SetValues\(\)](#), [Core.ConstructionMonitorInit.\\_\\_SetValues\(\)](#), [Core.UtahRealEstateInit.\\_\\_SetValues\(\)](#), [AuthUtil.AuthUtil.passFlagCm](#), and [AuthUtil.AuthUtil.passFlagUre](#).

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:



Here is the caller graph for this function:



## 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](#).

```

00043     def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict,
00044 ColumnSelection, valueObject):
00045         """
00046         The __init__ function is the constructor for a class. It is called when an
00047         object of that class
00048         is created, and it sets up the attributes of that object. In this case, we
00049         are setting up our
00050         object to have a dataframe attribute (which will be used to store all of our
00051         data), as well as
00052         attributes for each parameter in our ReST call.
00053
00054     Args:
00055         self: Represent the instance of the class
00056         RestDomain: Specify the domain of the rest api
00057         NumBatches: Determine how many batches of data to retrieve
00058         ParameterDict: Pass in the parameters that will be used to make the api
00059         call
00060         HeaderDict: Pass the header dictionary from the main function to this
00061         class
00062         ColumnSelection: Determine which columns to pull from the api
00063         valueObject: Pass in the value object that is used to determine what
00064         values are returned
00065
00066     Returns:
00067         An object of the class
00068
00069     Doc Author:
00070         Willem van der Schans, Trelent AI
00071
00072     """
00073     self.dataframe = None
00074     self.__numBatches = NumBatches
00075     self.__parameterDict = ParameterDict
00076     self.__restDomain = RestDomain
00077     self.__headerDict = HeaderDict
00078     self.__columnSelection = ColumnSelection
00079     self.valueObject = valueObject
00080     self.__maxRequests = 10000
00081     self.__requestCount = math.ceil(self.__numBatches /
00082 (self.__maxRequests / int(self.__parameterDict['size'])))
00083     self.__requestCalls = math.ceil(self.__maxRequests /
00084 int(self.__parameterDict['size']))
00085     self.__dateTracker = None
00086

```

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](#),

---

## 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
00098         object called __df (which is local). This
00099         process will be repeated until all the data has been collected from
00100         ConstructionMonitor.com's ReST API, at which point __df will contain all
00101
00102         Args:
00103         self: Represent the instance of the object itself
00104         valueObject: Update the progress bar in the gui
00105
00106         Returns:
00107         A dataframe
00108
00109         Doc Author:
00110         Willem van der Schans, Trelent AI
00111         """
00112         __df = None
00113         for callNum in range(0, self.__requestCount):
00114             self.__parameterDict["from"] = 0
00115
00116             if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00117                 __batchNum = self.__requestCalls
00118                 if __df is None:
00119                     self.__dateTracker = str(date.today())
00120                 else:
00121                     self.__dateTracker =
00122                     min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00123             elif self.__requestCount == 1:
00124                 __batchNum = self.__numBatches
00125                 self.__dateTracker = str(date.today())
00126             else:
00127                 __batchNum = self.__numBatches / (self.__maxRequests /
00128                 int(self.__parameterDict['size'])) - (
00129                     self.__requestCount - 1)
00130                 self.__dateTracker =
00131                 min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00132
00133             self.__parameterDict['dateEnd'] = self.__dateTracker
00134
00135             for record in range(0, int(math.ceil(__batchNum))):
00136                 if record != 0:
```

```

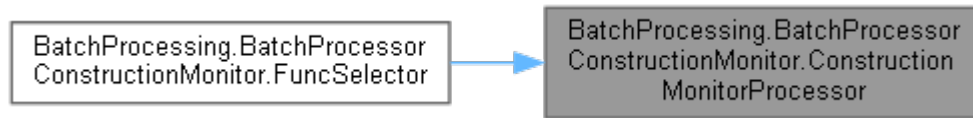
00132             self.__parameterDict["from"] = record *
int(self.__parameterDict["size"])
00133
00134             response = requests.post(url=self.__restDomain,
00135                                     headers=self.__headerDict,
00136                                     json=self.__parameterDict)
00137
00138             counter = 0
00139             try:
00140                 response = response.json()['hits']['hits']
00141             except KeyError as e:
00142                 # Logging
00143                 print(
00144                     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}")
00145                 continue
00146
00147             valueObject.setValue(valueObject.getValue() + 1)
00148
00149             if record == 0 and callNum == 0:
00150                 __df = pd.json_normalize(response[counter][["_source"]])
00151                 __df["id"] = response[counter]['_id']
00152                 __df["county"] =
response[counter][["_source"]]['county']['county_name']
00153                 counter += 1
00154
00155                 for i in range(counter, len(response)):
00156                     __tdf = pd.json_normalize(response[i][["_source"]])
00157                     __tdf["id"] = response[i]['_id']
00158                     __tdf["county"] =
response[i][["_source"]]['county']['county_name']
00159                     __df = pd.concat([__df, __tdf], ignore_index=True)
00160
00161                 if self.__columnSelection is not None:
00162                     __col_list = StringToList(self.__columnSelection)
00163                     __col_list.append("id")
00164                     __col_list.append("county")
00165                 else:
00166                     pass
00167
00168                 self.dataframe = __df
00169                 valueObject.setValue(-999)
00170
00171

```

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:



**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](#).

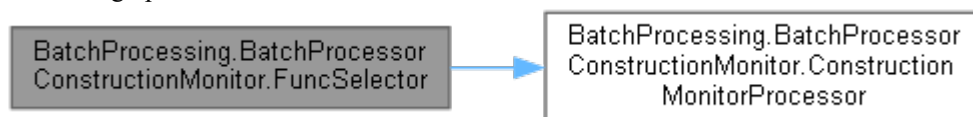
```
00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes
00081         it to the ConstructionMonitorProcessor function.
00082         The ConstructionMonitorProcessor function then uses this valueObject to
00083         determine which of its functions should be called.
00084
00085         Args:
00086             self: Represent the instance of the class
00087
00088         Returns:
00089             The result of the constructionmonitorprocessor function
00090
00091         Doc Author:
00092             Willem van der Schans, Trelent AI
00093         """
00094         self.ConstructionMonitorProcessor(self.valueObject)
```

#### References

[BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor\(\)](#),  
[BatchProcessing.BatchProcessorConstructionMonitor.valueObject](#),  
[BatchProcessing.BatchProcessorUtahRealEstate.valueObject](#).

and

Here is the call graph for this function:



---

## 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
00177     object of that class is instantiated, and it sets up the attributes of that object. In this case,
00178     we are setting up
00179     the dataframe attribute to be None (which will be set later), and we are also
00179     setting up some
00179     other attributes which will help us make our API calls.
```

```

00180
00181     Args:
00182         self: Represent the instance of the class
00183         RestDomain: Specify the domain of the rest api
00184         NumBatches: Determine how many batches of data to pull from the api
00185         ParameterString: Pass the parameters to the rest api
00186         HeaderDict: Pass in the header information for the api call
00187         valueObject: Create a dataframe from the json response
00188
00189     Returns:
00190         The instance of the class
00191
00192     Doc Author:
00193         Willem van der Schans, Trelent AI
00194     """
00195     self.dataframe = None
00196     self.__numBatches = NumBatches
00197     self.__parameterString = ParameterString
00198     self.__restDomain = RestDomain
00199     self.__headerDict = HeaderDict
00200     self.valueObject = valueObject
00201

```

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:

```

Definition at line 219 of file [BatchProcessing.py](#).

```

00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221         The BatchProcessingUtahRealestateCom function is a function that takes in
00222         the valueObject and uses it to
00223         update the progress bar. It also takes in self, which contains all the
00224         necessary information for this
00225         function to work properly. The BatchProcessingUtahRealestateCom function
00226         will then use requests to get data from
00227         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
00228         called __df (which is local). This
00229         process will be repeated until all the data has been collected from
00230         UtahRealestate.com's ReST API, at which point __df will contain all
00231
00232     Args:
00233         self: Represent the instance of the class
00234         valueObject: Pass the value of a progress bar to the function
00235
00236     Returns:
00237         A dataframe of the scraped data
00238
00239     Doc Author:
00240         Willem van der Schans, Trelent AI
00241     """
00242     __df = pd.DataFrame()
00243
00244     for batch in range(self.__numBatches):
00245
00246         if batch == 0:
00247             response =
00248             requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00249                         headers=self.__headerDict)
00250
00251             response_temp = response.json()
00252             __df = pd.json_normalize(response_temp, record_path=['value'])
00253
00254         else:
00255             response =
00256             requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
00257 200}",
00258                         headers=self.__headerDict)
00259
00260             response_temp = response.json()
00261             response_temp = pd.json_normalize(response_temp,
00262             record_path=['value'])
00263             __df = pd.concat([__df, response_temp], ignore_index=True)
00264
00265             valueObject.setValue(valueObject.getValue() + 1)
00266
00267     self.dataframe = __df
00268     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:



## 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](#).

```
00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an
00205         argument and then calls the appropriate
00206         function based on what was selected in the dropdown menu. The
00207         valueObject is passed to each of these functions
00208         so that they can access all of its attributes.
00209
00210     Args:
00211         self: Represent the instance of the class
00212
00213     Returns:
00214         The function that is selected by the user
00215
00216     Doc Author:
00217         Willem van der Schans, Trelent AI
00218     """
00219     self.BatchProcessingUtahRealestateCom(self.valueObject)
```

### References

[BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom\(\)](#),  
[BatchProcessing.BatchProcessorConstructionMonitor.valueObject](#),  
[BatchProcessing.BatchProcessorUtahRealEstate.valueObject](#).

and

Here is the call graph for this function:



## 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

#### [dataframe](#)Private 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](#).

```

00019     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type,
00020                   ColumnSelection=None):
00021         """
00022         The __init__ function is the first function that gets called when an object
00023         of this class is created.
00024         It initializes all the variables and sets up a layout for the GUI. It also
00025         creates a window to display
00026         the dataframe in.
00027
00028         Args:
00029             self: Represent the instance of the class
00030             BatchesNum: Determine the number of batches that will be created
00031             RestDomain: Specify the domain of the rest api
00032             ParameterDict: Pass the parameters of the request to the class
00033             HeaderDict: Store the headers of the dataframe
00034             Type: Determine the type of dataframe that is being created
00035             ColumnSelection: Select the columns to be displayed in the gui
00036
00037         Returns:
00038             Nothing
00039
00040         Doc Author:
00041             Willem van der Schans, Trelent AI
00042
00043         """
00044         self.__parameterDict = ParameterDict
00045         self.__restDomain = RestDomain
00046         self.__headerDict = HeaderDict
00047         self.__columnSelection = ColumnSelection
00048         self.__type = Type
00049         self.dataframe = None
00050
00051         self.__layout = None
00052         self.__batches = BatchesNum
00053         self.__window = None
00054         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

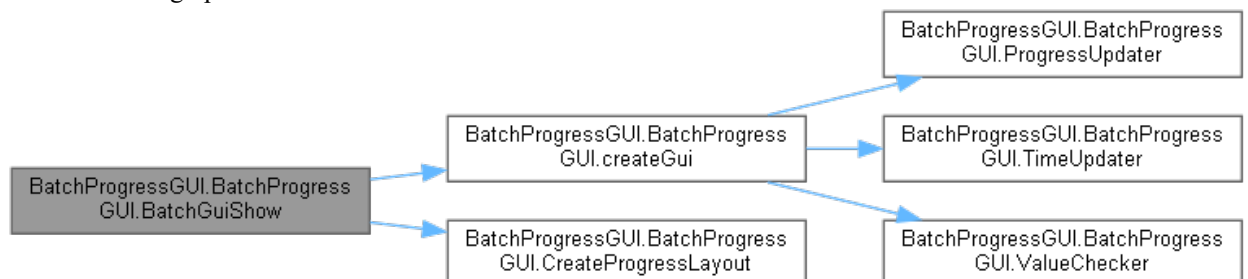
Doc Author:  
Willem van der Schans, Trelent AI

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

References [BatchProgressGUI.BatchProgressGUI.\\_\\_type](#), [PopupWrapped.PopupWrapped.\\_\\_type](#), [BatchProgressGUI.BatchProgressGUI.createGui\(\)](#), and [BatchProgressGUI.BatchProgressGUI.CreateProgressLayout\(\)](#).

Here is the call graph for this function:



**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](#).

```

00104     def createGui(self, Sourcetype):
00105
00106         """
00107         The createGui function is the main function that creates the GUI.
00108         It takes in a type parameter which determines what kind of batch processor
00109         to use.
00110         The createGui function then sets up all the variables and objects needed for
00111         the program to run, including: window, start_time, update_text, valueObj
00112         (DataTransfer),
00113         processorObject (BatchProcessorConstructionMonitor or
00114         BatchProcessorUtahRealestate),
00115         and threading objects for TimeUpdater and ValueChecker functions. The
00116         createGui function also starts these threads.
00117
00118         Args:
00119             self: Access the object itself
00120             Sourcetype: Determine which batch processor to use
00121
00122         Returns:
00123             The dataframe
00124
00125         Doc Author:
00126             Willem van der Schans, Trelent AI
00127         """
00128         self.__window = sg.Window('Progress', self.__layout, finalize=True,
00129         icon=ImageLoader("taskbar_icon.ico"))
00130
00131         start_time = datetime.datetime.now().replace(microsecond=0)
00132         update_text = f"Batch {0} completed"
00133         self.__window['--progress_text--'].update(update_text)
00134         self.__window['--progress_bar--'].update(0)
00135         self.__window['--time_est--'].update("Est time needed 00:00:00")
00136
00137         valueObj = DataTransfer()
00138         valueObj.setValue(0)
00139
00140         if Sourcetype == "construction_monitor":
00141
00142             processorObject =
00143             BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00144             NumBatches=self.__batches,
00145             ParameterDict=self.__parameterDict,
00146             HeaderDict=self.__headerDict,
00147             ColumnSelection=self.__columnSelection,
00148             valueObject=valueObj)
00149         elif Sourcetype == "utah_real_estate":
00150             processorObject =
00151             BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00152             NumBatches=self.__batches,
00153             ParameterString=self.__parameterDict,
00154             HeaderDict=self.__headerDict,
00155             valueObject=valueObj)
00156
00157         threading.Thread(target=self.TimeUpdater,
00158             args=(start_time,),
00159             daemon=True).start()
00160         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00161         %H:%M:%S.%f')}[:-3]} | TimeUpdater Thread Successfully Started")
00162
00163         batchFuncThread =
00164         threading.Thread(target=processorObject.FuncSelector,
00165             daemon=False)
00166         batchFuncThread.start()
00167         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00168         %H:%M:%S.%f')}[:-3]} | BatchFunc Thread Successfully Started")
00169         threading.Thread(target=self.ValueChecker,
00170             args=(valueObj,),
00171             daemon=False).start()

```

```

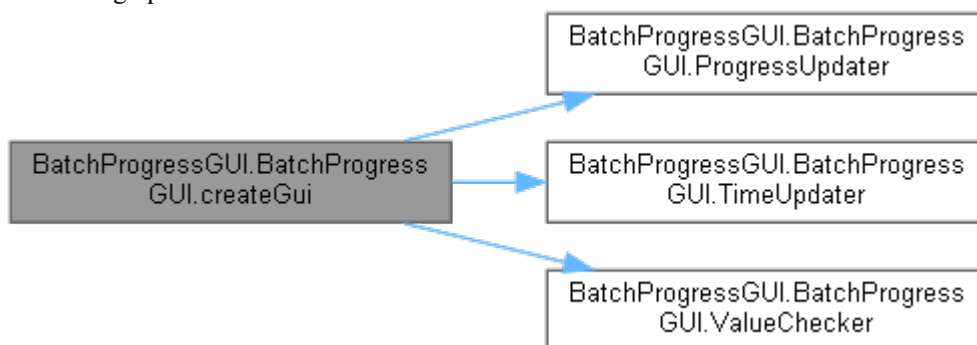
00162         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | ValueChecker Thread Successfully Started")
00163
00164         while True:
00165
00166             self.ProgressUpdater(valueObj)
00167
00168             if valueObj.getValue() == -999:
00169                 break
00170
00171             window, event, values = sg.read_all_windows()
00172             if event.startswith('update'):
00173                 __key_to_update = event[len('update'):]
00174                 window[__key_to_update].update(values[event])
00175                 window.refresh()
00176                 pass
00177
00178             if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179                 break
00180
00181             time.sleep(0.1)
00182
00183             self.dataframe = processorObject.dataframe
00184             self.__window.close()
00185
00186             PopupWrapped(text="Api Request Completed", windowType="notice")
00187

```

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:



Here is the caller graph for this function:



**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](#).

```
00069     def CreateProgressLayout(self):
00070
00071         """
00072         The CreateProgressLayout function creates the layout for the progress window.
00073         The function takes in self as a parameter and returns nothing.
00074
00075         Parameters:
00076             self (object): The object that is calling this function.
00077
00078         Args:
00079             self: Access the class variables and methods
00080
00081         Returns:
00082             A list of lists
00083
00084         Doc Author:
00085             Willem van der Schans, Trelent AI
00086         """
00087         sg.theme('Default1')
00088
00089         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
00090                 sg.Push()]
00091
00092         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
00093                 sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]
00094
00095         __Line3 = [
00096             sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
00097                             key='--progress_bar--')]
00098
00099
00100         layout = [__Line1, __Line2, __Line3]
00101
00102         self.__layout = layout
00103
```

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:



**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](#).

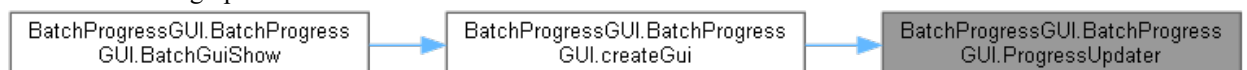
```

00188     def ProgressUpdater(self, valueObj):
00189         """
00190         The ProgressUpdater function is a callback function that updates the progress
00191         bar and text
00192         in the GUI. It takes in one argument, which is an object containing information
00193         about the
00194         current batch number. The ProgressUpdater function then checks if this value
00195         has changed from
00196         the last time it was called (i.e., if we are on a new batch). If so, it updates
00197         both the progress
00198         bar and text with this new information.
00199
00200     Args:
00201         self: Make the progressupdater function an instance method
00202         valueObj: Get the current value of the batch counter
00203
00204     Returns:
00205         The value of the batch counter
00206
00207     Doc Author:
00208         Willem van der Schans, Trelent AI
00209     """
00210     if valueObj.getValue() != self.__batch_counter:
00211         self.__batch_counter = valueObj.getValue()
00212         __update_text = f"Batch {self.__batch_counter}/{self.__batches}
00213         completed"
00214         self.__window.write_event_value('update--progress_bar--',
00215         self.__batch_counter)
00216         self.__window.write_event_value('update--progress_text--',
00217         update_text)
00218     else:
00219         pass
00220 
```

References [BatchProgressGUI.BatchProgressGUI.\\_\\_batch\\_counter](#), and [BatchProgressGUI.BatchProgressGUI.\\_\\_window](#).

Referenced by [BatchProgressGUI.BatchProgressGUI.createGui\(\)](#).

Here is the caller graph for this function:



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



```

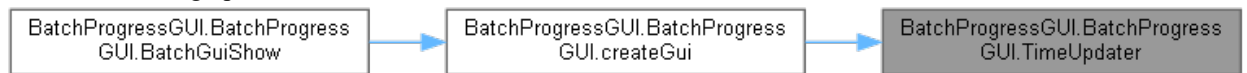
00257         seconds=(__passed_time * self.__batches -
00258         __passed_time)).seconds
00259         __time_est = time.strftime('%H:%M:%S',
00260         time.gmtime(__time_est))
00261         __end_string = f"Est time needed {__time_est}"
00262         self.__window.write_event_value('update--time_est--',
00263         __end_string)
00264         else:
00265         __end_string = f"Est time needed 00:00:00"
00266         self.__window.write_event_value('update--time_est--',
00267         __end_string)
00268         time.sleep(0.25)

```

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:



**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](#).

```

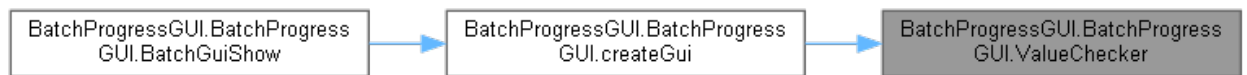
00268     def ValueChecker(self, ObjectVal):
00269         """
00270         The ValueChecker function is a thread that checks the value of an object.
00271         It will check if the value has changed, and if it has, it will return
00272         True.
00273         If not, then it returns False.
00274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

```

References [BatchProgressGUI.BatchProgressGUI.\\_\\_batch\\_counter](#).

Referenced by [BatchProgressGUI.BatchProgressGUI.createGui\(\)](#).

Here is the caller graph for this function:



## 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\\_arg](#)[year\\_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](#).

```
00016     def __init__(self, state_arg=None, year_arg=None):
00017         """
00018         The __init__ function is called when the class is instantiated.
00019         Its job is to initialize the object with some default values, and do any other
00020         setup that might be necessary.
00021         The __init__ function can take arguments, but it doesn't have to.
00022
00023         Args:
00024             self: Represent the instance of the class
00025             state_arg: Set the state_arg attribute of the class
00026             year_arg: Set the year of data to be retrieved
00027
00028         Returns:
00029             A popupwrapped object
00030
00031         Doc Author:
00032             Willem van der Schans, Trelent AI
00033         """
00034         self.state_arg = state_arg
00035         self.year_arg = year_arg
```

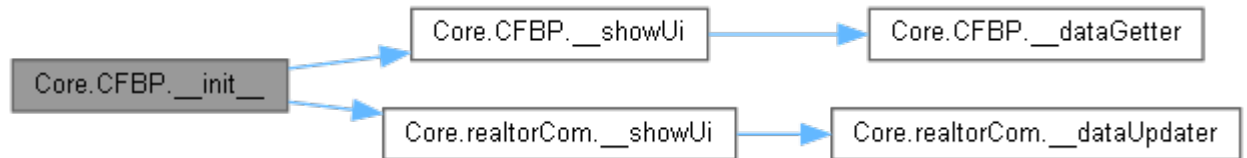
```

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("cfpb", 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.cfbp.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

```

```

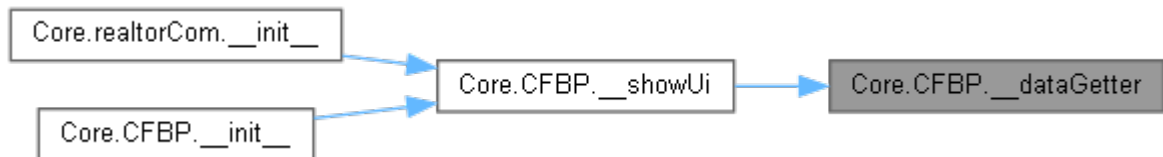
00095
00096     Doc Author:
00097         Willem van der Schans, Trelent AI
00098     """
00099     arg_dict_bu = locals()
00100
00101     link = "https://ffiec.cfbp.gov/v2/data-browser-api/view/csv?"
00102
00103     if self.state_arg is None:
00104         self.state_arg = "UT"
00105     else:
00106         pass
00107
00108     if self.year_arg is None:
00109         self.year_arg = str(date.today().year - 1)
00110     else:
00111         pass
00112
00113     passFlag = False
00114
00115     while not passFlag:
00116
00117         self.link =
00118         "https://ffiec.cfbp.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
00119         + f"&years={self.year_arg}"
00120
00121         response = requests.get(self.link)
00122
00123         if response.status_code == 400:
00124             self.year_arg = int(self.year_arg) - 1
00125         else:
00126             passFlag = True
00127
00128         RESTError(response)
00129         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](#).

```

00057     def __showUi(self):
00058
00059     """
00060     The __showUi function is a function that creates a progress bar window.
00061     The __showUi function takes class variables and returns a windowobj.
00062
00063
00064     Args:

```

```

00065         self: Represent the instance of the class
00066
00067     Returns:
00068         The uiobj variable
00069
00070     Doc Author:
00071         Willem van der Schans, Trelent AI
00072     """
00073     uiObj = PopupWrapped(text="Cenus Request running",
00074 windowType="progress", error=None)
00075
00076     threadGui = threading.Thread(target=self.__dataGetter,
00077                                 daemon=False)
00078     threadGui.start()
00079
00080     while threadGui.is alive():
00081         uiObj.textUpdate()
00082         uiObj.windowPush()
00083     else:
00084         uiObj.stopWindow()

```

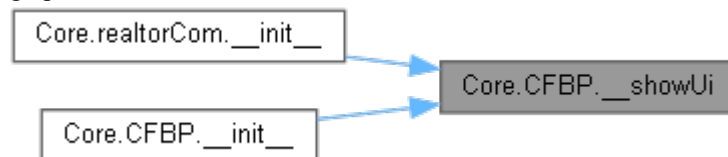
References [Core.CFBP.\\_\\_dataGetter\(\)](#).

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](#).

```
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the variables that will be used by other functions in this class.
00031
00032
00033         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             None
00038
00039         Doc Author:
```

```

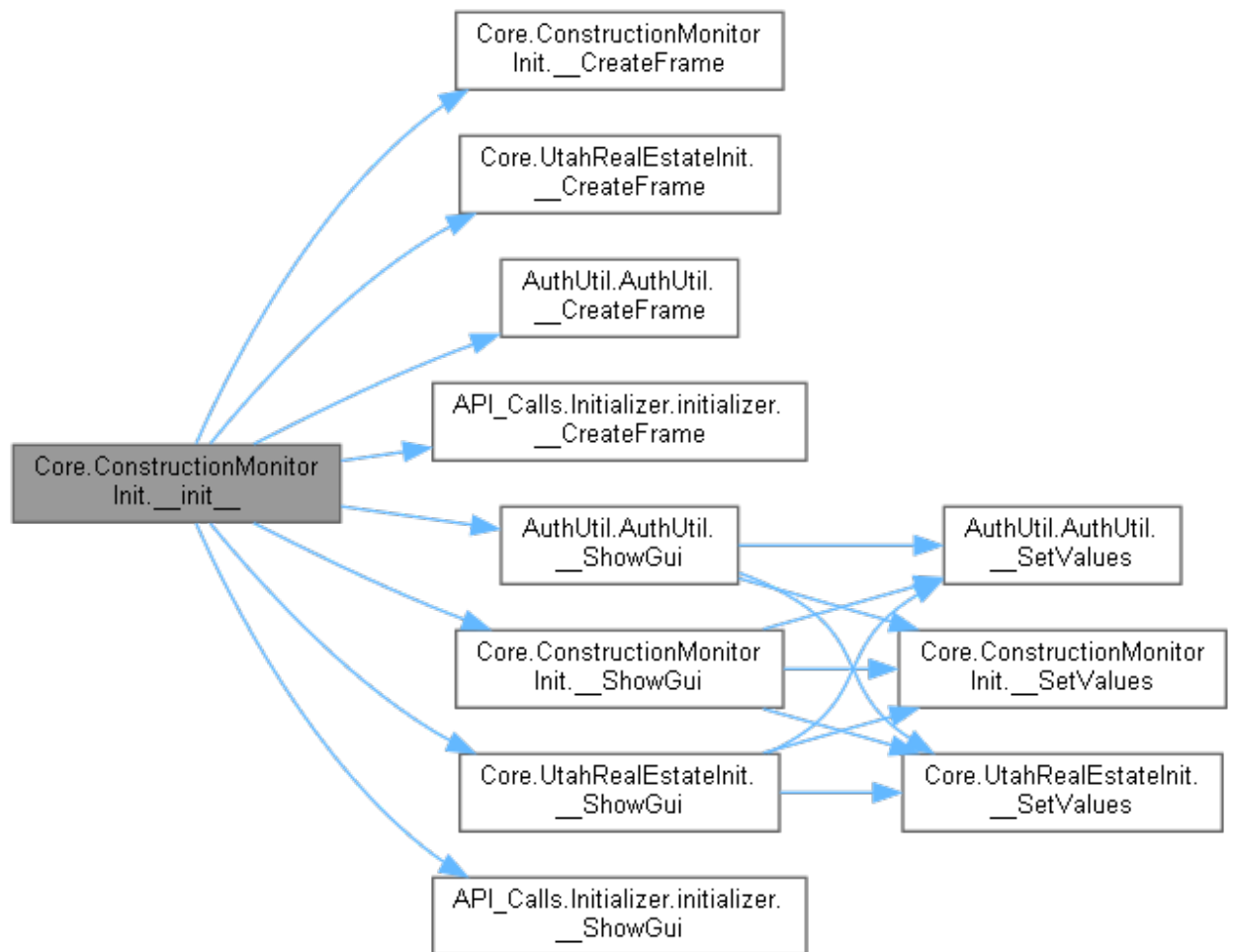
00040 Willem van der Schans, Trelent AI
00041 """
00042     self.size = None
00043     self.SourceInclude = None
00044     self.dateStart = None
00045     self.dateEnd = None
00046     self.rest_domain = None
00047     self.auth_key = None
00048     self.ui_flag = None
00049     self.append_file = None
00050
00051     passFlag = False
00052
00053     while not passFlag:
00054         if
os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpat
h(
00055             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
00056
Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00057     "Security").joinpath("auth.json")):
00058         try:
00059             f =
open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00060             "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00061             key = f.readline()
00062             f.close()
00063             f =
open(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00064             "Security").joinpath("auth.json"), "rb")
00065             authDict = json.load(f)
00066             fernet = Fernet(key)
00067             self.auth_key =
fernet.decrypt(authDict["cm"]["auth"]).decode()
00068             passFlag = True
00069         except Exception as e:
00070             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")
00071             AuthUtil()
00072         else:
00073             AuthUtil()
00074
00075     self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00076

```

#### References

<a href="#">Core.UtahRealEstateInit. CreateFrame()</a> ,	<a href="#">Core.ConstructionMonitorInit. CreateFrame()</a> ,
<a href="#">API Calls.Initializer.initializer. CreateFrame()</a> ,	<a href="#">AuthUtil.AuthUtil. CreateFrame()</a> ,
<a href="#">API Calls.Initializer.initializer. ShowGui()</a> ,	<a href="#">AuthUtil.AuthUtil. ShowGui()</a> ,
<a href="#">Core.UtahRealEstateInit. ShowGui()</a> ,	<a href="#">Core.ConstructionMonitorInit. ShowGui()</a> ,
<a href="#">Core.ConstructionMonitorInit.append_file,</a>	<a href="#">AuthUtil.AuthUtil.append_file,</a>
<a href="#">Core.UtahRealEstateInit.append_file,</a>	<a href="#">Core.UtahRealEstateInit.append_file,</a>
<a href="#">Core.ConstructionMonitorInit.auth_key,</a>	<a href="#">Core.ConstructionMonitorInit.dateEnd,</a>
<a href="#">Core.UtahRealEstateInit.dateEnd,</a>	<a href="#">Core.ConstructionMonitorInit.dateStart,</a>
<a href="#">Core.UtahRealEstateInit.dateStart,</a>	<a href="#">Core.ConstructionMonitorInit.rest_domain,</a>
<a href="#">Core.ConstructionMonitorInit.size,</a>	<a href="#">Core.ConstructionMonitorInit.SourceInclude,</a>
<a href="#">Core.ConstructionMonitorInit.ui_flag,</a>	and

Here is the call graph for this function:



## 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
00121         be displayed in the GUI window.
00122         This is done by creating each line as a list and then appending it to
00123         another list which will contain all lines.
00124
00125         Args:
  
```

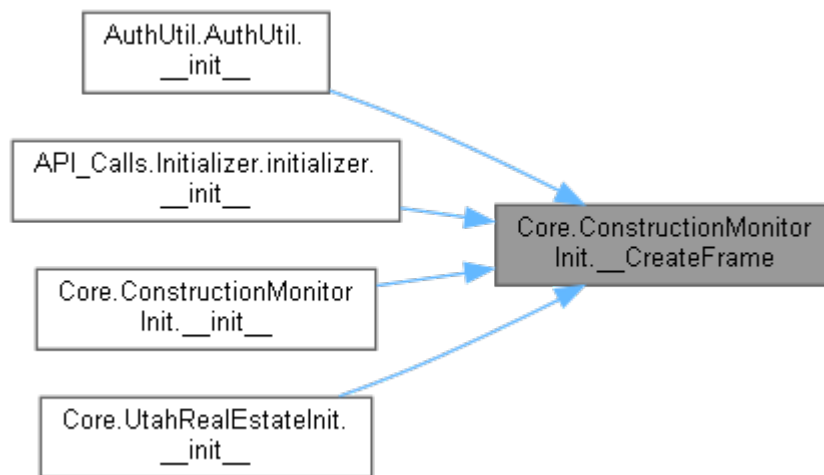
```

00124
00125     Returns:
00126         The layout for the gui
00127
00128     Doc Author:
00129         Willem van der Schans, Trelent AI
00130     """
00131     sg.theme('Default1')
00132
00133     line00 = [sg.HSeparator()]
00134
00135     line0 = [sg.Image(ImageLoader("logo.png")),
00136              sg.Push(),
00137              sg.Text("Construction Monitor Utility", font=("Helvetica",
00138 12, "bold"), justification="center"),
00138              sg.Push(),
00139              sg.Push()]
00140
00141     line1 = [sg.HSeparator()]
00142
00143     line3 = [sg.Text("Start Date : ", size=(15, None),
00144 justification="Right"),
00144              sg.Input(default_text=(date.today() -
00145 timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
00146 size=(20, 1)),
00147              sg.CalendarButton("Select Date", format="%Y-%m-%d",
00148 key="-start_date-", target="-Cal-")]
00149
00150     line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00151              sg.Input(default_text=date.today().strftime("%Y-%m-%d"),
00152 key="-EndCal-",
00153 size=(20, 1)),
00154              sg.CalendarButton("Select Date", format="%Y-%m-%d",
00155 key="-start_date-", target="-EndCal-")]
00156
00157     line5 = [sg.HSeparator()]
00158
00159     line6 = [sg.Push(),
00160              sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00161 justification="center"),
00162              sg.Push()]
00163
00164     line7 = [sg.HSeparator()]
00165
00166     line8 = [sg.Text("Appending File : ", size=(15, None),
00167 justification="Right"),
00168              sg.Input(default_text="", key="-AppendingFile-",
00169 disabled=True,
00170 size=(20, 1)),
00171              sg.FileBrowse("Browse File", file_types=[("csv files",
00172 "*.csv")], key="-append_file-",
00173 target="-AppendingFile-")]
00174
00175     line9 = [sg.HSeparator()]
00176
00177     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00178
00179     layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8,
00180 line9, line10]
00181
00182     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
00179         used in the __GetData function.
00180         The __SetValues function takes a dictionary as an argument, and then sets
00181         each variable based on what is passed into
00182         the dictionary. The keys for this dictionary are defined by the user when
00183         they create their own instance of this class.
00184
00185         Args:
00186             self: Represent the instance of the class
00187             values: Pass in the values from the ui
00188
00189         Returns:
00190             A dictionary of values
00191
00192         Doc Author:
00193             Willem van der Schans, Trelent AI
00194         """
00195         self.size = 1000
00196
00197         if values["-Cal-"] != "":
00198             self.dateStart = values["-Cal-"]
00199         else:
00200             self.dateStart = (date.today() -
00201                             timedelta(days=14)).strftime("%Y-%m-%d")
00202
00203         if values["-EndCal-"] != "":
00204             self.dateEnd = values["-EndCal-"]
00205         else:
00206             self.dateEnd = date.today().strftime("%Y-%m-%d")
  
```

```

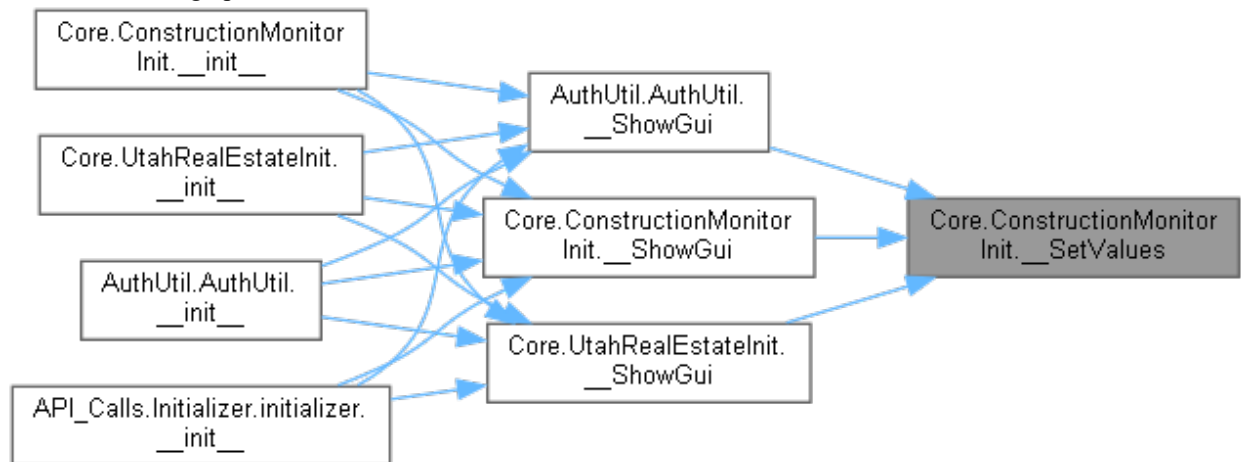
00203
00204     self.rest_domain =
00205     "https://api.constructionmonitor.com/v2/powersearch/"
00206
00207     self.SourceInclude = None
00208
00209     if values["-append_file-"] != "":
00210         self.append_file = str(values["-append_file-"])
00211     else:
00212         self.append_file = None
00213
00214     self.ui_flag = True
00215

```

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:



```
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](#).

```

00077     def __ShowGui(self, layout, text):
00078
00079         """
00080         The __ShowGui function is the main function that creates and displays the
00081         GUI.
00082         It takes in a layout, which is a list of lists containing all the elements
00083         to be displayed on screen.

```

```

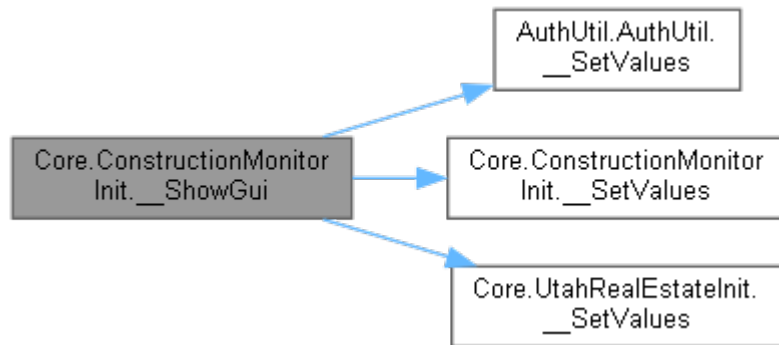
00082     The text parameter specifies what title should appear at the top of the window.
00083
00084     Args:
00085         self: Refer to the current instance of a class
00086         layout: Determine what the gui will look like
00087         text: Set the title of the window
00088
00089     Returns:
00090         A dictionary of values
00091
00092     Doc Author:
00093         Willem van der Schans, Trelent AI
00094     """
00095     window = sg.Window(text, layout, grab_anywhere=False,
00096 return_keyboard_events=True,
00097                             finalize=True,
00098                             icon=ImageLoader("taskbar_icon.ico"))
00099
00100     while True:
00101         event, values = window.read()
00102
00103         if event == "Submit":
00104             try:
00105                 self.__SetValues(values)
00106                 break
00107             except Exception as e:
00108                 print(e)
00109                 RESTError(993)
00110                 raise SystemExit(933)
00111         elif event == sg.WIN_CLOSED or event == "Quit":
00112             break
00113
00114     window.close()

```

References [AuthUtil.AuthUtil.\\_\\_SetValues\(\)](#), [Core.ConstructionMonitorInit.\\_\\_SetValues\(\)](#), and [Core.UtahRealEstateInit.\\_\\_SetValues\(\)](#).

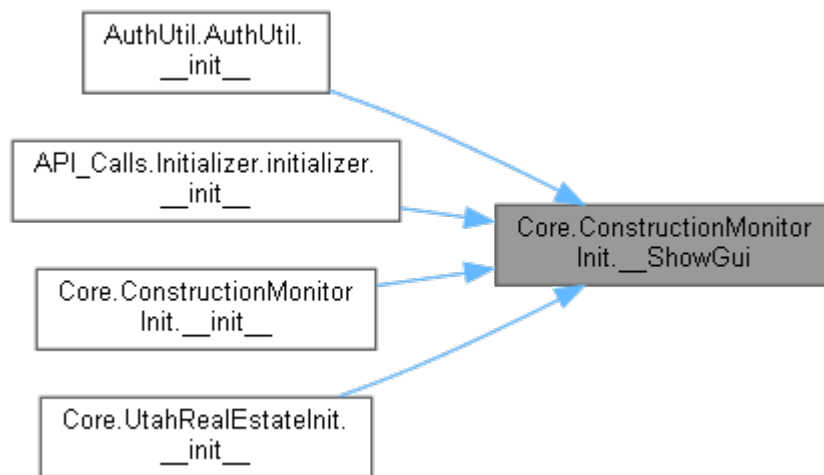
Referenced by [AuthUtil.AuthUtil.\\_\\_init\\_\\_\(\)](#), [API Calls.Initializer.initializer.\\_\\_init\\_\\_\(\)](#), [Core.ConstructionMonitorInit.\\_\\_init\\_\\_\(\)](#), and [Core.UtahRealEstateInit.\\_\\_init\\_\\_\(\)](#).

Here is the call graph for this function:



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\\_\\_\(\)](#), [Core.ConstructionMonitorInit.\\_\\_SetValues\(\)](#), and

### 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

### [dataframe](#)Private 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](#).

```
00218     def __init__(self, siteClass):
00219
00220         """
00221         The __init__ function is the first function that runs when an object of this
00222         class is created.
00223         It sets up all the variables and functions needed for this class to run
00224         properly.
```

```

00223
00224
00225     Args:
00226         self: Represent the instance of the class
00227         siteClass: Identify the site that is being used
00228
00229     Returns:
00230         Nothing
00231
00232     Doc Author:
00233         Willem van der Schans, Trelent AI
00234     """
00235     self.__siteClass = siteClass
00236     self.__restDomain = None
00237     self.__headerDict = None
00238     self.__columnSelection = None
00239     self.__appendFile = None
00240
00241     self.__parameterDict = {}
00242     self.__search_id = None
00243     self.__record_val = 0
00244     self.__batches = 0
00245
00246     self.__ui_flag = None
00247
00248     self.dataframe = None
00249
00250     try:
00251         self.mainFunc()
00252     except SystemError as e:
00253         if "Status Code = 1000 | Catastrophic Error" in str(getattr(e,
00254         'message', repr(e))):
00255             print(
00256                 f"ConstructionMonitor/Core.py | Error = {e} | Coerced
00257         SystemError in ConstructionMonitorMain class")
00258             pass
00259         except AttributeError as e:
00260             # This allows for user cancellation of the program using the quit
00261         button
00262             if "'NoneType' object has no attribute 'json'" in str(getattr(e,
00263         'message', repr(e))):
00264                 RESTError(1101)
00265                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00266         %H:%M:%S.%f')}[:-3]} | Error {e}")
00267                 pass
00268             elif e is not None:
00269                 print(
00270                     f"ConstructionMonitor/Core.py | Error = {e} |
00271         Authentication Error | Please update keys in AuthUtil")
00272                 RESTError(401)
00273                 print(e)
00274                 pass
00275             else:
00276                 pass
00277         except Exception as e:
00278             print(e)
00279             RESTError(1001)
00280             raise SystemExit(1001)
00281

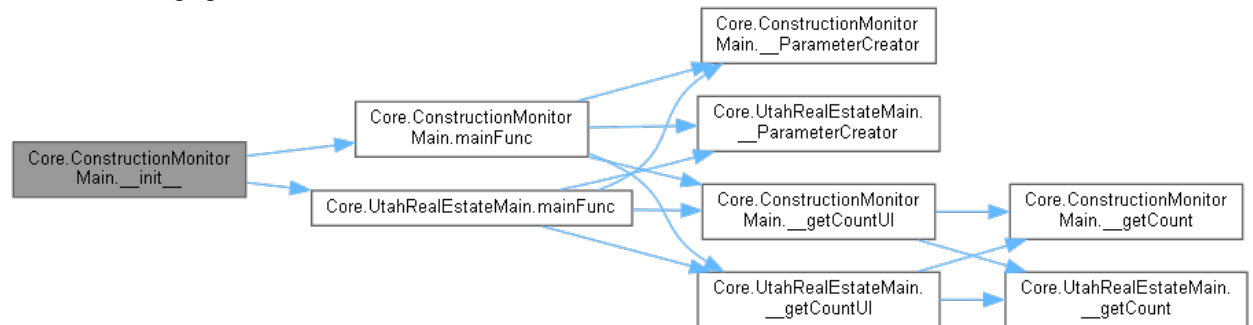
```

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:



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

```

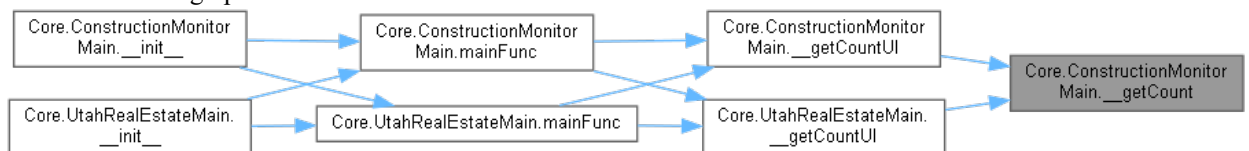
00390
00391         __count_resp = requests.post(url=self.__restDomain,
00392                                     headers=self.__headerDict,
00393                                     json=__temp_param_dict)
00394
00395     except requests.exceptions.Timeout as e:
00396         print(e)
00397         RESTError(790)
00398         raise SystemExit(790)
00399     except requests.exceptions.TooManyRedirects as e:
00400         print(e)
00401         RESTError(791)
00402         raise SystemExit(791)
00403     except requests.exceptions.MissingSchema as e:
00404         print(e)
00405         RESTError(1101)
00406     except requests.exceptions.RequestException as e:
00407         print(e)
00408         RESTError(405)
00409         raise SystemExit(405)
00410
00411         count_resp = count_resp.json()
00412
00413         self.__record_val = __count_resp["hits"]["total"]["value"]
00414
00415         del __count_resp, __temp_param_dict
00416

```

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:



**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](#).

```

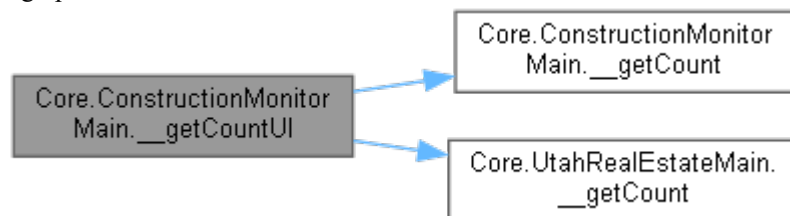
00417     def __getCountUI(self):
00418         """
00419         The __getCountUI function is a wrapper for the __getCount function.
00420         It allows the user to run __getCount in a separate thread, so that they can
00421         continue working while it runs.
00422         The function will display a progress bar and update with text as it progresses
00423         through its tasks.
00424         Args:
00425             self: Access the class variables and methods
00426         Returns:
00427             The count of the number of records in the database
00428         Doc Author:
00429             Willem van der Schans, Trelent AI
00430         """
00431         if self.__ui_flag:
00432             uiObj = PopupWrapped(text="Batch request running",
00433                                 windowType="progress", error=None)
00434             threadGui = threading.Thread(target=self.__getCount,
00435                                         daemon=False)
00436             threadGui.start()
00437             while threadGui.is_alive():
00438                 uiObj.textUpdate()
00439                 uiObj.windowPush()
00440             else:
00441                 uiObj.stopWindow()
00442         else:
00443             self.__getCount()

```

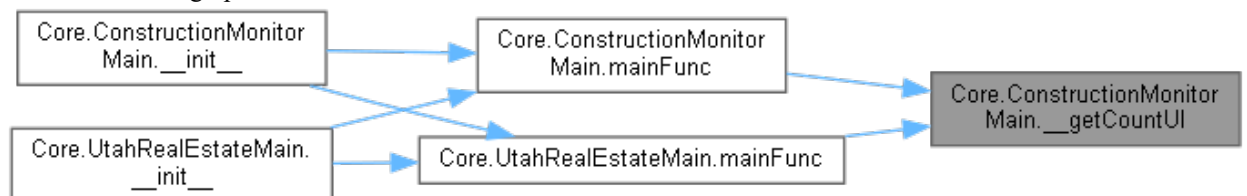
References [Core.ConstructionMonitorMain.\\_\\_getCount\(\)](#), [Core.UtahRealEstateMain.\\_\\_getCount\(\)](#), and [Core.ConstructionMonitorMain.\\_\\_ui\\_flag](#).

Referenced by [Core.ConstructionMonitorMain.mainFunc\(\)](#), and [Core.UtahRealEstateMain.mainFunc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**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](#).

```

00332     def __ParameterCreator(self):
00333         """
00334         The ParameterCreator function is used to create the parameter dictionary
00335         that will be passed into the
00336         __Request function. The function takes in a siteClass object and extracts
00337         all of its attributes, except for
00338         those that start with '__' or are callable. It then creates a dictionary
00339         from these attributes and stores it as
00340         self.__parameterDict.
00341
00342         Args:
00343         self: Make the function a method of the class
00344
00345         Returns:
00346         A dictionary of parameters and a list of non parameter variables
00347
00348         Doc Author:
00349         Willem van der Schans, Trelent AI
00350         """
00351         __Source_dict = {key: value for key, value in
00352         self.__siteClass.__dict__.items() if
00353         not key.startswith('__') and not callable(key)}
00354
00355         self.__restDomain = __Source_dict["rest_domain"]
00356         __Source_dict.pop("rest_domain")
00357         self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00358         __Source_dict.pop("auth_key")
00359         self.__columnSelection = __Source_dict["SourceInclude"]
00360         __Source_dict.pop("SourceInclude")
00361         self.__ui_flag = __Source_dict["ui_flag"]
00362         __Source_dict.pop("ui_flag")
00363         self.__appendFile = __Source_dict["append_file"]
00364         __Source_dict.pop("append_file")
00365
00366         temp_dict = copy.copy(__Source_dict)
00367         for key, value in temp_dict.items():
00368             if value is None:
00369                 __Source_dict.pop(key)
00370             else:
00371                 pass
00372
00373         self.__parameterDict = copy.copy(__Source_dict)

```

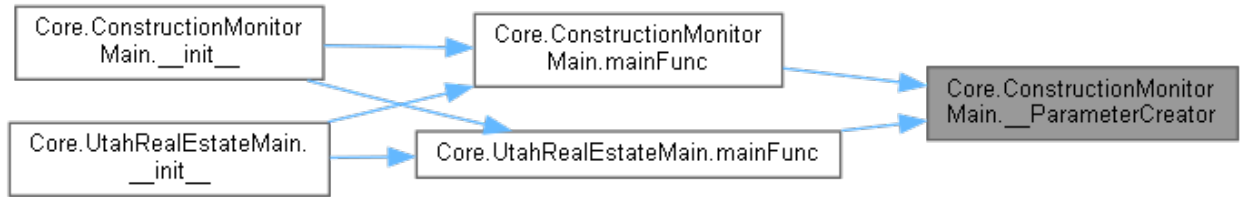
References

[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\(\)](#), [Core.UtahRealEstateMain.mainFunc\(\)](#), and

Here is the caller graph for this function:



**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](#).

```

00276     def mainFunc(self):
00277         """
00278         The mainFunc function is the main function of this module. It will be called
00279         by the GUI or CLI to execute
00280         the code in this module. The mainFunc function will first create a parameter
00281         dictionary using the __ParameterCreator
00282         method, then it will get a count of all records that match its parameters
00283         using the __getCountUI method, and then
00284         it will calculate how many batches are needed to retrieve all records with
00285         those parameters using BatchCalculator.
00286         After that it asks if you want to continue with retrieving data from Salesforce
00287         (if running in GUI mode). Then it shows
00288         a progress bar for each
00289
00290         Args:
00291         self: Refer to the current object
00292
00293         Returns:
00294         The dataframe
00295
00296         Doc Author:
00297         Willem van der Schans, Trelent AI
00298         """
00299         self.__ParameterCreator()
00300
00301         print(
00302             f"{datetime.datetime.today().strftime('%m-%d-%Y
00303             %H:%M:%S.%f')}[:-3]} | Param Dict = {self.__parameterDict}")
00304         print(
00305             f"{datetime.datetime.today().strftime('%m-%d-%Y
00306             %H:%M:%S.%f')}[:-3]} | Rest Domain = {self.__restDomain}")
00307
00308         self.__getCountUI()
00309
  
```

```

00303         self.__batches = BatchCalculator(self.__record_val,
00304 self.__parameterDict)
00305         print(
00306             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")
00307
00308         if self.__batches != 0:
00309             startTime = datetime.datetime.now().replace(microsecond=0)
00310             eventReturn = BatchInputGui(self.__batches, self.__record_val)
00311             if eventReturn == "Continue":
00312                 print(
00313                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} batches sent to server")
00314                 BatchGuiObject =
BatchProgressGUI(RestDomain=self.__restDomain,
00315 ParameterDict=self.__parameterDict,
00316 HeaderDict=self.__headerDict,
00317 ColumnSelection=self.__columnSelection,
00318 BatchesNum=self.__batches,
00319 Type="construction_monitor")
00320                 BatchGuiObject.BatchGuiShow()
00321                 self.dataframe = BatchGuiObject.dataframe
00322                 print(
00323                     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()))}")
00324                 FileSaver("cm", self.dataframe, self.__appendFile)
00325             else:
00326                 print(
00327                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} batches canceled by user")
00328             else:
00329                 RESTError(994)
00330                 raise SystemExit(994)
00331

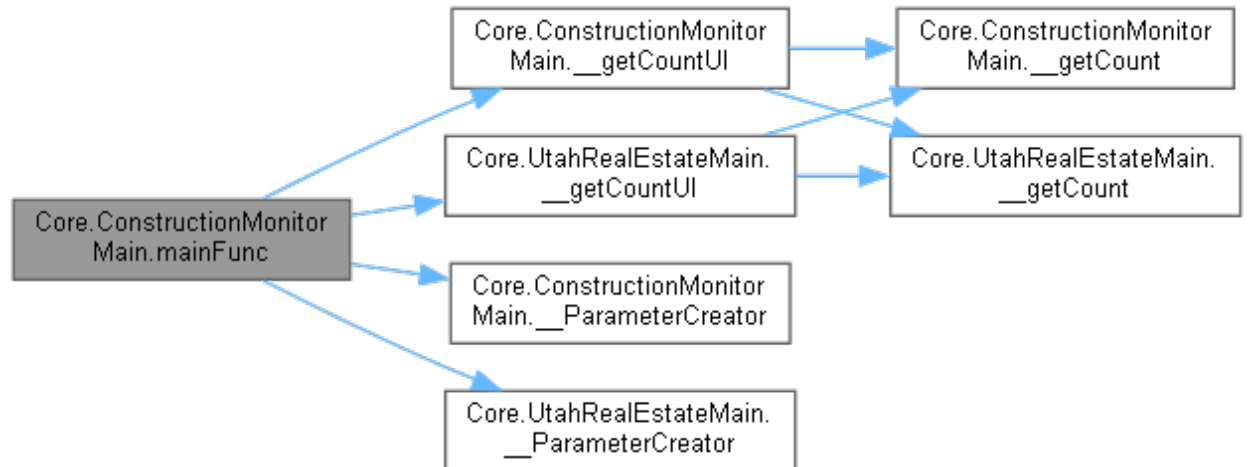
```

References

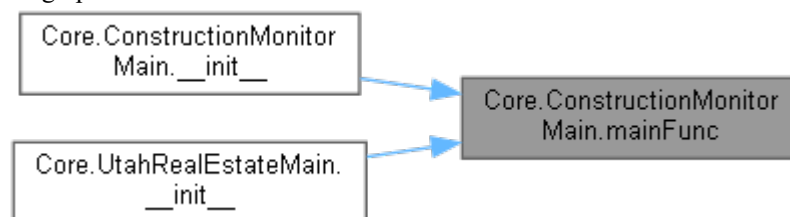
[Core.UtahRealEstateMain.\\_\\_appendFile](#), [Core.ConstructionMonitorMain.\\_\\_appendFile](#),  
[Core.ConstructionMonitorMain.\\_\\_batches](#), [BatchProgressGUI.BatchProgressGUI.\\_\\_batches](#),  
[BatchProcessing.BatchProcessorConstructionMonitor.\\_\\_columnSelection](#), [Core.UtahRealEstateMain.\\_\\_batches](#),  
[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:



Here is the caller graph for this function:



## 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.BatchProcessorConstructionMonitor.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](#).

```
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
```

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](#).

```

00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054

```

References [DataTransfer.DataTransfer.\\_\\_value](#).

Referenced by [DataTransfer.DataTransfer.whileValue\(\)](#).

Here is the caller graph for this function:



**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](#).

```

00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026         Args:
00027             self: Represent the instance of the class
00028             value: Set the value of the instance variable __value
00029
00030         Returns:
00031             The value that was passed to it
00032
00033         Doc Author:
00034             Willem van der Schans, Trelent AI
00035         """
00036         self.__value = value
00037
00038

```

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, Trement AI
```

Definition at line [55](#) of file [DataTransfer.py](#).

```
00055     def whileValue(self):
00056         """
00057         The whileValue function is a function that will run the getValue function
00058         until it is told to stop.
00059         This allows for the program to constantly be checking for new values from
00060         the sensor.
00061         Args:
00062         self: Refer to the current instance of the class
00063         Returns:
00064         The value of the input
00065         Doc Author:
00066         Willem van der Schans, Trement AI
00067         """
00068         while True:
00069             self.getValue()
00070
```

References [DataTransfer.DataTransfer.getValue\(\)](#).

Here is the call graph for this function:



---

## 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](#).

```
00015     def __init__(self, method, outputDF, AppendingPath=None):
00016         """
00017         The __init__ function is called when the class is instantiated.
00018         It sets up the instance of the class, and defines all variables that will
00019         be used by other functions in this class.
00020         The __init__ function takes two arguments: self and method. The first
00021         argument, self, refers to an instance of a
00022         class (in this case it's an instance of DataFrameSaver). The second argument,
00023         method refers to a string value that
00024         is passed into DataFrameSaver when it's instantiated.
00025
00026     Args:
```

```

00024         self: Represent the instance of the class
00025         method: Determine which dataframe to append the new data to
00026         outputDF: Pass in the dataframe that will be saved to a csv file
00027         AppendingPath: Specify the path to an existing csv file that you want
to append your dataframe to
00028
00029         Returns:
00030             Nothing
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034         """
00035         self.docPath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00036             datetime.datetime.today().strftime('%m%d%Y'))
00037         self.data = outputDF
00038         self.dataAppending = None
00039         self.appendFlag = True
00040         self.fileName =
f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00041         self.uiFlag = True
00042
00043         if method.lower() == "ure":
00044             self.primaryKey = "ListingKeyNumeric"
00045         elif method.lower() == "cm":
00046             self.primaryKey = "id"
00047         elif "realtor" in method.lower():
00048             self.primaryKey = None
00049             self.uiFlag = False
00050         elif method.lower() == "cfbp":
00051             self.primaryKey = None
00052             self.uiFlag = False
00053         else:
00054             raise ValueError("method input is invalid choice one of 4 options:
URE, CM, Realtor, CFBP")
00055
00056         if AppendingPath is None:
00057             self.appendFlag = False
00058         else:
00059             self.dataAppending = pd.read_csv(AppendingPath)
00060
00061         if self.appendFlag:
00062             if self.primaryKey is not None:
00063                 # Due to low_memory loading the columns are not typed properly,
00064                 # since we are comparing this will be an issue since we need to
do type comparisons,
00065                 # so here we coerce the types of the primary keys to numeric.
00066                 # If another primary key is ever chosen make sure to core to the
right data type.
00067                 self.dataAppending[self.primaryKey] =
pd.to_numeric(self.dataAppending[self.primaryKey])
00068                 self.data[self.primaryKey] =
pd.to_numeric(self.data[self.primaryKey])
00069
00070                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00071 keep="last")
00072             else:
00073                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(keep="last")
00074             else:
00075                 self.outputFrame = self.data
00076
00077             if os.path.exists(self.docPath):
00078                 self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00079             else:
00080                 os.mkdir(self.docPath)
00081                 self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00082
00083             if self.uiFlag:
00084                 if self.appendFlag:
00085                     PopupWrapped(text=f"File Appended and Saved to
{self.docPath.joinpath(self.fileName)}",
00086 windowType="savedLarge")

```

```

00087
00088         # Logging
00089         print(
00090             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")
00091         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]}")
00092     else:
00093         PopupWrapped(text=f"File Saved to
{self.docPath.joinpath(self.fileName)}", windowType="savedLarge")
00094
00095         # Logging
00096         print(
00097             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")
00098     else:
00099         pass
00100

```

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](#).

```

00101     def getPath(self):
00102         """
00103         The getPath function returns the path to the file.
00104         It is a string, and it joins the docPath with the fileName.
00105
00106         Args:
00107             self: Represent the instance of the class
00108
00109         Returns:
00110             The path to the file
00111
00112         Doc Author:
00113             Willem van der Schans, Trelent AI
00114         """
00115         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

### [classObj](#)Private 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](#).

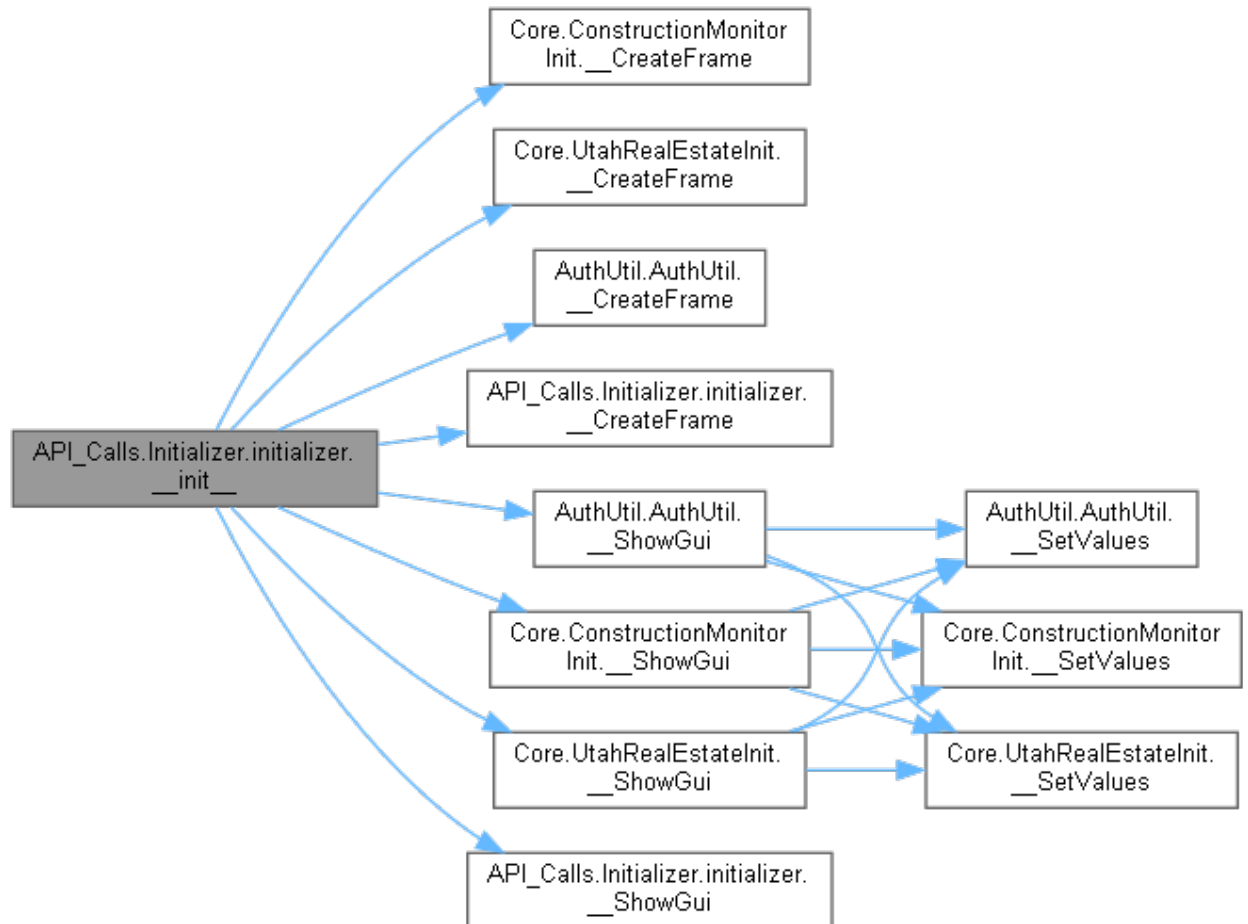
```
00024     def __init__(self):
00025
00026         """
00027         The __init__ function is called when the class is instantiated.
00028         It sets up the logging, calls the __ShowGui function to create and display
00029         the GUI, and then calls __CreateFrame to create a frame for displaying
00030         widgets.
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\(\)](#),  
[API\\_Calls.Initializer.initializer.\\_\\_CreateFrame\(\)](#),  
[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:



## 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
```

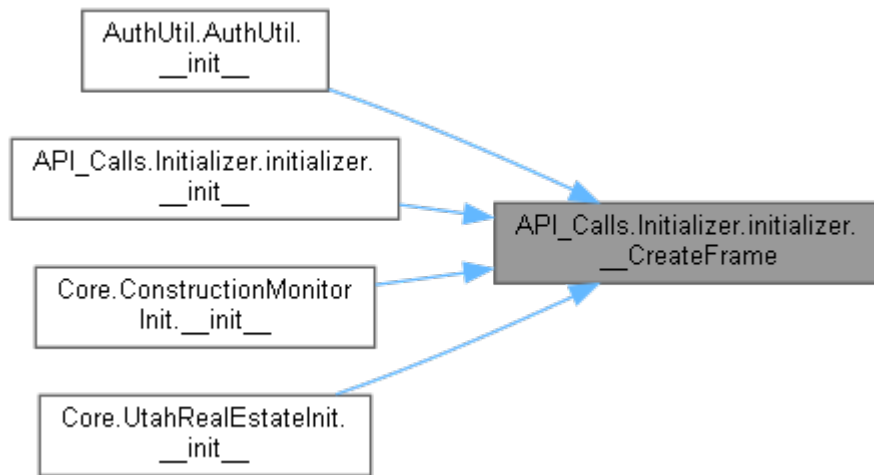
```

00137     """
00138     The __CreateFrame function is a helper function that creates the layout for
the main window.
00139     It returns a list of lists, which is then passed to sg.Window() as its layout
parameter.
00140
00141     Args:
00142         self: Represent the instance of the class
00143
00144     Returns:
00145         A list of lists, which is then passed to the sg
00146
00147     Doc Author:
00148         Willem van der Schans, Trelent AI
00149     """
00150     sg.theme('Default1')
00151
00152     line0 = [sg.HSeparator()]
00153
00154     line1 = [sg.Image(ImageLoader("logo.png")),
00155             sg.Push(),
00156             sg.Text("Gardner Data Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00157             sg.Push(),
00158             sg.Push()]
00159
00160     line3 = [sg.HSeparator()]
00161
00162     line4 = [sg.Push(),
00163             sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
justification="center"),
00164             sg.Push()]
00165
00166     line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20,
None)), sg.Push(),
00167             sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00168
00169     line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)),
sg.Push(),
00170             sg.Button("CFPB Mortgage", size=(20, None)),
00171             sg.Push()]]
00172
00173     line8 = [sg.HSeparator()]
00174
00175     line9 = [sg.Push(),
00176             sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
00177             sg.Push()]
00178
00179     line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20,
None)),
00180             sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00181
00182     line11 = [sg.HSeparator()]
00183
00184     layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]
00185
00186     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 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., `"Text"`, `"InputText"`, 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
00056         containing a tuple with three elements:
00057             1) The type of element to be displayed (e.g., "Text",
00058             "InputText", etc.)
00059             2) A dictionary containing any additional parameters for that element
00060             (e.g., size, default value, etc.)
00061             3) An optional key name for the element (used in event handling). If no
00062             key name is provided then one will be generated automatically by PySimpleGUIQt based
00063             on its position in the layout list
00064
00065         Args:
00066             self: Represent the instance of the class
00067             layout: Pass the layout of the window to be created
00068             text: Set the title of the window
00069
00070         Returns:
00071             A window object
00072
00073         Doc Author:
00074             Willem van der Schans, Trelent AI
00075         """
00076         versionChecker()
00077
00078
  
```



```

00073         window = sg.Window(text, layout, grab_anywhere=False,
00074         return_keyboard_events=True,
00075         finalize=True,
00076         icon=ImageLoader("taskbar_icon.ico"))
00077     while True:
00078         event, values = window.read()
00079
00080         if event == "Construction Monitor":
00081             print(
00082                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Construction Monitor API
Call-----")
00083             ConstructionMonitorMain(ConstructionMonitorInit())
00084             print(
00085                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Construction Monitor API
Call-----\n")
00086         elif event == "Utah Real Estate":
00087             print(
00088                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Utah Real Estate API
Call-----")
00089             UtahRealEstateMain(UtahRealEstateInit())
00090             print(
00091                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Utah Real Estate API
Call-----\n")
00092         elif event == "Realtor.Com":
00093             print(
00094                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Realtor.com API Call-----")
00095             realtorCom()
00096             print(
00097                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Realtor.com API
Call-----\n")
00098         elif event == "CFPB Mortgage":
00099             print(
00100                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating ffiec.cfpb API Call-----")
00101             CFBP()
00102             print(
00103                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing ffiec.cfpb API
Call-----\n")
00104         elif event == "Authorization Utility":
00105             print(
00106                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Authorization
Utility-----")
00107             AuthUtil()
00108             print(
00109                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Authorization
Utility-----\n")
00110         elif event == "Open Data Folder":
00111             print(
00112                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Data Folder Opened-----")
00113             try:
00114                 os.system(f"start
{Path(os.path.expanduser('~\\Documents')).joinpath('GardnerUtilData')}")
00115             except:
00116                 try:
00117                     os.system(f"start
{Path(os.path.expanduser('~\\Documents'))}")
00118                 except Exception as e:
00119                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
00120                     PopupWrapped(
00121                         text="Documents folder not found. Please create a
Windows recognized documents folder",
00122                         windowType="errorLarge")
00123
00124         elif event in ('Exit', None):

```

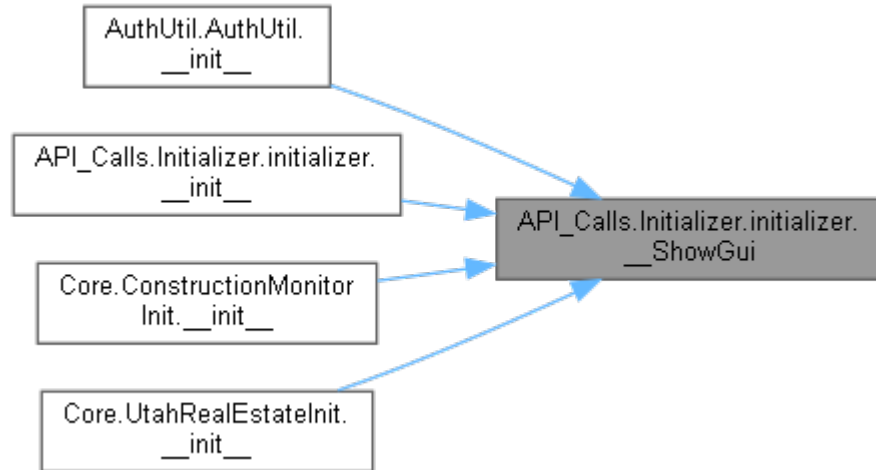
```

00125         try:
00126             break
00127         except Exception as e:
00128             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.")
00129             break
00130         elif event == sg.WIN_CLOSED or event == "Quit":
00131             break
00132
00133     window.close()
00134

```

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:




---

## 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
00019         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
```

```

00025     Returns:
00026         Nothing
00027     Doc Author:
00028         Willem van der Schans, Trelent AI
00029     """
00030     self.__text = text
00031     self.__type = windowType
00032     self.__error = error
00033     self.__layout = []
00034     self.__windowObj = None
00035     self.__thread = None
00036     self.__counter = 0
00037     self.__docpath = None
00038     self.__errorFlag = False
00039
00040     try:
00041         if "File Appended and Saved to " in self.__text:
00042             self.__docpath = str(self.__text[27:])
00043         elif "File Saved to " in self.__text:
00044             self.__docpath = str(self.__text[14:])
00045         else:
00046             pass
00047     except Exception as e:
00048         if self.__type == "savedLarge":
00049             print(
00050                 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")
00051             self.__errorFlag = True
00052         else:
00053             pass
00054
00055     self.__createWindow()
00056

```

References

<a href="#">PopupWrapped.PopupWrapped.__createWindow()</a> ,	<a href="#">PopupWrapped.PopupWrapped.__counter</a> ,
<a href="#">PopupWrapped.PopupWrapped.__error</a> ,	<a href="#">PopupWrapped.PopupWrapped.__docpath</a> ,
<a href="#">BatchProgressGUI.BatchProgressGUI.__layout</a> ,	<a href="#">PopupWrapped.PopupWrapped.__errorFlag</a> ,
<a href="#">PopupWrapped.PopupWrapped.__text</a> ,	<a href="#">PopupWrapped.PopupWrapped.__layout</a> ,
<a href="#">BatchProgressGUI.BatchProgressGUI.__type</a> ,	<a href="#">PopupWrapped.PopupWrapped.__thread</a> ,
<a href="#">PopupWrapped.PopupWrapped.__windowObj</a> .	<a href="#">PopupWrapped.PopupWrapped.__type</a> , and

Here is the call graph for this function:



## 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](#).

```

00057     def __createLayout(self):
00058         """

```

```

00059     The __createLayout function is used to create the layout of the window.
00060     The function takes class variables and returns a window layout.
00061     It uses a series of if statements to determine what type of window it is,
then creates a layout based on that information.
00062     Args:
00063         self: Refer to the current instance of a class
00064     Returns:
00065         A list of lists
00066     Doc Author:
00067         Willem van der Schans, Trelent AI
00068     """
00069     sg.theme('Default1')
00070     __Line1 = None
00071     __Line2 = None
00072
00073     if self.__type == "notice":
00074         __Line1 = [sg.Push(),
00075 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00076 sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00077         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00078     elif self.__type == "noticeLarge":
00079         __Line1 = [sg.Push(),
00080 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00081 sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00082         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00083     elif self.__type == "savedLarge":
00084         if self.__errorFlag:
00085             __Line1 = [sg.Push(),
00086 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00087 sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00088             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)),
sg.Push()]
00089         else:
00090             __Line1 = [sg.Push(),
00091 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00092 sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00093             __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)),
sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00094     elif self.__type == "errorLarge":
00095         __Line1 = [sg.Push(),
00096 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00097 sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00098         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00099     elif self.__type == "FatalErrorLarge":
00100         __Line1 = [sg.Push(),
00101 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00102 sg.Text(self.__text, justification="left",
key="-textField-"), sg.Push()]
00103         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00104     elif self.__type == "error":
00105         __Line1 = [sg.Push(),
00106 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00107 sg.Text(f"{self.__text}: {self.__error}",
justification="center", key="-textField-"),
00108 sg.Push()]
00109         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00110     elif self.__type == "AuthError":
00111         __Line1 = [sg.Push(),
00112 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00113 sg.Text(f"{self.__text}", justification="center",
key="-textField-"),
00114 sg.Push()]
00115         __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool
[Web Browser]"),

```

```

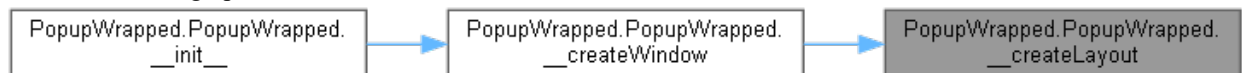
00116             sg.Ok(button_text="Return", focus=True, size=(10, 1)),
sg.Push()]
00117         elif self.__type == "versionWindow":
00118             __Line1 = [sg.Push(),
00119                 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00120                 sg.Text(f"{self.__text}", 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:



**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

```

```

00146         if self.__type == "progress":
00147             self.__windowObj = sg.Window(title=self.__type.capitalize(),
00148 layout=self.__layout, finalize=True,
00149                                     modal=True,
00150                                     keep_on_top=True,
00151                                     disable_close=False,
00152 icon=ImageLoader("taskbar_icon.ico"),
00153                                     size=(290, 50))
00154         elif self.__type == "noticeLarge":
00155             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
00156 layout=self.__layout, finalize=True,
00157                                     modal=True,
00158                                     keep_on_top=True,
00159                                     disable_close=False,
00160 icon=ImageLoader("taskbar_icon.ico"))
00161         elif self.__type == "savedLarge":
00162             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
00163 layout=self.__layout, finalize=True,
00164                                     modal=True,
00165                                     keep_on_top=False,
00166                                     disable_close=False,
00167 icon=ImageLoader("taskbar_icon.ico"))
00168         elif self.__type == "errorLarge":
00169             self.__windowObj = sg.Window(title="Error", layout=self.__layout,
00170 layout=self.__layout, finalize=True,
00171                                     modal=True,
00172                                     keep_on_top=True,
00173                                     disable_close=False,
00174 icon=ImageLoader("taskbar_icon.ico"))
00175         elif self.__type == "FatalErrorLarge":
00176             self.__windowObj = sg.Window(title="Fatal Error",
00177 layout=self.__layout, finalize=True,
00178                                     modal=True,
00179                                     keep_on_top=True,
00180                                     disable_close=False,
00181 icon=ImageLoader("taskbar_icon.ico"))
00182         elif self.__type == "AuthError":
00183             self.__windowObj = sg.Window(title="Authentication Error",
00184 layout=self.__layout, finalize=True,
00185                                     modal=True,
00186                                     keep_on_top=True,
00187                                     disable_close=False,
00188 icon=ImageLoader("taskbar_icon.ico"))
00189         elif self.__type == "versionWindow":
00190             self.__windowObj = sg.Window(title="Update", layout=self.__layout,
00191 layout=self.__layout, finalize=True,
00192                                     modal=True,
00193                                     keep_on_top=True,
00194                                     disable_close=False,
00195 icon=ImageLoader("taskbar_icon.ico"),
00196                                     size=(290, 80))
00197         if self.__type != "progress" or self.__type.startswith("perm"):
00198             print("Here")
00199             timer = 0
00200             while timer < 100:
00201                 event, values = self.__windowObj.read()
00202                 print(event)
00203                 if event == "Ok" or event == sg.WIN_CLOSED or event == "Return"
00204 or event == "Continue":
00205                     break
00206                 elif event == "Open Generation Tool [Web Browser]":

```

```

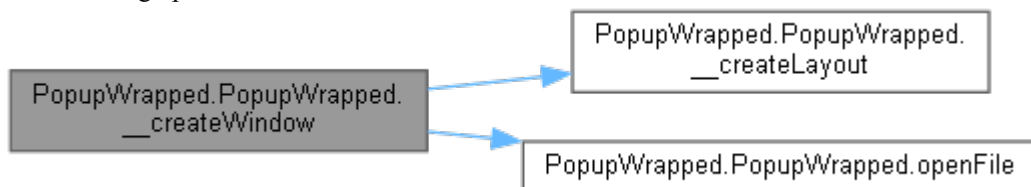
00206
webbrowser.open('https://www.debugbear.com/basic-auth-header-generator', new=2,
autorange=True)
00207
pass
00208
elif event == "Open File":
00209
threadFile = threading.Thread(target=self.openFile,
daemon=False)
00210
threadFile.start()
00211
time.sleep(3)
00212
break
00213
elif event == "Download":
00214
webbrowser.open('https://github.com/Kydoimos97/GardnerApiUtility/releases/latest',
new=2,
00215
autorange=True)
00216
pass
00217
time.sleep(0.1)
00218
if self.__type == "FatalErrorLarge":
00219
try:
00220
os.system(
00221
f"start
00222
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
00223
)
00224
except Exception as e:
00225
print(
00226
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")
00227
self.__windowObj.close()
00228
00229

```

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:



Here is the caller graph for this function:



**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:



**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
00233         in the startWindow function.
00234         This is done by calling the close() method on self.__windowObj, which will
00235         cause it to be destroyed.
00236         Args:
00237             self: Represent the instance of the class
00238         Returns:
00239             The window object
00240         Doc Author:
00241             Willem van der Schans, Trelent AI
00242         """
00243         self.__windowObj.close()
00244

```

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
00275         event starts with 'update'.
00276         If it does, it will take everything after 'update' as a key for updating
that specific
00277         value.
00278         It will then update that value using its key and refresh the window.
00279     Args:
00280         self: Reference the object that is calling the function
00281     Returns:
00282         A tuple containing the event and values
00283     Doc Author:
00284         Willem van der Schans, Trelent AI
00285     """
00286     event, values = self.__windowObj.read()
00287
00288     if event.startswith('update'):
00289         __key_to_update = event[len('update'):]
00290         self.__windowObj[__key_to_update].update(values[event])
00291         self.__windowObj.refresh()
00292

```

References [PopupWrapped.PopupWrapped.\\_\\_windowObj](#).

---

## Member Data Documentation

### PopupWrapped.PopupWrapped.\_\_counter[private]

Definition at line [36](#) of file [PopupWrapped.py](#).

Referenced by [PopupWrapped.PopupWrapped.\\_\\_init\\_\\_\(\)](#), [PopupWrapped.PopupWrapped.textUpdate\(\)](#), and

### PopupWrapped.PopupWrapped.\_\_docpath[private]

Definition at line [37](#) of file [PopupWrapped.py](#).

Referenced by [PopupWrapped.PopupWrapped.\\_\\_init\\_\\_\(\)](#), [PopupWrapped.PopupWrapped.openFile\(\)](#), and

### 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\(\)](#), [PopupWrapped.PopupWrapped.\\_\\_init\\_\\_\(\)](#), and

### 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\(\)](#), [PopupWrapped.PopupWrapped.\\_\\_init\\_\\_\(\)](#), and

### 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
00021         needs to run before anything else in your class.
00022         Args:
00023             self: Represent the instance of the class
00024         Returns:
00025             A new object
00026         Doc Author:
00027             Willem van der Schans, Trelent AI
00029
```

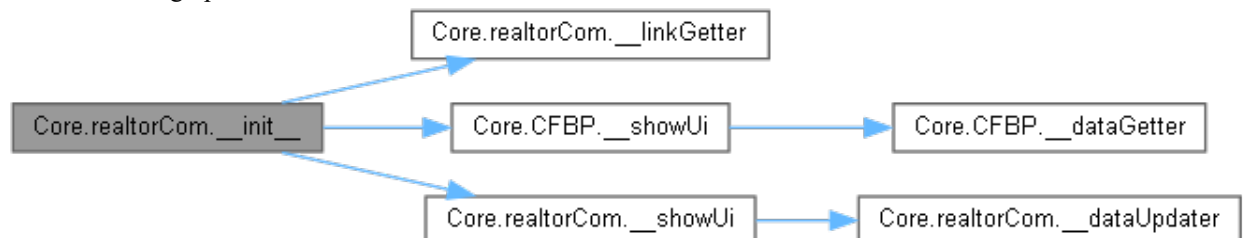
```

00030     """
00031         self.__page_html = None
00032         self.__update_date = None
00033         self.__last_date = None
00034         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00035         self.__linkDict = {}
00036         self.dfState = None
00037         self.dfCounty = None
00038         self.dfZip = None
00039         self.uiString = "Files Saved to \n"
00040
00041         eventReturn = confirmDialog()
00042         if eventReturn == "Continue":
00043             page_html =
requests.get("https://www.realtor.com/research/data/").text
00044             self.__page_html = BeautifulSoup(page_html, "html.parser")
00045             startTime = datetime.datetime.now().replace(microsecond=0)
00046             self.__linkGetter()
00047             print(
00048                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Link Dictionary = {self.__idDict}")
00049             self.__showUi()
00050             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00051             print(
00052                 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()))}")
00053             else:
00054                 print(
00055                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | User Canceled Request")
00056                 pass
00057

```

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
00117         for each of the three
00118         types of realtor data. It takes class variables and return the path to
00119         the saved file. The function first creates an empty
00120         dictionary called tempdf, then iterates through each key in self.__idDict
00121         (which contains all three ids).
00122         For each key, it reads in a csv file from the link associated with that
00123         id and saves it to tempdf as a pandas
00124         DataFrame object. Then, depending on which type of realtor data we are
00125         dealing with (State/County/Zip), we save
00126
00127         Args:
00128             self: Access the attributes and methods of the class
00129
00130         Returns:
00131             The path of the saved file
00132
00133         Doc Author:
00134             Willem van der Schans, Trelent AI
00135         """
00136         for key, value in self.__idDict.items():
00137             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00138
00139             if key == "State":
00140                 self.dfState = tempdf
00141             elif key == "County":
00142                 self.dfCounty = tempdf
00143             elif key == "Zip":
00144                 self.dfZip = tempdf
00145
00146         FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00147         self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()}"
00148         \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
00090         dictionary and adds
00091         a link to each entry in the dictionary. The link is used to access historical
00092         data for each
00093         scope symbol.
00094
00095         Args:
00096         self: Refer to the object itself
00097
00098         Returns:
00099         A dictionary of all the links to the history pages
00100
00101         Doc Author:
00102         Willem van der Schans, Trelent AI
00103         """
00104         for key, value in self.__idDict.items():
00105             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00106                 try:
00107                     for nestedRow in row.find_all("a"):
00108                         if "History" in str(nestedRow.get("href")) and key in
00109                         str(nestedRow.get("href")):
00110                             self.__idDict[key] = {"id": value, "link":
00111                             nestedRow.get("href")}
00112                     except Exception as e:
00113                         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00114                         %H:%M:%S.%f')}[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
00115                         for realtor.com")
00116                         RESTError(801)
00117                         raise SystemExit(801)
00118                 
```

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
00062         progress window.
00063         It also starts the dataUpdater thread, which will update the progress bar
00064         as it runs.
00065
00066         Args:
```



```

00066         self: Represent the instance of the class
00067
00068     Returns:
00069         A popupwrapped object
00070
00071     Doc Author:
00072         Willem van der Schans, Trelent AI
00073     """
00074     uiObj = PopupWrapped(text="Request running", windowType="progress",
00075 error=None)
00076
00077     threadGui = threading.Thread(target=self.__dataUpdater,
00078                                 daemon=False)
00079     threadGui.start()
00080
00081     while threadGui.is alive():
00082         uiObj.textUpdate()
00083         uiObj.windowPush()
00084     else:
00085         uiObj.stopWindow()

```

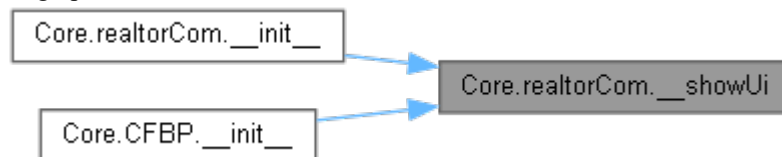
References [Core.realtorCom.\\_\\_dataUpdater\(\)](#).

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](#).

```
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the initial state of the object.
00031
00032
00033         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             The __createframe function
00038
00039         Doc Author:
00040             Willem van der Schans, Trelent AI
```

```

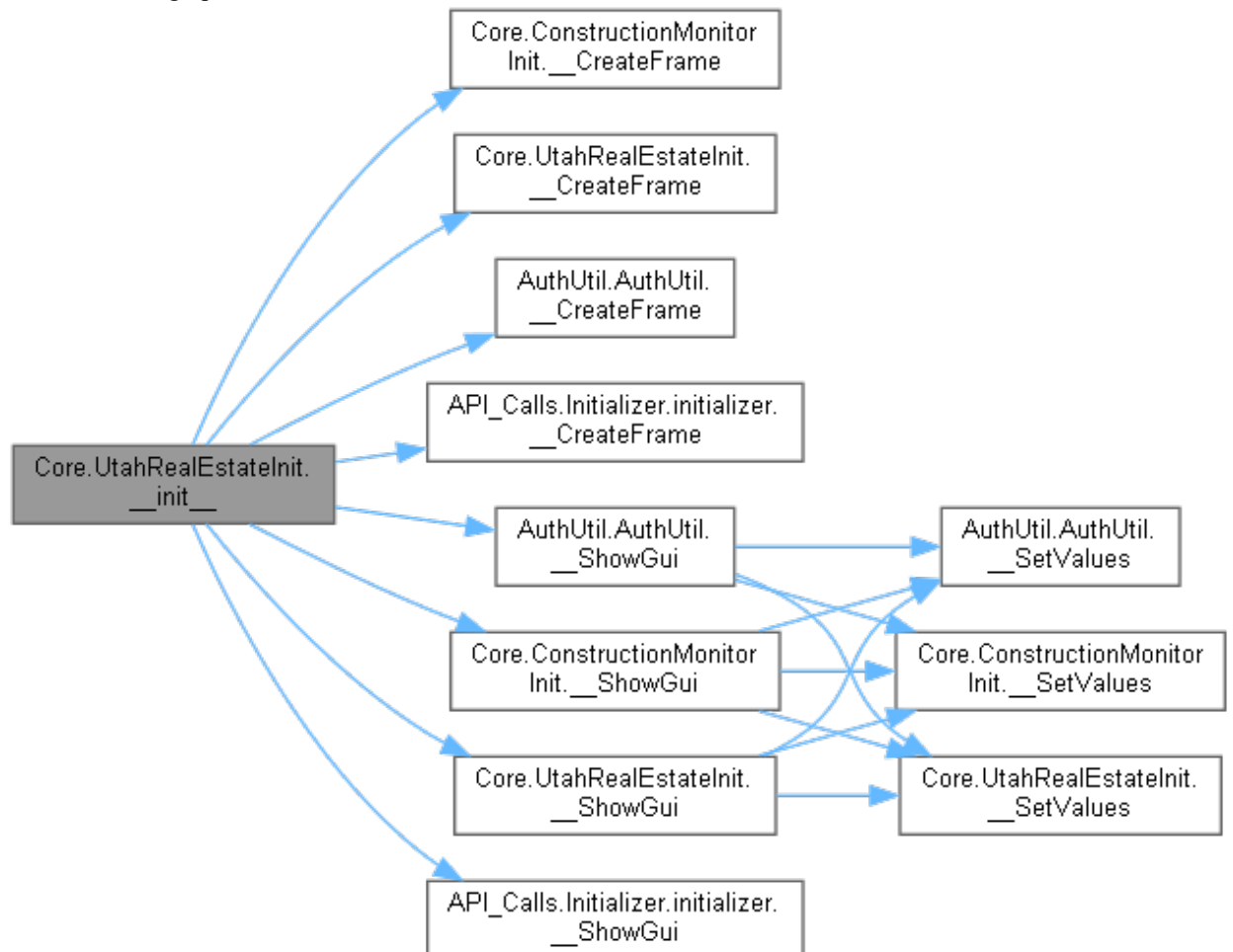
00041     """
00042         self.StandardStatus = None
00043         self.ListedOrModified = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.select = None
00047         self.file_name = None
00048         self.append_file = None
00049
00050         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00051

```

References

<a href="#">Core.UtahRealEstateInit.__CreateFrame()</a> ,	<a href="#">Core.ConstructionMonitorInit.__CreateFrame()</a> ,
<a href="#">API_Calls.Initializer.initializer.__CreateFrame()</a> ,	<a href="#">AuthUtil.AuthUtil.__CreateFrame()</a> ,
<a href="#">API_Calls.Initializer.initializer.__ShowGui()</a> ,	<a href="#">AuthUtil.AuthUtil.__ShowGui()</a> ,
<a href="#">Core.UtahRealEstateInit.__ShowGui()</a> ,	<a href="#">Core.ConstructionMonitorInit.__ShowGui()</a> ,
<a href="#">Core.ConstructionMonitorInit.append_file</a> ,	<a href="#">AuthUtil.AuthUtil.append_file</a> ,
<a href="#">Core.ConstructionMonitorInit.dateEnd</a> ,	<a href="#">Core.UtahRealEstateInit.append_file</a> ,
<a href="#">Core.ConstructionMonitorInit.dateStart</a> ,	<a href="#">Core.UtahRealEstateInit.dateEnd</a> ,
<a href="#">AuthUtil.AuthUtil.file_name</a> ,	<a href="#">Core.UtahRealEstateInit.dateStart</a> ,
<a href="#">AuthUtil.AuthUtil.ListedOrModified</a> ,	<a href="#">Core.UtahRealEstateInit.file_name</a> ,
<a href="#">Core.UtahRealEstateInit.select</a> ,	<a href="#">Core.UtahRealEstateInit.ListedOrModified</a> ,
<a href="#">Core.UtahRealEstateInit.StandardStatus</a> ,	<a href="#">AuthUtil.AuthUtil.StandardStatus</a> ,
	and

Here is the call graph for this function:



## 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
00096         be displayed in the window.
00097         Each element is defined by its type and any additional parameters needed
00098         to define it.
00099         Args:
00100         Returns:
00101             A list of lists, which is used to create the gui
00102         Doc Author:
00103             Willem van der Schans, Trelent AI
00104         """
00105         sg.theme('Default1')
00106         line0 = [sg.HSeparator()]
00107         line0 = [sg.Image(ImageLoader("logo.png")),
00108                 sg.Push(),
00109                 sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
00110 "bold"), justification="center"),
00111                 sg.Push(),
00112                 sg.Push()]
00113         line1 = [sg.HSeparator()]
00114         line2 = [sg.Text("MLS Status : ", size=(15, None),
00115 justification="Right"),
00116                 sg.DropDown(default_value="Active", values=["Active",
00117 "Closed"], key="-status-", size=(31, 1))]
00118         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00119                 sg.DropDown(default_value="Listing Date", values=["Listing
00120 Date", "Modification Date", "Close Date"],
00121 key="-type-", size=(31, 1))]
00122         line4 = [sg.Text("Start Date : ", size=(15, None),
00123 justification="Right"),
00124                 sg.Input(default_text=(date.today() -
00125 timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00126 disabled=False, size=(20, 1)),
00127                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00128 key='-start_date-', target="-DateStart-")]
00129         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00130                 sg.Input(default_text=(date.today()).strftime("%Y-%m-%d"),
00131 key="-DateEnd-", disabled=False,
00132 size=(20, 1)),
00133                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00134 key='-end_date-', target="-DateEnd-")]
00135
```

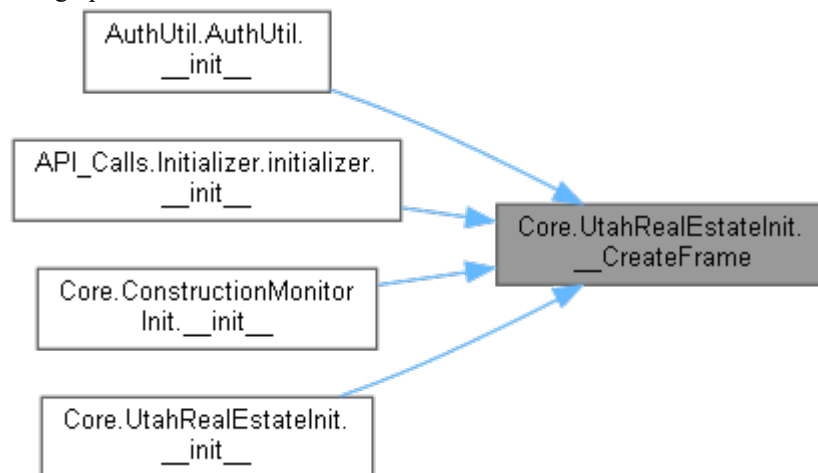
```

00135         line7 = [sg.HSeparator()]
00136
00137         line8 = [sg.Push(),
00138                 sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00139                        justification="center"),
00140                 sg.Push()]
00141
00142         line9 = [sg.HSeparator()]
00143
00144         line10 = [sg.Text("Appending File : ", size=(15, None),
00145                        justification="Right"),
00146                  sg.Input(default_text="", key="-AppendingFile-",
00147                         disabled=True,
00148                             size=(20, 1)),
00149                  sg.FileBrowse("Browse File", file_types=[("csv files",
00150                        "*.csv")], key='-append file-',
00151                             target="-AppendingFile-")]
00152
00153         line11 = [sg.HSeparator()]
00154
00155         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00156
00157         layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8,
00158                 line9, line10, line11,
00159                 line12]
00160
00161         return layout
00162
00163

```

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

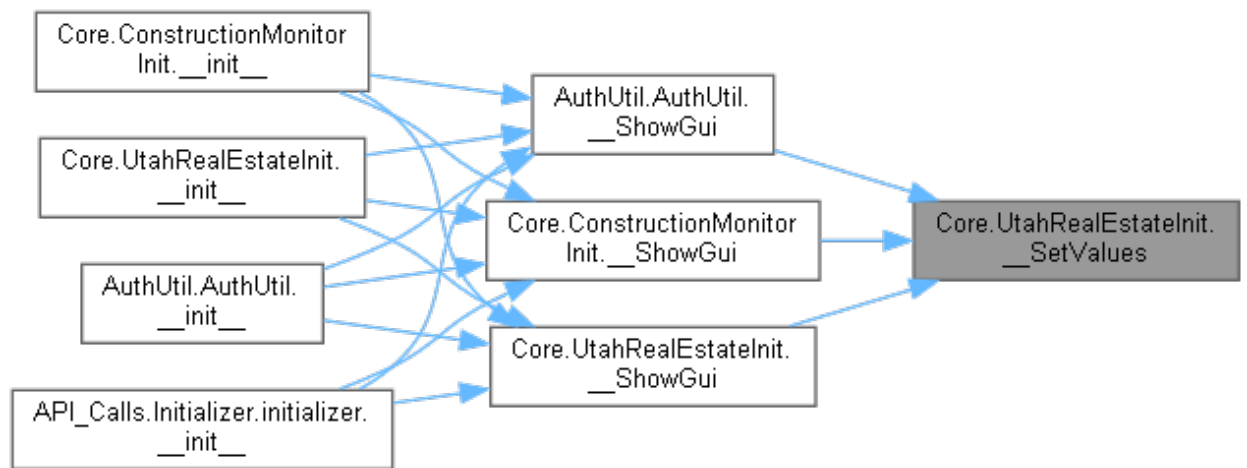
Definition at line [158](#) of file [UtahRealEstate/Core.py](#).

```
00158     def __SetValues(self, values):
00159
00160         """
00161         The __SetValues function is used to set the values of the variables that are
00162         used in the __GetData function. The values are passed from a dictionary called
00163         'values' which is created by parsing through an XML file using ElementTree. This function also sets
00164         default values for some of these variables if they were not specified in the XML file.
00165
00166         Args:
00167             self: Represent the instance of the class
00168             values: Pass the values from the gui to this function
00169
00170         Returns:
00171             A dictionary with the following keys:
00172
00173         Doc Author:
00174             Willem van der Schans, Trelent AI
00175         """
00176         self.StandardStatus = values["-status-"]
00177
00178         self.ListedOrModified = values["-type-"]
00179
00180         if values["-DateStart-"] != "":
00181             self.dateStart = values["-DateStart-"]
00182         else:
00183             self.dateStart = (date.today() -
00184                             timedelta(days=14)).strftime("%Y-%m-%d")
00185
00186         if values["-DateEnd-"] != "":
00187             self.dateEnd = values["-DateEnd-"]
00188         else:
00189             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00190
00191         self.select = None
00192
00193         if values["-append_file-"] != "":
00194             self.append_file = str(values["-append_file-"])
00195         else:
00196             self.append_file = None
00197
```

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:



```
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
00056         displays it to the user.
00057         It takes in two parameters: layout, which is a list of lists containing all
00058         the elements for each row;
00059         and text, which is a string containing what will be displayed as the title
00060         of the window. The __ShowGui
00061         method then uses these parameters to create an instance of sg.Window with
00062         all its attributes set accordingly.
00063
00064         Args:
00065             self: Refer to the current class instance
00066             layout: Pass the layout of the window to be created
00067             text: Set the title of the window
00068
00069         Returns:
00070             A dictionary of values
00071
00072         Doc Author:
00073             Willem van der Schans, Trelent AI
00074         """
00075         window = sg.Window(text, layout, grab_anywhere=False,
00076                             return_keyboard_events=True,
00077                             finalize=True,
00078                             icon=ImageLoader("taskbar_icon.ico"))
00079         while True:
```



```

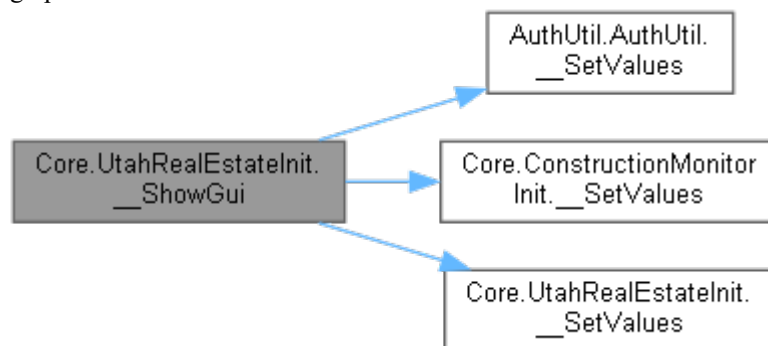
00076         event, values = window.read()
00077
00078         if event == "Submit":
00079             try:
00080                 self.__SetValues(values)
00081                 break
00082             except Exception as e:
00083                 print(e)
00084                 RESTError(993)
00085                 raise SystemExit(993)
00086         elif event == sg.WIN_CLOSED or event == "Quit":
00087             break
00088
00089         window.close()
00090

```

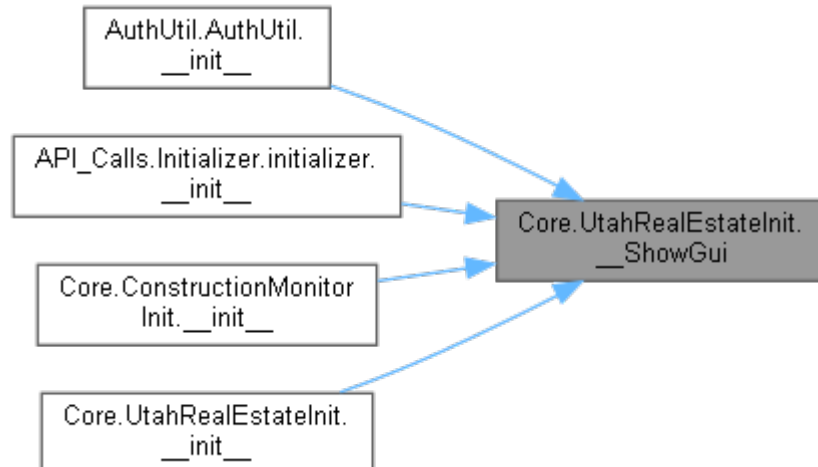
References [AuthUtil.AuthUtil.\\_\\_SetValues\(\)](#), [Core.ConstructionMonitorInit.\\_\\_SetValues\(\)](#), and [Core.UtahRealEstateInit.\\_\\_SetValues\(\)](#).

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:



Here is the caller graph for this function:



## 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.
```

```

00204     It sets up all the variables and functions needed for this class to work
properly.
00205
00206     Args:
00207         self: Represent the instance of the class
00208         siteClass: Determine which site to pull data from
00209
00210     Returns:
00211         Nothing
00212
00213     Doc Author:
00214         Willem van der Schans, Trelent AI
00215     """
00216         self.dataframe = None
00217         self.__batches = 0
00218         self.__siteClass = siteClass
00219         self.__headerDict = None
00220         self.__parameterString = ""
00221         self.__appendFile = None
00222         self.__dateStart = None
00223         self.__dateEnd = None
00224         self.__restDomain =
'https://resoapi.utahrealstate.com/reso/odata/Property?'
00225         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00226             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00227         self.filePath =
Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00228             "Security").joinpath("auth.json")
00229         self.key = None
00230         self.__record_val = None
00231
00232         try:
00233             self.mainFunc()
00234         except KeyError as e:
00235             # This allows for user cancellation of the program using the quit
button
00236             if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00237                 RESTError(1101)
00238                 print(e)
00239                 pass
00240             else:
00241                 pass
00242         except Exception as e:
00243             print(e)
00244             RESTError(1001)
00245             raise SystemExit(1001)
00246

```

## References

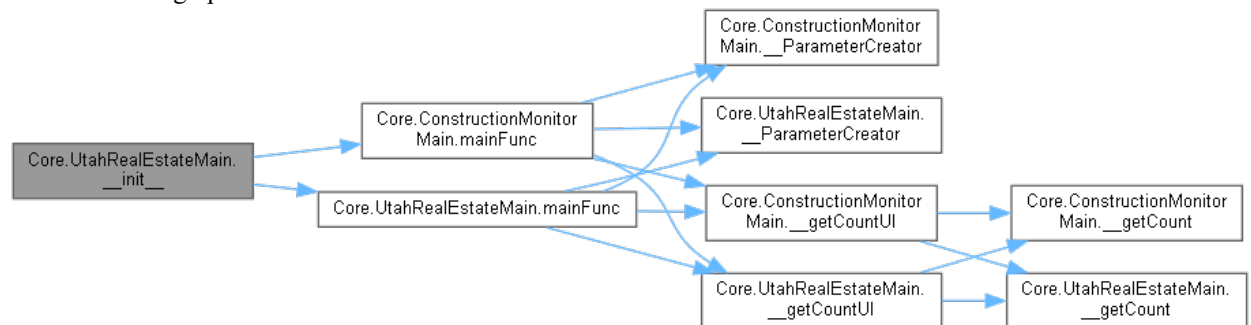
[Core.ConstructionMonitorMain.\\_\\_appendFile](#),  
[Core.UtahRealEstateMain.\\_\\_appendFile](#),  
[Core.ConstructionMonitorMain.\\_\\_batches](#),  
[BatchProgressGUI.BatchProgressGUI.\\_\\_batches](#),  
[Core.UtahRealEstateMain.\\_\\_dateEnd](#),  
[Core.UtahRealEstateMain.\\_\\_dateStart](#),  
[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.UtahRealEstateMain.mainFunc\(\).](#)

[Core.ConstructionMonitorMain.mainFunc\(\),](#)

and

Here is the call graph for this function:



## 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](#).

```
00365     def __getCount(self):
00366         """
00367         The __getCount function is used to determine the number of records that will
00368         be returned by the query.
00369         This function is called when a user calls the count() method on a ReST object.
00370         The __getCount function uses
00371         the $count parameter in OData to return only an integer value representing
00372         how many records would be returned
00373         by the query.
00374
00375         Args:
00376         self: Represent the instance of the class
00377
00378         Returns:
00379         The number of records in the data set
00380
00381         Doc Author:
00382         Willem van der Schans, Trelent AI
00383         """
00384         __count_resp = None
00385         try:
00386             __count_resp =
00387             requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00388                         headers=self.__headerDict)
00389         except requests.exceptions.Timeout as e:
00390             print(e)
00391             RESTError(790)
00392             raise SystemExit(790)
```

```

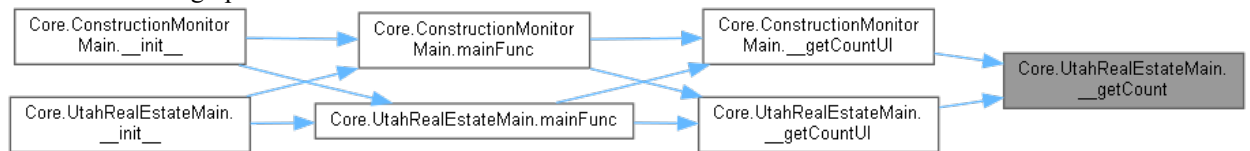
00391         except requests.exceptions.TooManyRedirects as e:
00392             print(e)
00393             RESTError(791)
00394             raise SystemExit(791)
00395         except requests.exceptions.MissingSchema as e:
00396             print(e)
00397             RESTError(1101)
00398         except requests.exceptions.RequestException as e:
00399             print(e)
00400             RESTError(405)
00401             raise SystemExit(405)
00402
00403         self.__record_val = int(__count_resp.json()["@odata.count"])
00404

```

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:



**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](#).

```

00405     def __getCountUI(self):
00406
00407         """
00408         The __getCountUI function is a wrapper for the __getCount function.
00409         It creates a progress window and updates it while the __getCount function
00410         runs.
00411         The purpose of this is to keep the GUI responsive while running long processes.
00412         Args:
00413             self: Represent the instance of the class
00414         Returns:
00415             A popupwrapped object
00416         Doc Author:
00417             Willem van der Schans, Trelent AI
00418         """
00419         uiObj = PopupWrapped(text="Batch request running",
00420                               windowType="progress", error=None)
00421
00422         threadGui = threading.Thread(target=self.__getCount,
00423                                     daemon=False)
00424         threadGui.start()
00425
00426

```

```

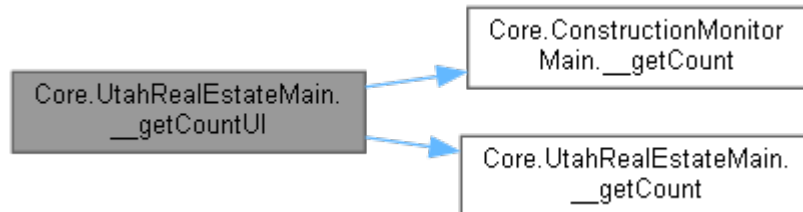
00427         while threadGui.is_alive():
00428             uiObj.textUpdate()
00429             uiObj.windowPush()
00430         else:
00431             uiObj.stopWindow()

```

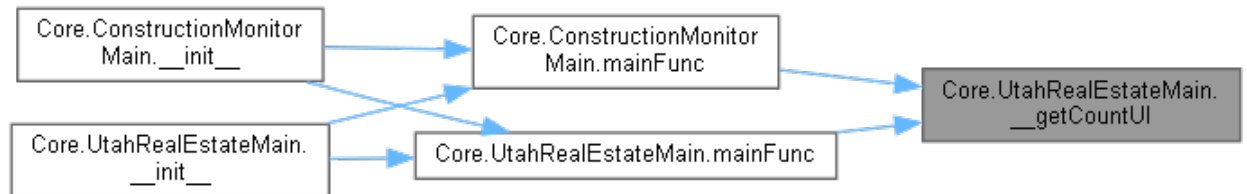
References [Core.ConstructionMonitorMain.\\_\\_getCount\(\)](#), and  
[Core.UtahRealEstateMain.\\_\\_getCount\(\)](#).

Referenced by [Core.ConstructionMonitorMain.mainFunc\(\)](#), and  
[Core.UtahRealEstateMain.mainFunc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**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](#).

```

00324     def __ParameterCreator(self):
00325         """
00326         The __ParameterCreator function is used to create the filter string for the
00327         ReST API call.
00328         The function takes in a siteClass object and extracts all of its parameters
00329         into a dictionary.
00330         It then creates an appropriate filter string based on those parameters.
00331
00332         Args:
00333             self: Bind the object to the class
00334
00335         Returns:
00336             A string to be used as the parameter in the api call
00337
00338         Doc Author:
00339             Willem van der Schans, Trelent AI
00340         """
00341         filter_string = ""
00342         __Source_dict = {key: value for key, value in
self.__siteClass.__dict__.items() if
not key.startswith('__') and not callable(key)}

```

```

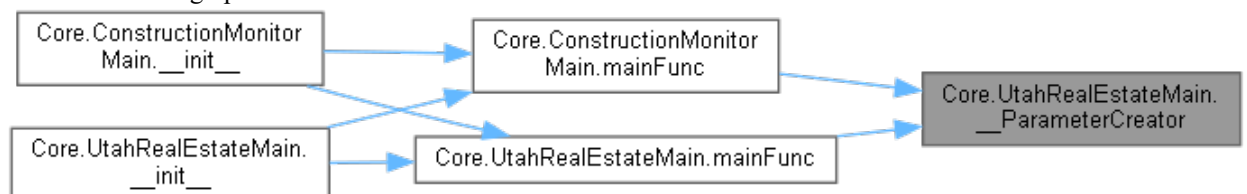
00343
00344     self.__appendFile = __Source_dict["append_file"]
00345     __Source_dict.pop("append_file")
00346
00347     temp_dict = copy.copy(__Source_dict)
00348     for key, value in temp_dict.items():
00349         if value is None:
00350             __Source_dict.pop(key)
00351         else:
00352             pass
00353
00354     if __Source_dict["ListedOrModified"] == "Listing Date":
00355         filter_string =
00356         f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingC
00357         ontractDate%20le%20{__Source_dict['dateEnd']}"
00358     elif __Source_dict["ListedOrModified"] == "Modification Date":
00359         filter_string =
00360         f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20a
00361         nd%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00362     elif __Source_dict["ListedOrModified"] == "Close Date":
00363         filter_string =
00364         f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{
00365         __Source_dict['dateEnd']}"
00366
00367     filter_string = filter_string +
00368     f"%20and%20StandardStatus%20has%20data.Models.StandardStatus'{__Source_dict['Stan
00369     dardStatus']}'"
00370
00371     self.__parameterString = filter_string
00372

```

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:



**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 "Run" 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         """

```



```

00250     The mainFunc function is the main function of this module. It will be called
by the GUI when a user clicks on
00251     the &quot;Run&quot; button in the GUI. The mainFunc function should contain
all of your code for running your program, and it
00252     should return a dataframe that contains all the data you want to display in
your final report.
00253
00254     Args:
00255         self: Reference the object itself
00256
00257     Returns:
00258         A dataframe
00259
00260     Doc Author:
00261         Willem van der Schans, Trelent AI
00262     """
00263     passFlag = False
00264
00265     while not passFlag:
00266         if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00267             try:
00268                 f = open(self.keyPath, "rb")
00269                 key = f.readline()
00270                 f.close()
00271                 f = open(self.filePath, "rb")
00272                 authDict = json.load(f)
00273                 fernet = Fernet(key)
00274                 authkey =
fernet.decrypt(authDict["ure"]["auth"]).decode()
00275                 self.__headerDict = {authDict["ure"]["parameter"] :
authkey}
00276                 passFlag = True
00277             except Exception as e:
00278                 print(
00279                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening
AuthUtil")
00280                 AuthUtil()
00281             else:
00282                 AuthUtil()
00283
00284         self.__ParameterCreator()
00285
00286         print(
00287             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Param String = {self.__parameterString}")
00288         print(
00289             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Rest Domain = {self.__restDomain}")
00290
00291         self.__getCountUI()
00292
00293         if self.__record_val is None:
00294             self.__record_val = 0
00295
00296         self.__batches = BatchCalculator(self.__record_val, None)
00297
00298         print(
00299             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")
00300
00301         if self.__batches != 0:
00302             startTime = datetime.datetime.now().replace(microsecond=0)
00303             eventReturn = BatchInputGui(self.__batches, self.__record_val)
00304             if eventReturn == "Continue":
00305                 print(
00306                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} batches sent to server")
00307                 BatchGuiObject =
BatchProgressGUI(RestDomain=self.__restDomain,
00308
ParameterDict=self.__parameterString,
00309
HeaderDict=self.__headerDict,
00310
BatchesNum=self.__batches,
00311
Type="utah_real_estate")
00312                 BatchGuiObject.BatchGuiShow()

```

```

00313         self.dataframe = BatchGuiObject.dataframe
00314         print(
00315             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()))})")
00316         FileSaver("ure", self.dataframe, self.__appendFile)
00317     else:
00318         print(
00319             f"({datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]) | Request for {self.__batches} batches canceled by user")
00320     else:
00321         RESTError(994)
00322         raise SystemExit(994)
00323

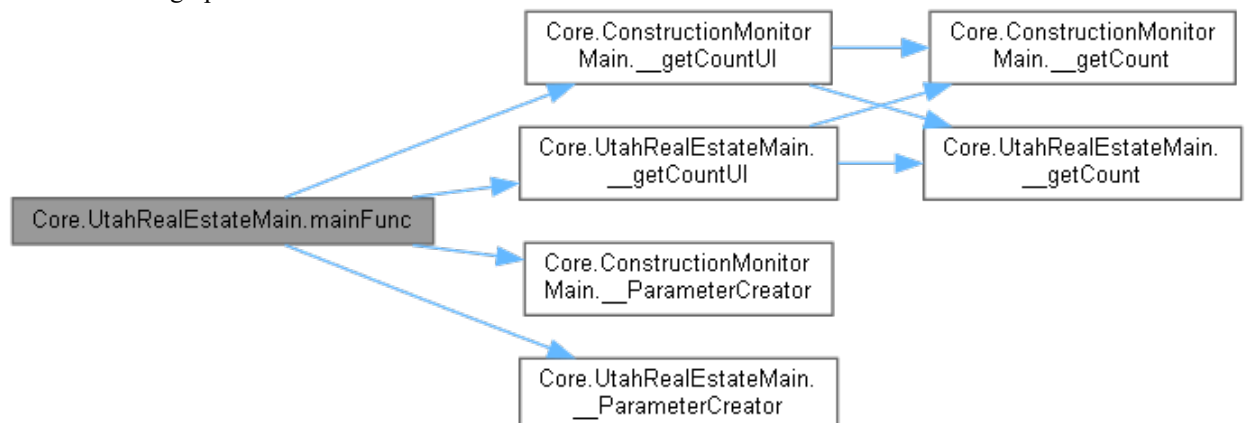
```

#### References

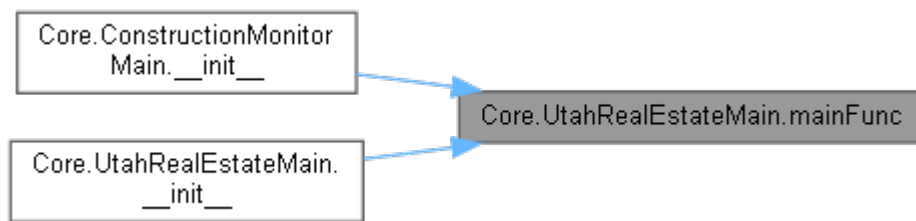
[Core.ConstructionMonitorMain.\\_\\_appendFile](#),  
[Core.UtahRealEstateMain.\\_\\_appendFile](#),  
[Core.ConstructionMonitorMain.batches](#),  
[Core.UtahRealEstateMain.batches](#),  
[Core.ConstructionMonitorMain.getCountUI\(\)](#),  
[Core.UtahRealEstateMain.getCountUI\(\)](#),  
[BatchProcessing.BatchProcessorConstructionMonitor.headerDict](#),  
[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](#),  
[Core.UtahRealEstateMain.keyPath](#).

Referenced by [Core.ConstructionMonitorMain.init\(\)](#),  
[Core.UtahRealEstateMain.init\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:




---

## 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**

```
00001 /* Generated by Cython 0.29.32 */
00002
00003 #ifndef PY_SSIZE_T_CLEAN
00004 #define PY_SSIZE_T_CLEAN
00005 #endif /* PY_SSIZE_T_CLEAN */
00006 #include "Python.h"
00007 #ifndef Py_PYTHON_H
00008     #error Python headers needed to compile C extensions, please install development
    version of Python.
00009 #elif PY_VERSION_HEX < 0x02060000 || (0x03000000 <= PY_VERSION_HEX && PY_VERSION_HEX
    < 0x03030000)
00010     #error Cython requires Python 2.6+ or Python 3.3+.
00011 #else
00012 #define CYTHON_ABI "0_29_32"
00013 #define CYTHON_HEX_VERSION 0x001D20F0
00014 #define CYTHON_FUTURE_DIVISION 0
00015 #include <stddef.h>
00016 #ifndef offsetof
00017     #define offsetof(type, member) ( (size_t) &((type*)0) -> member )
00018 #endif
00019 #if !defined(WIN32) && !defined(MS_WINDOWS)
00020     #ifndef __stdcall
00021         #define __stdcall
00022     #endif
00023     #ifndef __cdecl
00024         #define __cdecl
00025     #endif
00026     #ifndef __fastcall
00027         #define __fastcall
00028     #endif
00029 #endif
00030 #ifndef DL_IMPORT
00031     #define DL_IMPORT(t) t
00032 #endif
00033 #ifndef DL_EXPORT
00034     #define DL_EXPORT(t) t
00035 #endif
00036 #define __PYX_COMMA ,
00037 #ifndef HAVE_LONG_LONG
00038     #if PY_VERSION_HEX >= 0x02070000
00039         #define HAVE_LONG_LONG
00040     #endif
00041 #endif
00042 #ifndef PY_LONG_LONG
00043     #define PY_LONG_LONG LONG_LONG
00044 #endif
00045 #ifndef Py_HUGE_VAL
00046     #define Py_HUGE_VAL HUGE_VAL
00047 #endif
00048 #ifdef PYPY_VERSION
00049     #define CYTHON_COMPILING_IN_PYPY 1
00050     #define CYTHON_COMPILING_IN_PYSTON 0
00051     #define CYTHON_COMPILING_IN_CPYTHON 0
00052     #define CYTHON_COMPILING_IN_NOGIL 0
00053     #undef CYTHON_USE_TYPE_SLOTS
00054     #define CYTHON_USE_TYPE_SLOTS 0
00055     #undef CYTHON_USE_PYTYPE_LOOKUP
00056     #define CYTHON_USE_PYTYPE_LOOKUP 0
00057     #if PY_VERSION_HEX < 0x03050000
00058         #undef CYTHON_USE_ASYNC_SLOTS
00059         #define CYTHON_USE_ASYNC_SLOTS 0
00060     #elif !defined(CYTHON_USE_ASYNC_SLOTS)
00061         #define CYTHON_USE_ASYNC_SLOTS 1
00062     #endif
00063     #undef CYTHON_USE_PYLIST_INTERNALS
00064     #define CYTHON_USE_PYLIST_INTERNALS 0
00065     #undef CYTHON_USE_UNICODE_INTERNALS
00066     #define CYTHON_USE_UNICODE_INTERNALS 0
00067     #undef CYTHON_USE_UNICODE_WRITER
00068     #define CYTHON_USE_UNICODE_WRITER 0
00069     #undef CYTHON_USE_PYLONG_INTERNALS
00070     #define CYTHON_USE_PYLONG_INTERNALS 0
00071     #undef CYTHON_AVOID_BORROWED_REFS
```

```

00072 #define CYTHON_AVOID_BORROWED_REFS 1
00073 #undef CYTHON_ASSUME_SAFE_MACROS
00074 #define CYTHON_ASSUME_SAFE_MACROS 0
00075 #undef CYTHON_UNPACK_METHODS
00076 #define CYTHON_UNPACK_METHODS 0
00077 #undef CYTHON_FAST_THREAD_STATE
00078 #define CYTHON_FAST_THREAD_STATE 0
00079 #undef CYTHON_FAST_PYCALL
00080 #define CYTHON_FAST_PYCALL 0
00081 #undef CYTHON_PEP489_MULTI_PHASE_INIT
00082 #define CYTHON_PEP489_MULTI_PHASE_INIT 0
00083 #undef CYTHON_USE_TP_FINALIZE
00084 #define CYTHON_USE_TP_FINALIZE 0
00085 #undef CYTHON_USE_DICT_VERSIONS
00086 #define CYTHON_USE_DICT_VERSIONS 0
00087 #undef CYTHON_USE_EXC_INFO_STACK
00088 #define CYTHON_USE_EXC_INFO_STACK 0
00089 #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
00090     #define CYTHON_UPDATE_DESCRIPTOR_DOC (PYPY_VERSION_HEX >= 0x07030900)
00091 #endif
00092 #elif defined(PYSTON_VERSION)
00093     #define CYTHON_COMPILING_IN_PYPY 0
00094     #define CYTHON_COMPILING_IN_PYSTON 1
00095     #define CYTHON_COMPILING_IN_CPYTHON 0
00096     #define CYTHON_COMPILING_IN_NOGIL 0
00097     #ifndef CYTHON_USE_TYPE_SLOTS
00098         #define CYTHON_USE_TYPE_SLOTS 1
00099     #endif
00100     #undef CYTHON_USE_PYTYPE_LOOKUP
00101     #define CYTHON_USE_PYTYPE_LOOKUP 0
00102     #undef CYTHON_USE_ASYNC_SLOTS
00103     #define CYTHON_USE_ASYNC_SLOTS 0
00104     #undef CYTHON_USE_PYLIST_INTERNALS
00105     #define CYTHON_USE_PYLIST_INTERNALS 0
00106     #ifndef CYTHON_USE_UNICODE_INTERNALS
00107         #define CYTHON_USE_UNICODE_INTERNALS 1
00108     #endif
00109     #undef CYTHON_USE_UNICODE_WRITER
00110     #define CYTHON_USE_UNICODE_WRITER 0
00111     #undef CYTHON_USE_PYLONG_INTERNALS
00112     #define CYTHON_USE_PYLONG_INTERNALS 0
00113     #ifndef CYTHON_AVOID_BORROWED_REFS
00114         #define CYTHON_AVOID_BORROWED_REFS 0
00115     #endif
00116     #ifndef CYTHON_ASSUME_SAFE_MACROS
00117         #define CYTHON_ASSUME_SAFE_MACROS 1
00118     #endif
00119     #ifndef CYTHON_UNPACK_METHODS
00120         #define CYTHON_UNPACK_METHODS 1
00121     #endif
00122     #undef CYTHON_FAST_THREAD_STATE
00123     #define CYTHON_FAST_THREAD_STATE 0
00124     #undef CYTHON_FAST_PYCALL
00125     #define CYTHON_FAST_PYCALL 0
00126     #undef CYTHON_PEP489_MULTI_PHASE_INIT
00127     #define CYTHON_PEP489_MULTI_PHASE_INIT 0
00128     #undef CYTHON_USE_TP_FINALIZE
00129     #define CYTHON_USE_TP_FINALIZE 0
00130     #undef CYTHON_USE_DICT_VERSIONS
00131     #define CYTHON_USE_DICT_VERSIONS 0
00132     #undef CYTHON_USE_EXC_INFO_STACK
00133     #define CYTHON_USE_EXC_INFO_STACK 0
00134     #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
00135         #define CYTHON_UPDATE_DESCRIPTOR_DOC 0
00136     #endif
00137 #elif defined(PY_NOGIL)
00138     #define CYTHON_COMPILING_IN_PYPY 0
00139     #define CYTHON_COMPILING_IN_PYSTON 0
00140     #define CYTHON_COMPILING_IN_CPYTHON 0
00141     #define CYTHON_COMPILING_IN_NOGIL 1
00142     #ifndef CYTHON_USE_TYPE_SLOTS
00143         #define CYTHON_USE_TYPE_SLOTS 1
00144     #endif
00145     #undef CYTHON_USE_PYTYPE_LOOKUP
00146     #define CYTHON_USE_PYTYPE_LOOKUP 0
00147     #ifndef CYTHON_USE_ASYNC_SLOTS
00148         #define CYTHON_USE_ASYNC_SLOTS 1

```



```

00149 #endif
00150 #undef CYTHON_USE_PYLIST_INTERNALS
00151 #define CYTHON_USE_PYLIST_INTERNALS 0
00152 #ifndef CYTHON_USE_UNICODE_INTERNALS
00153     #define CYTHON_USE_UNICODE_INTERNALS 1
00154 #endif
00155 #undef CYTHON_USE_UNICODE_WRITER
00156 #define CYTHON_USE_UNICODE_WRITER 0
00157 #undef CYTHON_USE_PYLONG_INTERNALS
00158 #define CYTHON_USE_PYLONG_INTERNALS 0
00159 #ifndef CYTHON_AVOID_BORROWED_REFS
00160     #define CYTHON_AVOID_BORROWED_REFS 0
00161 #endif
00162 #ifndef CYTHON_ASSUME_SAFE_MACROS
00163     #define CYTHON_ASSUME_SAFE_MACROS 1
00164 #endif
00165 #ifndef CYTHON_UNPACK_METHODS
00166     #define CYTHON_UNPACK_METHODS 1
00167 #endif
00168 #undef CYTHON_FAST_THREAD_STATE
00169 #define CYTHON_FAST_THREAD_STATE 0
00170 #undef CYTHON_FAST_PYCALL
00171 #define CYTHON_FAST_PYCALL 0
00172 #ifndef CYTHON_PEP489_MULTI_PHASE_INIT
00173     #define CYTHON_PEP489_MULTI_PHASE_INIT 1
00174 #endif
00175 #ifndef CYTHON_USE_TP_FINALIZE
00176     #define CYTHON_USE_TP_FINALIZE 1
00177 #endif
00178 #undef CYTHON_USE_DICT_VERSIONS
00179 #define CYTHON_USE_DICT_VERSIONS 0
00180 #undef CYTHON_USE_EXC_INFO_STACK
00181 #define CYTHON_USE_EXC_INFO_STACK 0
00182 #else
00183     #define CYTHON_COMPILING_IN_PYPY 0
00184     #define CYTHON_COMPILING_IN_PYSTON 0
00185     #define CYTHON_COMPILING_IN_CPYTHON 1
00186     #define CYTHON_COMPILING_IN_NOGIL 0
00187     #ifndef CYTHON_USE_TYPE_SLOTS
00188         #define CYTHON_USE_TYPE_SLOTS 1
00189     #endif
00190     #if PY_VERSION_HEX < 0x02070000
00191         #undef CYTHON_USE_PYTYPE_LOOKUP
00192         #define CYTHON_USE_PYTYPE_LOOKUP 0
00193     #elif !defined(CYTHON_USE_PYTYPE_LOOKUP)
00194         #define CYTHON_USE_PYTYPE_LOOKUP 1
00195     #endif
00196     #if PY_MAJOR_VERSION < 3
00197         #undef CYTHON_USE_ASYNC_SLOTS
00198         #define CYTHON_USE_ASYNC_SLOTS 0
00199     #elif !defined(CYTHON_USE_ASYNC_SLOTS)
00200         #define CYTHON_USE_ASYNC_SLOTS 1
00201     #endif
00202     #if PY_VERSION_HEX < 0x02070000
00203         #undef CYTHON_USE_PYLONG_INTERNALS
00204         #define CYTHON_USE_PYLONG_INTERNALS 0
00205     #elif !defined(CYTHON_USE_PYLONG_INTERNALS)
00206         #define CYTHON_USE_PYLONG_INTERNALS 1
00207     #endif
00208     #ifndef CYTHON_USE_PYLIST_INTERNALS
00209         #define CYTHON_USE_PYLIST_INTERNALS 1
00210     #endif
00211     #ifndef CYTHON_USE_UNICODE_INTERNALS
00212         #define CYTHON_USE_UNICODE_INTERNALS 1
00213     #endif
00214     #if PY_VERSION_HEX < 0x030300F0 || PY_VERSION_HEX >= 0x030B00A2
00215         #undef CYTHON_USE_UNICODE_WRITER
00216         #define CYTHON_USE_UNICODE_WRITER 0
00217     #elif !defined(CYTHON_USE_UNICODE_WRITER)
00218         #define CYTHON_USE_UNICODE_WRITER 1
00219     #endif
00220     #ifndef CYTHON_AVOID_BORROWED_REFS
00221         #define CYTHON_AVOID_BORROWED_REFS 0
00222     #endif
00223     #ifndef CYTHON_ASSUME_SAFE_MACROS
00224         #define CYTHON_ASSUME_SAFE_MACROS 1
00225     #endif

```

```

00226 #ifndef CYTHON_UNPACK_METHODS
00227     #define CYTHON_UNPACK_METHODS 1
00228 #endif
00229 #if PY_VERSION_HEX >= 0x030B00A4
00230     #undef CYTHON_FAST_THREAD_STATE
00231     #define CYTHON_FAST_THREAD_STATE 0
00232     #elif !defined(CYTHON_FAST_THREAD_STATE)
00233         #define CYTHON_FAST_THREAD_STATE 1
00234     #endif
00235 #ifndef CYTHON_FAST_PYCALL
00236     #define CYTHON_FAST_PYCALL (PY_VERSION_HEX < 0x030A0000)
00237 #endif
00238 #ifndef CYTHON_PEP489_MULTI_PHASE_INIT
00239     #define CYTHON_PEP489_MULTI_PHASE_INIT (PY_VERSION_HEX >= 0x03050000)
00240 #endif
00241 #ifndef CYTHON_USE_TP_FINALIZE
00242     #define CYTHON_USE_TP_FINALIZE (PY_VERSION_HEX >= 0x030400a1)
00243 #endif
00244 #ifndef CYTHON_USE_DICT_VERSIONS
00245     #define CYTHON_USE_DICT_VERSIONS (PY_VERSION_HEX >= 0x030600B1)
00246 #endif
00247 #if PY_VERSION_HEX >= 0x030B00A4
00248     #undef CYTHON_USE_EXC_INFO_STACK
00249     #define CYTHON_USE_EXC_INFO_STACK 0
00250     #elif !defined(CYTHON_USE_EXC_INFO_STACK)
00251         #define CYTHON_USE_EXC_INFO_STACK (PY_VERSION_HEX >= 0x030700A3)
00252     #endif
00253 #ifndef CYTHON_UPDATE_DESCRIPTOR_DOC
00254     #define CYTHON_UPDATE_DESCRIPTOR_DOC 1
00255 #endif
00256 #endif
00257 #if !defined(CYTHON_FAST_PYCCALL)
00258     #define CYTHON_FAST_PYCCALL (CYTHON_FAST_PYCALL && PY_VERSION_HEX >= 0x030600B1)
00259 #endif
00260 #if CYTHON_USE_PYLONG_INTERNALS
00261     #if PY_MAJOR_VERSION < 3
00262         #include "longintrepr.h"
00263     #endif
00264     #undef SHIFT
00265     #undef BASE
00266     #undef MASK
00267     #ifdef SIZEOF_VOID_P
00268         enum { __pyx_check_sizeof_voidp = 1 / (int) (SIZEOF_VOID_P == sizeof(void*)) };
00269     #endif
00270 #endif
00271 #ifndef __has_attribute
00272     #define __has_attribute(x) 0
00273 #endif
00274 #ifndef __has_cpp_attribute
00275     #define __has_cpp_attribute(x) 0
00276 #endif
00277 #ifndef CYTHON_RESTRICT
00278     #if defined(__GNUC__)
00279         #define CYTHON_RESTRICT __restrict
00280     #elif defined(_MSC_VER) && _MSC_VER >= 1400
00281         #define CYTHON_RESTRICT __restrict
00282     #elif defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
00283         #define CYTHON_RESTRICT restrict
00284     #else
00285         #define CYTHON_RESTRICT
00286     #endif
00287 #endif
00288 #ifndef CYTHON_UNUSED
00289     #if defined(__GNUC__)
00290         #if !(defined(__cplusplus)) || (__GNUC__ > 3 || (__GNUC__ == 3 && __GNUC_MINOR__
00291 >= 4))
00292             #define CYTHON_UNUSED __attribute__((__unused__))
00293         #else
00294             #define CYTHON_UNUSED
00295         #endif
00296     #elif defined(__ICC) || (defined(__INTEL_COMPILER) && !defined(_MSC_VER))
00297         #define CYTHON_UNUSED __attribute__((__unused__))
00298     #else
00299         #define CYTHON_UNUSED
00300     #endif
00301 #ifndef CYTHON_MAYBE_UNUSED_VAR

```

```

00302 # if defined(__cplusplus)
00303     template<class T> void CYTHON_MAYBE_UNUSED_VAR( const T& ) { }
00304 # else
00305 #     define CYTHON_MAYBE_UNUSED_VAR(x) (void)(x)
00306 # endif
00307 #endif
00308 #ifndef CYTHON_NCP_UNUSED
00309 # if CYTHON_COMPILING_IN_CPYTHON
00310 #     define CYTHON_NCP_UNUSED
00311 # else
00312 #     define CYTHON_NCP_UNUSED CYTHON_UNUSED
00313 # endif
00314 #endif
00315 #define __Pyx_void_to_None(void_result) ((void)(void_result), Py_INCREF(Py_None),
Py_None)
00316 #ifdef MSC_VER
00317     #ifndef _MSC_STDINT_H_
00318         #if _MSC_VER < 1300
00319             typedef unsigned char    uint8_t;
00320             typedef unsigned int      uint32_t;
00321         #else
00322             typedef unsigned    int8    uint8_t;
00323             typedef unsigned    __int32 uint32_t;
00324         #endif
00325     #endif
00326 #else
00327     #include <stdint.h>
00328 #endif
00329 #ifndef CYTHON_FALLTHROUGH
00330     #if defined(__cplusplus) && __cplusplus >= 201103L
00331         #if __has_cpp_attribute(fallthrough)
00332             #define CYTHON_FALLTHROUGH [[fallthrough]]
00333         #elif __has_cpp_attribute(clang::fallthrough)
00334             #define CYTHON_FALLTHROUGH [[clang::fallthrough]]
00335         #elif __has_cpp_attribute(gnu::fallthrough)
00336             #define CYTHON_FALLTHROUGH [[gnu::fallthrough]]
00337         #endif
00338     #endif
00339     #ifndef CYTHON_FALLTHROUGH
00340         #if __has_attribute(fallthrough)
00341             #define CYTHON_FALLTHROUGH __attribute__((fallthrough))
00342         #else
00343             #define CYTHON_FALLTHROUGH
00344         #endif
00345     #endif
00346     #if defined(__clang__) && defined(__apple_build_version__)
00347         #if __apple_build_version__ < 7000000
00348             #undef CYTHON_FALLTHROUGH
00349             #define CYTHON_FALLTHROUGH
00350         #endif
00351     #endif
00352 #endif
00353
00354 #ifndef CYTHON_INLINE
00355     #if defined(__clang__)
00356         #define CYTHON_INLINE __inline__ __attribute__((__unused__))
00357     #elif defined(__GNUC__)
00358         #define CYTHON_INLINE __inline__
00359     #elif defined(_MSC_VER)
00360         #define CYTHON_INLINE __inline
00361     #elif defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
00362         #define CYTHON_INLINE inline
00363     #else
00364         #define CYTHON_INLINE
00365     #endif
00366 #endif
00367
00368 #if CYTHON_COMPILING_IN_PYPY && PY_VERSION_HEX < 0x02070600 &&
!defined(Py_OptimizeFlag)
00369     #define Py_OptimizeFlag 0
00370 #endif
00371 #define __PYX_BUILD_PY_SSIZE_T "n"
00372 #define CYTHON_FORMAT_SSIZE_T "z"
00373 #if PY_MAJOR_VERSION < 3
00374     #define __Pyx_BUILTIN_MODULE_NAME "__builtin__"
00375     #define __Pyx_PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline,
lnos)\

```

```

00376         PyCode_New(a+k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)
00377     #define __Pyx_DefaultClassType PyClass_Type
00378 #else
00379     #define __Pyx_BUILTIN_MODULE_NAME "builtins"
00380     #define __Pyx_DefaultClassType PyType_Type
00381 #if PY_VERSION_HEX >= 0x030B00A1
00382     static CYTHON_INLINE PyCodeObject* __Pyx_PyCode_New(int a, int k, int l, int s,
00383 int f,
00384                                     PyObject *code, PyObject *c,
00385 PyObject* n, PyObject *v,
00386                                     PyObject *fv, PyObject *cell,
00387 PyObject* fn,
00388                                     PyObject *name, int fline,
00389 PyObject *lnos) {
00390     PyObject *kwds=NULL, *argcount=NULL, *posonlyargcount=NULL,
00391 *kwnonlyargcount=NULL;
00392     PyObject *nlocals=NULL, *stacksize=NULL, *flags=NULL, *replace=NULL,
00393 *call_result=NULL, *empty=NULL;
00394     const char *fn_cstr=NULL;
00395     const char *name_cstr=NULL;
00396     PyCodeObject* co=NULL;
00397     PyObject *type, *value, *traceback;
00398     PyErr_Fetch(&type, &value, &traceback);
00399     if (!(kwds=PyDict_New())) goto end;
00400     if (!(argcount=PyLong_FromLong(a))) goto end;
00401     if (PyDict_SetItemString(kwds, "co_argcount", argcount) != 0) goto end;
00402     if (!(posonlyargcount=PyLong_FromLong(0))) goto end;
00403     if (PyDict_SetItemString(kwds, "co_posonlyargcount", posonlyargcount) !=
00404 0) goto end;
00405     if (!(kwnonlyargcount=PyLong_FromLong(k))) goto end;
00406     if (PyDict_SetItemString(kwds, "co_kwnonlyargcount", kwnonlyargcount) != 0)
00407 goto end;
00408     if (!(nlocals=PyLong_FromLong(l))) goto end;
00409     if (PyDict_SetItemString(kwds, "co_nlocals", nlocals) != 0) goto end;
00410     if (!(stacksize=PyLong_FromLong(s))) goto end;
00411     if (PyDict_SetItemString(kwds, "co_stacksize", stacksize) != 0) goto end;
00412     if (!(flags=PyLong_FromLong(f))) goto end;
00413     if (PyDict_SetItemString(kwds, "co_flags", flags) != 0) goto end;
00414     if (PyDict_SetItemString(kwds, "co_code", code) != 0) goto end;
00415     if (PyDict_SetItemString(kwds, "co_consts", c) != 0) goto end;
00416     if (PyDict_SetItemString(kwds, "co_names", n) != 0) goto end;
00417     if (PyDict_SetItemString(kwds, "co_varnames", v) != 0) goto end;
00418     if (PyDict_SetItemString(kwds, "co_freevars", fv) != 0) goto end;
00419     if (PyDict_SetItemString(kwds, "co_cellvars", cell) != 0) goto end;
00420     if (PyDict_SetItemString(kwds, "co_linetable", lnos) != 0) goto end;
00421     if (!(fn_cstr=PyUnicode_AsUTF8AndSize(fn, NULL))) goto end;
00422     if (!(name_cstr=PyUnicode_AsUTF8AndSize(name, NULL))) goto end;
00423     if (!(co = PyCode_NewEmpty(fn_cstr, name_cstr, fline))) goto end;
00424     if (!(replace = PyObject_GetAttrString((PyObject*)co, "replace"))) goto
00425 cleanup_code_too;
00426     if (!(empty = PyTuple_New(0))) goto cleanup_code_too; // unfortunately
00427 __pyx_empty_tuple isn't available here
00428     if (!(call_result = PyObject_Call(replace, empty, kwds))) goto
00429 cleanup_code_too;
00430     Py_XDECREF((PyObject*)co);
00431     co = (PyCodeObject*)call_result;
00432     call_result = NULL;
00433     if (0) {
00434         cleanup_code_too:
00435         Py_XDECREF((PyObject*)co);
00436         co = NULL;
00437     }
00438     end:
00439     Py_XDECREF(kwds);
00440     Py_XDECREF(argcount);
00441     Py_XDECREF(posonlyargcount);
00442     Py_XDECREF(kwnonlyargcount);
00443     Py_XDECREF(nlocals);
00444     Py_XDECREF(stacksize);
00445     Py_XDECREF(replace);
00446     Py_XDECREF(call_result);
00447     Py_XDECREF(empty);
00448     if (type) {
00449         PyErr_Restore(type, value, traceback);
00450     }
00451     return co;
00452 }

```

```

00442 #else
00443     #define __Pyx_PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline,
lnos)\
00444         PyCode_New(a, k, l, s, f, code, c, n, v, fv, cell, fn, name, fline, lnos)
00445 #endif
00446     #define __Pyx_DefaultClassType PyType_Type
00447 #endif
00448 #ifndef Py_TPFLAGS_CHECKTYPES
00449     #define Py_TPFLAGS_CHECKTYPES 0
00450 #endif
00451 #ifndef Py_TPFLAGS_HAVE_INDEX
00452     #define Py_TPFLAGS_HAVE_INDEX 0
00453 #endif
00454 #ifndef Py_TPFLAGS_HAVE_NEWBUFFER
00455     #define Py_TPFLAGS_HAVE_NEWBUFFER 0
00456 #endif
00457 #ifndef Py_TPFLAGS_HAVE_FINALIZE
00458     #define Py_TPFLAGS_HAVE_FINALIZE 0
00459 #endif
00460 #ifndef METH_STACKLESS
00461     #define METH_STACKLESS 0
00462 #endif
00463 #if PY_VERSION_HEX <= 0x030700A3 || !defined(METH_FASTCALL)
00464     #ifndef METH_FASTCALL
00465         #define METH_FASTCALL 0x80
00466     #endif
00467     typedef PyObject *(*__Pyx_PyCFunctionFast) (PyObject *self, PyObject *const
*args, Py_ssize_t nargs);
00468     typedef PyObject *(*__Pyx_PyCFunctionFastWithKeywords) (PyObject *self, PyObject
*const *args,
00469                                                             Py_ssize_t nargs,
PyObject *kwnames);
00470 #else
00471     #define __Pyx_PyCFunctionFast_PyCFunctionFast
00472     #define __Pyx_PyCFunctionFastWithKeywords_PyCFunctionFastWithKeywords
00473 #endif
00474 #if CYTHON_FAST_PYCCALL
00475     #define __Pyx_PyFastCFunction_Check(func)\
00476         ((PyCFunction_Check(func) && (METH_FASTCALL == (PyCFunction_GET_FLAGS(func) &
~(METH_CLASS | METH_STATIC | METH_COEXIST | METH_KEYWORDS | METH_STACKLESS))))
00477 #else
00478     #define __Pyx_PyFastCFunction_Check(func) 0
00479 #endif
00480 #if CYTHON_COMPILING_IN_PYPY && !defined(PyObject_Malloc)
00481     #define PyObject_Malloc(s)    PyMem_Malloc(s)
00482     #define PyObject_Free(p)      PyMem_Free(p)
00483     #define PyObject_Realloc(p)   PyMem_Realloc(p)
00484 #endif
00485 #if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX < 0x030400A1
00486     #define PyMem_RawMalloc(n)    PyMem_Malloc(n)
00487     #define PyMem_RawRealloc(p, n) PyMem_Realloc(p, n)
00488     #define PyMem_RawFree(p)      PyMem_Free(p)
00489 #endif
00490 #if CYTHON_COMPILING_IN_PYSTON
00491     #define __Pyx_PyCode_HasFreeVars(co) PyCode_HasFreeVars(co)
00492     #define __Pyx_PyFrame_SetLineNumber(frame, lineno) PyFrame_SetLineNumber(frame,
lineno)
00493 #else
00494     #define __Pyx_PyCode_HasFreeVars(co) (PyCode_GetNumFree(co) > 0)
00495     #define __Pyx_PyFrame_SetLineNumber(frame, lineno) (frame)->f_lineno = (lineno)
00496 #endif
00497 #if !CYTHON_FAST_THREAD_STATE || PY_VERSION_HEX < 0x02070000
00498     #define __Pyx_PyThreadState_Current PyThreadState_GET()
00499 #elif PY_VERSION_HEX >= 0x03060000
00500     #define __Pyx_PyThreadState_Current _PyThreadState_UncheckedGet()
00501 #elif PY_VERSION_HEX >= 0x03000000
00502     #define __Pyx_PyThreadState_Current PyThreadState_GET()
00503 #else
00504     #define __Pyx_PyThreadState_Current _PyThreadState_Current
00505 #endif
00506 #if PY_VERSION_HEX < 0x030700A2 && !defined(PyThread_tss_create) &&
!defined(Py_tss_NEEDS_INIT)
00507 #include "pythread.h"
00508 #define Py_tss_NEEDS_INIT 0
00509 typedef int Py_tss_t;
00510 static CYTHON_INLINE int PyThread_tss_create(Py_tss_t *key) {
00511     *key = PyThread_create_key();

```

```

00512     return 0;
00513 }
00514 static CYTHON_INLINE Py_tss_t * PyThread_tss_alloc(void) {
00515     Py_tss_t *key = (Py_tss_t *)PyObject_Malloc(sizeof(Py_tss_t));
00516     *key = Py_tss_NEEDS_INIT;
00517     return key;
00518 }
00519 static CYTHON_INLINE void PyThread_tss_free(Py_tss_t *key) {
00520     PyObject_Free(key);
00521 }
00522 static CYTHON_INLINE int PyThread_tss_is_created(Py_tss_t *key) {
00523     return *key != Py_tss_NEEDS_INIT;
00524 }
00525 static CYTHON_INLINE void PyThread_tss_delete(Py_tss_t *key) {
00526     PyThread_delete_key(*key);
00527     *key = Py_tss_NEEDS_INIT;
00528 }
00529 static CYTHON_INLINE int PyThread_tss_set(Py_tss_t *key, void *value) {
00530     return PyThread_set_key_value(*key, value);
00531 }
00532 static CYTHON_INLINE void * PyThread_tss_get(Py_tss_t *key) {
00533     return PyThread_get_key_value(*key);
00534 }
00535 #endif
00536 #if CYTHON_COMPILING_IN_CPYTHON || defined(_PyDict_NewPresized)
00537 #define __Pyx_PyDict_NewPresized(n)  ((n <= 8) ? PyDict_New() :
_PyDict_NewPresized(n))
00538 #else
00539 #define __Pyx_PyDict_NewPresized(n)  PyDict_New()
00540 #endif
00541 #if PY_MAJOR_VERSION >= 3 || CYTHON_FUTURE_DIVISION
00542     #define __Pyx_PyNumber_Divide(x,y)      PyNumber_TrueDivide(x,y)
00543     #define __Pyx_PyNumber_InPlaceDivide(x,y)  PyNumber_InPlaceTrueDivide(x,y)
00544 #else
00545     #define __Pyx_PyNumber_Divide(x,y)      PyNumber_Divide(x,y)
00546     #define __Pyx_PyNumber_InPlaceDivide(x,y)  PyNumber_InPlaceDivide(x,y)
00547 #endif
00548 #if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX >= 0x030500A1 &&
CYTHON_USE_UNICODE_INTERNALS
00549 #define __Pyx_PyDict_GetItemStr(dict, name)  _PyDict_GetItem_KnownHash(dict, name,
((PyASCIIObject *) name)->hash)
00550 #else
00551 #define __Pyx_PyDict_GetItemStr(dict, name)  PyDict_GetItem(dict, name)
00552 #endif
00553 #if PY_VERSION_HEX > 0x03030000 && defined(PyUnicode_KIND)
00554     #define CYTHON_PEP393_ENABLED 1
00555     #if defined(PyUnicode_IS_READY)
00556         #define __Pyx_PyUnicode_READY(op)           (likely(PyUnicode_IS_READY(op)) ?\
00557             0 : _PyUnicode_Ready((PyObject)
00558 *) (op)))
00559     #else
00560         #define __Pyx_PyUnicode_READY(op)           (0)
00561     #endif
00562     #define __Pyx_PyUnicode_GET_LENGTH(u)           PyUnicode_GET_LENGTH(u)
00563     #define __Pyx_PyUnicode_READ_CHAR(u, i)         PyUnicode_READ_CHAR(u, i)
00564     #define __Pyx_PyUnicode_MAX_CHAR_VALUE(u)       PyUnicode_MAX_CHAR_VALUE(u)
00565     #define __Pyx_PyUnicode_KIND(u)                 PyUnicode_KIND(u)
00566     #define __Pyx_PyUnicode_DATA(u)                 PyUnicode_DATA(u)
00567     #define __Pyx_PyUnicode_READ(k, d, i)           PyUnicode_READ(k, d, i)
00568     #define __Pyx_PyUnicode_WRITE(k, d, i, ch)      PyUnicode_WRITE(k, d, i, ch)
00569     #if defined(PyUnicode_IS_READY) && defined(PyUnicode_GET_SIZE)
00570         #define __Pyx_PyUnicode_IS_TRUE(u)          (0 != (likely(PyUnicode_IS_READY(u)) ?
PyUnicode_GET_LENGTH(u) : (PyCompactUnicodeObject *) (u))->wstr_length))
00571     #else
00572         #define __Pyx_PyUnicode_IS_TRUE(u)          (0 != (likely(PyUnicode_IS_READY(u)) ?
PyUnicode_GET_LENGTH(u) : PyUnicode_GET_SIZE(u)))
00573     #endif
00574     #else
00575         #define __Pyx_PyUnicode_IS_TRUE(u)          (0 != PyUnicode_GET_LENGTH(u))
00576     #endif
00577 #else
00578     #define CYTHON_PEP393_ENABLED 0
00579     #define PyUnicode_1BYTE_KIND 1
00580     #define PyUnicode_2BYTE_KIND 2
00581     #define PyUnicode_4BYTE_KIND 4
00582     #define __Pyx_PyUnicode_READY(op)               (0)

```

```

00583 #define __Pyx_PyUnicode_GET_LENGTH(u) PyUnicode_GET_SIZE(u)
00584 #define __Pyx_PyUnicode_READ_CHAR(u, i) ((Py_UCS4)(PyUnicode_AS_UNICODE(u)[i]))
00585 #define __Pyx_PyUnicode_MAX_CHAR_VALUE(u) ((sizeof(Py_UNICODE) == 2) ? 65535 :
1114111)
00586 #define __Pyx_PyUnicode_KIND(u) (sizeof(Py_UNICODE))
00587 #define __Pyx_PyUnicode_DATA(u) ((void*)PyUnicode_AS_UNICODE(u))
00588 #define __Pyx_PyUnicode_READ(k, d, i) ((void)(k),
(Py_UCS4)((Py_UNICODE*)d)[i]))
00589 #define __Pyx_PyUnicode_WRITE(k, d, i, ch) (((void)(k)), ((Py_UNICODE*)d)[i] =
ch)
00590 #define __Pyx_PyUnicode_IS_TRUE(u) (0 != PyUnicode_GET_SIZE(u))
00591 #endif
00592 #if CYTHON_COMPILING_IN_PYPY
00593 #define __Pyx_PyUnicode_Concat(a, b) PyNumber_Add(a, b)
00594 #define __Pyx_PyUnicode_ConcatSafe(a, b) PyNumber_Add(a, b)
00595 #else
00596 #define __Pyx_PyUnicode_Concat(a, b) PyUnicode_Concat(a, b)
00597 #define __Pyx_PyUnicode_ConcatSafe(a, b) ((unlikely((a) == Py_None) ||
unlikely((b) == Py_None)) ?\
PyNumber_Add(a, b) : __Pyx_PyUnicode_Concat(a, b))
00598 #endif
00599 #endif
00600 #if CYTHON_COMPILING_IN_PYPY && !defined(PyUnicode_Contains)
00601 #define PyUnicode_Contains(u, s) PySequence_Contains(u, s)
00602 #endif
00603 #if CYTHON_COMPILING_IN_PYPY && !defined(PyByteArray_Check)
00604 #define PyByteArray_Check(obj) PyObject_TypeCheck(obj, &PyByteArray_Type)
00605 #endif
00606 #if CYTHON_COMPILING_IN_PYPY && !defined(PyObject_Format)
00607 #define PyObject_Format(obj, fmt) PyObject_CallMethod(obj, "__format__", "O",
fmt)
00608 #endif
00609 #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))
00610 #define __Pyx_PyUnicode_FormatSafe(a, b) ((unlikely((a) == Py_None ||
(PyUnicode_Check(b) && !PyUnicode_CheckExact(b)))) ? PyNumber_Remainder(a, b) :
PyUnicode_Format(a, b))
00611 #if PY_MAJOR_VERSION >= 3
00612 #define __Pyx_PyString_Format(a, b) PyUnicode_Format(a, b)
00613 #else
00614 #define __Pyx_PyString_Format(a, b) PyString_Format(a, b)
00615 #endif
00616 #if PY_MAJOR_VERSION < 3 && !defined(PyObject_ASCII)
00617 #define PyObject_ASCII(o) PyObject_Repr(o)
00618 #endif
00619 #if PY_MAJOR_VERSION >= 3
00620 #define PyBaseString_Type PyUnicode_Type
00621 #define PyStringObject PyUnicodeObject
00622 #define PyString_Type PyUnicode_Type
00623 #define PyString_Check PyUnicode_Check
00624 #define PyString_CheckExact PyUnicode_CheckExact
00625 #ifndef PyObject_Unicode
00626 #define PyObject_Unicode PyObject_Str
00627 #endif
00628 #endif
00629 #if PY_MAJOR_VERSION >= 3
00630 #define __Pyx_PyBaseString_Check(obj) PyUnicode_Check(obj)
00631 #define __Pyx_PyBaseString_CheckExact(obj) PyUnicode_CheckExact(obj)
00632 #else
00633 #define __Pyx_PyBaseString_Check(obj) (PyString_Check(obj) ||
PyUnicode_Check(obj))
00634 #define __Pyx_PyBaseString_CheckExact(obj) (PyString_CheckExact(obj) ||
PyUnicode_CheckExact(obj))
00635 #endif
00636 #ifndef PySet_CheckExact
00637 #define PySet_CheckExact(obj) ((Py_TYPE(obj) == &PySet_Type)
00638 #endif
00639 #if PY_VERSION_HEX >= 0x030900A4
00640 #define __Pyx_SET_REFCNT(obj, refcnt) Py_SET_REFCNT(obj, refcnt)
00641 #define __Pyx_SET_SIZE(obj, size) Py_SET_SIZE(obj, size)
00642 #else
00643 #define __Pyx_SET_REFCNT(obj, refcnt) Py_REFCNT(obj) = (refcnt)
00644 #define __Pyx_SET_SIZE(obj, size) Py_SIZE(obj) = (size)
00645 #endif
00646 #if CYTHON_ASSUME_SAFE_MACROS
00647 #define __Pyx_PySequence_SIZE(seq) Py_SIZE(seq)
00648 #else

```



```

00649 #define __Pyx_PySequence_SIZE(seq) PySequence_Size(seq)
00650 #endif
00651 #if PY_MAJOR_VERSION >= 3
00652     #define PyIntObject PyLongObject
00653     #define PyInt_Type PyLong_Type
00654     #define PyInt_Check(op) PyLong_Check(op)
00655     #define PyInt_CheckExact(op) PyLong_CheckExact(op)
00656     #define PyInt_FromString PyLong_FromString
00657     #define PyInt_FromUnicode PyLong_FromUnicode
00658     #define PyInt_FromLong PyLong_FromLong
00659     #define PyInt_FromSize_t PyLong_FromSize_t
00660     #define PyInt_FromSsize_t PyLong_FromSsize_t
00661     #define PyInt_AsLong PyLong_AsLong
00662     #define PyInt_AS_LONG PyLong_AS_LONG
00663     #define PyInt_AsSsize_t PyLong_AsSsize_t
00664     #define PyInt_AsUnsignedLongMask PyLong_AsUnsignedLongMask
00665     #define PyInt_AsUnsignedLongLongMask PyLong_AsUnsignedLongLongMask
00666     #define PyNumber_Int PyNumber_Long
00667 #endif
00668 #if PY_MAJOR_VERSION >= 3
00669     #define PyBoolObject PyLongObject
00670 #endif
00671 #if PY_MAJOR_VERSION >= 3 && CYTHON_COMPILING_IN_PYPY
00672     #ifndef PyUnicode_InternFromString
00673         #define PyUnicode_InternFromString(s) PyUnicode_FromString(s)
00674     #endif
00675 #endif
00676 #if PY_VERSION_HEX < 0x030200A4
00677     typedef long Py_hash_t;
00678     #define __Pyx_PyInt_FromHash_t PyInt_FromLong
00679     #define __Pyx_PyInt_AsHash_t __Pyx_PyIndex_AsHash_t
00680 #else
00681     #define __Pyx_PyInt_FromHash_t PyInt_FromSsize_t
00682     #define __Pyx_PyInt_AsHash_t __Pyx_PyIndex_AsSsize_t
00683 #endif
00684 #if PY_MAJOR_VERSION >= 3
00685     #define __Pyx_PyMethod_New(func, self, klass) ((self) ? ((void)(klass),
PyMethod_New(func, self)) : __Pyx_NewRef(func))
00686 #else
00687     #define __Pyx_PyMethod_New(func, self, klass) PyMethod_New(func, self, klass)
00688 #endif
00689 #if CYTHON_USE_ASYNC_SLOTS
00690     #if PY_VERSION_HEX >= 0x030500B1
00691         #define __Pyx_PyAsyncMethodsStruct PyAsyncMethods
00692         #define __Pyx_PyType_AsAsync(obj) (Py_TYPE(obj)->tp_as_async)
00693     #else
00694         #define __Pyx_PyType_AsAsync(obj) ((__Pyx_PyAsyncMethodsStruct*)
(Py_TYPE(obj)->tp_reserved))
00695     #endif
00696 #else
00697     #define __Pyx_PyType_AsAsync(obj) NULL
00698 #endif
00699 #ifndef __Pyx_PyAsyncMethodsStruct
00700     typedef struct {
00701         unaryfunc am_await;
00702         unaryfunc am_aiter;
00703         unaryfunc am_anext;
00704     } __Pyx_PyAsyncMethodsStruct;
00705 #endif
00706
00707 #if defined(_WIN32) || defined(WIN32) || defined(MS_WINDOWS)
00708     #if !defined(_USE_MATH_DEFINES)
00709         #define _USE_MATH_DEFINES
00710     #endif
00711 #endif
00712 #include <math.h>
00713 #ifndef NAN
00714     #define __PYX_NAN() ((float) NAN)
00715 #else
00716     static CYTHON_INLINE float __PYX_NAN() {
00717         float value;
00718         memset(&value, 0xFF, sizeof(value));
00719         return value;
00720     }
00721 #endif
00722 #if defined(__CYGWIN__) && defined(_LDBL_EQ_DBL)
00723     #define __Pyx_trunc1 trunc

```



```

00724 #else
00725 #define __Pyx_truncl truncl
00726 #endif
00727
00728 #define __PYX_MARK_ERR_POS(f_index, lineno) \
00729 { __pyx_filename = __pyx_f[f_index]; (void)__pyx_filename; __pyx_lineno =
lineno; (void)__pyx_lineno; __pyx_clineno = __LINE__; (void)__pyx_clineno; }
00730 #define __PYX_ERR(f_index, lineno, Ln_error) \
00731 { __PYX_MARK_ERR_POS(f_index, lineno) goto Ln_error; }
00732
00733 #ifndef __PYX_EXTERN_C
00734 #ifdef __cplusplus
00735 #define __PYX_EXTERN_C extern "C"
00736 #else
00737 #define __PYX_EXTERN_C extern
00738 #endif
00739 #endif
00740
00741 #define __PYX_HAVE_API_Calls__main__
00742 #define __PYX_HAVE_API__API_Calls__main__
00743 /* Early includes */
00744 #ifdef OPENMP
00745 #include <omp.h>
00746 #endif /* _OPENMP */
00747
00748 #if defined(PYREX_WITHOUT_ASSERTIONS) && !defined(CYTHON_WITHOUT_ASSERTIONS)
00749 #define CYTHON_WITHOUT_ASSERTIONS
00750 #endif
00751
00752 typedef struct {PyObject **p; const char *s; const Py_ssize_t n; const char* encoding;
00753               const char is_unicode; const char is_str; const char intern; }
__Pyx_StringTabEntry;
00754
00755 #define __PYX_DEFAULT_STRING_ENCODING_IS_ASCII 0
00756 #define __PYX_DEFAULT_STRING_ENCODING_IS_UTF8 0
00757 #define __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT (PY_MAJOR_VERSION >= 3 &&
__PYX_DEFAULT_STRING_ENCODING_IS_UTF8)
00758 #define __PYX_DEFAULT_STRING_ENCODING ""
00759 #define __Pyx_PyObject_FromString __Pyx_PyBytes_FromString
00760 #define __Pyx_PyObject_FromStringAndSize __Pyx_PyBytes_FromStringAndSize
00761 #define __Pyx_uchar_cast(c) ((unsigned char)c)
00762 #define __Pyx_long_cast(x) ((long)x)
00763 #define __Pyx_fits_Py_ssize_t(v, type, is_signed) (\
00764     (sizeof(type) < sizeof(Py_ssize_t)) ||\
00765     (sizeof(type) > sizeof(Py_ssize_t) &&\
00766      likely(v < (type)PY_SSIZE_T_MAX ||\
00767            v == (type)PY_SSIZE_T_MAX) &&\
00768     (!is_signed || likely(v > (type)PY_SSIZE_T_MIN ||\
00769                          v == (type)PY_SSIZE_T_MIN))) ||\
00770     (sizeof(type) == sizeof(Py_ssize_t) &&\
00771      (is_signed || likely(v < (type)PY_SSIZE_T_MAX ||\
00772                          v == (type)PY_SSIZE_T_MAX)))) )
00773 static CYTHON_INLINE int __Pyx_is_valid_index(Py_ssize_t i, Py_ssize_t limit) {
00774     return (size_t) i < (size_t) limit;
00775 }
00776 #if defined (__cplusplus) && __cplusplus >= 201103L
00777 #include <cstdlib>
00778 #define __Pyx_sst_abs(value) std::abs(value)
00779 #elif SIZEOF_INT >= SIZEOF_SIZE_T
00780 #define __Pyx_sst_abs(value) abs(value)
00781 #elif SIZEOF_LONG >= SIZEOF_SIZE_T
00782 #define __Pyx_sst_abs(value) labs(value)
00783 #elif defined (_MSC_VER)
00784 #define __Pyx_sst_abs(value) ((Py_ssize_t) _abs64(value))
00785 #elif defined (__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
00786 #define __Pyx_sst_abs(value) llabs(value)
00787 #elif defined (__GNUC__)
00788 #define __Pyx_sst_abs(value) __builtin_llabs(value)
00789 #else
00790 #define __Pyx_sst_abs(value) ((value<0) ? -value : value)
00791 #endif
00792 static CYTHON_INLINE const char* __Pyx_PyObject_AsString(PyObject*);
00793 static CYTHON_INLINE const char* __Pyx_PyObject_AsStringAndSize(PyObject*,
Py_ssize_t* length);
00794 #define __Pyx_PyByteArray_FromString(s) PyByteArray_FromStringAndSize((const
char*)s, strlen((const char*)s))

```

```

00795 #define __Pyx_PyByteArray_FromStringAndSize(s, l)
PyByteArray_FromStringAndSize((const char*)s, l)
00796 #define __Pyx_PyBytes_FromString PyBytes_FromString
00797 #define __Pyx_PyBytes_FromStringAndSize PyBytes_FromStringAndSize
00798 static CYTHON_INLINE PyObject* __Pyx_PyUnicode_FromString(const char*);
00799 #if PY_MAJOR_VERSION < 3
00800 #define __Pyx_PyStr_FromString __Pyx_PyBytes_FromString
00801 #define __Pyx_PyStr_FromStringAndSize __Pyx_PyBytes_FromStringAndSize
00802 #else
00803 #define __Pyx_PyStr_FromString __Pyx_PyUnicode_FromString
00804 #define __Pyx_PyStr_FromStringAndSize __Pyx_PyUnicode_FromStringAndSize
00805 #endif
00806 #define __Pyx_PyBytes_AsWritableString(s) ((char*) PyBytes_AS_STRING(s))
00807 #define __Pyx_PyBytes_AsWritableSString(s) ((signed char*)
PyBytes_AS_STRING(s))
00808 #define __Pyx_PyBytes_AsWritableUString(s) ((unsigned char*)
PyBytes_AS_STRING(s))
00809 #define __Pyx_PyBytes_AsString(s) ((const char*) PyBytes_AS_STRING(s))
00810 #define __Pyx_PyBytes_AsSString(s) ((const signed char*) PyBytes_AS_STRING(s))
00811 #define __Pyx_PyBytes_AsUString(s) ((const unsigned char*)
PyBytes_AS_STRING(s))
00812 #define __Pyx_PyObject_AsWritableString(s) ((char*)
__Pyx_PyObject_AsString(s))
00813 #define __Pyx_PyObject_AsWritableSString(s) ((signed char*)
__Pyx_PyObject_AsString(s))
00814 #define __Pyx_PyObject_AsWritableUString(s) ((unsigned char*)
__Pyx_PyObject_AsString(s))
00815 #define __Pyx_PyObject_AsSString(s) ((const signed char*)
__Pyx_PyObject_AsString(s))
00816 #define __Pyx_PyObject_AsUString(s) ((const unsigned char*)
__Pyx_PyObject_AsString(s))
00817 #define __Pyx_PyObject_FromCString(s) __Pyx_PyObject_FromString((const char*)s)
00818 #define __Pyx_PyBytes_FromCString(s) __Pyx_PyBytes_FromString((const char*)s)
00819 #define __Pyx_PyByteArray_FromCString(s) __Pyx_PyByteArray_FromString((const
char*)s)
00820 #define __Pyx_PyStr_FromCString(s) __Pyx_PyStr_FromString((const char*)s)
00821 #define __Pyx_PyUnicode_FromCString(s) __Pyx_PyUnicode_FromString((const char*)s)
00822 static CYTHON_INLINE size_t __Pyx_Py_UNICODE_strlen(const Py_UNICODE *u) {
00823     const Py_UNICODE *u_end = u;
00824     while (*u_end++) ;
00825     return (size_t)(u_end - u - 1);
00826 }
00827 #define __Pyx_PyUnicode_FromUnicode(u) PyUnicode_FromUnicode(u,
__Pyx_Py_UNICODE_strlen(u))
00828 #define __Pyx_PyUnicode_FromUnicodeAndLength PyUnicode_FromUnicode
00829 #define __Pyx_PyUnicode_AsUnicode PyUnicode_AsUnicode
00830 #define __Pyx_NewRef(obj) (Py_INCREF(obj), obj)
00831 #define __Pyx_Owned_Py_None(b) __Pyx_NewRef(Py_None)
00832 static CYTHON_INLINE PyObject * __Pyx_PyBool_FromLong(long b);
00833 static CYTHON_INLINE int __Pyx_PyObject_IsTrue(PyObject*);
00834 static CYTHON_INLINE int __Pyx_PyObject_IsTrueAndDecref(PyObject*);
00835 static CYTHON_INLINE PyObject* __Pyx_PyNumber_IntOrLong(PyObject* x);
00836 #define __Pyx_PySequence_Tuple(obj)\
00837     (likely(PyTuple_CheckExact(obj)) ? __Pyx_NewRef(obj) : PySequence_Tuple(obj))
00838 static CYTHON_INLINE Py_ssize_t __Pyx_PyIndex_AsSsize_t(PyObject*);
00839 static CYTHON_INLINE PyObject * __Pyx_PyInt_FromSize_t(size_t);
00840 static CYTHON_INLINE Py_hash_t __Pyx_PyIndex_AsHash_t(PyObject*);
00841 #if CYTHON_ASSUME_SAFE_MACROS
00842 #define __pyx_PyFloat_AsDouble(x) (PyFloat_CheckExact(x) ? PyFloat_AS_DOUBLE(x) :
PyFloat_AsDouble(x))
00843 #else
00844 #define __pyx_PyFloat_AsDouble(x) PyFloat_AsDouble(x)
00845 #endif
00846 #define __pyx_PyFloat_AsFloat(x) ((float) __pyx_PyFloat_AsDouble(x))
00847 #if PY_MAJOR_VERSION >= 3
00848 #define __Pyx_PyNumber_Int(x) (PyLong_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Long(x))
00849 #else
00850 #define __Pyx_PyNumber_Int(x) (PyInt_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Int(x))
00851 #endif
00852 #define __Pyx_PyNumber_Float(x) (PyFloat_CheckExact(x) ? __Pyx_NewRef(x) :
PyNumber_Float(x))
00853 #if PY_MAJOR_VERSION < 3 && __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
00854 static int __Pyx_sys_getdefaultencoding_not_ascii;
00855 static int __Pyx_init_sys_getdefaultencoding_params(void) {
00856     PyObject* sys;

```

```

00857 PyObject* default_encoding = NULL;
00858 PyObject* ascii_chars_u = NULL;
00859 PyObject* ascii_chars_b = NULL;
00860 const char* default_encoding_c;
00861 sys = PyImport_ImportModule("sys");
00862 if (!sys) goto bad;
00863 default_encoding = PyObject_CallMethod(sys, (char*) "getdefaultencoding",
NULL);
00864 Py_DECREF(sys);
00865 if (!default_encoding) goto bad;
00866 default_encoding_c = PyBytes_AsString(default_encoding);
00867 if (!default_encoding_c) goto bad;
00868 if (strcmp(default_encoding_c, "ascii") == 0) {
00869     __Pyx_sys_getdefaultencoding_not_ascii = 0;
00870 } else {
00871     char ascii_chars[128];
00872     int c;
00873     for (c = 0; c < 128; c++) {
00874         ascii_chars[c] = c;
00875     }
00876     __Pyx_sys_getdefaultencoding_not_ascii = 1;
00877     ascii_chars_u = PyUnicode_DecodeASCII(ascii_chars, 128, NULL);
00878     if (!ascii_chars_u) goto bad;
00879     ascii_chars_b = PyUnicode_AsEncodedString(ascii_chars_u,
default_encoding_c, NULL);
00880     if (!ascii_chars_b || !PyBytes_Check(ascii_chars_b) || memcmp(ascii_chars,
PyBytes_AS_STRING(ascii_chars_b), 128) != 0) {
00881         PyErr_Format(
00882             PyExc_ValueError,
00883             "This module compiled with c_string_encoding=ascii, but default
encoding '%.200s' is not a superset of ascii.",
00884             default_encoding_c);
00885         goto bad;
00886     }
00887     Py_DECREF(ascii_chars_u);
00888     Py_DECREF(ascii_chars_b);
00889 }
00890 Py_DECREF(default_encoding);
00891 return 0;
00892 bad:
00893 Py_XDECREF(default_encoding);
00894 Py_XDECREF(ascii_chars_u);
00895 Py_XDECREF(ascii_chars_b);
00896 return -1;
00897 }
00898 #endif
00899 #if __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT && PY_MAJOR_VERSION >= 3
00900 #define __Pyx_PyUnicode_FromStringAndSize(c_str, size) PyUnicode_DecodeUTF8(c_str,
size, NULL)
00901 #else
00902 #define __Pyx_PyUnicode_FromStringAndSize(c_str, size) PyUnicode_Decode(c_str,
size, __PYX_DEFAULT_STRING_ENCODING, NULL)
00903 #if __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
00904 static char* __PYX_DEFAULT_STRING_ENCODING;
00905 static int __Pyx_init_sys_getdefaultencoding_params(void) {
00906     PyObject* sys;
00907     PyObject* default_encoding = NULL;
00908     char* default_encoding_c;
00909     sys = PyImport_ImportModule("sys");
00910     if (!sys) goto bad;
00911     default_encoding = PyObject_CallMethod(sys, (char*) (const char*)
"getdefaultencoding", NULL);
00912     Py_DECREF(sys);
00913     if (!default_encoding) goto bad;
00914     default_encoding_c = PyBytes_AsString(default_encoding);
00915     if (!default_encoding_c) goto bad;
00916     __PYX_DEFAULT_STRING_ENCODING = (char*) malloc(strlen(default_encoding_c) +
1);
00917     if (!__PYX_DEFAULT_STRING_ENCODING) goto bad;
00918     strcpy(__PYX_DEFAULT_STRING_ENCODING, default_encoding_c);
00919     Py_DECREF(default_encoding);
00920     return 0;
00921 bad:
00922     Py_XDECREF(default_encoding);
00923     return -1;
00924 }
00925 #endif

```

```

00926 #endif
00927
00928
00929 /* Test for GCC > 2.95 */
00930 #if defined(__GNUC__)      && (__GNUC__ > 2 || (__GNUC__ == 2 && (__GNUC_MINOR__ >
95)))
00931     #define likely(x)      __builtin_expect(!!(x), 1)
00932     #define unlikely(x)    __builtin_expect(!!(x), 0)
00933 #else /* !__GNUC__ or GCC < 2.95 */
00934     #define likely(x)      (x)
00935     #define unlikely(x)    (x)
00936 #endif /* __GNUC__ */
00937 static CYTHON_INLINE void __Pyx_pretend_to_initialize(void* ptr) { (void)ptr; }
00938
00939 static PyObject* __pyx_m = NULL;
00940 static PyObject* __pyx_d;
00941 static PyObject* __pyx_b;
00942 static PyObject* __pyx_cython_runtime = NULL;
00943 static PyObject* __pyx_empty_tuple;
00944 static PyObject* __pyx_empty_bytes;
00945 static PyObject* __pyx_empty_unicode;
00946 static int __pyx_lineno;
00947 static int __pyx_clineno = 0;
00948 static const char* __pyx_filename = __FILE__;
00949 static const char* __pyx_filename;
00950
00951
00952 static const char* __pyx_f[] = {
00953     "_main.py",
00954 };
00955
00956 /*--- Type declarations ---*/
00957
00958 /* --- Runtime support code (head) --- */
00959 /* Refnanny.proto */
00960 #ifndef CYTHON_REFNANNY
00961     #define CYTHON_REFNANNY 0
00962 #endif
00963 #if CYTHON_REFNANNY
00964     typedef struct {
00965         void (*INCRREF)(void*, PyObject*, int);
00966         void (*DECREF)(void*, PyObject*, int);
00967         void (*GOTREF)(void*, PyObject*, int);
00968         void (*GIVEREF)(void*, PyObject*, int);
00969         void* (*SetupContext)(const char*, int, const char*);
00970         void (*FinishContext)(void**);
00971     } __Pyx_RefNannyAPIStruct;
00972     static __Pyx_RefNannyAPIStruct* __Pyx_RefNanny = NULL;
00973     static __Pyx_RefNannyAPIStruct* __Pyx_RefNannyImportAPI(const char* modname);
00974     #define __Pyx_RefNannyDeclarations void* __pyx_refnanny = NULL;
00975 #ifdef WITH_THREAD
00976     #define __Pyx_RefNannySetupContext(name, acquire_gil)\
00977         if (acquire_gil) {\
00978             PyGILState_STATE __pyx_gilstate_save = PyGILState_Ensure();\
00979             __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__,\
__FILE__);\
00980             PyGILState_Release(__pyx_gilstate_save);\
00981         } else {\
00982             __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__,\
__FILE__);\
00983         }\
00984 #else
00985     #define __Pyx_RefNannySetupContext(name, acquire_gil)\
00986         __pyx_refnanny = __Pyx_RefNanny->SetupContext((name), __LINE__,\
__FILE__)\
00987 #endif
00988     #define __Pyx_RefNannyFinishContext()\
00989         __Pyx_RefNanny->FinishContext(&__pyx_refnanny)
00990     #define __Pyx_INCREF(r)      __Pyx_RefNanny->INCRREF(__pyx_refnanny, (PyObject
*) (r), __LINE__)
00991     #define __Pyx_DECREF(r)      __Pyx_RefNanny->DECREF(__pyx_refnanny, (PyObject
*) (r), __LINE__)
00992     #define __Pyx_GOTREF(r)      __Pyx_RefNanny->GOTREF(__pyx_refnanny, (PyObject
*) (r), __LINE__)
00993     #define __Pyx_GIVEREF(r)     __Pyx_RefNanny->GIVEREF(__pyx_refnanny, (PyObject
*) (r), __LINE__)
00994     #define __Pyx_XINCRREF(r)    do { if((r) != NULL) {__Pyx_INCREF(r); }} while(0)

```

```

00995 #define __Pyx_XDECREF(r) do { if((r) != NULL) {__Pyx_DECREF(r); }} while(0)
00996 #define __Pyx_XGOTREF(r) do { if((r) != NULL) {__Pyx_GOTREF(r); }} while(0)
00997 #define __Pyx_XGIVEREF(r) do { if((r) != NULL) {__Pyx_GIVEREF(r);}} while(0)
00998 #else
00999 #define __Pyx_RefNannyDeclarations
01000 #define __Pyx_RefNannySetupContext(name, acquire_gil)
01001 #define __Pyx_RefNannyFinishContext()
01002 #define __Pyx_INCREF(r) Py_INCREF(r)
01003 #define __Pyx_DECREF(r) Py_DECREF(r)
01004 #define __Pyx_GOTREF(r)
01005 #define __Pyx_GIVEREF(r)
01006 #define __Pyx_XINCREASE(r) Py_XINCREASE(r)
01007 #define __Pyx_XDECREF(r) Py_XDECREF(r)
01008 #define __Pyx_XGOTREF(r)
01009 #define __Pyx_XGIVEREF(r)
01010 #endif
01011 #define __Pyx_XDECREF_SET(r, v) do {\
01012     PyObject *tmp = (PyObject *) r;\
01013     r = v; __Pyx_XDECREF(tmp);\
01014 } while (0)
01015 #define __Pyx_DECREF_SET(r, v) do {\
01016     PyObject *tmp = (PyObject *) r;\
01017     r = v; __Pyx_DECREF(tmp);\
01018 } while (0)
01019 #define __Pyx_CLEAR(r) do { PyObject* tmp = ((PyObject*)(r)); r = NULL;\
__Pyx_DECREF(tmp);} while(0)
01020 #define __Pyx_XCLEAR(r) do { if((r) != NULL) {PyObject* tmp = ((PyObject*)(r));\
r = NULL; __Pyx_DECREF(tmp);} } while(0)
01021
01022 /* PyObjectGetAttrStr.proto */
01023 #if CYTHON_USE_TYPE_SLOTS
01024 static CYTHON_INLINE PyObject* __Pyx_PyObject_GetAttrStr(PyObject* obj, PyObject*
attr_name);
01025 #else
01026 #define __Pyx_PyObject_GetAttrStr(o,n) PyObject_GetAttr(o,n)
01027 #endif
01028
01029 /* Import.proto */
01030 static PyObject* __Pyx_Import(PyObject* name, PyObject* from_list, int level);
01031
01032 /* ImportFrom.proto */
01033 static PyObject* __Pyx_ImportFrom(PyObject* module, PyObject* name);
01034
01035 /* GetBuiltinName.proto */
01036 static PyObject* __Pyx_GetBuiltinName(PyObject* name);
01037
01038 /* PyDictVersioning.proto */
01039 #if CYTHON_USE_DICT_VERSIONS && CYTHON_USE_TYPE_SLOTS
01040 #define __PYX_DICT_VERSION_INIT ((PY_UINT64_T) -1)
01041 #define __PYX_GET_DICT_VERSION(dict) (((PyDictObject*)(dict))->ma_version_tag)
01042 #define __PYX_UPDATE_DICT_CACHE(dict, value, cache_var, version_var)\
01043     (version_var) = __PYX_GET_DICT_VERSION(dict);\
01044     (cache_var) = (value);
01045 #define __PYX_PY_DICT_LOOKUP_IF_MODIFIED(VAR, DICT, LOOKUP) {\
01046     static PY_UINT64_T __pyx_dict_version = 0;\
01047     static PyObject* __pyx_dict_cached_value = NULL;\
01048     if (likely(__PYX_GET_DICT_VERSION(DICT) == __pyx_dict_version)) {\
01049         (VAR) = __pyx_dict_cached_value;\
01050     } else {\
01051         (VAR) = __pyx_dict_cached_value = (LOOKUP);\
01052         __pyx_dict_version = __PYX_GET_DICT_VERSION(DICT);\
01053     }\
01054 }
01055 static CYTHON_INLINE PY_UINT64_T __Pyx_get_tp_dict_version(PyObject* obj);
01056 static CYTHON_INLINE PY_UINT64_T __Pyx_get_object_dict_version(PyObject* obj);
01057 static CYTHON_INLINE int __Pyx_object_dict_version_matches(PyObject* obj,
PY_UINT64_T tp_dict_version, PY_UINT64_T obj_dict_version);
01058 #else
01059 #define __PYX_GET_DICT_VERSION(dict) (0)
01060 #define __PYX_UPDATE_DICT_CACHE(dict, value, cache_var, version_var)
01061 #define __PYX_PY_DICT_LOOKUP_IF_MODIFIED(VAR, DICT, LOOKUP) (VAR) = (LOOKUP);
01062 #endif
01063
01064 /* GetModuleGlobalName.proto */
01065 #if CYTHON_USE_DICT_VERSIONS
01066 #define __Pyx_GetModuleGlobalName(var, name) {\
01067     static PY_UINT64_T __pyx_dict_version = 0;\

```

```

01068     static PyObject * __pyx_dict_cached_value = NULL;\
01069     (var) = (likely(__pyx_dict_version == __PYX_GET_DICT_VERSION(__pyx_d))) ?\
01070     (likely(__pyx_dict_cached_value) ? __Pyx_NewRef(__pyx_dict_cached_value)
: __Pyx_GetBuiltinName(name)) :\
01071     __Pyx_GetModuleGlobalName(name, &__pyx_dict_version,
&__pyx_dict_cached_value);\
01072 }
01073 #define __Pyx_GetModuleGlobalNameUncached(var, name) {\
01074     PY_UINT64_T __pyx_dict_version;\
01075     PyObject * __pyx_dict_cached_value;\
01076     (var) = __Pyx_GetModuleGlobalName(name, &__pyx_dict_version,
&__pyx_dict_cached_value);\
01077 }
01078 static PyObject * __Pyx_GetModuleGlobalName(PyObject *name, PY_UINT64_T
*dict_version, PyObject **dict_cached_value);
01079 #else
01080 #define __Pyx_GetModuleGlobalName(var, name) (var) =
__Pyx_GetModuleGlobalName(name)
01081 #define __Pyx_GetModuleGlobalNameUncached(var, name) (var) =
__Pyx_GetModuleGlobalName(name)
01082 static CYTHON_INLINE PyObject * __Pyx_GetModuleGlobalName(PyObject *name);
01083 #endif
01084
01085 /* PyFunctionFastCall.proto */
01086 #if CYTHON_FAST_PYCALL
01087 #define __Pyx_PyFunction_FastCall(func, args, nargs)\
01088     __Pyx_PyFunction_FastCallDict((func), (args), (nargs), NULL)
01089 #if 1 || PY_VERSION_HEX < 0x030600B1
01090 static PyObject * __Pyx_PyFunction_FastCallDict(PyObject *func, PyObject **args,
Py_ssize_t nargs, PyObject *kwargs);
01091 #else
01092 #define __Pyx_PyFunction_FastCallDict(func, args, nargs, kwargs)
__PyFunction_FastCallDict(func, args, nargs, kwargs)
01093 #endif
01094 #define __Pyx_BUILD_ASSERT_EXPR(cond)\
01095     (sizeof(char [1 - 2*(cond)]) - 1)
01096 #ifndef Py_MEMBER_SIZE
01097 #define Py_MEMBER_SIZE(type, member) sizeof(((type *)0)->member)
01098 #endif
01099 #if CYTHON_FAST_PYCALL
01100     static size_t __pyx_pyframe_localsplus_offset = 0;
01101     #include "frameobject.h"
01102     #if PY_VERSION_HEX >= 0x030b00a6
01103     #ifndef Py_BUILD_CORE
01104     #define Py_BUILD_CORE 1
01105     #endif
01106     #include "internal/pycore_frame.h"
01107 #endif
01108     #define __Pxy_PyFrame_Initialize_Offsets()\
01109         ((void)__Pyx_BUILD_ASSERT_EXPR(sizeof(PyFrameObject) ==
offsetof(PyFrameObject, f_localsplus) + Py_MEMBER_SIZE(PyFrameObject, f_localsplus)),\
01110         (void)(__pyx_pyframe_localsplus_offset = ((size_t)PyFrame_Type.tp_basicsize)
- Py_MEMBER_SIZE(PyFrameObject, f_localsplus)))
01111     #define __Pyx_PyFrame_GetLocalsplus(frame)\
01112         (assert(__pyx_pyframe_localsplus_offset), (PyObject **)(((char *) (frame)) +
__pyx_pyframe_localsplus_offset))
01113 #endif // CYTHON_FAST_PYCALL
01114 #endif
01115
01116 /* PyObjectCall.proto */
01117 #if CYTHON_COMPILING_IN_CPYTHON
01118 static CYTHON_INLINE PyObject* __Pyx_PyObject_Call(PyObject *func, PyObject *arg,
PyObject *kw);
01119 #else
01120 #define __Pyx_PyObject_Call(func, arg, kw) PyObject_Call(func, arg, kw)
01121 #endif
01122
01123 /* PyObjectCallMethO.proto */
01124 #if CYTHON_COMPILING_IN_CPYTHON
01125 static CYTHON_INLINE PyObject* __Pyx_PyObject_CallMethO(PyObject *func, PyObject
*arg);
01126 #endif
01127
01128 /* PyObjectCallNoArg.proto */
01129 #if CYTHON_COMPILING_IN_CPYTHON
01130 static CYTHON_INLINE PyObject* __Pyx_PyObject_CallNoArg(PyObject *func);
01131 #else

```

```

01132 #define __Pyx_PyObject_CallNoArg(func) __Pyx_PyObject_Call(func,
__pyx_empty_tuple, NULL)
01133 #endif
01134
01135 /* PyThreadStateGet.proto */
01136 #if CYTHON_FAST_THREAD_STATE
01137 #define __Pyx_PyThreadState_declare PyThreadState *__pyx_tstate;
01138 #define __Pyx_PyThreadState_assign __pyx_tstate = __Pyx_PyThreadState_Current;
01139 #define __Pyx_PyErr_Occurred() __pyx_tstate->curexc_type
01140 #else
01141 #define __Pyx_PyThreadState_declare
01142 #define __Pyx_PyThreadState_assign
01143 #define __Pyx_PyErr_Occurred() PyErr_Occurred()
01144 #endif
01145
01146 /* PyErrFetchRestore.proto */
01147 #if CYTHON_FAST_THREAD_STATE
01148 #define __Pyx_PyErr_Clear() __Pyx_ErrRestore(NULL, NULL, NULL)
01149 #define __Pyx_ErrRestoreWithState(type, value, tb)
__Pyx_ErrRestoreInState(PyThreadState_GET(), type, value, tb)
01150 #define __Pyx_ErrFetchWithState(type, value, tb)
Pyx_ErrFetchInState(PyThreadState_GET(), type, value, tb)
01151 #define __Pyx_ErrRestore(type, value, tb) __Pyx_ErrRestoreInState(__pyx_tstate,
type, value, tb)
01152 #define __Pyx_ErrFetch(type, value, tb) __Pyx_ErrFetchInState(__pyx_tstate,
type, value, tb)
01153 static CYTHON_INLINE void __Pyx_ErrRestoreInState(PyThreadState *tstate, PyObject
*type, PyObject *value, PyObject *tb);
01154 static CYTHON_INLINE void __Pyx_ErrFetchInState(PyThreadState *tstate, PyObject
**type, PyObject **value, PyObject **tb);
01155 #if CYTHON_COMPILING_IN_CPYTHON
01156 #define __Pyx_PyErr_SetNone(exc) (Py_INCREF(exc), __Pyx_ErrRestore((exc), NULL,
NULL))
01157 #else
01158 #define __Pyx_PyErr_SetNone(exc) PyErr_SetNone(exc)
01159 #endif
01160 #else
01161 #define __Pyx_PyErr_Clear() PyErr_Clear()
01162 #define __Pyx_PyErr_SetNone(exc) PyErr_SetNone(exc)
01163 #define __Pyx_ErrRestoreWithState(type, value, tb) PyErr_Restore(type, value, tb)
01164 #define __Pyx_ErrFetchWithState(type, value, tb) PyErr_Fetch(type, value, tb)
01165 #define __Pyx_ErrRestoreInState(tstate, type, value, tb) PyErr_Restore(type,
value, tb)
01166 #define __Pyx_ErrFetchInState(tstate, type, value, tb) PyErr_Fetch(type, value,
tb)
01167 #define __Pyx_ErrRestore(type, value, tb) PyErr_Restore(type, value, tb)
01168 #define __Pyx_ErrFetch(type, value, tb) PyErr_Fetch(type, value, tb)
01169 #endif
01170
01171 /* CLineInTraceback.proto */
01172 #ifndef CYTHON_CLINE_IN_TRACEBACK
01173 #define __Pyx_CLineForTraceback(tstate, c_line) (((CYTHON_CLINE_IN_TRACEBACK)) ?
c_line : 0)
01174 #else
01175 static int __Pyx_CLineForTraceback(PyThreadState *tstate, int c_line);
01176 #endif
01177
01178 /* CodeObjectCache.proto */
01179 typedef struct {
01180     PyCodeObject* code_object;
01181     int code_line;
01182 } __Pyx_CodeObjectCacheEntry;
01183 struct __Pyx_CodeObjectCache {
01184     int count;
01185     int max_count;
01186     __Pyx_CodeObjectCacheEntry* entries;
01187 };
01188 static struct __Pyx_CodeObjectCache __pyx_code_cache = {0,0,NULL};
01189 static int __pyx_bisect_code_objects(__Pyx_CodeObjectCacheEntry* entries, int
count, int code_line);
01190 static PyCodeObject* __pyx_find_code_object(int code_line);
01191 static void __pyx_insert_code_object(int code_line, PyCodeObject* code_object);
01192
01193 /* AddTraceback.proto */
01194 static void __Pyx_AddTraceback(const char *funcname, int c_line,
int py_line, const char *filename);
01195
01196

```



```

01197 /* GCCDiagnostics.proto */
01198 #if defined(__GNUC__) && (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 6))
01199 #define __Pyx_HAS_GCC_DIAGNOSTIC
01200 #endif
01201
01202 /* CIntToPy.proto */
01203 static CYTHON_INLINE PyObject* __Pyx_PyInt_From_long(long value);
01204
01205 /* CIntFromPy.proto */
01206 static CYTHON_INLINE long __Pyx_PyInt_As_long(PyObject *);
01207
01208 /* CIntFromPy.proto */
01209 static CYTHON_INLINE int __Pyx_PyInt_As_int(PyObject *);
01210
01211 /* FastTypeChecks.proto */
01212 #if CYTHON_COMPILING_IN_CPYTHON
01213 #define __Pyx_TypeCheck(obj, type) __Pyx_IsSubtype(Py_TYPE(obj), (PyTypeObject
*)type)
01214 static CYTHON_INLINE int __Pyx_IsSubtype(PyTypeObject *a, PyTypeObject *b);
01215 static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches(PyObject *err, PyObject
*type);
01216 static CYTHON_INLINE int Pyx_PyErr_GivenExceptionMatches2(PyObject *err, PyObject
*type1, PyObject *type2);
01217 #else
01218 #define __Pyx_TypeCheck(obj, type) PyObject_TypeCheck(obj, (PyTypeObject *)type)
01219 #define __Pyx_PyErr_GivenExceptionMatches(err, type)
PyErr_GivenExceptionMatches(err, type)
01220 #define __Pyx_PyErr_GivenExceptionMatches2(err, type1, type2)
(PyErr_GivenExceptionMatches(err, type1) || PyErr_GivenExceptionMatches(err, type2))
01221 #endif
01222 #define __Pyx_PyException_Check(obj) __Pyx_TypeCheck(obj, PyExc_Exception)
01223
01224 /* CheckBinaryVersion.proto */
01225 static int __Pyx_check_binary_version(void);
01226
01227 /* InitStrings.proto */
01228 static int __Pyx_InitStrings(__Pyx_StringTabEntry *t);
01229
01230
01231 /* Module declarations from 'API_Calls.main' */
01232 #define __Pyx_MODULE_NAME "API_Calls.main"
01233 extern int __pyx_module_is_main_API_Calls__main_;
01234 int __pyx_module_is_main_API_Calls__main_ = 0;
01235
01236 /* Implementation of 'API_Calls.main' */
01237 static const char __pyx_k_main[] = "__main__";
01238 static const char __pyx_k_name[] = "__name__";
01239 static const char __pyx_k_test[] = "__test__";
01240 static const char __pyx_k_import[] = "__import__";
01241 static const char __pyx_k_Initializer[] = "Initializer";
01242 static const char __pyx_k_initializer[] = "initializer";
01243 static const char __pyx_k_cline_in_traceback[] = "cline_in_traceback";
01244 static PyObject* __pyx_n_s_Initializer;
01245 static PyObject* __pyx_n_s_cline_in_traceback;
01246 static PyObject* __pyx_n_s_import;
01247 static PyObject* __pyx_n_s_initializer;
01248 static PyObject* __pyx_n_s_main;
01249 static PyObject* __pyx_n_s_name;
01250 static PyObject* __pyx_n_s_test;
01251 /* Late includes */
01252
01253 static PyMethodDef __pyx_methods[] = {
01254     {0, 0, 0, 0}
01255 };
01256
01257 #if PY_MAJOR_VERSION >= 3
01258 #if CYTHON_PEP489_MULTI_PHASE_INIT
01259 static PyObject* __pyx_pymod_create(PyObject *spec, PyModuleDef *def); /*proto*/
01260 static int __pyx_pymod_exec_main(PyObject* module); /*proto*/
01261 static PyModuleDef_Slot __pyx_moduledef_slots[] = {
01262     {Py_mod_create, (void*)__pyx_pymod_create},
01263     {Py_mod_exec, (void*)__pyx_pymod_exec_main},
01264     {0, NULL}
01265 };
01266 #endif
01267
01268 static struct PyModuleDef __pyx_moduledef = {

```



```

01269     PyModuleDef_HEAD_INIT,
01270     "main",
01271     0, /* m_doc */
01272     #if CYTHON_PEP489_MULTI_PHASE_INIT
01273     0, /* m_size */
01274     #else
01275     -1, /* m_size */
01276     #endif
01277     __pyx_methods /* m_methods */,
01278     #if CYTHON_PEP489_MULTI_PHASE_INIT
01279     __pyx_moduledef_slots, /* m_slots */
01280     #else
01281     NULL, /* m_reload */
01282     #endif
01283     NULL, /* m_traverse */
01284     NULL, /* m_clear */
01285     NULL /* m_free */
01286 };
01287 #endif
01288 #ifndef CYTHON_SMALL_CODE
01289 #if defined(__clang__)
01290     #define CYTHON_SMALL_CODE
01291 #elif defined(__GNUC__) && (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 3))
01292     #define CYTHON_SMALL_CODE __attribute__((cold))
01293 #else
01294     #define CYTHON_SMALL_CODE
01295 #endif
01296 #endif
01297
01298 static __Pyx_StringTabEntry __pyx_string_tab[] = {
01299     {&__pyx_n_s_Initializer, __pyx_k_Initializer, sizeof(__pyx_k_Initializer), 0, 0,
01300     1, 1},
01301     {&__pyx_n_s_cline_in_traceback, __pyx_k_cline_in_traceback,
01302     sizeof(__pyx_k_cline_in_traceback), 0, 0, 1, 1},
01303     {&__pyx_n_s_import, __pyx_k_import, sizeof(__pyx_k_import), 0, 0, 1, 1},
01304     {&__pyx_n_s_initializer, __pyx_k_initializer, sizeof(__pyx_k_initializer), 0, 0,
01305     1, 1},
01306     {&__pyx_n_s_main, __pyx_k_main, sizeof(__pyx_k_main), 0, 0, 1, 1},
01307     {&__pyx_n_s_name, __pyx_k_name, sizeof(__pyx_k_name), 0, 0, 1, 1},
01308     {&__pyx_n_s_test, __pyx_k_test, sizeof(__pyx_k_test), 0, 0, 1, 1},
01309     {0, 0, 0, 0, 0, 0, 0}
01310 };
01311
01312 static CYTHON_SMALL_CODE int __Pyx_InitCachedBuiltins(void) {
01313     return 0;
01314 }
01315
01316 static CYTHON_SMALL_CODE int __Pyx_InitCachedConstants(void) {
01317     __Pyx_RefNannyDeclarations
01318     __Pyx_RefNannySetupContext("__Pyx_InitCachedConstants", 0);
01319     __Pyx_RefNannyFinishContext();
01320     return 0;
01321 }
01322
01323 static CYTHON_SMALL_CODE int __Pyx_InitGlobals(void) {
01324     if (__Pyx_InitStrings(__pyx_string_tab) < 0) __PYX_ERR(0, 1, __pyx_L1_error);
01325     return 0;
01326 }
01327
01328 __pyx_L1_error:;
01329     return -1;
01330 }
01331
01332 static CYTHON_SMALL_CODE int __Pyx_modinit_global_init_code(void); /*proto*/
01333 static CYTHON_SMALL_CODE int __Pyx_modinit_variable_export_code(void); /*proto*/
01334 static CYTHON_SMALL_CODE int __Pyx_modinit_function_export_code(void); /*proto*/
01335 static CYTHON_SMALL_CODE int __Pyx_modinit_type_init_code(void); /*proto*/
01336 static CYTHON_SMALL_CODE int __Pyx_modinit_type_import_code(void); /*proto*/
01337 static CYTHON_SMALL_CODE int __Pyx_modinit_variable_import_code(void); /*proto*/
01338 static CYTHON_SMALL_CODE int __Pyx_modinit_function_import_code(void); /*proto*/
01339
01340 static int __Pyx_modinit_global_init_code(void) {
01341     __Pyx_RefNannyDeclarations
01342     __Pyx_RefNannySetupContext("__Pyx_modinit_global_init_code", 0);
01343     /*--- Global init code ---*/
01344     __Pyx_RefNannyFinishContext();
01345     return 0;
01346 }
01347
01348 static int __Pyx_modinit_variable_export_code(void) {

```

```

01343 __Pyx_RefNannyDeclarations
01344 __Pyx_RefNannySetupContext("__Pyx_modinit_variable_export_code", 0);
01345 /*--- Variable export code ---*/
01346 __Pyx_RefNannyFinishContext();
01347 return 0;
01348 }
01349
01350 static int __Pyx_modinit_function_export_code(void) {
01351     __Pyx_RefNannyDeclarations
01352     __Pyx_RefNannySetupContext("__Pyx_modinit_function_export_code", 0);
01353     /*--- Function export code ---*/
01354     __Pyx_RefNannyFinishContext();
01355     return 0;
01356 }
01357
01358 static int __Pyx_modinit_type_init_code(void) {
01359     __Pyx_RefNannyDeclarations
01360     __Pyx_RefNannySetupContext("__Pyx_modinit_type_init_code", 0);
01361     /*--- Type init code ---*/
01362     __Pyx_RefNannyFinishContext();
01363     return 0;
01364 }
01365
01366 static int __Pyx_modinit_type_import_code(void) {
01367     __Pyx_RefNannyDeclarations
01368     __Pyx_RefNannySetupContext("__Pyx_modinit_type_import_code", 0);
01369     /*--- Type import code ---*/
01370     __Pyx_RefNannyFinishContext();
01371     return 0;
01372 }
01373
01374 static int __Pyx_modinit_variable_import_code(void) {
01375     __Pyx_RefNannyDeclarations
01376     __Pyx_RefNannySetupContext("__Pyx_modinit_variable_import_code", 0);
01377     /*--- Variable import code ---*/
01378     __Pyx_RefNannyFinishContext();
01379     return 0;
01380 }
01381
01382 static int __Pyx_modinit_function_import_code(void) {
01383     __Pyx_RefNannyDeclarations
01384     __Pyx_RefNannySetupContext("__Pyx_modinit_function_import_code", 0);
01385     /*--- Function import code ---*/
01386     __Pyx_RefNannyFinishContext();
01387     return 0;
01388 }
01389
01390
01391 #ifndef CYTHON_NO_PYINIT_EXPORT
01392 #define __Pyx_PyMODINIT_FUNC PyMODINIT_FUNC
01393 #elif PY_MAJOR_VERSION < 3
01394 #ifdef __cplusplus
01395 #define __Pyx_PyMODINIT_FUNC extern "C" void
01396 #else
01397 #define __Pyx_PyMODINIT_FUNC void
01398 #endif
01399 #else
01400 #ifdef __cplusplus
01401 #define __Pyx_PyMODINIT_FUNC extern "C" PyObject *
01402 #else
01403 #define __Pyx_PyMODINIT_FUNC PyObject *
01404 #endif
01405 #endif
01406
01407
01408 #if PY_MAJOR_VERSION < 3
01409 __Pyx_PyMODINIT_FUNC init_main(void) CYTHON_SMALL_CODE; /*proto*/
01410 __Pyx_PyMODINIT_FUNC init_main(void)
01411 #else
01412 __Pyx_PyMODINIT_FUNC PyInit__main(void) CYTHON_SMALL_CODE; /*proto*/
01413 __Pyx_PyMODINIT_FUNC PyInit__main(void)
01414 #if CYTHON_PEP489_MULTI_PHASE_INIT
01415 {
01416     return PyModuleDef_Init(&__pyx_moduledef);
01417 }
01418 static CYTHON_SMALL_CODE int __Pyx_check_single_interpreter(void) {
01419     #if PY_VERSION_HEX >= 0x030700A1

```

```

01420     static PY_INT64_T main_interpreter_id = -1;
01421     PY_INT64_T current_id =
PyInterpreterState_GetID(PyThreadState_Get()->interp);
01422     if (main_interpreter_id == -1) {
01423         main_interpreter_id = current_id;
01424         return (unlikely(current_id == -1)) ? -1 : 0;
01425     } else if (unlikely(main_interpreter_id != current_id))
01426         #else
01427         static PyInterpreterState *main_interpreter = NULL;
01428         PyInterpreterState *current_interpreter = PyThreadState_Get()->interp;
01429         if (!main_interpreter) {
01430             main_interpreter = current_interpreter;
01431         } else if (unlikely(main_interpreter != current_interpreter))
01432             #endif
01433         {
01434             PyErr_SetString(
01435                 PyExc_ImportError,
01436                 "Interpreter change detected - this module can only be loaded into one
interpreter per process.");
01437             return -1;
01438         }
01439         return 0;
01440     }
01441     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) {
01442         PyObject *value = PyObject_GetAttrString(spec, from_name);
01443         int result = 0;
01444         if (likely(value)) {
01445             if (allow_none || value != Py_None) {
01446                 result = PyDict_SetItemString(moddict, to_name, value);
01447             }
01448             Py_DECREF(value);
01449         } else if (PyErr_ExceptionMatches(PyExc_AttributeError)) {
01450             PyErr_Clear();
01451         } else {
01452             result = -1;
01453         }
01454         return result;
01455     }
01456     static CYTHON_SMALL_CODE PyObject* __pyx_pymod_create(PyObject *spec, CYTHON_UNUSED
PyModuleDef *def) {
01457         PyObject *module = NULL, *moddict, *modname;
01458         if (__Pyx_check_single_interpreter())
01459             return NULL;
01460         if (__pyx_m)
01461             return __Pyx_NewRef(__pyx_m);
01462         modname = PyObject_GetAttrString(spec, "name");
01463         if (unlikely(!modname)) goto bad;
01464         module = PyModule_NewObject(modname);
01465         Py_DECREF(modname);
01466         if (unlikely(!module)) goto bad;
01467         moddict = PyModule_GetDict(module);
01468         if (unlikely(!moddict)) goto bad;
01469         if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "loader", "__loader__",
1) < 0)) goto bad;
01470         if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "origin", "__file__", 1)
< 0)) goto bad;
01471         if (unlikely(__Pyx_copy_spec_to_module(spec, moddict, "parent", "__package__",
1) < 0)) goto bad;
01472         if (unlikely(__Pyx_copy_spec_to_module(spec, moddict,
"submodule_search_locations", "__path__", 0) < 0)) goto bad;
01473         return module;
01474     bad:
01475         Py_XDECREF(module);
01476         return NULL;
01477     }
01478
01479
01480     static CYTHON_SMALL_CODE int __pyx_pymod_exec_main_(PyObject
*__pyx_pyinit_module)
01481     #endif
01482     #endif
01483     {
01484         PyObject *__pyx_t_1 = NULL;
01485         PyObject *__pyx_t_2 = NULL;
01486         int __pyx_lineno = 0;
01487         const char *__pyx_filename = NULL;

```

```

01488     int __pyx_clineno = 0;
01489     __Pyx_RefNannyDeclarations
01490     #if CYTHON_PEP489_MULTI_PHASE_INIT
01491     if (__pyx_m) {
01492         if (__pyx_m == __pyx_pyinit_module) return 0;
01493         PyErr_SetString(PyExc_RuntimeError, "Module '__main__' has already been imported.
Re-initialisation is not supported.");
01494         return -1;
01495     }
01496     #elif PY_MAJOR_VERSION >= 3
01497     if (__pyx_m) return __Pyx_NewRef(__pyx_m);
01498     #endif
01499     #if CYTHON_REFNANNY
01500     __Pyx_RefNanny = __Pyx_RefNannyImportAPI("refnanny");
01501     if (!__Pyx_RefNanny) {
01502         PyErr_Clear();
01503         __Pyx_RefNanny = __Pyx_RefNannyImportAPI("Cython.Runtime.refnanny");
01504         if (!__Pyx_RefNanny)
01505             Py_FatalError("failed to import 'refnanny' module");
01506     }
01507     #endif
01508     Pyx_RefNannySetupContext(" Pyx PyMODINIT FUNC PyInit main (void)", 0);
01509     if (__Pyx_check_binary_version() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01510     #ifdef __Pxy_PyFrame_Initialize_Offsets
01511     __Pxy_PyFrame_Initialize_Offsets();
01512     #endif
01513     __pyx_empty_tuple = PyTuple_New(0); if (unlikely(!__pyx_empty_tuple))
__PYX_ERR(0, 1, __pyx_L1_error)
01514     __pyx_empty_bytes = PyBytes_FromStringAndSize("", 0); if
(unlikely(!__pyx_empty_bytes)) __PYX_ERR(0, 1, __pyx_L1_error)
01515     __pyx_empty_unicode = PyUnicode_FromStringAndSize("", 0); if
(unlikely(!__pyx_empty_unicode)) __PYX_ERR(0, 1, __pyx_L1_error)
01516     #ifdef __Pyx_CyFunction_USED
01517     if (__pyx_CyFunction_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01518     #endif
01519     #ifdef __Pyx_FusedFunction_USED
01520     if (__pyx_FusedFunction_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01521     #endif
01522     #ifdef __Pyx_Coroutine_USED
01523     if (__pyx_Coroutine_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01524     #endif
01525     #ifdef __Pyx_Generator_USED
01526     if (__pyx_Generator_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01527     #endif
01528     #ifdef __Pyx_AsyncGen_USED
01529     if (__pyx_AsyncGen_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01530     #endif
01531     #ifdef __Pyx_StopAsyncIteration_USED
01532     if (__pyx_StopAsyncIteration_init() < 0) __PYX_ERR(0, 1, __pyx_L1_error)
01533     #endif
01534     /*--- Library function declarations ---*/
01535     /*--- Threads initialization code ---*/
01536     #if defined(WITH_THREAD) && PY_VERSION_HEX < 0x030700F0 &&
defined(__PYX_FORCE_INIT_THREADS) && __PYX_FORCE_INIT_THREADS
01537     PyEval_InitThreads();
01538     #endif
01539     /*--- Module creation code ---*/
01540     #if CYTHON_PEP489_MULTI_PHASE_INIT
01541     __pyx_m = __pyx_pyinit_module;
01542     Py_INCREF(__pyx_m);
01543     #else
01544     #if PY_MAJOR_VERSION < 3
01545     __pyx_m = Py_InitModule4("__main__", __pyx_methods, 0, 0, PYTHON_API_VERSION);
Py_XINCREF(__pyx_m);
01546     #else
01547     __pyx_m = PyModule_Create(&__pyx_moduledef);
01548     #endif
01549     if (unlikely(!__pyx_m)) __PYX_ERR(0, 1, __pyx_L1_error)
01550     #endif
01551     __pyx_d = PyModule_GetDict(__pyx_m); if (unlikely(!__pyx_d)) __PYX_ERR(0, 1,
__pyx_L1_error)
01552     Py_INCREF(__pyx_d);
01553     __pyx_b = PyImport_AddModule(__Pyx_BUILTIN_MODULE_NAME); if (unlikely(!__pyx_b))
__PYX_ERR(0, 1, __pyx_L1_error)
01554     Py_INCREF(__pyx_b);
01555     __pyx_cython_runtime = PyImport_AddModule((char *) "cython_runtime"); if
(unlikely(!__pyx_cython_runtime)) __PYX_ERR(0, 1, __pyx_L1_error)

```

[illegible]

[illegible]

```

01693 PyObject *empty_list = 0;
01694 PyObject *module = 0;
01695 PyObject *global_dict = 0;
01696 PyObject *empty_dict = 0;
01697 PyObject *list;
01698 #if PY_MAJOR_VERSION < 3
01699 PyObject *py_import;
01700 py_import = __Pyx_PyObject_GetAttrStr(__pyx_b, __pyx_n_s_import);
01701 if (!py_import)
01702     goto bad;
01703 #endif
01704 if (from_list)
01705     list = from_list;
01706 else {
01707     empty_list = PyList_New(0);
01708     if (!empty_list)
01709         goto bad;
01710     list = empty_list;
01711 }
01712 global_dict = PyModule_GetDict(__pyx_m);
01713 if (!global_dict)
01714     goto bad;
01715 empty_dict = PyDict_New();
01716 if (!empty_dict)
01717     goto bad;
01718 {
01719     #if PY_MAJOR_VERSION >= 3
01720     if (level == -1) {
01721         if ((1) && (strchr(__Pyx_MODULE_NAME, '.'))) {
01722             module = PyImport_ImportModuleLevelObject(
01723                 name, global_dict, empty_dict, list, 1);
01724             if (!module) {
01725                 if (!PyErr_ExceptionMatches(PyExc_ImportError))
01726                     goto bad;
01727                 PyErr_Clear();
01728             }
01729         }
01730         level = 0;
01731     }
01732     #endif
01733     if (!module) {
01734         #if PY_MAJOR_VERSION < 3
01735         PyObject *py_level = PyInt_FromLong(level);
01736         if (!py_level)
01737             goto bad;
01738         module = PyObject_CallFunctionObjArgs(py_import,
01739             name, global_dict, empty_dict, list, py_level, (PyObject *)NULL);
01740         Py_DECREF(py_level);
01741         #else
01742         module = PyImport_ImportModuleLevelObject(
01743             name, global_dict, empty_dict, list, level);
01744         #endif
01745     }
01746 }
01747 bad:
01748 #if PY_MAJOR_VERSION < 3
01749 Py_XDECREF(py_import);
01750 #endif
01751 Py_XDECREF(empty_list);
01752 Py_XDECREF(empty_dict);
01753 return module;
01754 }
01755
01756 /* ImportFrom */
01757 static PyObject* __Pyx_ImportFrom(PyObject* module, PyObject* name) {
01758     PyObject* value = __Pyx_PyObject_GetAttrStr(module, name);
01759     if (unlikely(!value) && PyErr_ExceptionMatches(PyExc_AttributeError)) {
01760         PyErr_Format(PyExc_ImportError,
01761             #if PY_MAJOR_VERSION < 3
01762             "cannot import name %.230s", PyString_AS_STRING(name));
01763             #else
01764             "cannot import name %S", name);
01765             #endif
01766     }
01767     return value;
01768 }
01769

```

```

01770 /* GetBuiltinName */
01771 static PyObject* __Pyx_GetBuiltinName(PyObject* name) {
01772     PyObject* result = __Pyx_PyObject_GetAttrStr(__pyx_b, name);
01773     if (unlikely(!result)) {
01774         PyErr_Format(PyExc_NameError,
01775             "name '%U' is not defined", name);
01776     }
01777     #else
01778         "name '%.200s' is not defined", PyString_AS_STRING(name));
01779     #endif
01780 }
01781 return result;
01782 }
01783
01784 /* PyDictVersioning */
01785 #if CYTHON_USE_DICT_VERSIONS && CYTHON_USE_TYPE_SLOTS
01786 static CYTHON_INLINE PY_UINT64_T __Pyx_get_tp_dict_version(PyObject* obj) {
01787     PyObject* dict = Py_TYPE(obj)->tp_dict;
01788     return likely(dict) ? __PYX_GET_DICT_VERSION(dict) : 0;
01789 }
01790 static CYTHON_INLINE PY_UINT64_T __Pyx_get_object_dict_version(PyObject* obj) {
01791     PyObject** dictptr = NULL;
01792     Py_ssize_t offset = Py_TYPE(obj)->tp_dictoffset;
01793     if (offset) {
01794         #if CYTHON_COMPILING_IN_CPYTHON
01795             dictptr = (likely(offset > 0)) ? (PyObject**) ((char*)obj + offset) :
01796                 _PyObject_GetDictPtr(obj);
01797         #else
01798             dictptr = _PyObject_GetDictPtr(obj);
01799         #endif
01800     }
01801     return (dictptr && *dictptr) ? __PYX_GET_DICT_VERSION(*dictptr) : 0;
01802 }
01803 static CYTHON_INLINE int __Pyx_object_dict_version_matches(PyObject* obj,
01804     PY_UINT64_T tp_dict_version, PY_UINT64_T obj_dict_version) {
01805     PyObject* dict = Py_TYPE(obj)->tp_dict;
01806     if (unlikely(!dict) || unlikely(tp_dict_version !=
01807         __PYX_GET_DICT_VERSION(dict)))
01808         return 0;
01809     return obj_dict_version == __Pyx_get_object_dict_version(obj);
01810 }
01811 #endif
01812
01813 /* GetModuleGlobalName */
01814 #if CYTHON_USE_DICT_VERSIONS
01815 static PyObject* __Pyx_GetModuleGlobalName(PyObject* name, PY_UINT64_T
01816     *dict_version, PyObject** dict_cached_value)
01817 #else
01818 static CYTHON_INLINE PyObject* __Pyx_GetModuleGlobalName(PyObject* name)
01819 #endif
01820 {
01821     PyObject* result;
01822     #if !CYTHON_AVOID_BORROWED_REFS
01823     #if CYTHON_COMPILING_IN_CPYTHON && PY_VERSION_HEX >= 0x030500A1
01824         result = _PyDict_GetItem_KnownHash(__pyx_d, name, ((PyASCIIObject*)
01825             name)->hash);
01826         __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
01827         if (likely(result)) {
01828             return __Pyx_NewRef(result);
01829         } else if (unlikely(PyErr_Occurred())) {
01830             return NULL;
01831         }
01832     #else
01833         result = PyDict_GetItem(__pyx_d, name);
01834         __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
01835         if (likely(result)) {
01836             return __Pyx_NewRef(result);
01837         }
01838     #endif
01839     #endif
01840     result = PyObject_GetItem(__pyx_d, name);
01841     __PYX_UPDATE_DICT_CACHE(__pyx_d, result, *dict_cached_value, *dict_version)
01842     if (likely(result)) {
01843         return __Pyx_NewRef(result);
01844     }
01845     PyErr_Clear();
01846 #endif

```



```

01842     return __Pyx_GetBuiltinName(name);
01843 }
01844
01845 /* PyFunctionFastCall */
01846 #if CYTHON_FAST_PYCALL
01847 static PyObject* __Pyx_PyFunction_FastCallNoKw(PyCodeObject *co, PyObject **args,
Py_ssize_t na,
01848                                           PyObject *globals) {
01849     PyFrameObject *f;
01850     PyThreadState *tstate = __Pyx_PyThreadState_Current;
01851     PyObject **fastlocals;
01852     Py_ssize_t i;
01853     PyObject *result;
01854     assert(globals != NULL);
01855     /* XXX Perhaps we should create a specialized
01856        PyFrame New() that doesn't take locals, but does
01857        take builtins without sanity checking them.
01858        */
01859     assert(tstate != NULL);
01860     f = PyFrame_New(tstate, co, globals, NULL);
01861     if (f == NULL) {
01862         return NULL;
01863     }
01864     fastlocals = __Pyx_PyFrame_GetLocalsplus(f);
01865     for (i = 0; i < na; i++) {
01866         Py_INCREF(*args);
01867         fastlocals[i] = *args++;
01868     }
01869     result = PyEval_EvalFrameEx(f, 0);
01870     ++tstate->recursion_depth;
01871     Py_DECREF(f);
01872     --tstate->recursion_depth;
01873     return result;
01874 }
01875 #if 1 || PY_VERSION_HEX < 0x030600B1
01876 static PyObject* __Pyx_PyFunction_FastCallDict(PyObject *func, PyObject **args,
Py_ssize_t nargs, PyObject *kwargs) {
01877     PyCodeObject *co = (PyCodeObject *)PyFunction_GET_CODE(func);
01878     PyObject *globals = PyFunction_GET_GLOBALS(func);
01879     PyObject *argdefs = PyFunction_GET_DEFAULTS(func);
01880     PyObject *closure;
01881     #if PY_MAJOR_VERSION >= 3
01882     PyObject *kwdefs;
01883     #endif
01884     PyObject *kwtuple, **k;
01885     PyObject **d;
01886     Py_ssize_t nd;
01887     Py_ssize_t nk;
01888     PyObject *result;
01889     assert(kwargs == NULL || PyDict_Check(kwargs));
01890     nk = kwargs ? PyDict_Size(kwargs) : 0;
01891     if (Py_EnterRecursiveCall((char*)" while calling a Python object")) {
01892         return NULL;
01893     }
01894     if (
01895     #if PY_MAJOR_VERSION >= 3
01896         co->co_kwonlyargcount == 0 &&
01897     #endif
01898         likely(kwargs == NULL || nk == 0) &&
01899         co->co_flags == (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)) {
01900         if (argdefs == NULL && co->co_argcount == nargs) {
01901             result = __Pyx_PyFunction_FastCallNoKw(co, args, nargs, globals);
01902             goto done;
01903         }
01904         else if (nargs == 0 && argdefs != NULL
01905                 && co->co_argcount == Py_SIZE(argdefs)) {
01906             /* function called with no arguments, but all parameters have
01907                a default value: use default values as arguments .*/
01908             args = &PyTuple_GET_ITEM(argdefs, 0);
01909             result = __Pyx_PyFunction_FastCallNoKw(co, args, Py_SIZE(argdefs),
globals);
01910             goto done;
01911         }
01912     }
01913     if (kwargs != NULL) {
01914         Py_ssize_t pos, i;
01915         kwtuple = PyTuple_New(2 * nk);

```

```

01916         if (kwtuple == NULL) {
01917             result = NULL;
01918             goto done;
01919         }
01920         k = &PyTuple_GET_ITEM(kwtuple, 0);
01921         pos = i = 0;
01922         while (PyDict_Next(kwargs, &pos, &k[i], &k[i+1])) {
01923             Py_INCREF(k[i]);
01924             Py_INCREF(k[i+1]);
01925             i += 2;
01926         }
01927         nk = i / 2;
01928     }
01929     else {
01930         kwtuple = NULL;
01931         k = NULL;
01932     }
01933     closure = PyFunction_GET_CLOSURE(func);
01934     #if PY_MAJOR_VERSION >= 3
01935     kwdefs = PyFunction_GET_KW_DEFAULTS(func);
01936 #endif
01937     if (argdefs != NULL) {
01938         d = &PyTuple_GET_ITEM(argdefs, 0);
01939         nd = Py_SIZE(argdefs);
01940     }
01941     else {
01942         d = NULL;
01943         nd = 0;
01944     }
01945     #if PY_MAJOR_VERSION >= 3
01946     result = PyEval_EvalCodeEx((PyObject*)co, globals, (PyObject *)NULL,
01947                                args, (int)nargs,
01948                                k, (int)nk,
01949                                d, (int)nd, kwdefs, closure);
01950 #else
01951     result = PyEval_EvalCodeEx(co, globals, (PyObject *)NULL,
01952                                args, (int)nargs,
01953                                k, (int)nk,
01954                                d, (int)nd, closure);
01955 #endif
01956     Py_XDECREF(kwtuple);
01957 done:
01958     Py_LeaveRecursiveCall();
01959     return result;
01960 }
01961 #endif
01962 #endif
01963
01964 /* PyObjectCall */
01965 #if CYTHON_COMPILING_IN_CPYTHON
01966 static CYTHON_INLINE PyObject* __Pyx_PyObject_Call(PyObject *func, PyObject *arg,
PyObject *kw) {
01967     PyObject *result;
01968     ternaryfunc call = Py_TYPE(func)->tp_call;
01969     if (unlikely(!call))
01970         return PyObject_Call(func, arg, kw);
01971     if (unlikely(Py_EnterRecursiveCall((char*)" while calling a Python object")))
01972         return NULL;
01973     result = (*call)(func, arg, kw);
01974     Py_LeaveRecursiveCall();
01975     if (unlikely(!result) && unlikely(!PyErr_Occurred())) {
01976         PyErr_SetString(
01977             PyExc_SystemError,
01978             "NULL result without error in PyObject_Call");
01979     }
01980     return result;
01981 }
01982 #endif
01983
01984 /* PyObjectCallMethO */
01985 #if CYTHON_COMPILING_IN_CPYTHON
01986 static CYTHON_INLINE PyObject* __Pyx_PyObject_CallMethO(PyObject *func, PyObject
*arg) {
01987     PyObject *self, *result;
01988     PyCFunction cfunc;
01989     cfunc = PyCFunction_GET_FUNCTION(func);
01990     self = PyCFunction_GET_SELF(func);

```

```

01991     if (unlikely(Py_EnterRecursiveCall((char*)" while calling a Python object")))
01992         return NULL;
01993     result = cfunc(self, arg);
01994     Py_LeaveRecursiveCall();
01995     if (unlikely(!result) && unlikely(!PyErr_Occurred())) {
01996         PyErr_SetString(
01997             PyExc_SystemError,
01998             "NULL result without error in PyObject_Call");
01999     }
02000     return result;
02001 }
02002 #endif
02003
02004 /* PyObjectCallNoArg */
02005 #if CYTHON_COMPILING_IN_CPYTHON
02006 static CYTHON_INLINE PyObject* Pyx PyObject CallNoArg(PyObject *func) {
02007 #if CYTHON_FAST_PYCALL
02008     if (PyFunction_Check(func)) {
02009         return __Pyx_PyFunction_FastCall(func, NULL, 0);
02010     }
02011 #endif
02012 #ifdef Pyx_CyFunction_USED
02013     if (likely(PyCFunction_Check(func) || __Pyx_CyFunction_Check(func)))
02014 #else
02015     if (likely(PyCFunction_Check(func)))
02016 #endif
02017     {
02018         if (likely(PyCFunction_GET_FLAGS(func) & METH_NOARGS)) {
02019             return __Pyx_PyObject_CallMethO(func, NULL);
02020         }
02021     }
02022     return __Pyx_PyObject_Call(func, __pyx_empty_tuple, NULL);
02023 }
02024 #endif
02025
02026 /* PyErrFetchRestore */
02027 #if CYTHON_FAST_THREAD_STATE
02028 static CYTHON_INLINE void __Pyx_ErrRestoreInState(PyThreadState *tstate, PyObject
02029 *type, PyObject *value, PyObject *tb) {
02030     PyObject *tmp_type, *tmp_value, *tmp_tb;
02031     tmp_type = tstate->curexc_type;
02032     tmp_value = tstate->curexc_value;
02033     tmp_tb = tstate->curexc_traceback;
02034     tstate->curexc_type = type;
02035     tstate->curexc_value = value;
02036     tstate->curexc_traceback = tb;
02037     Py_XDECREF(tmp_type);
02038     Py_XDECREF(tmp_value);
02039     Py_XDECREF(tmp_tb);
02040 }
02041 static CYTHON_INLINE void __Pyx_ErrFetchInState(PyThreadState *tstate, PyObject
02042 **type, PyObject **value, PyObject **tb) {
02043     *type = tstate->curexc_type;
02044     *value = tstate->curexc_value;
02045     *tb = tstate->curexc_traceback;
02046     tstate->curexc_type = 0;
02047     tstate->curexc_value = 0;
02048     tstate->curexc_traceback = 0;
02049 }
02050 #endif
02051
02052 /* CLineInTraceback */
02053 #ifndef CYTHON_CLINE_IN_TRACEBACK
02054 static int __Pyx_CLineForTraceback(CYTHON_NCP_UNUSED PyThreadState *tstate, int
02055 c_line) {
02056     PyObject *use_cline;
02057     PyObject *ptype, *pvalue, *ptraceback;
02058 #if CYTHON_COMPILING_IN_CPYTHON
02059     PyObject **cython_runtime_dict;
02060 #endif
02061 #endif
02062     if (unlikely(!__pyx_cython_runtime)) {
02063         return c_line;
02064     }
02065     __Pyx_ErrFetchInState(tstate, &ptype, &pvalue, &ptraceback);
02066 #if CYTHON_COMPILING_IN_CPYTHON
02067     cython_runtime_dict = _PyObject_GetDictPtr(__pyx_cython_runtime);
02068     if (likely(cython_runtime_dict)) {

```

```

02065     __PYX_PY_DICT_LOOKUP_IF_MODIFIED(
02066         use_cline, *cython_runtime_dict,
02067         __Pyx_PyDict_GetItemStr(*cython_runtime_dict,
__pyx_n_s_cline_in_traceback))
02068     } else
02069 #endif
02070     {
02071         PyObject *use_cline_obj = __Pyx_PyObject_GetAttrStr(__pyx_cython_runtime,
__pyx_n_s_cline_in_traceback);
02072         if (use_cline_obj) {
02073             use_cline = PyObject_Not(use_cline_obj) ? Py_False : Py_True;
02074             Py_DECREF(use_cline_obj);
02075         } else {
02076             PyErr_Clear();
02077             use_cline = NULL;
02078         }
02079     }
02080     if (!use_cline) {
02081         c_iline = 0;
02082         (void) PyObject_SetAttr(__pyx_cython_runtime,
__pyx_n_s_cline_in_traceback, Py_False);
02083     }
02084     else if (use_cline == Py_False || (use_cline != Py_True &&
PyObject_Not(use_cline) != 0)) {
02085         c_iline = 0;
02086     }
02087     __Pyx_ErrRestoreInState(tstate, ptype, pvalue, ptraceback);
02088     return c_iline;
02089 }
02090 #endif
02091
02092 /* CodeObjectCache */
02093 static int __pyx_bisect_code_objects(__Pyx_CodeObjectCacheEntry* entries, int
count, int code_line) {
02094     int start = 0, mid = 0, end = count - 1;
02095     if (end >= 0 && code_line > entries[end].code_line) {
02096         return count;
02097     }
02098     while (start < end) {
02099         mid = start + (end - start) / 2;
02100         if (code_line < entries[mid].code_line) {
02101             end = mid;
02102         } else if (code_line > entries[mid].code_line) {
02103             start = mid + 1;
02104         } else {
02105             return mid;
02106         }
02107     }
02108     if (code_line <= entries[mid].code_line) {
02109         return mid;
02110     } else {
02111         return mid + 1;
02112     }
02113 }
02114 static PyCodeObject* __pyx_find_code_object(int code_line) {
02115     PyCodeObject* code_object;
02116     int pos;
02117     if (unlikely(!code_line) || unlikely(!__pyx_code_cache.entries)) {
02118         return NULL;
02119     }
02120     pos = __pyx_bisect_code_objects(__pyx_code_cache.entries,
__pyx_code_cache.count, code_line);
02121     if (unlikely(pos >= __pyx_code_cache.count) ||
unlikely(__pyx_code_cache.entries[pos].code_line != code_line)) {
02122         return NULL;
02123     }
02124     code_object = __pyx_code_cache.entries[pos].code_object;
02125     Py_INCREF(code_object);
02126     return code_object;
02127 }
02128 static void __pyx_insert_code_object(int code_line, PyCodeObject* code_object) {
02129     int pos, i;
02130     __Pyx_CodeObjectCacheEntry* entries = __pyx_code_cache.entries;
02131     if (unlikely(!code_line)) {
02132         return;
02133     }
02134     if (unlikely(!entries)) {

```

```

02135     entries  

02136     (__Pyx_CodeObjectCacheEntry*) PyMem_Malloc(64*sizeof(__Pyx_CodeObjectCacheEntry));  

02137     if (likely(entries)) {  

02138         __pyx_code_cache.entries = entries;  

02139         __pyx_code_cache.max_count = 64;  

02140         __pyx_code_cache.count = 1;  

02141         entries[0].code_line = code_line;  

02142         entries[0].code_object = code_object;  

02143         Py_INCREF(code_object);  

02144     }  

02145     return;  

02146     pos = __pyx_bisect_code_objects(__pyx_code_cache.entries,  

__pyx_code_cache.count, code_line);  

02147     if ((pos < __pyx_code_cache.count) &&  

unlikely( __pyx_code_cache.entries[pos].code_line == code_line)) {  

02148         PyObject* tmp = entries[pos].code_object;  

02149         entries[pos].code_object = code_object;  

02150         Py_DECREF(tmp);  

02151         return;  

02152     }  

02153     if ( __pyx_code_cache.count == __pyx_code_cache.max_count) {  

02154         int new_max = __pyx_code_cache.max_count + 64;  

02155         entries = (__Pyx_CodeObjectCacheEntry*) PyMem_Realloc(  

02156             __pyx_code_cache.entries, ((size_t)new_max) *  

sizeof(__Pyx_CodeObjectCacheEntry));  

02157         if (unlikely(!entries)) {  

02158             return;  

02159         }  

02160         __pyx_code_cache.entries = entries;  

02161         __pyx_code_cache.max_count = new_max;  

02162     }  

02163     for (i=__pyx_code_cache.count; i>pos; i--) {  

02164         entries[i] = entries[i-1];  

02165     }  

02166     entries[pos].code_line = code_line;  

02167     entries[pos].code_object = code_object;  

02168     __pyx_code_cache.count++;  

02169     Py_INCREF(code_object);  

02170 }  

02171  

02172 /* AddTraceback */  

02173 #include "compile.h"  

02174 #include "frameobject.h"  

02175 #include "traceback.h"  

02176 #if PY_VERSION_HEX >= 0x030b00a6  

02177     #ifndef Py_BUILD_CORE  

02178         #define Py_BUILD_CORE 1  

02179     #endif  

02180     #include "internal/pycore_frame.h"  

02181 #endif  

02182 static PyObject* __Pyx_CreateCodeObjectForTraceback(  

02183     const char *funcname, int c_line,  

02184     int py_line, const char *filename) {  

02185     PyObject* py_code = NULL;  

02186     PyObject* py_funcname = NULL;  

02187     #if PY_MAJOR_VERSION < 3  

02188     PyObject* py_srcfile = NULL;  

02189     py_srcfile = PyString_FromString(filename);  

02190     if (!py_srcfile) goto bad;  

02191     #endif  

02192     if (c_line) {  

02193         #if PY_MAJOR_VERSION < 3  

02194         py_funcname = PyString_FromFormat( "%s (%s:%d)", funcname, __pyx_cfilenm,  

c_line);  

02195         if (!py_funcname) goto bad;  

02196         #else  

02197         py_funcname = PyUnicode_FromFormat( "%s (%s:%d)", funcname, __pyx_cfilenm,  

c_line);  

02198         if (!py_funcname) goto bad;  

02199         funcname = PyUnicode_AsUTF8(py_funcname);  

02200         if (!funcname) goto bad;  

02201         #endif  

02202     }  

02203     else{  

02204         #if PY_MAJOR_VERSION < 3  

02205         py_funcname = PyString_FromString(funcname);  


```

```

02206         if (!py_funcname) goto bad;
02207         #endif
02208     }
02209     #if PY_MAJOR_VERSION < 3
02210     py_code = __Pyx_PyCode_New(
02211         0,
02212         0,
02213         0,
02214         0,
02215         0,
02216         __pyx_empty_bytes, /*PyObject *code,*/
02217         __pyx_empty_tuple, /*PyObject *consts,*/
02218         __pyx_empty_tuple, /*PyObject *names,*/
02219         __pyx_empty_tuple, /*PyObject *varnames,*/
02220         __pyx_empty_tuple, /*PyObject *freevars,*/
02221         __pyx_empty_tuple, /*PyObject *cellvars,*/
02222         py_srcfile, /*PyObject *filename,*/
02223         py_funcname, /*PyObject *name,*/
02224         py_line,
02225         __pyx_empty_bytes /*PyObject *notab*/
02226     );
02227     Py_DECREF(py_srcfile);
02228     #else
02229     py_code = PyCode_NewEmpty(filename, funcname, py_line);
02230     #endif
02231     Py_XDECREF(py_funcname); // XDECREF since it's only set on Py3 if cline
02232     return py_code;
02233 bad:
02234     Py_XDECREF(py_funcname);
02235     #if PY_MAJOR_VERSION < 3
02236     Py_XDECREF(py_srcfile);
02237     #endif
02238     return NULL;
02239 }
02240 static void __Pyx_AddTraceback(const char *funcname, int c_line,
02241                               int py_line, const char *filename) {
02242     PyCodeObject *py_code = 0;
02243     PyFrameObject *py_frame = 0;
02244     PyThreadState *tstate = __Pyx_PyThreadState_Current;
02245     PyObject *ptype, *pvalue, *ptraceback;
02246     if (c_line) {
02247         c_line = __Pyx_CLineForTraceback(tstate, c_line);
02248     }
02249     py_code = __pyx_find_code_object(c_line ? -c_line : py_line);
02250     if (!py_code) {
02251         __Pyx_ErrFetchInState(tstate, &ptype, &pvalue, &ptraceback);
02252         py_code = __Pyx_CreateCodeObjectForTraceback(
02253             funcname, c_line, py_line, filename);
02254         if (!py_code) {
02255             /* If the code object creation fails, then we should clear the
02256                fetched exception references and propagate the new exception */
02257             Py_XDECREF(ptype);
02258             Py_XDECREF(pvalue);
02259             Py_XDECREF(ptraceback);
02260             goto bad;
02261         }
02262         __Pyx_ErrRestoreInState(tstate, ptype, pvalue, ptraceback);
02263         __pyx_insert_code_object(c_line ? -c_line : py_line, py_code);
02264     }
02265     py_frame = PyFrame_New(
02266         tstate, /*PyThreadState *tstate,*/
02267         py_code, /*PyObject *code,*/
02268         __pyx_d, /*PyObject *globals,*/
02269         0 /*PyObject *locals*/
02270     );
02271     if (!py_frame) goto bad;
02272     __Pyx_PyFrame_SetLineNumber(py_frame, py_line);
02273     PyTraceBack_Here(py_frame);
02274 bad:
02275     Py_XDECREF(py_code);
02276     Py_XDECREF(py_frame);
02277 }
02278
02279 /* MainFunction */
02280 #ifdef __FreeBSD__
02281 #include <floatingpoint.h>
02282 #endif

```

```

02283 #if PY_MAJOR_VERSION < 3
02284 int main(int argc, char** argv) {
02285 #elif defined(WIN32) || defined(MS_WINDOWS)
02286 int wmain(int argc, wchar_t **argv) {
02287 #else
02288 static int __Pyx_main(int argc, wchar_t **argv) {
02289 #endif
02290 /* 754 requires that FP exceptions run in "no stop" mode by default,
02291  * and until C vendors implement C99's ways to control FP exceptions,
02292  * Python requires non-stop mode. Alas, some platforms enable FP
02293  * exceptions by default. Here we disable them.
02294  */
02295 #ifdef __FreeBSD__
02296     fp_except_t m;
02297     m = fpgetmask();
02298     fpsetmask(m & ~FP_X OFL);
02299 #endif
02300     if (argc && argv)
02301         Py_SetProgramName(argv[0]);
02302     Py_Initialize();
02303     if (argc && argv)
02304         PySys_SetArgv(argc, argv);
02305     {
02306         PyObject* m = NULL;
02307         __pyx_module_is_main_API_Calls__main_ = 1;
02308         #if PY_MAJOR_VERSION < 3
02309             init_main_();
02310         #elif CYTHON_PEP489_MULTI_PHASE_INIT
02311             m = PyInit__main_();
02312             if (!PyModule_Check(m)) {
02313                 PyModuleDef *mdef = (PyModuleDef *) m;
02314                 PyObject *modname = PyUnicode_FromString("__main__");
02315                 m = NULL;
02316                 if (modname) {
02317                     m = PyModule_NewObject(modname);
02318                     Py_DECREF(modname);
02319                     if (m) PyModule_ExecDef(m, mdef);
02320                 }
02321             }
02322         #else
02323             m = PyInit__main_();
02324         #endif
02325         if (PyErr_Occurred()) {
02326             PyErr_Print();
02327             #if PY_MAJOR_VERSION < 3
02328                 if (Py_FlushLine()) PyErr_Clear();
02329             #endif
02330             return 1;
02331         }
02332         Py_XDECREF(m);
02333     }
02334     #if PY_VERSION_HEX < 0x03060000
02335         Py_Finalize();
02336     #else
02337         if (Py_FinalizeEx() < 0)
02338             return 2;
02339     #endif
02340     return 0;
02341 }
02342 #if PY_MAJOR_VERSION >= 3 && !defined(WIN32) && !defined(MS_WINDOWS)
02343 #include <locale.h>
02344 static wchar_t*
02345 __Pyx_char2wchar(char* arg)
02346 {
02347     wchar_t *res;
02348 #ifdef HAVE_BROKEN_MBSTOWCS
02349     /* Some platforms have a broken implementation of
02350     * mbstowcs which does not count the characters that
02351     * would result from conversion. Use an upper bound.
02352     */
02353     size_t argsize = strlen(arg);
02354 #else
02355     size_t argsize = mbstowcs(NULL, arg, 0);
02356 #endif
02357     size_t count;
02358     unsigned char *in;
02359     wchar_t *out;

```

```

02360 #ifdef HAVE_MBRTOWC
02361     mbstate_t mbs;
02362 #endif
02363     if (argsize != (size_t)-1) {
02364         res = (wchar_t *)malloc((argsize+1)*sizeof(wchar_t));
02365         if (!res)
02366             goto oom;
02367         count = mbstowcs(res, arg, argsize+1);
02368         if (count != (size_t)-1) {
02369             wchar_t *tmp;
02370             /* Only use the result if it contains no
02371              surrogate characters. */
02372             for (tmp = res; *tmp != 0 &&
02373                  (*tmp < 0xd800 || *tmp > 0xdfff); tmp++)
02374                 ;
02375             if (*tmp == 0)
02376                 return res;
02377         }
02378         free(res);
02379     }
02380 #ifdef HAVE_MBRTOWC
02381     /* Overallocate; as multi-byte characters are in the argument, the
02382      actual output could use less memory. */
02383     argsize = strlen(arg) + 1;
02384     res = (wchar_t *)malloc(argsize*sizeof(wchar_t));
02385     if (!res) goto oom;
02386     in = (unsigned char*)arg;
02387     out = res;
02388     memset(&mbs, 0, sizeof mbs);
02389     while (argsize) {
02390         size_t converted = mbrtowc(out, (char*)in, argsize, &mbs);
02391         if (converted == 0)
02392             break;
02393         if (converted == (size_t)-2) {
02394             /* Incomplete character. This should never happen,
02395              since we provide everything that we have -
02396              unless there is a bug in the C library, or I
02397              misunderstood how mbrtowc works. */
02398             fprintf(stderr, "unexpected mbrtowc result -2\\n");
02399             free(res);
02400             return NULL;
02401         }
02402         if (converted == (size_t)-1) {
02403             /* Conversion error. Escape as UTF-8b, and start over
02404              in the initial shift state. */
02405             *out++ = 0xdc00 + *in++;
02406             argsize--;
02407             memset(&mbs, 0, sizeof mbs);
02408             continue;
02409         }
02410         if (*out >= 0xd800 && *out <= 0xdfff) {
02411             /* Surrogate character. Escape the original
02412              byte sequence with surrogateescape. */
02413             argsize -= converted;
02414             while (converted--)
02415                 *out++ = 0xdc00 + *in++;
02416             continue;
02417         }
02418         in += converted;
02419         argsize -= converted;
02420         out++;
02421     }
02422 #else
02423     /* Cannot use C locale for escaping; manually escape as if charset
02424      is ASCII (i.e. escape all bytes > 128. This will still roundtrip
02425      correctly in the locale's charset, which must be an ASCII superset. */
02426     res = (wchar_t *)malloc((strlen(arg)+1)*sizeof(wchar_t));
02427     if (!res) goto oom;
02428     in = (unsigned char*)arg;
02429     out = res;
02430     while(*in)
02431         if(*in < 128)
02432             *out++ = *in++;
02433         else
02434             *out++ = 0xdc00 + *in++;
02435     *out = 0;
02436 #endif

```



```

02437     return res;
02438 oom:
02439     fprintf(stderr, "out of memory\\n");
02440     return NULL;
02441 }
02442 int
02443 main(int argc, char **argv)
02444 {
02445     if (!argc) {
02446         return __Pyx_main(0, NULL);
02447     }
02448     else {
02449         int i, res;
02450         wchar_t **argv_copy = (wchar_t **)malloc(sizeof(wchar_t*)*argc);
02451         wchar_t **argv_copy2 = (wchar_t **)malloc(sizeof(wchar_t*)*argc);
02452         char *oldloc = strdup(setlocale(LC_ALL, NULL));
02453         if (!argv_copy || !argv_copy2 || !oldloc) {
02454             fprintf(stderr, "out of memory\\n");
02455             free(argv_copy);
02456             free(argv_copy2);
02457             free(oldloc);
02458             return 1;
02459         }
02460         res = 0;
02461         setlocale(LC_ALL, "");
02462         for (i = 0; i < argc; i++) {
02463             argv_copy2[i] = argv_copy[i] = __Pyx_char2wchar(argv[i]);
02464             if (!argv_copy[i]) res = 1;
02465         }
02466         setlocale(LC_ALL, oldloc);
02467         free(oldloc);
02468         if (res == 0)
02469             res = __Pyx_main(argc, argv_copy);
02470         for (i = 0; i < argc; i++) {
02471             #if PY_VERSION_HEX < 0x03050000
02472                 free(argv_copy2[i]);
02473             #else
02474                 PyMem_RawFree(argv_copy2[i]);
02475             #endif
02476         }
02477         free(argv_copy);
02478         free(argv_copy2);
02479         return res;
02480     }
02481 }
02482 #endif
02483
02484 /* CIntToPy */
02485 static CYTHON_INLINE PyObject* __Pyx_PyInt_From_long(long value) {
02486 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02487 #pragma GCC diagnostic push
02488 #pragma GCC diagnostic ignored "-Wconversion"
02489 #endif
02490     const long neg_one = (long) -1, const_zero = (long) 0;
02491 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02492 #pragma GCC diagnostic pop
02493 #endif
02494     const int is_unsigned = neg_one > const_zero;
02495     if (is_unsigned) {
02496         if (sizeof(long) < sizeof(unsigned long)) {
02497             return PyInt_FromLong((long) value);
02498         } else if (sizeof(long) <= sizeof(unsigned long)) {
02499             return PyLong_FromUnsignedLong((unsigned long) value);
02500 #ifdef HAVE_LONG_LONG
02501         } else if (sizeof(long) <= sizeof(unsigned PY_LONG_LONG)) {
02502             return PyLong_FromUnsignedLongLong((unsigned PY_LONG_LONG) value);
02503 #endif
02504         }
02505     } else {
02506         if (sizeof(long) <= sizeof(long)) {
02507             return PyInt_FromLong((long) value);
02508 #ifdef HAVE_LONG_LONG
02509         } else if (sizeof(long) <= sizeof(PY_LONG_LONG)) {
02510             return PyLong_FromLongLong((PY_LONG_LONG) value);
02511 #endif
02512         }
02513     }

```

```

02514     {
02515         int one = 1; int little = (int)*(unsigned char *)&one;
02516         unsigned char *bytes = (unsigned char *)&value;
02517         return _PyLong_FromByteArray(bytes, sizeof(long),
02518                                     little, !is_unsigned);
02519     }
02520 }
02521
02522 /* CIntFromPyVerify */
02523 #define __PYX_VERIFY_RETURN_INT(target_type, func_type, func_value)\
02524     __PYX_VERIFY_RETURN_INT(target_type, func_type, func_value, 0)
02525 #define __PYX_VERIFY_RETURN_INT_EXC(target_type, func_type, func_value)\
02526     __PYX_VERIFY_RETURN_INT(target_type, func_type, func_value, 1)
02527 #define __PYX_VERIFY_RETURN_INT(target_type, func_type, func_value, exc)\
02528     {\
02529         func_type value = func_value;\
02530         if (sizeof(target_type) < sizeof(func_type)) {\
02531             if (unlikely(value != (func_type)(target_type) value)) {\
02532                 func_type zero = 0;\
02533                 if (exc && unlikely(value == (func_type)-1 && PyErr_Occurred()))\
02534                     return (target_type) -1;\
02535                 if (is_unsigned && unlikely(value < zero))\
02536                     goto raise_neg_overflow;\
02537                 else\
02538                     goto raise_overflow;\
02539             }\
02540         }\
02541         return (target_type) value;\
02542     }
02543
02544 /* CIntFromPy */
02545 static CYTHON_INLINE long __Pyx_PyInt_As_long(PyObject *x) {
02546 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02547 #pragma GCC diagnostic push
02548 #pragma GCC diagnostic ignored "-Wconversion"
02549 #endif
02550     const long neg_one = (long) -1, const_zero = (long) 0;
02551 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02552 #pragma GCC diagnostic pop
02553 #endif
02554     const int is_unsigned = neg_one > const_zero;
02555 #if PY_MAJOR_VERSION < 3
02556     if (likely(PyInt_Check(x))) {
02557         if (sizeof(long) < sizeof(long)) {
02558             __PYX_VERIFY_RETURN_INT(long, long, PyInt_AS_LONG(x))
02559         } else {
02560             long val = PyInt_AS_LONG(x);
02561             if (is_unsigned && unlikely(val < 0)) {
02562                 goto raise_neg_overflow;
02563             }
02564             return (long) val;
02565         }
02566     } else
02567 #endif
02568     if (likely(PyLong_Check(x))) {
02569         if (is_unsigned) {
02570 #if CYTHON_USE_PYLONG_INTERNALS
02571             const digit* digits = ((PyLongObject*)x)->ob_digit;
02572             switch (Py_SIZE(x)) {
02573                 case 0: return (long) 0;
02574                 case 1: __PYX_VERIFY_RETURN_INT(long, digit, digits[0])
02575                 case 2:
02576                     if (8 * sizeof(long) > 1 * PyLong_SHIFT) {
02577                         if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02578                             __PYX_VERIFY_RETURN_INT(long, unsigned long,
02579                                                         (((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0]))
02579                         } else if (8 * sizeof(long) >= 2 * PyLong_SHIFT) {
02580                             return (long) (((long)digits[1]) << PyLong_SHIFT) |
02581                                     (long)digits[0]);
02582                         }
02583                     }
02584                     break;
02585                 case 3:
02586                     if (8 * sizeof(long) > 2 * PyLong_SHIFT) {
02587                         if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {

```

```

02587         __PYX_VERIFY_RETURN_INT(long, unsigned long,
((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
PyLong_SHIFT) | (unsigned long)digits[0])))
02588     } else if (8 * sizeof(long) >= 3 * PyLong_SHIFT) {
02589         return (long) (((((((long)digits[2]) << PyLong_SHIFT)
| (long)digits[1]) << PyLong_SHIFT) | (long)digits[0]));
02590     }
02591 }
02592 break;
02593 case 4:
02594     if (8 * sizeof(long) > 3 * PyLong_SHIFT) {
02595         if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02596             __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])))
02597         } else if (8 * sizeof(long) >= 4 * PyLong_SHIFT) {
02598             return (long) (((((((((long)digits[3]) <<
PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) |
(long)digits[0]));
02599         }
02600     }
02601     break;
02602 }
02603 #endif
02604 #if CYTHON_COMPILING_IN_CPYTHON
02605     if (unlikely(Py_SIZE(x) < 0)) {
02606         goto raise_neg_overflow;
02607     }
02608 #else
02609     {
02610         int result = PyObject_RichCompareBool(x, Py_False, Py_LT);
02611         if (unlikely(result < 0))
02612             return (long) -1;
02613         if (unlikely(result == 1))
02614             goto raise_neg_overflow;
02615     }
02616 #endif
02617     if (sizeof(long) <= sizeof(unsigned long)) {
02618         __PYX_VERIFY_RETURN_INT_EXC(long, unsigned long,
PyLong_AsUnsignedLong(x))
02619     } #ifdef HAVE_LONG_LONG
    } else if (sizeof(long) <= sizeof(unsigned PY_LONG_LONG)) {
02620         __PYX_VERIFY_RETURN_INT_EXC(long, unsigned PY_LONG_LONG,
PyLong_AsUnsignedLongLong(x))
02621     } #endif
02622 }
02623 } else {
02624     #if CYTHON_USE_PYLONG_INTERNALS
02625         const digit* digits = ((PyLongObject*)x)->ob_digit;
02626         switch (Py_SIZE(x)) {
02627             case 0: return (long) 0;
02628             case -1: __PYX_VERIFY_RETURN_INT(long, sdigit, (sdigit)
(-(sdigit)digits[0]))
02629             case 1: __PYX_VERIFY_RETURN_INT(long, digit, +digits[0])
02630             case -2:
02631                 if (8 * sizeof(long) - 1 > 1 * PyLong_SHIFT) {
02632                     if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02633                         __PYX_VERIFY_RETURN_INT(long, long, -(long)
((((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02634                     } else if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
02635                         return (long) (((long)-1)*((((long)digits[1]) <<
PyLong_SHIFT) | (long)digits[0]));
02636                     }
02637                 }
02638             }
02639             break;
02640             case 2:
02641                 if (8 * sizeof(long) > 1 * PyLong_SHIFT) {
02642                     if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02643                         __PYX_VERIFY_RETURN_INT(long, unsigned long,
((((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02644                     } else if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
02645                         return (long) (((((((long)digits[1]) << PyLong_SHIFT) |
(long)digits[0])));
02646                     }
02647                 }
02648             }
02649             break;
02650             case -3:

```

```

02650         if (8 * sizeof(long) - 1 > 2 * PyLong_SHIFT) {
02651             if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
02652                 __PYX_VERIFY_RETURN_INT(long, long, -(long)
02653 ((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
02654 PyLong_SHIFT) | (unsigned long)digits[0])))
02655             } else if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
02656                 return (long) (((long)-1)*((((((long)digits[2]) <<
02657 PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0])));
02658             }
02659             break;
02660         case 3:
02661             if (8 * sizeof(long) > 2 * PyLong_SHIFT) {
02662                 if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
02663                     __PYX_VERIFY_RETURN_INT(long, unsigned long,
02664 ((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
02665 PyLong_SHIFT) | (unsigned long)digits[0])))
02666                 } else if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
02667                     return (long) ((((((long)digits[2]) << PyLong_SHIFT)
02668 | (long)digits[1]) << PyLong_SHIFT) | (long)digits[0]));
02669                 }
02670             }
02671             break;
02672         case -4:
02673             if (8 * sizeof(long) - 1 > 3 * PyLong_SHIFT) {
02674                 if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02675                     __PYX_VERIFY_RETURN_INT(long, long, -(long)
02676 (((((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) <<
02677 PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02678                 } else if (8 * sizeof(long) - 1 > 4 * PyLong_SHIFT) {
02679                     return (long) ((((((long)-1)*(((((((long)digits[3]) <<
02680 PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) |
02681 (long)digits[0]))));
02682                 }
02683             }
02684             break;
02685         case 4:
02686             if (8 * sizeof(long) > 3 * PyLong_SHIFT) {
02687                 if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02688                     __PYX_VERIFY_RETURN_INT(long, unsigned long,
02689 (((((((((unsigned long)digits[3]) << PyLong_SHIFT) | (unsigned long)digits[2]) <<
02690 PyLong_SHIFT) | (unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02691                 } else if (8 * sizeof(long) - 1 > 4 * PyLong_SHIFT) {
02692                     return (long) ((((((((((long)digits[3]) <<
02693 PyLong_SHIFT) | (long)digits[2]) << PyLong_SHIFT) | (long)digits[1]) << PyLong_SHIFT) |
02694 (long)digits[0]))));
02695                 }
02696             }
02697             break;
02698     #endif
02699     if (sizeof(long) <= sizeof(long)) {
02700         __PYX_VERIFY_RETURN_INT_EXC(long, long, PyLong_AsLong(x))
02701     #ifdef HAVE_LONG_LONG
02702     } else if (sizeof(long) <= sizeof(PY_LONG_LONG)) {
02703         __PYX_VERIFY_RETURN_INT_EXC(long, PY_LONG_LONG,
02704 PyLong_AsLongLong(x))
02705     #endif
02706     }
02707     {
02708     #if CYTHON_COMPILING_IN_PYPY && !defined(_PyLong_AsByteArray)
02709         PyErr_SetString(PyExc_RuntimeError,
02710             "_PyLong_AsByteArray() not available in PyPy, cannot
02711 convert large numbers");
02712     #else
02713         long val;
02714         PyObject *v = __Pyx_PyNumber_IntOrLong(x);
02715         #if PY_MAJOR_VERSION < 3
02716             if (likely(v) && !PyLong_Check(v)) {
02717                 PyObject *tmp = v;
02718                 v = PyNumber_Long(tmp);
02719                 Py_DECREF(tmp);
02720             }
02721         #endif
02722         if (likely(v)) {
02723             int one = 1; int is_little = (int)*(unsigned char *)&one;

```

```

02711         unsigned char *bytes = (unsigned char *)&val;
02712         int ret = _PyLong_AsByteArray((PyLongObject *)v,
02713                                     bytes, sizeof(val),
02714                                     is_little, !is_unsigned);
02715         Py_DECREF(v);
02716         if (likely(!ret))
02717             return val;
02718     }
02719 #endif
02720     return (long) -1;
02721 }
02722 } else {
02723     long val;
02724     PyObject *tmp = __Pyx_PyNumber_IntOrLong(x);
02725     if (!tmp) return (long) -1;
02726     val = Pyx_PyInt_As_long(tmp);
02727     Py_DECREF(tmp);
02728     return val;
02729 }
02730 raise_overflow:
02731     PyErr_SetString(PyExc_OverflowError,
02732                     "value too large to convert to long");
02733     return (long) -1;
02734 raise_neg_overflow:
02735     PyErr_SetString(PyExc_OverflowError,
02736                     "can't convert negative value to long");
02737     return (long) -1;
02738 }
02739
02740 /* CIntFromPy */
02741 static CYTHON_INLINE int __Pyx_PyInt_As_int(PyObject *x) {
02742 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02743 #pragma GCC diagnostic push
02744 #pragma GCC diagnostic ignored "-Wconversion"
02745 #endif
02746     const int neg_one = (int) -1, const_zero = (int) 0;
02747 #ifdef __Pyx_HAS_GCC_DIAGNOSTIC
02748 #pragma GCC diagnostic pop
02749 #endif
02750     const int is_unsigned = neg_one > const_zero;
02751 #if PY_MAJOR_VERSION < 3
02752     if (likely(PyInt_Check(x))) {
02753         if (sizeof(int) < sizeof(long)) {
02754             __PYX_VERIFY_RETURN_INT(int, long, PyInt_AS_LONG(x))
02755         } else {
02756             long val = PyInt_AS_LONG(x);
02757             if (is_unsigned && unlikely(val < 0)) {
02758                 goto raise_neg_overflow;
02759             }
02760             return (int) val;
02761         }
02762     } else
02763 #endif
02764     if (likely(PyLong_Check(x))) {
02765         if (is_unsigned) {
02766 #if CYTHON_USE_PYLONG_INTERNALS
02767             const digit* digits = ((PyLongObject*)x)->ob_digit;
02768             switch (Py_SIZE(x)) {
02769                 case 0: return (int) 0;
02770                 case 1: __PYX_VERIFY_RETURN_INT(int, digit, digits[0])
02771                 case 2:
02772                     if (8 * sizeof(int) > 1 * PyLong_SHIFT) {
02773                         if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02774                             __PYX_VERIFY_RETURN_INT(int, unsigned long,
02775                                                         (((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02776                         } else if (8 * sizeof(int) >= 2 * PyLong_SHIFT) {
02777                             return (int) (((((int)digits[1]) << PyLong_SHIFT) |
02778                                             (int)digits[0]));
02779                         }
02780                     }
02781                     break;
02782                 case 3:
02783                     if (8 * sizeof(int) > 2 * PyLong_SHIFT) {
02784                         if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
02785                             __PYX_VERIFY_RETURN_INT(int, unsigned long,
02786                                                         ((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
02787                                                         PyLong_SHIFT) | (unsigned long)digits[0])))
02788                         }

```

```

02784         } else if (8 * sizeof(int) >= 3 * PyLong_SHIFT) {
02785             return (int) ((((((int)digits[2]) << PyLong_SHIFT) |
(int)digits[1]) << PyLong_SHIFT) | (int)digits[0]));
02786         }
02787     }
02788     break;
02789     case 4:
02790         if (8 * sizeof(int) > 3 * PyLong_SHIFT) {
02791             if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02792                 __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])))
02793             } else if (8 * sizeof(int) >= 4 * PyLong_SHIFT) {
02794                 return (int) (((((((((int)digits[3]) << PyLong_SHIFT)
| (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0]));
02795             }
02796         }
02797         break;
02798     }
02799 #endif
02800 #if CYTHON_COMPILING_IN_CPYTHON
02801     if (unlikely(Py_SIZE(x) < 0)) {
02802         goto raise_neg_overflow;
02803     }
02804 #else
02805     {
02806         int result = PyObject_RichCompareBool(x, Py_False, Py_LT);
02807         if (unlikely(result < 0))
02808             return (int) -1;
02809         if (unlikely(result == 1))
02810             goto raise_neg_overflow;
02811     }
02812 #endif
02813     if (sizeof(int) <= sizeof(unsigned long)) {
02814         __PYX_VERIFY_RETURN_INT_EXC(int, unsigned long,
PyLong_AsUnsignedLong(x))
02815     } #ifdef HAVE_LONG_LONG
02816     } else if (sizeof(int) <= sizeof(unsigned PY_LONG_LONG)) {
02817         __PYX_VERIFY_RETURN_INT_EXC(int, unsigned PY_LONG_LONG,
PyLong_AsUnsignedLongLong(x))
02818     } #endif
02819     }
02820     } else {
02821     #if CYTHON_USE_PYLONG_INTERNALS
02822         const digit* digits = ((PyLongObject*)x)->ob_digit;
02823         switch (Py_SIZE(x)) {
02824             case 0: return (int) 0;
02825             case -1: __PYX_VERIFY_RETURN_INT(int, sdigit, (sdigit)
(-(sdigit)digits[0]))
02826             case 1: __PYX_VERIFY_RETURN_INT(int, digit, +digits[0])
02827             case -2:
02828                 if (8 * sizeof(int) - 1 > 1 * PyLong_SHIFT) {
02829                     if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02830                         __PYX_VERIFY_RETURN_INT(int, long, -(long)
(((((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02831                     } else if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
02832                         return (int) (((int)-1)*((((((int)digits[1]) <<
PyLong_SHIFT) | (int)digits[0])));
02833                     }
02834                 }
02835                 break;
02836             case 2:
02837                 if (8 * sizeof(int) > 1 * PyLong_SHIFT) {
02838                     if (8 * sizeof(unsigned long) > 2 * PyLong_SHIFT) {
02839                         __PYX_VERIFY_RETURN_INT(int, unsigned long,
(((((((unsigned long)digits[1]) << PyLong_SHIFT) | (unsigned long)digits[0])))
02840                     } else if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
02841                         return (int) (((((((((int)digits[1]) << PyLong_SHIFT) |
(int)digits[0])));
02842                     }
02843                 }
02844                 break;
02845             case -3:
02846                 if (8 * sizeof(int) - 1 > 2 * PyLong_SHIFT) {
02847                     if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {

```

```

02848         __PYX_VERIFY_RETURN_INT(int, long, -(long)
((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
PyLong_SHIFT) | (unsigned long)digits[0])))
02849     } else if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
02850         return (int) (((int)-1)*((((((int)digits[2]) <<
PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0]]));
02851     }
02852 }
02853 break;
02854 case 3:
02855     if (8 * sizeof(int) > 2 * PyLong_SHIFT) {
02856         if (8 * sizeof(unsigned long) > 3 * PyLong_SHIFT) {
02857             __PYX_VERIFY_RETURN_INT(int, unsigned long,
((((((unsigned long)digits[2]) << PyLong_SHIFT) | (unsigned long)digits[1]) <<
PyLong_SHIFT) | (unsigned long)digits[0])))
02858         } else if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
02859             return (int) (((((((((int)digits[2]) << PyLong_SHIFT) |
(int)digits[1]) << PyLong_SHIFT) | (int)digits[0]]));
02860         }
02861     }
02862 break;
02863 case -4:
02864     if (8 * sizeof(int) - 1 > 3 * PyLong_SHIFT) {
02865         if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02866             __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])))
02867         } else if (8 * sizeof(int) - 1 > 4 * PyLong_SHIFT) {
02868             return (int) (((int)-1)*((((((((((int)digits[3]) <<
PyLong_SHIFT) | (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) |
(int)digits[0]]));
02869         }
02870     }
02871 break;
02872 case 4:
02873     if (8 * sizeof(int) > 3 * PyLong_SHIFT) {
02874         if (8 * sizeof(unsigned long) > 4 * PyLong_SHIFT) {
02875             __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])))
02876         } else if (8 * sizeof(int) - 1 > 4 * PyLong_SHIFT) {
02877             return (int) (((((((((((((int)digits[3]) << PyLong_SHIFT)
| (int)digits[2]) << PyLong_SHIFT) | (int)digits[1]) << PyLong_SHIFT) | (int)digits[0]]));
02878         }
02879     }
02880 break;
02881 }
02882 #endif
02883 if (sizeof(int) <= sizeof(long)) {
02884     __PYX_VERIFY_RETURN_INT_EXC(int, long, PyLong_AsLong(x))
02885 #ifdef HAVE_LONG_LONG
02886 } else if (sizeof(int) <= sizeof(PY_LONG_LONG)) {
02887     __PYX_VERIFY_RETURN_INT_EXC(int, PY_LONG_LONG,
PyLong_AsLongLong(x))
02888 #endif
02889 }
02890 }
02891 {
02892 #if CYTHON_COMPILING_IN_PYPY && !defined(_PyLong_AsByteArray)
02893     PyErr_SetString(PyExc_RuntimeError,
02894         "_PyLong_AsByteArray() not available in PyPy, cannot
convert large numbers");
02895 #else
02896     int val;
02897     PyObject *v = __Pyx_PyNumber_IntOrLong(x);
02898     #if PY_MAJOR_VERSION < 3
02899         if (likely(v) && !PyLong_Check(v)) {
02900             PyObject *tmp = v;
02901             v = PyNumber_Long(tmp);
02902             Py_DECREF(tmp);
02903         }
02904     #endif
02905     if (likely(v)) {
02906         int one = 1; int is_little = (int)*(unsigned char *)&one;
02907         unsigned char *bytes = (unsigned char *)&val;
02908         int ret = _PyLong_AsByteArray((PyLongObject *)v,
02909             bytes, sizeof(val),

```

```

02910                                     is_little, !is_unsigned);
02911         Py_DECREF(v);
02912         if (likely(!ret))
02913             return val;
02914     }
02915 #endif
02916     return (int) -1;
02917 }
02918 } else {
02919     int val;
02920     PyObject *tmp = __Pyx_PyNumber_IntOrLong(x);
02921     if (!tmp) return (int) -1;
02922     val = __Pyx_PyInt_As_int(tmp);
02923     Py_DECREF(tmp);
02924     return val;
02925 }
02926 raise_overflow:
02927     PyErr_SetString(PyExc_OverflowError,
02928         "value too large to convert to int");
02929     return (int) -1;
02930 raise_neg_overflow:
02931     PyErr_SetString(PyExc_OverflowError,
02932         "can't convert negative value to int");
02933     return (int) -1;
02934 }
02935
02936 /* FastTypeChecks */
02937 #if CYTHON_COMPILING_IN_CPYTHON
02938 static int __Pyx_InBases(PyTypeObject *a, PyTypeObject *b) {
02939     while (a) {
02940         a = a->tp_base;
02941         if (a == b)
02942             return 1;
02943     }
02944     return b == &PyBaseObject_Type;
02945 }
02946 static CYTHON_INLINE int __Pyx_IsSubtype(PyTypeObject *a, PyTypeObject *b) {
02947     PyObject *mro;
02948     if (a == b) return 1;
02949     mro = a->tp_mro;
02950     if (likely(mro)) {
02951         Py_ssize_t i, n;
02952         n = PyTuple_GET_SIZE(mro);
02953         for (i = 0; i < n; i++) {
02954             if (PyTuple_GET_ITEM(mro, i) == (PyObject *)b)
02955                 return 1;
02956         }
02957         return 0;
02958     }
02959     return __Pyx_InBases(a, b);
02960 }
02961 #if PY_MAJOR_VERSION == 2
02962 static int __Pyx_inner_PyErr_GivenExceptionMatches2(PyObject *err, PyObject*
exc_type1, PyObject* exc_type2) {
02963     PyObject *exception, *value, *tb;
02964     int res;
02965     __Pyx_PyThreadState_declare
02966     __Pyx_PyThreadState_assign
02967     __Pyx_ErrFetch(&exception, &value, &tb);
02968     res = exc_type1 ? PyObject_IsSubclass(err, exc_type1) : 0;
02969     if (unlikely(res == -1)) {
02970         PyErr_WriteUnraisable(err);
02971         res = 0;
02972     }
02973     if (!res) {
02974         res = PyObject_IsSubclass(err, exc_type2);
02975         if (unlikely(res == -1)) {
02976             PyErr_WriteUnraisable(err);
02977             res = 0;
02978         }
02979     }
02980     __Pyx_ErrRestore(exception, value, tb);
02981     return res;
02982 }
02983 #else
02984 static CYTHON_INLINE int __Pyx_inner_PyErr_GivenExceptionMatches2(PyObject *err,
PyObject* exc_type1, PyObject *exc_type2) {

```



```

02985     int res = exc_type1 ? __Pyx_IsSubtype((PyTypeObject*)err,
(PyTypeObject*)exc_type1) : 0;
02986     if (!res) {
02987         res = __Pyx_IsSubtype((PyTypeObject*)err, (PyTypeObject*)exc_type2);
02988     }
02989     return res;
02990 }
02991 #endif
02992 static int __Pyx_PyErr_GivenExceptionMatchesTuple(PyObject *exc_type, PyObject
*tuple) {
02993     Py_ssize_t i, n;
02994     assert(PyExceptionClass_Check(exc_type));
02995     n = PyTuple_GET_SIZE(tuple);
02996     #if PY_MAJOR_VERSION >= 3
02997     for (i=0; i<n; i++) {
02998         if (exc_type == PyTuple_GET_ITEM(tuple, i)) return 1;
02999     }
03000 #endif
03001     for (i=0; i<n; i++) {
03002         PyObject *t = PyTuple_GET_ITEM(tuple, i);
03003         #if PY_MAJOR_VERSION < 3
03004         if (likely(exc_type == t)) return 1;
03005         #endif
03006         if (likely(PyExceptionClass_Check(t))) {
03007             if (__Pyx_inner_PyErr_GivenExceptionMatches2(exc_type, NULL, t))
return 1;
03008         } else {
03009         }
03010     }
03011     return 0;
03012 }
03013 static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches(PyObject *err, PyObject*
exc_type) {
03014     if (likely(err == exc_type)) return 1;
03015     if (likely(PyExceptionClass_Check(err))) {
03016         if (likely(PyExceptionClass_Check(exc_type))) {
03017             return __Pyx_inner_PyErr_GivenExceptionMatches2(err, NULL, exc_type);
03018         } else if (likely(PyTuple_Check(exc_type))) {
03019             return __Pyx_PyErr_GivenExceptionMatchesTuple(err, exc_type);
03020         } else {
03021         }
03022     }
03023     return PyErr_GivenExceptionMatches(err, exc_type);
03024 }
03025 static CYTHON_INLINE int __Pyx_PyErr_GivenExceptionMatches2(PyObject *err, PyObject
*exc_type1, PyObject *exc_type2) {
03026     assert(PyExceptionClass_Check(exc_type1));
03027     assert(PyExceptionClass_Check(exc_type2));
03028     if (likely(err == exc_type1 || err == exc_type2)) return 1;
03029     if (likely(PyExceptionClass_Check(err))) {
03030         return __Pyx_inner_PyErr_GivenExceptionMatches2(err, exc_type1,
exc_type2);
03031     }
03032     return (PyErr_GivenExceptionMatches(err, exc_type1) ||
PyErr_GivenExceptionMatches(err, exc_type2));
03033 }
03034 #endif
03035
03036 /* CheckBinaryVersion */
03037 static int __Pyx_check_binary_version(void) {
03038     char ctversion[5];
03039     int same=1, i, found_dot;
03040     const char* rt_from_call = Py_GetVersion();
03041     PyOS_snprintf(ctversion, 5, "%d.%d", PY_MAJOR_VERSION, PY_MINOR_VERSION);
03042     found_dot = 0;
03043     for (i = 0; i < 4; i++) {
03044         if (!ctversion[i]) {
03045             same = (rt_from_call[i] < '0' || rt_from_call[i] > '9');
03046             break;
03047         }
03048         if (rt_from_call[i] != ctversion[i]) {
03049             same = 0;
03050             break;
03051         }
03052     }
03053     if (!same) {
03054         char rtversion[5] = {'\0'};

```

```

03055     char message[200];
03056     for (i=0; i<4; ++i) {
03057         if (rt_from_call[i] == '.') {
03058             if (found_dot) break;
03059             found_dot = 1;
03060         } else if (rt_from_call[i] < '0' || rt_from_call[i] > '9') {
03061             break;
03062         }
03063         rtversion[i] = rt_from_call[i];
03064     }
03065     PyOS_snprintf(message, sizeof(message),
03066                  "compiletime version %s of module '%.100s' "
03067                  "does not match runtime version %s",
03068                  ctversion, __Pyx_MODULE_NAME, rtversion);
03069     return PyErr_WarnEx(NULL, message, 1);
03070 }
03071 return 0;
03072 }
03073
03074 /* InitStrings */
03075 static int __Pyx_InitStrings(__Pyx_StringTabEntry *t) {
03076     while (t->p) {
03077         #if PY_MAJOR_VERSION < 3
03078         if (t->is_unicode) {
03079             *t->p = PyUnicode_DecodeUTF8(t->s, t->n - 1, NULL);
03080         } else if (t->intern) {
03081             *t->p = PyString_InternFromString(t->s);
03082         } else {
03083             *t->p = PyString_FromStringAndSize(t->s, t->n - 1);
03084         }
03085         #else
03086         if (t->is_unicode | t->is_str) {
03087             if (t->intern) {
03088                 *t->p = PyUnicode_InternFromString(t->s);
03089             } else if (t->encoding) {
03090                 *t->p = PyUnicode_Decode(t->s, t->n - 1, t->encoding, NULL);
03091             } else {
03092                 *t->p = PyUnicode_FromStringAndSize(t->s, t->n - 1);
03093             }
03094         } else {
03095             *t->p = PyBytes_FromStringAndSize(t->s, t->n - 1);
03096         }
03097         #endif
03098         if (!*t->p)
03099             return -1;
03100         if (PyObject_Hash(*t->p) == -1)
03101             return -1;
03102         ++t;
03103     }
03104     return 0;
03105 }
03106
03107 static CYTHON_INLINE PyObject* __Pyx_PyUnicode_FromString(const char* c_str) {
03108     return __Pyx_PyUnicode_FromStringAndSize(c_str, (Py_ssize_t)strlen(c_str));
03109 }
03110 static CYTHON_INLINE const char* __Pyx_PyObject_AsString(PyObject* o) {
03111     Py_ssize_t ignore;
03112     return __Pyx_PyObject_AsStringAndSize(o, &ignore);
03113 }
03114 #if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII ||
03115     __PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
03116 #if !CYTHON_PEP393_ENABLED
03117 static const char* __Pyx_PyUnicode_AsStringAndSize(PyObject* o, Py_ssize_t *length)
03118 {
03119     char* defenc_c;
03120     PyObject* defenc = _PyUnicode_AsDefaultEncodedString(o, NULL);
03121     if (!defenc) return NULL;
03122     defenc_c = PyBytes_AS_STRING(defenc);
03123     #if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
03124     {
03125         char* end = defenc_c + PyBytes_GET_SIZE(defenc);
03126         char* c;
03127         for (c = defenc_c; c < end; c++) {
03128             if ((unsigned char) (*c) >= 128) {
03129                 PyUnicode_AsASCIIString(o);
03130                 return NULL;
03131             }
03132         }
03133     }
03134     #endif

```

```

03130     }
03131     }
03132 #endif
03133     *length = PyBytes_GET_SIZE(defenc);
03134     return defenc_c;
03135 }
03136 #else
03137 static CYTHON_INLINE const char* __Pyx_PyUnicode_AsStringAndSize(PyObject* o,
Py_ssize_t *length) {
03138     if (unlikely(__Pyx_PyUnicode_READY(o) == -1)) return NULL;
03139 #if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
03140     if (likely(PyUnicode_IS_ASCII(o))) {
03141         *length = PyUnicode_GET_LENGTH(o);
03142         return PyUnicode_AsUTF8(o);
03143     } else {
03144         PyUnicode AsASCIIString(o);
03145         return NULL;
03146     }
03147 #else
03148     return PyUnicode_AsUTF8AndSize(o, length);
03149 #endif
03150 }
03151 #endif
03152 #endif
03153 static CYTHON_INLINE const char* __Pyx_PyObject_AsStringAndSize(PyObject* o,
Py_ssize_t *length) {
03154 #if __PYX_DEFAULT_STRING_ENCODING_IS_ASCII ||
__PYX_DEFAULT_STRING_ENCODING_IS_DEFAULT
03155     if (
03156 #if PY_MAJOR_VERSION < 3 && __PYX_DEFAULT_STRING_ENCODING_IS_ASCII
__Pyx_sys_getdefaultencoding_not_ascii &&
03157 #endif
03158         PyUnicode_Check(o)) {
03159         return __Pyx_PyUnicode_AsStringAndSize(o, length);
03160     } else
03161 #endif
03162 #if (!CYTHON_COMPILING_IN_PYPY) || (defined(PyByteArray_AS_STRING) &&
defined(PyByteArray_GET_SIZE))
03163     if (PyByteArray_Check(o)) {
03164         *length = PyByteArray_GET_SIZE(o);
03165         return PyByteArray_AS_STRING(o);
03166     } else
03167 #endif
03168 #endif
03169     {
03170         char* result;
03171         int r = PyBytes_AsStringAndSize(o, &result, length);
03172         if (unlikely(r < 0)) {
03173             return NULL;
03174         } else {
03175             return result;
03176         }
03177     }
03178 }
03179 static CYTHON_INLINE int __Pyx_PyObject_IsTrue(PyObject* x) {
03180     int is_true = x == Py_True;
03181     if (is_true | (x == Py_False) | (x == Py_None)) return is_true;
03182     else return PyObject_IsTrue(x);
03183 }
03184 static CYTHON_INLINE int __Pyx_PyObject_IsTrueAndDecref(PyObject* x) {
03185     int retval;
03186     if (unlikely(!x)) return -1;
03187     retval = __Pyx_PyObject_IsTrue(x);
03188     Py_DECREF(x);
03189     return retval;
03190 }
03191 static PyObject* __Pyx_PyNumber_IntOrLongWrongResultType(PyObject* result, const
char* type_name) {
03192 #if PY_MAJOR_VERSION >= 3
03193     if (PyLong_Check(result)) {
03194         if (PyErr_WarnFormat(PyExc_DeprecationWarning, 1,
03195             "__int__ returned non-int (type %.200s). "
03196             "The ability to return an instance of a strict subclass of int "
03197             "is deprecated, and may be removed in a future version of Python.",
03198             Py_TYPE(result)->tp_name)) {
03199             Py_DECREF(result);
03200             return NULL;
03201         }
03202     }

```

```

03202         return result;
03203     }
03204 #endif
03205     PyErr_Format(PyExc_TypeError,
03206                 "___%.4s__ returned non-%.4s (type %.200s)",
03207                 type_name, type_name, Py_TYPE(result)->tp_name);
03208     Py_DECREF(result);
03209     return NULL;
03210 }
03211 static CYTHON_INLINE PyObject* __Pyx_PyNumber_IntOrLong(PyObject* x) {
03212     #if CYTHON_USE_TYPE_SLOTS
03213         PyNumberMethods *m;
03214     #endif
03215     const char *name = NULL;
03216     PyObject *res = NULL;
03217     #if PY_MAJOR_VERSION < 3
03218     if (likely(PyInt_Check(x) || PyLong_Check(x)))
03219     #else
03220     if (likely(PyLong_Check(x)))
03221     #endif
03222         return __Pyx_NewRef(x);
03223     #if CYTHON_USE_TYPE_SLOTS
03224     m = Py_TYPE(x)->tp_as_number;
03225     #if PY_MAJOR_VERSION < 3
03226     if (m && m->nb_int) {
03227         name = "int";
03228         res = m->nb_int(x);
03229     }
03230     else if (m && m->nb_long) {
03231         name = "long";
03232         res = m->nb_long(x);
03233     }
03234     #else
03235     if (likely(m && m->nb_int)) {
03236         name = "int";
03237         res = m->nb_int(x);
03238     }
03239     #endif
03240     #else
03241     if (!PyBytes_CheckExact(x) && !PyUnicode_CheckExact(x)) {
03242         res = PyNumber_Int(x);
03243     }
03244     #endif
03245     if (likely(res)) {
03246     #if PY_MAJOR_VERSION < 3
03247         if (unlikely(!PyInt_Check(res) && !PyLong_Check(res))) {
03248     #else
03249         if (unlikely(!PyLong_CheckExact(res))) {
03250     #endif
03251             return __Pyx_PyNumber_IntOrLongWrongResultType(res, name);
03252         }
03253     }
03254     else if (!PyErr_Occurred()) {
03255         PyErr_SetString(PyExc_TypeError,
03256                         "an integer is required");
03257     }
03258     return res;
03259 }
03260 static CYTHON_INLINE Py_ssize_t __Pyx_PyIndex_AsSsize_t(PyObject* b) {
03261     Py_ssize_t ival;
03262     PyObject *x;
03263     #if PY_MAJOR_VERSION < 3
03264     if (likely(PyInt_CheckExact(b))) {
03265         if (sizeof(Py_ssize_t) >= sizeof(long))
03266             return PyInt_AS_LONG(b);
03267         else
03268             return PyInt_AsSsize_t(b);
03269     }
03270     #endif
03271     if (likely(PyLong_CheckExact(b))) {
03272         #if CYTHON_USE_PYLONG_INTERNALS
03273         const digit* digits = ((PyLongObject*)b)->ob_digit;
03274         const Py_ssize_t size = Py_SIZE(b);
03275         if (likely(__Pyx_sst_abs(size) <= 1)) {
03276             ival = likely(size) ? digits[0] : 0;
03277             if (size == -1) ival = -ival;
03278             return ival;

```

```

03279     } else {
03280         switch (size) {
03281             case 2:
03282                 if (8 * sizeof(Py_ssize_t) > 2 * PyLong_SHIFT) {
03283                     return (Py_ssize_t) (((size_t)digits[1]) << PyLong_SHIFT) |
03284                     (size_t)digits[0]);
03285                 break;
03286             case -2:
03287                 if (8 * sizeof(Py_ssize_t) > 2 * PyLong_SHIFT) {
03288                     return -(Py_ssize_t) (((size_t)digits[1]) << PyLong_SHIFT) |
03289                     (size_t)digits[0]);
03290                 break;
03291             case 3:
03292                 if (8 * sizeof(Py_ssize_t) > 3 * PyLong_SHIFT) {
03293                     return (Py_ssize_t) (((size_t)digits[2]) << PyLong_SHIFT) |
03294                     (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]);
03295                 break;
03296             case -3:
03297                 if (8 * sizeof(Py_ssize_t) > 3 * PyLong_SHIFT) {
03298                     return -(Py_ssize_t) (((size_t)digits[2]) << PyLong_SHIFT) |
03299                     (size_t)digits[1]) << PyLong_SHIFT) | (size_t)digits[0]);
03300                 break;
03301             case 4:
03302                 if (8 * sizeof(Py_ssize_t) > 4 * PyLong_SHIFT) {
03303                     return (Py_ssize_t) (((size_t)digits[3]) << PyLong_SHIFT) |
03304                     (size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) |
03305                     (size_t)digits[0]);
03306                 break;
03307             case -4:
03308                 if (8 * sizeof(Py_ssize_t) > 4 * PyLong_SHIFT) {
03309                     return -(Py_ssize_t) (((size_t)digits[3]) << PyLong_SHIFT) |
03310                     (size_t)digits[2]) << PyLong_SHIFT) | (size_t)digits[1]) << PyLong_SHIFT) |
03311                     (size_t)digits[0]);
03312                 break;
03313             }
03314         }
03315         #endif
03316         return PyLong_AsSsize_t(b);
03317     }
03318     x = PyNumber_Index(b);
03319     if (!x) return -1;
03320     ival = PyInt_AsSsize_t(x);
03321     Py_DECREF(x);
03322     return ival;
03323 }
03324 static CYTHON_INLINE Py_hash_t __Pyx_PyIndex_AsHash_t(PyObject* o) {
03325     if (sizeof(Py_hash_t) == sizeof(Py_ssize_t)) {
03326         return (Py_hash_t) __Pyx_PyIndex_AsSsize_t(o);
03327     }
03328     #if PY_MAJOR_VERSION < 3
03329     } else if (likely(PyInt_CheckExact(o))) {
03330         return PyInt_AS_LONG(o);
03331     }
03332     #endif
03333     } else {
03334         Py_ssize_t ival;
03335         PyObject *x;
03336         x = PyNumber_Index(o);
03337         if (!x) return -1;
03338         ival = PyInt_AsLong(x);
03339         Py_DECREF(x);
03340         return ival;
03341     }
03342 }
03343 static CYTHON_INLINE PyObject * __Pyx_PyBool_FromLong(long b) {
03344     return b ? __Pyx_NewRef(Py_True) : __Pyx_NewRef(Py_False);
03345 }
03346 static CYTHON_INLINE PyObject * __Pyx_PyInt_FromSize_t(size_t ival) {
03347     return PyInt_FromSize_t(ival);
03348 }
03349 #endif /* Py_PYTHON_H */

```



## **\_main\_.py**

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 from Initializer import initializer
00005
00006 initializer()
```

## AuthUtil.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import ctypes
00005 import datetime
00006 import json
00007 import os
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 from cryptography.fernet import Fernet
00012
00013 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00014 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00015 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00016
00017
00018 class AuthUtil:
00019
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.
00025
00026         Args:
00027             self: Represent the instance of the class
00028
00029         Returns:
00030             None
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034         """
00035         self.StandardStatus = None
00036         self.ListedOrModified = None
00037         self.file_name = None
00038         self.append_file = None
00039         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00040         self.filePath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath("Security
")
00041         self.k = None
00042         self.keyFlag = True
00043         self.jsonDict = {}
00044         self.passFlagUre = False
00045         self.passFlagCm = False
00046         self.outcomeText = "Please input the plain text keys in the input boxes above
\n " \
00047
00048         "Submitting will overwrite any old values in an
unrecoverable manner."
00049
00050         if os.path.exists(self.filePath):
00051             pass
00052         else:
00053             if
os.path.exists(Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData")):
00054                 os.mkdir(self.filePath)
00055             else:
00056                 os.mkdir(Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData"))
00057                 os.mkdir(self.filePath)
00058
00059         if os.path.exists(self.keyPath):
00060             pass
00061         else:
00062             if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00063                 os.mkdir(self.keyPath)
00064             else:
```



```

00064         os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))
00065                    os.mkdir(self.keyPath)
00066
00067         if
00068         os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00069             try:
00070                 f =
00071                 open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00072                 self.k = f.readline()
00073                 f.close()
00074             except Exception as e:
00075                 print(e)
00076                 RESTError(402)
00077                 raise SystemExit(402)
00078         else:
00079             self.k = Fernet.generate_key()
00080             f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"),
00081                    "wb")
00082             f.write(self.k)
00083             f.close()
00084
00085             try:
00086                 os.remove(self.filePath.joinpath("auth.json"))
00087             except Exception as e:
00088                 # Logging
00089                 print(
00090                     f"{datetime.datetime.today().strftime('%m-%d-%Y
00091 %H:%M:%S.%f')}[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file - This
00092 can be due to the file not existing. Continuing...")
00093                 pass
00094
00095             f = open(self.filePath.joinpath("auth.json"), "wb")
00096             f.close()
00097             self.keyFlag = False
00098
00099         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00100
00101         try:
00102             ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3
00103 ra3rvavcr3w"), 2)
00104         except Exception as e:
00105             # Logging
00106             print(
00107                 f"{datetime.datetime.today().strftime('%m-%d-%Y
00108 %H:%M:%S.%f')}[:-3]} | Authutil.py | Error = {e} | Error when setting the key file as hidden.
00109 This is either a Permission error or Input Error. Continuing...")
00110             pass
00111
00112         def __SetValues(self, values):
00113
00114             """
00115             The __SetValues function is called when the user clicks on the "OK"
00116             button in the window.
00117             It takes a dictionary of values as an argument, and then uses those values to
00118             update
00119             the auth.json file with new keys for both Utah Real Estate and Construction
00120             Monitor.
00121
00122             Args:
00123                 self: Make the function a method of the class
00124                 values: Store the values that are entered into the form
00125
00126             Returns:
00127                 A dictionary of the values entered by the user
00128
00129             Doc Author:
00130                 Willem van der Schans, Trelent AI
00131
00132             """
00133             ureCurrent = None
00134             cmCurrent = None
00135             keyFile = None
00136             self.popupFlag = False
00137
00138             fernet = Fernet(self.k)
00139
00140             try:

```

```

00129         f = open(self.filePath.joinpath("auth.json"), "r")
00130         keyFile = json.load(f)
00131         fileFlag = True
00132     except:
00133         fileFlag = False
00134
00135     # Try initial decoding, if fails pass and write new keys and files
00136     if fileFlag:
00137         try:
00138             ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00139         except Exception as e:
00140             # Logging
00141             print(
00142                 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")
00143             ureCurrent = None
00144
00145         try:
00146             cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00147         except Exception as e:
00148             # Logging
00149             print(
00150                 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")
00151             cmCurrent = None
00152
00153     if values["-ureAuth-"] != "":
00154         self.jsonDict.update(
00155             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00156         self.passFlagUre = True
00157     elif ureCurrent is not None:
00158         self.jsonDict.update(
00159             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00160         self.passFlagUre = True
00161     else:
00162         pass
00163
00164     if values["-cmAuth-"] != "":
00165         if values["-cmAuth-"].startswith("Basic"):
00166             self.jsonDict.update(
00167                 {"cm": {"parameter": "Authorization",
00168                     "auth":
fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00169             self.passFlagCm = True
00170         else:
00171             PopupWrapped("Please make sure you provide a HTTP Basic Auth key for
construction Monitor",
00172                         windowType="AuthError")
00173             self.popupFlag = True
00174             pass
00175     elif ureCurrent is not None:
00176         self.jsonDict.update(
00177             {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00178         self.passFlagUre = True
00179     else:
00180         pass
00181
00182     if not self.passFlagUre and not self.passFlagCm:
00183         PopupWrapped("Please make sure you provide keys for both Utah Real estate
and Construction Monitor",
00184                     windowType="errorLarge")
00185     if self.passFlagCm and not self.passFlagUre:
00186         PopupWrapped("Please make sure you provide a key for Utah Real estate",
00187                     windowType="errorLarge")
00187     if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
00188         PopupWrapped("Please make sure you provide a key for Construction
Monitor", windowType="errorLarge")
00189     if self.popupFlag:
00190         pass
00191     else:
00192         jsonOut = json.dumps(self.jsonDict, indent=4)
00193         f = open(self.filePath.joinpath("auth.json"), "w")

```

```

00194         f.write(jsonOut)
00195
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
00202         argument
00203         is simply what will be displayed at the top of the window.
00204
00205         Args:
00206             self: Represent the instance of the class
00207             layout: Pass the layout of the gui to be displayed
00208             text: Set the title of the window
00209
00210         Returns:
00211             A window object
00212         """
00213         window = sg.Window(text, layout, grab_anywhere=False,
00214                             return_keyboard_events=True,
00215                             finalize=True,
00216                             icon=ImageLoader("taskbar_icon.ico"))
00217
00218         while not self.passFlagUre or not self.passFlagCm:
00219             event, values = window.read()
00220
00221             if event == "Submit":
00222                 try:
00223                     self.__SetValues(values)
00224                 except Exception as e:
00225                     print(e)
00226                     RESError(993)
00227                 finally:
00228                     pass
00229             elif event == sg.WIN_CLOSED or event == "Quit":
00230                 break
00231             else:
00232                 pass
00233
00234         window.close()
00235
00236     def __CreateFrame(self):
00237
00238         """
00239         The __CreateFrame function creates the GUI layout for the Authentication Utility.
00240         It is called by __init__ and returns a list of lists that contains all the elements
00241         that will be displayed in the window.
00242
00243         Args:
00244             self: Access the class attributes and methods
00245
00246         Returns:
00247             A list of lists
00248
00249         Doc Author:
00250             Trelent
00251         """
00252         sg.theme('Default1')
00253
00254         line00 = [sg.HSeparator()]
00255
00256         line0 = [sg.Image(ImageLoader("logo.png")),
00257                  sg.Push(),
00258                  sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
00259                          justification="center"),
00260                  sg.Push(),
00261                  sg.Push()]
00262
00263         line1 = [sg.HSeparator()]
00264
00265         line2 = [sg.Push(),
00266                  sg.Text("Utah Real Estate API Key: ", justification="center"),
00267                  sg.Push()]
00268
00269         line3 = [sg.Push(),
00270                  sg.Input(default_text="123", key="-ureAuth-", disabled=False,

```

```

00268             size=(40, 1)),
00269             sg.Push()]
00270
00271         line4 = [sg.HSeparator()]
00272
00273         line5 = [sg.Push(),
00274                 sg.Text("Construction Monitor HTTP BASIC Key: ",
00275                 justification="center"),
00276                 sg.Push()]
00277
00278         line6 = [sg.Push(),
00279                 sg.Input(default_text="Basic 123", key="-cmAuth-",
00280                 disabled=False,
00281                 size=(40, 1)),
00282                 sg.Push()]
00283
00284         line7 = [sg.HSeparator()]
00285
00286         line8 = [sg.Push(),
00287                 sg.Text(self.outcomeText, justification="center"),
00288                 sg.Push()]
00289
00290         line9 = [sg.HSeparator()]
00291
00292         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00293
00294         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
00295                 line8, line9, line10]
00296
00297         return layout

```

## BatchProcessing.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import math
00006 from datetime import date
00007
00008 import pandas as pd
00009 import requests
00010
00011 from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList
00012
00013
00014 def BatchCalculator(TotalRecords, Argument_Dict):
00015     """
00016     The BatchCalculator function takes two arguments:
00017     1. TotalRecords - the total number of records in the database
00018     2. Argument_Dict - a dictionary containing all the arguments passed to this
    function by the user
00019
00020     Args:
00021         TotalRecords: Determine the number of batches that will be needed to complete
    the query
00022         Argument_Dict: Pass in the arguments that will be used to query the database
00023
00024     Returns:
00025         The total number of batches that will be made
00026
00027     Doc Author:
00028         Willem van der Schans, Trelent AI
00029     """
00030     try:
00031         document_limit = Argument_Dict["size"]
00032     except Exception as e:
00033         # Logging
00034         print(
00035             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")
00036         document_limit = 200
00037
00038     return int(math.ceil(float(TotalRecords) / float(document_limit)))
00039
00040
00041 class BatchProcessorConstructionMonitor:
00042
00043     def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict,
    ColumnSelection, valueObject):
00044
00045         """
00046         The __init__ function is the constructor for a class. It is called when an object
    of that class
00047         is created, and it sets up the attributes of that object. In this case, we are
    setting up our
00048         object to have a dataframe attribute (which will be used to store all of our data),
    as well as
00049         attributes for each parameter in our ReST call.
00050
00051     Args:
00052         self: Represent the instance of the class
00053         RestDomain: Specify the domain of the rest api
00054         NumBatches: Determine how many batches of data to retrieve
00055         ParameterDict: Pass in the parameters that will be used to make the api call
00056         HeaderDict: Pass the header dictionary from the main function to this class
00057         ColumnSelection: Determine which columns to pull from the api
00058         valueObject: Pass in the value object that is used to determine what values
    are returned
00059
00060     Returns:
00061         An object of the class
00062
00063     Doc Author:
```

```

00064         Willem van der Schans, Trelent AI
00065         """
00066         self.dataframe = None
00067         self.__numBatches = NumBatches
00068         self.__parameterDict = ParameterDict
00069         self.__restDomain = RestDomain
00070         self.__headerDict = HeaderDict
00071         self.__columnSelection = ColumnSelection
00072         self.valueObject = valueObject
00073         self.__maxRequests = 10000
00074         self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])))
00075         self.__requestCalls = math.ceil(self.__maxRequests /
int(self.__parameterDict['size']))
00076         self.__dateTracker = None
00077
00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes
it to the ConstructionMonitorProcessor function.
00081         The ConstructionMonitorProcessor function then uses this valueObject to
determine which of its functions should be called.
00082
00083         Args:
00084             self: Represent the instance of the class
00085
00086         Returns:
00087             The result of the constructionmonitorprocessor function
00088
00089         Doc Author:
00090             Willem van der Schans, Trelent AI
00091         """
00092         self.ConstructionMonitorProcessor(self.valueObject)
00093
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:

```

```

00132         self.__parameterDict["from"] = record *
int(self.__parameterDict["size"])
00133
00134         response = requests.post(url=self.__restDomain,
00135                                 headers=self.__headerDict,
00136                                 json=self.__parameterDict)
00137
00138         counter = 0
00139         try:
00140             response = response.json()['hits']['hits']
00141         except KeyError as e:
00142             # Logging
00143             print(
00144                 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}")
00145             continue
00146
00147         valueObject.setValue(valueObject.getValue() + 1)
00148
00149         if record == 0 and callNum == 0:
00150             __df = pd.json_normalize(response[counter][["_source"]])
00151             __df["id"] = response[counter]['_id']
00152             __df["county"] =
response[counter][["_source"]]['county']['county_name']
00153             counter += 1
00154
00155             for i in range(counter, len(response)):
00156                 __tdf = pd.json_normalize(response[i][["_source"]])
00157                 __tdf["id"] = response[i]['_id']
00158                 __tdf["county"] =
response[i][["_source"]]['county']['county_name']
00159                 __df = pd.concat([__df, __tdf], ignore_index=True)
00160
00161             if self.__columnSelection is not None:
00162                 __col_list = StringToList(self.__columnSelection)
00163                 __col_list.append("id")
00164                 __col_list.append("county")
00165             else:
00166                 pass
00167
00168             self.dataframe = __df
00169             valueObject.setValue(-999)
00170
00171
00172 class BatchProcessorUtahRealEstate:
00173
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.
00180
00181         Args:
00182             self: Represent the instance of the class
00183             RestDomain: Specify the domain of the rest api
00184             NumBatches: Determine how many batches of data to pull from the api
00185             ParameterString: Pass the parameters to the rest api
00186             HeaderDict: Pass in the header information for the api call
00187             valueObject: Create a dataframe from the json response
00188
00189         Returns:
00190             The instance of the class
00191
00192         Doc Author:
00193             Willem van der Schans, Trelent AI
00194         """
00195         self.dataframe = None
00196         self.__numBatches = NumBatches
00197         self.__parameterString = ParameterString
00198         self.__restDomain = RestDomain

```

```

00199         self.__headerDict = HeaderDict
00200         self.valueObject = valueObject
00201
00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an argument
00205         and then calls the appropriate
00206         function based on what was selected in the dropdown menu. The valueObject
00207         is passed to each of these functions
00208         so that they can access all of its attributes.
00209
00210         Args:
00211             self: Represent the instance of the class
00212
00213         Returns:
00214             The function that is selected by the user
00215
00216         Doc Author:
00217             Willem van der Schans, Trelent AI
00218         """
00219         self.BatchProcessingUtahRealestateCom(self.valueObject)
00220
00221     def BatchProcessingUtahRealestateCom(self, valueObject):
00222         """
00223         The BatchProcessingUtahRealestateCom function is a function that takes in the
00224         valueObject and uses it to
00225         update the progress bar. It also takes in self, which contains all the
00226         necessary information for this
00227         function to work properly. The BatchProcessingUtahRealestateCom function
00228         will then use requests to get data from
00229         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
00230         called __df (which is local). This
00231         process will be repeated until all the data has been collected from
00232         UtahRealestate.com's ReST API, at which point __df will contain all
00233
00234         Args:
00235             self: Represent the instance of the class
00236             valueObject: Pass the value of a progress bar to the function
00237
00238         Returns:
00239             A dataframe of the scraped data
00240
00241         Doc Author:
00242             Willem van der Schans, Trelent AI
00243         """
00244         __df = pd.DataFrame()
00245
00246         for batch in range(self.__numBatches):
00247             if batch == 0:
00248                 response =
00249                 requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00250                             headers=self.__headerDict)
00251
00252                 response_temp = response.json()
00253                 __df = pd.json_normalize(response_temp, record_path=['value'])
00254             else:
00255                 response =
00256                 requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
00257                             200}",
00258                             headers=self.__headerDict)
00259
00260                 response_temp = response.json()
00261                 response_temp = pd.json_normalize(response_temp,
00262                                                     record_path=['value'])
00263                 __df = pd.concat([__df, response_temp], ignore_index=True)
00264
00265                 valueObject.setValue(valueObject.getValue() + 1)
00266
00267         self.dataframe = __df
00268         valueObject.setValue(-999)

```



## DataSupportFunctions.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 def StringToList(string):
00005     """
00006     The StringToList function takes a string and converts it into a list.
00007     The function is used to convert the input from the user into a list of strings,
which can then be iterated through.
00008
00009     Args:
00010         string: Split the string into a list
00011
00012     Returns:
00013         A list of strings
00014
00015     Doc Author:
00016         Willem van der Schans, Trelent AI
00017     """
00018     listOut = list(string.split(", "))
00019     return listOut
```

## FileSaver.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 from pathlib import Path
00007
00008 import pandas as pd
00009
00010 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00011
00012
00013 class FileSaver:
00014
00015     def __init__(self, method, outputDF, AppendingPath=None):
00016         """
00017         The __init__ function is called when the class is instantiated.
00018         It sets up the instance of the class, and defines all variables that will be used
00019         by other functions in this class.
00020         The __init__ function takes two arguments: self and method. The first argument,
00021         self, refers to an instance of a
00022         class (in this case it's an instance of DataFrameSaver). The second argument,
00023         method refers to a string value that
00024         is passed into DataFrameSaver when it's instantiated.
00025
00026         Args:
00027         self: Represent the instance of the class
00028         method: Determine which dataframe to append the new data to
00029         outputDF: Pass in the dataframe that will be saved to a csv file
00030         AppendingPath: Specify the path to an existing csv file that you want to
00031         append your dataframe to
00032
00033         Returns:
00034         Nothing
00035
00036         Doc Author:
00037         Willem van der Schans, Trelent AI
00038
00039         """
00040         self.docPath =
00041         Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00042             datetime.datetime.today().strftime('%m%d%Y'))
00043         self.data = outputDF
00044         self.dataAppending = None
00045         self.appendFlag = True
00046         self.fileName =
00047         f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00048         self.uiFlag = True
00049
00050         if method.lower() == "ure":
00051             self.primaryKey = "ListingKeyNumeric"
00052         elif method.lower() == "cm":
00053             self.primaryKey = "id"
00054         elif "realtor" in method.lower():
00055             self.primaryKey = None
00056             self.uiFlag = False
00057         elif method.lower() == "cfbp":
00058             self.primaryKey = None
00059             self.uiFlag = False
00060         else:
00061             raise ValueError("method input is invalid choice one of 4 options: URE,
00062             CM, Realtor, CFBP")
00063
00064         if AppendingPath is None:
00065             self.appendFlag = False
00066         else:
00067             self.dataAppending = pd.read_csv(AppendingPath)
00068
00069         if self.appendFlag:
00070             if self.primaryKey is not None:
00071                 # Due to low memory loading the columns are not typed properly,
00072                 # since we are comparing this will be an issue since we need to do
00073                 type comparisons,
```

```

00065         # so here we coerce the types of the primary keys to numeric.
00066         # If another primary key is ever chosen make sure to core to the right
data_type.
00067         self.dataAppending[self.primaryKey] =
pd.to_numeric(self.dataAppending[self.primaryKey])
00068         self.data[self.primaryKey] =
pd.to_numeric(self.data[self.primaryKey])
00069
00070         self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00071 keep="last")
00072         else:
00073         self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(keep="last")
00074         else:
00075         self.outputFrame = self.data
00076
00077         if os.path.exists(self.docPath):
00078         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00079         else:
00080         os.mkdir(self.docPath)
00081         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00082
00083         if self.uiFlag:
00084         if self.appendFlag:
00085         PopupWrapped(text=f"File Appended and Saved to
{self.docPath.joinpath(self.fileName)}",
00086 windowType="savedLarge")
00087
00088         # Logging
00089         print(
00090         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")
00091         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]}")
00092         else:
00093         PopupWrapped(text=f"File Saved to
{self.docPath.joinpath(self.fileName)}", windowType="savedLarge")
00094
00095         # Logging
00096         print(
00097         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")
00098         else:
00099         pass
00100
00101     def getPath(self):
00102     """
00103     The getPath function returns the path to the file.
00104     It is a string, and it joins the docPath with the fileName.
00105
00106     Args:
00107         self: Represent the instance of the class
00108
00109     Returns:
00110         The path to the file
00111
00112     Doc Author:
00113         Willem van der Schans, Trelent AI
00114     """
00115     return str(self.docPath.joinpath(self.fileName))

```

## versionChecker.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002 import requests
00003
00004 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00005
00006
00007 def versionChecker():
00008     """
00009     The versionChecker function is used to check if the current version of the program
    is up-to-date.
00010     It does this by comparing the latest release on GitHub.
00011     If they are not equal, it will pop up a window telling you that there's an update
    available.
00012
00013     Args:
00014
00015     Returns:
00016         A popup window with the current version and latest version
00017
00018     Doc Author:
00019         Willem van der Schans, Trelent AI
00020     """
00021     current_version = "1.1.0"
00022     response =
requests.get("https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/late
st")
00023     latest_version = response.json()['name']
00024     text_string = f"A new version is available \n" \
00025                 f"Running version: {current_version} \n" \
00026                 f"Latest version: {latest_version}"
00027     print(text_string)
00028
00029     if current_version != latest_version:
00030         PopupWrapped(text_string, windowType="versionWindow")
```

## ErrorPopup.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00005
00006
00007 def ErrorPopup(textString):
00008     """
00009     The ErrorPopup function is used to display a popup window with an error message.
00010     It takes one argument, textString, which is the string that will be displayed in the
00011     popup window.
00012     The function also opens up the log folder upon program exit.
00013     Args:
00014         textString: Display the error message
00015     Returns:
00016         Nothing, but it does print an error message to the console
00017     Doc Author:
00018         Willem van der Schans, Trelent AI
00019     """
00020     PopupWrapped(
00021         f"ERROR @ {textString} \n"
00022         f"Log folder will be opened upon program exit",
00023         windowType="FatalErrorLarge")
```

## ErrorPrint.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005
00006
00007 def RESTErrorPrint(response):
00008     """
00009     The RESTErrorPrint function is used to print the response from a ReST API call.
00010     If the response is an integer, it will be printed as-is. If it's not an integer,
00011     it will be converted to text and then printed.
00012
00013     Args:
00014         response: Print the response from a rest api call
00015
00016     Returns:
00017         The response text
00018
00019     Doc Author:
00020         Willem van der Schans, Trelent AI
00021     """
00022     if isinstance(response, int):
00023         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Resource Response: {response}")
00024     else:
00025         response_txt = response.text
00026         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Resource Response: {response_txt}")
```

## Logger.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 import sys
00007 from pathlib import Path
00008
00009
00010 def logger():
00011     """
00012     The logger function creates a log file in the user's AppData directory.
00013     The function will create the directory if it does not exist.
00014     The function will also delete the oldest file when 100 logs have been saved to prevent
    bloat.
00015
00016     Args:
00017
00018     Returns:
00019         A file path to the log file that was created
00020
00021     Doc Author:
00022         Willem van der Schans, Trelent AI
00023     """
00024     dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
00025     if os.path.exists(dir_path):
00026         pass
00027     else:
00028         if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00029             os.mkdir(dir_path)
00030         else:
00031             os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00032             os.mkdir(dir_path)
00033
00034     filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
00035         f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
00036     sys.stdout = open(filePath, 'w')
00037     sys.stderr = sys.stdin = sys.stdout
00038
00039     def sorted_ls(path):
00040         """
00041         The sorted_ls function takes a path as an argument and returns the files in that
    directory sorted by modification time.
00042
00043     Args:
00044         path: Specify the directory to be sorted
00045
00046     Returns:
00047         A list of files in a directory sorted by modification time
00048
00049     Doc Author:
00050         Willem van der Schans, Trelent AI
00051     """
00052     mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
00053     return list(sorted(os.listdir(path), key=mtime))
00054
00055     del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
00056     for file in del_list:
00057         os.remove(dir_path.joinpath(file))
00058         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
    %H:%M:%S.%f')}[:-3]} | Log file {file} deleted")
```

## RESError.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005
00006 from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
00007 from API_Calls.Functions.ErrorFunc.ErrorPrint import RESErrorPrint
00008
00009
00010 def RESError(response):
00011     """
00012     The RESError function is a function that checks the status codes.
00013     If it is 200, then everything went well and nothing happens. If it isn't 200, then
    an error message will be printed to
00014     the console with information about what happened (i.e., if there was an authentication
    error or if the resource wasn't found).
00015     The function also raises an exception and opens an error popup for easy debugging.
00016
00017     Args:
00018         response: Print out the response from the server
00019
00020     Returns:
00021         A text string
00022
00023     Doc Author:
00024         Trelent
00025     """
00026     if isinstance(response, int):
00027         status_code = response
00028     else:
00029         status_code = response.status_code
00030
00031     if status_code == 200:
00032         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Api Request completed successfully"
00033         print(textString)
00034         pass
00035     elif status_code == 301:
00036         RESErrorPrint(response)
00037         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"
00038         ErrorPopup(textString)
00039         raise ValueError(textString)
00040     elif status_code == 400:
00041         RESErrorPrint(response)
00042         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Bad Request; check input arguments"
00043         ErrorPopup(textString)
00044         raise ValueError(textString)
00045     elif status_code == 401:
00046         RESErrorPrint(response)
00047         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Authentication Error: No keys found"
00048         ErrorPopup(textString)
00049         raise PermissionError(textString)
00050     elif status_code == 402:
00051         RESErrorPrint(response)
00052         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"
00053         ErrorPopup(textString)
00054         raise PermissionError(textString)
00055     elif status_code == 403:
00056         RESErrorPrint(response)
00057         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"
00058         ErrorPopup(textString)
00059         raise PermissionError(textString)
00060     elif status_code == 404:
00061         RESErrorPrint(response)
```



```

00062         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"
00063         ErrorPopup(textString)
00064         raise NameError(textString)
00065     elif status_code == 405:
00066         RESTErrorPrint(response)
00067         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"
00068         ErrorPopup(textString)
00069         raise ValueError(textString)
00070     elif status_code == 408:
00071         RESTErrorPrint(response)
00072         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout by server"
00073         ErrorPopup(textString)
00074         raise TimeoutError(textString)
00075     elif status_code == 503:
00076         RESTErrorPrint(response)
00077         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"
00078         ErrorPopup(textString)
00079         raise SystemError(textString)
00080     elif status_code == 701:
00081         RESTErrorPrint(response)
00082         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Error in coercing icon to bits
(Imgeloader.py)"
00083         ErrorPopup(textString)
00084         raise TypeError(textString)
00085     elif status_code == 801:
00086         RESTErrorPrint(response)
00087         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"
00088         ErrorPopup(textString)
00089         raise ValueError(textString)
00090     elif status_code == 790:
00091         RESTErrorPrint(response)
00092         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout within requests"
00093         ErrorPopup(textString)
00094         raise TimeoutError(textString)
00095     elif status_code == 791:
00096         RESTErrorPrint(response)
00097         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
00098         ErrorPopup(textString)
00099         raise ValueError(textString)
00100     elif status_code == 990:
00101         RESTErrorPrint(response)
00102         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No password input"
00103         ErrorPopup(textString)
00104         raise ValueError(textString)
00105     elif status_code == 991:
00106         RESTErrorPrint(response)
00107         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No username input"
00108         ErrorPopup(textString)
00109         raise ValueError(textString)
00110     elif status_code == 992:
00111         RESTErrorPrint(response)
00112         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)"
00113         ErrorPopup(textString)
00114         raise ValueError(textString)
00115     elif status_code == 993:
00116         RESTErrorPrint(response)
00117         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"
00118         ErrorPopup(textString)
00119         print(ValueError(textString))

```

```

00120     elif status_code == 994:
00121         RESTErrorPrint(response)
00122         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Submission Error: server returned no
documents"
00123         ErrorPopup(textString)
00124         raise ValueError(textString)
00125     elif status_code == 1000:
00126         RESTErrorPrint(response)
00127         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Catastrophic Error"
00128         ErrorPopup(textString)
00129         raise SystemError(textString)
00130     elif status_code == 1001:
00131         RESTErrorPrint(response)
00132         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Main Function Error Break"
00133         raise SystemError(textString)
00134     elif status_code == 1100:
00135         RESTErrorPrint(response)
00136         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | User has cancelled the program
execution"
00137         raise KeyboardInterrupt(textString)
00138     elif status_code == 1101:
00139         RESTErrorPrint(response)
00140         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"
00141         print(textString)
00142     else:
00143         RESTErrorPrint(response)
00144         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

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003 import PySimpleGUI as sg
00004
00005 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00006
00007
00008 def BatchInputGui(batches, documentCount=None):
00009     """
00010     The BatchInputGui function is a simple GUI that displays the number of batches and
pages
00011     that will be requested. It also gives the user an option to cancel or continue with
their request.
00012
00013
00014     Args:
00015         batches: Determine how many batches will be run
00016         documentCount: Determine how many documents will be retrieved
00017
00018     Returns:
00019         The event, which is the button that was pressed
00020
00021     Doc Author:
00022         Willem van der Schans, Trelent AI
00023     """
00024     event = None
00025     if documentCount is None:
00026         __text1 = f"This request will run {batches}"
00027     else:
00028         __text1 = f"This request will run {batches} batches and will retrieve
{documentCount} rows"
00029
00030     __text2 = "Press Continue to start request"
00031
00032     __Line1 = [sg.Push(),
00033                sg.Text(__text1, justification="center"),
00034                sg.Push()]
00035
00036     __Line2 = [sg.Push(),
00037                sg.Text(__text2, justification="center"),
00038                sg.Push()]
00039
00040     __Line3 = [sg.Push(),
00041                sg.Ok("Continue"),
00042                sg.Cancel(),
00043                sg.Push()]
00044
00045     window = sg.Window("Popup", [__Line1, __Line2, __Line3],
00046                        modal=True,
00047                        keep_on_top=True,
00048                        disable_close=True,
00049                        icon=ImageLoader("taskbar_icon.ico"))
00050
00051     while True:
00052         event, values = window.read()
00053         if event == "Continue":
00054             break
00055         elif event == sg.WIN_CLOSED or event == "Cancel":
00056             break
00057
00058     window.close()
00059
00060     return event
00061
00062
00063 def confirmDialog():
00064     """
00065     The confirmDialog function is a simple confirmation dialog that asks the user if they
want to continue with the request.
00066     The function takes no arguments and returns the button event to allow for process
confirmation.
00067
```

```

00068 Args:
00069
00070 Returns:
00071     The event that was triggered,
00072
00073 Doc Author:
00074     Willem van der Schans, Trelent AI
00075 """
00076     event = None
00077     __text1 = f"This request can take multiple minutes to complete"
00078     __text2 = "Press Continue to start the request"
00079
00080     __Line1 = [sg.Push(),
00081                sg.Text(__text1, justification="center"),
00082                sg.Push()]
00083
00084     __Line2 = [sg.Push(),
00085                sg.Text(__text2, justification="center"),
00086                sg.Push()]
00087
00088     __Line3 = [sg.Push(),
00089                sg.Ok("Continue"),
00090                sg.Cancel(),
00091                sg.Push()]
00092
00093     window = sg.Window("Popup", [__Line1, __Line2, __Line3],
00094                         modal=True,
00095                         keep_on_top=True,
00096                         disable_close=True,
00097                         icon=ImageLoader("taskbar_icon.ico"))
00098
00099     while True:
00100         event, values = window.read()
00101         if event == "Continue":
00102             break
00103         elif event == sg.WIN_CLOSED or event == "Cancel":
00104             break
00105
00106     window.close()
00107
00108     return event

```

## BatchProgressGUI.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003 import datetime
00004 import threading
00005 import time
00006
00007 import PySimpleGUI as sg
00008
00009 from API_Calls.Functions.DataFunc.BatchProcessing import
BatchProcessorConstructionMonitor, BatchProcessorUtahRealEstate
00010 from API_Calls.Functions.Gui.DataTransfer import DataTransfer
00011 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00012 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00013
00014 counter = 1
00015
00016
00017 class BatchProgressGUI:
00018
00019     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type,
ColumnSelection=None):
00020
00021         """
00022         The __init__ function is the first function that gets called when an object of
this class is created.
00023         It initializes all the variables and sets up a layout for the GUI. It also creates
a window to display
00024         the dataframe in.
00025
00026         Args:
00027             self: Represent the instance of the class
00028             BatchesNum: Determine the number of batches that will be created
00029             RestDomain: Specify the domain of the rest api
00030             ParameterDict: Pass the parameters of the request to the class
00031             HeaderDict: Store the headers of the dataframe
00032             Type: Determine the type of dataframe that is being created
00033             ColumnSelection: Select the columns to be displayed in the gui
00034
00035         Returns:
00036             Nothing
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
00042         self.__parameterDict = ParameterDict
00043         self.__restDomain = RestDomain
00044         self.__headerDict = HeaderDict
00045         self.__columnSelection = ColumnSelection
00046         self.__type = Type
00047         self.__dataframe = None
00048
00049         self.__layout = None
00050         self.__batches = BatchesNum
00051         self.__window = None
00052         self.__batch_counter = 0
00053
00054     def BatchGuiShow(self):
00055         """
00056         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.
00057
00058         Args:
00059             self: Represent the instance of the class
00060
00061         Returns:
00062             The __type of the batchgui class
00063
00064         Doc Author:
00065             Willem van der Schans, Trelent AI
00066
00067         """
00068         self.CreateProgressLayout()
```

```

00067         self.createGui(self.__type)
00068
00069     def CreateProgressLayout(self):
00070
00071         """
00072         The CreateProgressLayout function creates the layout for the progress window.
00073         The function takes in self as a parameter and returns nothing.
00074
00075         Parameters:
00076             self (object): The object that is calling this function.
00077
00078         Args:
00079             self: Access the class variables and methods
00080
00081         Returns:
00082             A list of lists
00083
00084         Doc Author:
00085             Willem van der Schans, Trelent AI
00086         """
00087         sg.theme('Default1')
00088
00089         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
00090                   sg.Push()]
00091
00092         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
00093                   sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]
00094
00095         __Line3 = [
00096             sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
00097                           key='--progress_bar--')]
00098
00099
00100         layout = [__Line1, __Line2, __Line3]
00101
00102         self.__layout = layout
00103
00104     def createGui(self, Sourcetype):
00105
00106         """
00107         The createGui function is the main function that creates the GUI.
00108         It takes in a type parameter which determines what kind of batch processor to
use.
00109         The createGui function then sets up all the variables and objects needed for
00110         the program to run, including: window, start_time, update_text, valueObj
(DataTransfer),
00111         processorObject (BatchProcessorConstructionMonitor or
BatchProcessorUtahRealestate),
00112         and threading objects for TimeUpdater and ValueChecker functions. The createGui
function also starts these threads.
00113
00114         Args:
00115             self: Access the object itself
00116             Sourcetype: Determine which batch processor to use
00117
00118         Returns:
00119             The dataframe
00120
00121         Doc Author:
00122             Willem van der Schans, Trelent AI
00123         """
00124         self.__window = sg.Window('Progress', self.__layout, finalize=True,
icon=ImageLoader("taskbar_icon.ico"))
00125
00126         start_time = datetime.datetime.now().replace(microsecond=0)
00127         update_text = f"Batch {0} completed"
00128         self.__window['--progress_text--'].update(update_text)
00129         self.__window['--progress_bar--'].update(0)
00130         self.__window['--time_est--'].update("Est time needed 00:00:00")
00131
00132         valueObj = DataTransfer()
00133         valueObj.setValue(0)
00134

```

```

00135         if Sourcetype == "construction_monitor":
00136
00137             processorObject =
BatchProcessorConstructionMonitor(RestDomain=self. restDomain,
00138
NumBatches=self. batches,
00139
ParameterDict=self. parameterDict,
00140
HeaderDict=self. headerDict,
00141
ColumnSelection=self. columnSelection,
00142
valueObject=valueObj)
00143         elif Sourcetype == "utah_real_estate":
00144             processorObject =
BatchProcessorUtahRealEstate(RestDomain=self. restDomain,
00145
NumBatches=self. batches,
00146
ParameterString=self. parameterDict,
00147
HeaderDict=self. headerDict,
00148
valueObject=valueObj)
00149
00150             threading.Thread(target=self. TimeUpdater,
00151                             args=(start_time,),
00152                             daemon=True).start()
00153             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | TimeUpdater Thread Successfully Started")
00154
00155             batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
00156                                                 daemon=False)
00157             batchFuncThread.start()
00158             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | BatchFunc Thread Successfully Started")
00159             threading.Thread(target=self. ValueChecker,
00160                             args=(valueObj,),
00161                             daemon=False).start()
00162             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | ValueChecker Thread Successfully Started")
00163
00164             while True:
00165
00166                 self. ProgressUpdater(valueObj)
00167
00168                 if valueObj.getValue() == -999:
00169                     break
00170
00171                 window, event, values = sg.read_all_windows()
00172                 if event.startswith('update'):
00173                     __key_to_update = event[len('update'):]
00174                     window[__key_to_update].update(values[event])
00175                     window.refresh()
00176                     pass
00177
00178                 if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179                     break
00180
00181                 time.sleep(0.1)
00182
00183                 self. dataframe = processorObject.dataframe
00184                 self. window.close()
00185
00186                 PopupWrapped(text="Api Request Completed", windowType="notice")
00187
00188             def ProgressUpdater(self, valueObj):
00189                 """
00190                 The ProgressUpdater function is a callback function that updates the progress
bar and text
00191                 in the GUI. It takes in one argument, which is an object containing information
about the
00192                 current batch number. The ProgressUpdater function then checks if this value has
changed from
00193                 the last time it was called (i.e., if we are on a new batch). If so, it updates
both the progress
00194                 bar and text with this new information.

```

```

00195
00196     Args:
00197         self: Make the progressupdater function an instance method
00198         valueObj: Get the current value of the batch counter
00199
00200     Returns:
00201         The value of the batch counter
00202
00203     Doc Author:
00204         Willem van der Schans, Trelent AI
00205     """
00206     if valueObj.getValue() != self.__batch_counter:
00207         self.__batch_counter = valueObj.getValue()
00208
00209     __update_text = f"Batch {self.__batch_counter}/{self.__batches}
completed"
00210
00211     self.__window.write_event_value('update--progress_bar--',
self.__batch_counter)
00212     self.__window.write_event_value('update--progress_text--',
__update_text)
00213     else:
00214         pass
00215
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(
seconds=(__passed_time * (self.__batches /
self.__batch_counter) - __passed_time)).seconds
00254             except:
00255                 __time_est = datetime.timedelta(
seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00256
00257
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
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

```

## DataTransfer.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 class DataTransfer:
00005
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026
00027         Args:
00028             self: Represent the instance of the class
00029             value: Set the value of the instance variable __value
00030
00031         Returns:
00032             The value that was passed to it
00033
00034         Doc Author:
00035             Willem van der Schans, Trelent AI
00036         """
00037         self.__value = value
00038
00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054
00055     def whileValue(self):
00056         """
00057         The whileValue function is a function that will run the getValue function until
00058         it is told to stop.
00059         This allows for the program to constantly be checking for new values from the
00060         sensor.
00061
00062         Args:
00063             self: Refer to the current instance of the class
00064
00065         Returns:
00066             The value of the input
00067
00068         Doc Author:
00069             Willem van der Schans, Trelent AI
00070         """
00071         while True:
00072             self.getValue()
```



## ImageLoader.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import base64
00005 import os
00006 from io import BytesIO
00007 from os.path import join, normpath
00008
00009 from PIL import Image
00010
00011
00012 def ImageLoader(file):
00013     """
00014     The ImageLoader function takes in a file name and returns the image as a base64 encoded
    string.
00015     This is used to send images to the API for processing.
00016
00017     Args:
00018         file: Specify the image file to be loaded
00019
00020     Returns:
00021         A base64 encoded image string
00022
00023     Doc Author:
00024         Willem van der Schans, Trelent AI
00025     """
00026     try:
00027         __path = normpath(join(str(os.getcwd()).split("API_Calls", 1)[0]),
    "API_Calls"))
00028         __path = normpath(join(__path, "External Files"))
00029         __path = normpath(join(__path, "Images"))
00030         __path = join(__path, file).replace("\\", "/")
00031
00032         image = Image.open(__path)
00033
00034         __buff = BytesIO()
00035
00036         image.save(__buff, format="png")
00037
00038         img_str = base64.b64encode(__buff.getvalue())
00039
00040         return img_str
00041     except Exception as e:
00042         # We cannot log this error like other errors due to circular imports
00043         raise e
```

## PopupWrapped.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002 import datetime
00003 import os
00004 import threading
00005 import time
00006 import webbrowser
00007 from pathlib import Path
00008
00009 import PySimpleGUI as sg
00010
00011 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00012
00013
00014 class PopupWrapped():
00015
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
00019         this class is created.
00020         It sets up all the variables and creates a window for us to use.
00021         Args:
00022             self: Represent the instance of the class
00023             text: Set the text of the window
00024             windowType: Determine what type of window to create
00025             error: Display the error message in the window
00026         Returns:
00027             Nothing
00028         Doc Author:
00029             Willem van der Schans, Trelent AI
00030         """
00031         self.__text = text
00032         self.__type = windowType
00033         self.__error = error
00034         self.__layout = []
00035         self.__windowObj = None
00036         self.__thread = None
00037         self.__counter = 0
00038         self.__docpath = None
00039         self.__errorFlag = False
00040
00041         try:
00042             if "File Appended and Saved to " in self.__text:
00043                 self.__docpath = str(self.__text[27:])
00044             elif "File Saved to " in self.__text:
00045                 self.__docpath = str(self.__text[14:])
00046             else:
00047                 pass
00048         except Exception as e:
00049             if self.__type == "savedLarge":
00050                 print(
00051                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3] | PopupWrapped.py | Error = {e} | Error creating self.__docpath open
00052                     file button not available")
00053                 self.__errorFlag = True
00054             else:
00055                 pass
00056
00057         self.__createWindow()
00058
00059     def __createLayout(self):
00060         """
00061         The __createLayout function is used to create the layout of the window.
00062         The function takes class variables and returns a window layout.
00063         It uses a series of if statements to determine what type of window it is, then
00064         creates a layout based on that information.
00065         Args:
00066             self: Refer to the current instance of a class
00067         Returns:
00068             A list of lists
00069         Doc Author:
00070             Willem van der Schans, Trelent AI
00071         """
```

```

00069         sg.theme('Default1')
00070         __Line1 = None
00071         __Line2 = None
00072
00073         if self.__type == "notice":
00074             __Line1 = [sg.Push(),
00075                 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00076                     justification="center"),
00077                 sg.Text(self.__text, justification="center",
00078                     key="-textField-"), sg.Push()]
00079             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00080         elif self.__type == "noticeLarge":
00081             __Line1 = [sg.Push(),
00082                 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00083                     justification="center"),
00084                 sg.Text(self.__text, justification="center",
00085                     key="-textField-"), sg.Push()]
00086             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00087         elif self.__type == "savedLarge":
00088             if self.__errorFlag:
00089                 __Line1 = [sg.Push(),
00090                     sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00091                         justification="center"),
00092                     sg.Text(self.__text, justification="center",
00093                         key="-textField-"), sg.Push()]
00094                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00095             else:
00096                 __Line1 = [sg.Push(),
00097                     sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00098                         justification="center"),
00099                     sg.Text(self.__text, justification="center",
00100                         key="-textField-"), sg.Push()]
00101                 __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)),
00102                     sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00103         elif self.__type == "errorLarge":
00104             __Line1 = [sg.Push(),
00105                 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
00106                     justification="center"),
00107                 sg.Text(self.__text, justification="center",
00108                     key="-textField-"), sg.Push()]
00109             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00110         elif self.__type == "FatalErrorLarge":
00111             __Line1 = [sg.Push(),
00112                 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
00113                     justification="center"),
00114                 sg.Text(f"{self.__text}: {self.__error}",
00115                     justification="center", key="-textField-"),
00116                 sg.Push()]
00117             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00118         elif self.__type == "AuthError":
00119             __Line1 = [sg.Push(),
00120                 sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
00121                     justification="center"),
00122                 sg.Text(f"{self.__text}", justification="center",
00123                     key="-textField-"), sg.Push()]
00124             __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web
00125                 Browser]"),
00126                 sg.Ok(button_text="Return", focus=True, size=(10, 1)),
00127                 sg.Push()]
00128         elif self.__type == "versionWindow":
00129             __Line1 = [sg.Push(),
00130                 sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00131                     justification="center"),
00132                 sg.Text(f"{self.__text}", justification="center",
00133                     key="-textField-"), sg.Push()]
00134             __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00135                 sg.Ok(button_text="Continue", focus=True, size=(10, 1)),
00136                 sg.Push()]

```

```

00124         elif self.__type == "progress":
00125             __Line1 = [sg.Push(),
00126                 sg.Text(self.__text, justification="center",
00127                     key="-textField-"), sg.Push()]
00128         if self.__type == "progress":
00129             self.__layout = [__Line1, ]
00130         else:
00131             self.__layout = [__Line1, __Line2]
00132
00133     def __createWindow(self):
00134         """
00135         The __createWindow function is used to create the window object that will be
00136         displayed. The function takes class variables and a window object. The function first calls
00137         createLayout, which creates the layout for the window based on what type of message it
00138         is (error, notice, progress). Then it uses PySimpleGUI's Window class to create a new window
00139         with that layout and some other parameters such as title and icon. If this is not a progress
00140         bar or permanent message then we start a timer loop that waits until either 100 iterations
00141         have passed or an event has been triggered (such as clicking "Ok" or closing the
00142         window). Once one of these events occurs
00143         Args:
00144             self: Reference the instance of the class
00145         Returns:
00146             A window object
00147         Doc Author:
00148             Willem van der Schans, Trelent AI
00149         """
00150         self.__createLayout()
00151
00152         if self.__type == "progress":
00153             self.__windowObj = sg.Window(title=self.__type.capitalize(),
00154                 layout=self.__layout, finalize=True,
00155                 modal=True,
00156                 keep_on_top=True,
00157                 disable_close=False,
00158                 icon=ImageLoader("taskbar_icon.ico"),
00159                 size=(290, 50))
00160
00161         elif self.__type == "noticeLarge":
00162             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
00163                 finalize=True,
00164                 modal=True,
00165                 keep_on_top=True,
00166                 disable_close=False,
00167                 icon=ImageLoader("taskbar_icon.ico"))
00168
00169         elif self.__type == "savedLarge":
00170             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
00171                 finalize=True,
00172                 modal=True,
00173                 keep_on_top=False,
00174                 disable_close=False,
00175                 icon=ImageLoader("taskbar_icon.ico"))
00176
00177         elif self.__type == "errorLarge":
00178             self.__windowObj = sg.Window(title="Error", layout=self.__layout,
00179                 finalize=True,
00180                 modal=True,
00181                 keep_on_top=True,
00182                 disable_close=False,
00183                 icon=ImageLoader("taskbar_icon.ico"))
00184
00185         elif self.__type == "FatalErrorLarge":
00186             self.__windowObj = sg.Window(title="Fatal Error",
00187                 layout=self.__layout, finalize=True,
00188                 modal=True,
00189                 keep_on_top=True,
00190                 disable_close=False,
00191                 icon=ImageLoader("taskbar_icon.ico"))
00192
00193         elif self.__type == "AuthError":
00194             self.__windowObj = sg.Window(title="Authentication Error",
00195                 layout=self.__layout, finalize=True,
00196                 modal=True,
00197                 keep_on_top=True,
00198                 disable_close=False,
00199                 icon=ImageLoader("taskbar_icon.ico"))
00200
00201         elif self.__type == "versionWindow":
00202             self.__windowObj = sg.Window(title="Update", layout=self.__layout,
00203                 finalize=True,
00204                 modal=True,

```

```

00186         keep_on_top=True,
00187         disable_close=False,
00188         icon=ImageLoader("taskbar_icon.ico"))
00189     else:
00190         self.__windowObj = sg.Window(title=self.__type.capitalize(),
layout=self.__layout, finalize=True,
00191         modal=True,
00192         keep_on_top=True,
00193         disable_close=False,
00194         icon=ImageLoader("taskbar_icon.ico"),
00195         size=(290, 80))
00196
00197     if self.__type != "progress" or self.__type.startswith("perm"):
00198         print("Here")
00199         timer = 0
00200         while timer < 100:
00201             event, values = self.__windowObj.read()
00202             print(event)
00203             if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or
event == "Continue":
00204                 break
00205             elif event == "Open Generation Tool [Web Browser]":
00206                 webbrowser.open('https://www.debugbear.com/basic-auth-header-generator', new=2,
autoraise=True)
00207                 pass
00208             elif event == "Open File":
00209                 threadFile = threading.Thread(target=self.openFile,
daemon=False)
00210                 threadFile.start()
00211                 time.sleep(3)
00212                 break
00213             elif event == "Download":
00214                 webbrowser.open('https://github.com/Kydoimos97/GardnerApiUtility/releases/latest',
new=2,
00215                 autoraise=True)
00216                 pass
00217                 time.sleep(0.1)
00218
00219         if self.__type == "FatalErrorLarge":
00220             try:
00221                 os.system(
00222                     f"start
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')})")
00223             except Exception as e:
00224                 print(
00225                     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")
00226
00227                 self.__windowObj.close()
00228
00229     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
00243     00243     def textUpdate(self, sleep=0.5):
00244         """
00245         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.
00246         Args:
00247             self: Refer to the object itself
00248             sleep: Control the speed of the text update
00249         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
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
00275         if the event starts with 'update'.
00276         If it does, it will take everything after 'update' as a key for updating that
00277         specific value.
00278         It will then update that value using its key and refresh the window.
00279     Args:
00280         self: Reference the object that is calling the function
00281     Returns:
00282         A tuple containing the event and values
00283     Doc Author:
00284         Willem van der Schans, Trelent AI
00285     """
00286     event, values = self.__windowObj.read()
00287
00288     if event.startswith('update'):
00289         __key_to_update = event[len('update'):]
00290         self.__windowObj[__key_to_update].update(values[event])
00291         self.__windowObj.refresh()
00292
00293     def openFile(self):
00294
00295         """
00296         The openFile function opens the file that is associated with the
00297         document object. It does this by calling os.system and passing it
00298         self.__docpath as an argument.
00299     Args:
00300         self: Represent the instance of the object itself
00301     Returns:
00302         The filepath of the document
00303     Doc Author:
00304         Willem van der Schans, Trelent AI
00305     """
00306     os.system(self.__docpath)

```

## Initializer.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 from pathlib import Path
00007
00008 import PySimpleGUI as sg
00009
00010 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00011 from API_Calls.Functions.DataFunc.versionChecker import versionChecker
00012 from API_Calls.Functions.ErrorFunc.Logger import logger
00013 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00014 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00015 from API_Calls.Sources.CFBP.Core import CFBP
00016 from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
00017     ConstructionMonitorMain
00018 from API_Calls.Sources.Realtor.Core import realtorCom
00019 from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain,
UtahRealEstateInit
00020
00021
00022 class initializer:
00023
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
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
00056         a tuple with three elements:
00057             1) The type of element to be displayed (e.g., "Text",
00058             "InputText", etc.)
00059             2) A dictionary containing any additional parameters for that element (e.g.,
00060             size, default value, etc.)
00061             3) An optional key name for the element (used in event handling). If no key
00062             name is provided then one will be generated automatically by PySimpleGUIQt based on its
00063             position in the layout list
00064
00065         Args:
00066             self: Represent the instance of the class
00067             layout: Pass the layout of the window to be created
00068             text: Set the title of the window
00069
00070         Returns:
00071             A window object
```

```

00067
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
00070     """
00071         versionChecker()
00072
00073         window = sg.Window(text, layout, grab_anywhere=False,
00074 return_keyboard_events=True,
00075                             finalize=True,
00076                             icon=ImageLoader("taskbar_icon.ico"))
00077
00078         while True:
00079             event, values = window.read()
00080
00081             if event == "Construction Monitor":
00082                 print(
00083                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Construction Monitor API
00084 Call-----")
00085                 ConstructionMonitorMain(ConstructionMonitorInit())
00086                 print(
00087                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Construction Monitor API
00088 Call-----\n")
00089             elif event == "Utah Real Estate":
00090                 print(
00091                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Utah Real Estate API
00092 Call-----")
00093                 UtahRealEstateMain(UtahRealEstateInit())
00094                 print(
00095                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Utah Real Estate API
00096 Call-----\n")
00097             elif event == "Realtor.Com":
00098                 print(
00099                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Realtor.com API Call-----")
00100                 realtorCom()
00101                 print(
00102                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Realtor.com API Call-----\n")
00103             elif event == "CFPB Mortgage":
00104                 print(
00105                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating ffiec.cfpb API Call-----")
00106                 CFBP()
00107                 print(
00108                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing ffiec.cfpb API Call-----\n")
00109             elif event == "Authorization Utility":
00110                 print(
00111                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Authorization Utility-----")
00112                 AuthUtil()
00113                 print(
00114                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Authorization
00115 Utility-----\n")
00116             elif event == "Open Data Folder":
00117                 print(
00118                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Data Folder Opened-----")
00119                 try:
00120                     os.system(f"start
{Path(os.path.expanduser('~\Documents')).joinpath('GardnerUtilData')}")
00121                 except:
00122                     try:
00123                         os.system(f"start
{Path(os.path.expanduser('~\Documents'))}")
00124                     except Exception as e:
00125                         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
00126                         PopupWrapped(
00127                             text="Documents folder not found. Please create a
Windows recognized documents folder",
00128                             windowType="errorLarge")

```

```

00123
00124         elif event in ('Exit', None):
00125             try:
00126                 break
00127             except Exception as e:
00128                 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.")
00129                 break
00130         elif event == sg.WIN_CLOSED or event == "Quit":
00131             break
00132
00133     window.close()
00134
00135     def __CreateFrame(self):
00136
00137         """
00138         The __CreateFrame function is a helper function that creates the layout for the
main window.
00139         It returns a list of lists, which is then passed to sg.Window() as its layout
parameter.
00140
00141         Args:
00142             self: Represent the instance of the class
00143
00144         Returns:
00145             A list of lists, which is then passed to the sg
00146
00147         Doc Author:
00148             Willem van der Schans, Trelent AI
00149         """
00150         sg.theme('Default1')
00151
00152         line0 = [sg.HSeparator()]
00153
00154         line1 = [sg.Image(ImageLoader("logo.png")),
00155                 sg.Push(),
00156                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"),
justification="center"),
00157                 sg.Push(),
00158                 sg.Push()]
00159
00160         line3 = [sg.HSeparator()]
00161
00162         line4 = [sg.Push(),
00163                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
justification="center"),
00164                 sg.Push()]
00165
00166         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)),
sg.Push(),
00167                 sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00168
00169         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
00170                 sg.Button("CFPB Mortgage", size=(20, None)),
00171                 sg.Push()]]
00172
00173         line8 = [sg.HSeparator()]
00174
00175         line9 = [sg.Push(),
00176                 sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
00177                 sg.Push()]
00178
00179         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00180                 sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00181
00182         line11 = [sg.HSeparator()]
00183
00184         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]
00185
00186         return layout

```

## CFBP/Core.py

```
00001 import datetime
00002 import threading
00003 import time
00004
00005 import pandas as pd
00006 import requests
00007
00008 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00009 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00010 from API_Calls.Functions.Gui.BatchGui import confirmDialog
00011 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00012
00013
00014 class CFBP:
00015
00016     def __init__(self, state_arg=None, year_arg=None):
00017         """
00018         The __init__ function is called when the class is instantiated.
00019         Its job is to initialize the object with some default values, and do any other
00020         setup that might be necessary.
00021         The __init__ function can take arguments, but it doesn't have to.
00022
00023         Args:
00024             self: Represent the instance of the class
00025             state_arg: Set the state_arg attribute of the class
00026             year_arg: Set the year of data to be retrieved
00027
00028         Returns:
00029             A popupwrapped object
00030
00031         Doc Author:
00032             Willem van der Schans, Trelent AI
00033
00034         """
00035         self.state_arg = state_arg
00036         self.year_arg = year_arg
00037         self.uiString = None
00038         self.link = None
00039
00040         eventReturn = confirmDialog()
00041         if eventReturn == "Continue":
00042             startime = datetime.datetime.now().replace(microsecond=0)
00043             self.__showUi()
00044             print(
00045                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | API Link = {self.link}")
00046             F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00047             print(
00048                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S',
00049                 time.gmtime((datetime.datetime.now().replace(microsecond=0) -
00050                 starttime).total_seconds()))}")
00051             self.uiString = (
00052                 f"ffiec.cfbp.gov (Mortgage API) request Completed \n
00053                 {self.year_arg} data retrieved \n Data Saved at {F.getPath()}")
00054             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00055         else:
00056             print(
00057                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | User Canceled Request")
00058             pass
00059
00060     def __showUi(self):
00061         """
00062         The __showUi function is a function that creates a progress bar window.
00063         The __showUi function takes class variables and returns a windowobj.
00064
00065         Args:
00066             self: Represent the instance of the class
```

```

00067     Returns:
00068         The uiobj variable
00069
00070     Doc Author:
00071         Willem van der Schans, Trelent AI
00072     """
00073     uiObj = PopupWrapped(text="Cenus Request running", windowType="progress",
error=None)
00074
00075     threadGui = threading.Thread(target=self.dataGetter,
00076                                 daemon=False)
00077     threadGui.start()
00078
00079     while threadGui.is_alive():
00080         uiObj.textUpdate()
00081         uiObj.windowPush()
00082     else:
00083         uiObj.stopWindow()
00084
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
00095
00096         Doc Author:
00097             Willem van der Schans, Trelent AI
00098         """
00099         arg_dict_bu = locals()
00100
00101         link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
00102
00103         if self.state\_arg is None:
00104             self.state\_arg = "UT"
00105         else:
00106             pass
00107
00108         if self.year\_arg is None:
00109             self.year\_arg = str(date.today().year - 1)
00110         else:
00111             pass
00112
00113         passFlag = False
00114
00115         while not passFlag:
00116             self.link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" +
f"states={self.state\_arg}" + f"&years={self.year\_arg}"
00117
00118             response = requests.get(self.link)
00119
00120             if response.status_code == 400:
00121                 self.year\_arg = int(self.year\_arg) - 1
00122             else:
00123                 passFlag = True
00124
00125         RESError(response)
00126         raise SystemExit(0)

```

## ConstructionMonitor/Core.py

```
00001 import copy
00002 import datetime
00003 import json
00004 import os
00005 import threading
00006 import time
00007 from datetime import date, timedelta
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00018 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00019 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00020 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00021 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00022
00023
00024 class ConstructionMonitorInit:
00025
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the variables that will be used by other functions in this class.
00031
00032         Args:
00033             self: Represent the instance of the class
00034
00035         Returns:
00036             None
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
00042         self.size = None
00043         self.SourceInclude = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.rest_domain = None
00047         self.auth_key = None
00048         self.ui_flag = None
00049         self.append_file = None
00050
00051         passFlag = False
00052
00053         while not passFlag:
00054             if
00055             os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00056                 "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
00057             Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00058                 "Security").joinpath("auth.json")):
00059                 try:
00060                     f =
00061                     open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00062                         "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00063                     key = f.readline()
00064                     f.close()
00065                     f =
00066                     open(Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00067                         "Security").joinpath("auth.json"), "rb")
00068                     authDict = json.load(f)
00069                     fernet = Fernet(key)
00070                     self.auth_key =
00071                     fernet.decrypt(authDict["cm"]["auth"]).decode()
00072                     passFlag = True
```

```

00069         except Exception as e:
00070             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")
00071             AuthUtil()
00072         else:
00073             AuthUtil()
00074
00075     self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00076
00077     def __ShowGui(self, layout, text):
00078
00079         """
00080         The __ShowGui function is the main function that creates and displays the GUI.
00081         It takes in a layout, which is a list of lists containing all the elements to
be displayed on screen.
00082         The text parameter specifies what title should appear at the top of the window.
00083
00084         Args:
00085             self: Refer to the current instance of a class
00086             layout: Determine what the gui will look like
00087             text: Set the title of the window
00088
00089         Returns:
00090             A dictionary of values
00091
00092         Doc Author:
00093             Willem van der Schans, Trelent AI
00094         """
00095         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00096                             finalize=True,
00097                             icon=ImageLoader("taskbar_icon.ico"))
00098
00099         while True:
00100             event, values = window.read()
00101
00102             if event == "Submit":
00103                 try:
00104                     self.__SetValues(values)
00105                     break
00106                 except Exception as e:
00107                     print(e)
00108                     RESTError(993)
00109                     raise SystemExit(933)
00110             elif event == sg.WIN_CLOSED or event == "Quit":
00111                 break
00112
00113         window.close()
00114
00115     @staticmethod
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:
00124
00125         Returns:
00126             The layout for the gui
00127
00128         Doc Author:
00129             Willem van der Schans, Trelent AI
00130         """
00131         sg.theme('Default1')
00132
00133         line00 = [sg.HSeparator()]
00134
00135         line0 = [sg.Image(ImageLoader("logo.png")),
00136                 sg.Push(),
00137                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00138                 sg.Push(),

```



```

00139         sg.Push()]
00140
00141         line1 = [sg.HSeparator()]
00142
00143         line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00144                 sg.Input(default_text=(date.today() -
00145                                timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
00146                                size=(20, 1)),
00147                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00148                                key="-start_date-", target="-Cal-")]
00149
00150         line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00151                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"),
00152                                size=(20, 1)),
00153                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00154                                key="-start_date-", target="-EndCal-")]
00155
00156         line5 = [sg.HSeparator()]
00157
00158         line6 = [sg.Push(),
00159                 sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00160                        justification="center"),
00161                 sg.Push()]
00162
00163         line7 = [sg.HSeparator()]
00164
00165         line8 = [sg.Text("Appending File : ", size=(15, None),
00166                        justification="Right"),
00167                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00168                        size=(20, 1)),
00169                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")],
00170                        key="-append_file-",
00171                        target="-AppendingFile-")]
00172
00173         line9 = [sg.HSeparator()]
00174
00175         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00176
00177         layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8,
00178                 line9, line10]
00179
00180         return layout
00181
00182     def __SetValues(self, values):
00183         """
00184         The __SetValues function is used to set the values of the variables that are used
00185         in the __GetData function.
00186         The __SetValues function takes a dictionary as an argument, and then sets each
00187         variable based on what is passed into
00188         the dictionary. The keys for this dictionary are defined by the user when they
00189         create their own instance of this class.
00190
00191         Args:
00192             self: Represent the instance of the class
00193             values: Pass in the values from the ui
00194
00195         Returns:
00196             A dictionary of values
00197
00198         Doc Author:
00199             Willem van der Schans, Trelent AI
00200
00201         """
00202         self.size = 1000
00203
00204         if values["-Cal-"] != "":
00205             self.dateStart = values["-Cal-"]
00206         else:
00207             self.dateStart = (date.today() -
00208                               timedelta(days=14)).strftime("%Y-%m-%d")
00209
00210         if values["-EndCal-"] != "":
00211             self.dateEnd = values["-EndCal-"]
00212         else:
00213             self.dateEnd = date.today().strftime("%Y-%m-%d")
00214

```

```

00204     self.rest_domain = "https://api.constructionmonitor.com/v2/powersearch/?"
00205
00206     self.SourceInclude = None
00207
00208     if values["-append_file-"] != "":
00209         self.append_file = str(values["-append_file-"])
00210     else:
00211         self.append_file = None
00212
00213     self.ui_flag = True
00214
00215
00216 class ConstructionMonitorMain:
00217
00218     def __init__(self, siteClass):
00219
00220         """
00221         The __init__ function is the first function that runs when an object of this class
00222         is created.
00223         It sets up all the variables and functions needed for this class to run properly.
00224
00225         Args:
00226             self: Represent the instance of the class
00227             siteClass: Identify the site that is being used
00228
00229         Returns:
00230             Nothing
00231
00232         Doc Author:
00233             Willem van der Schans, Trelent AI
00234         """
00235         self._siteClass = siteClass
00236         self._restDomain = None
00237         self._headerDict = None
00238         self._columnSelection = None
00239         self._appendFile = None
00240
00241         self._parameterDict = {}
00242         self._search_id = None
00243         self._record_val = 0
00244         self._batches = 0
00245
00246         self._ui_flag = None
00247
00248         self.dataframe = None
00249
00250         try:
00251             self.mainFunc()
00252         except SystemError as e:
00253             if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message',
repr(e))):
00254                 print(
00255                     f"ConstructionMonitor/Core.py | Error = {e} | Coerced
SystemError in ConstructionMonitorMain class")
00256                 pass
00257             except AttributeError as e:
00258                 # This allows for user cancellation of the program using the quit button
00259                 if "'NoneType' object has no attribute 'json'" in str(getattr(e,
'message', repr(e))):
00260                     RESTError(1101)
00261                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Error {e}")
00262                     pass
00263                 elif e is not None:
00264                     print(
00265                         f"ConstructionMonitor/Core.py | Error = {e} | Authentication
Error | Please update keys in AuthUtil")
00266                     RESTError(401)
00267                     print(e)
00268                     pass
00269                 else:
00270                     pass
00271             except Exception as e:
00272                 print(e)
00273                 RESTError(1001)
00274                 raise SystemExit(1001)

```

```

00275
00276     def mainFunc(self):
00277         """
00278         The mainFunc function is the main function of this module. It will be called by
the GUI or CLI to execute
00279         the code in this module. The mainFunc function will first create a parameter
dictionary using the __ParameterCreator
00280         method, then it will get a count of all records that match its parameters using
the __getCountUI method, and then
00281         it will calculate how many batches are needed to retrieve all records with those
parameters using BatchCalculator.
00282         After that it asks if you want to continue with retrieving data from Salesforce
(if running in GUI mode). Then it shows
00283         a progress bar for each
00284
00285         Args:
00286             self: Refer to the current object
00287
00288         Returns:
00289             The dataframe
00290
00291         Doc Author:
00292             Willem van der Schans, Trelent AI
00293         """
00294         self.__ParameterCreator()
00295
00296         print(
00297             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Param Dict = {self.__parameterDict}"
00298         )
00299         print(
00300             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Rest Domain = {self.__restDomain}"
00301         )
00302         self.__getCountUI()
00303
00304         self.batches = BatchCalculator(self.record_val, self.parameterDict)
00305
00306         print(
00307             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Batches = {self.batches} | Rows {self.record_val}"
00308         )
00309         if self.batches != 0:
00310             startTime = datetime.datetime.now().replace(microsecond=0)
00311             eventReturn = BatchInputGui(self.batches, self.record_val)
00312             if eventReturn == "Continue":
00313                 print(
00314                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Request for {self.batches} batches sent to server"
00315                 )
00316                 BatchGuiObject = BatchProgressGUI(RestDomain=self.restDomain,
ParameterDict=self.parameterDict,
HeaderDict=self.headerDict,
ColumnSelection=self.columnSelection,
BatchesNum=self.batches,
Type="construction_monitor")
00317
00318                 BatchGuiObject.BatchGuiShow()
00319                 self.dataframe = BatchGuiObject.dataframe
00320
00321                 print(
00322                     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()))}"
00323                 )
00324                 FileSaver("cm", self.dataframe, self.appendFile)
00325             else:
00326                 print(
00327                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Request for {self.batches} batches canceled by user"
00328                 )
00329             else:
00330                 RESTError(994)
00331                 raise SystemExit(994)
00332
00333     def __ParameterCreator(self):
00334         """
The __ParameterCreator function is used to create the parameter dictionary that
will be passed into the

```

```

00335     __Request function. The function takes in a siteClass object and extracts
00336     all of its attributes, except for
00337     those that start with '__' or are callable. It then creates a dictionary from
00338     these attributes and stores it as
00339     self.__parameterDict.
00340
00341     Args:
00342     self: Make the function a method of the class
00343
00344     Returns:
00345     A dictionary of parameters and a list of non parameter variables
00346
00347     Doc Author:
00348     Willem van der Schans, Trelent AI
00349     """
00350     Source dict = {key: value for key, value in
00351     self.__siteClass.__dict__.items() if
00352     not key.startswith('__') and not callable(key)}
00353
00354     self.__restDomain = __Source_dict["rest_domain"]
00355     __Source_dict.pop("rest_domain")
00356     self.__headerDict = {"Authorization": Source_dict["auth key"]}
00357     __Source_dict.pop("auth_key")
00358     self.__columnSelection = __Source_dict["SourceInclude"]
00359     __Source_dict.pop("SourceInclude")
00360     self.__ui_flag = __Source_dict["ui_flag"]
00361     __Source_dict.pop("ui_flag")
00362     self.__appendFile = __Source_dict["append_file"]
00363     __Source_dict.pop("append_file")
00364
00365     temp_dict = copy.copy(__Source_dict)
00366     for key, value in temp_dict.items():
00367         if value is None:
00368             __Source_dict.pop(key)
00369         else:
00370             pass
00371
00372     self.__parameterDict = copy.copy(__Source_dict)
00373
00374     def __getCount(self):
00375     """
00376     The __getCount function is used to get the total number of records that are
00377     returned from a query.
00378     This function is called by the __init__ function and sets the self.__record_val
00379     variable with this value.
00380
00381     Args:
00382     self: Represent the instance of the class
00383
00384     Returns:
00385     The total number of records in the database
00386
00387     Doc Author:
00388     Willem van der Schans, Trelent AI
00389     """
00390     __count_resp = None
00391
00392     try:
00393         __temp_param_dict = copy.copy(self.__parameterDict)
00394         __count_resp = requests.post(url=self.__restDomain,
00395                                     headers=self.__headerDict,
00396                                     json=__temp_param_dict)
00397
00398     except requests.exceptions.Timeout as e:
00399         print(e)
00400         RESTError(790)
00401         raise SystemExit(790)
00402     except requests.exceptions.TooManyRedirects as e:
00403         print(e)
00404         RESTError(791)
00405         raise SystemExit(791)
00406     except requests.exceptions.MissingSchema as e:
00407         print(e)
00408         RESTError(1101)
00409     except requests.exceptions.RequestException as e:

```

```

00407         print(e)
00408         RESTError(405)
00409         raise SystemExit(405)
00410
00411         __count_resp = __count_resp.json()
00412
00413         self.__record_val = __count_resp["hits"]["total"]["value"]
00414
00415         del __count_resp, __temp_param_dict
00416
00417     def __getCountUI(self):
00418
00419         """
00420         The __getCountUI function is a wrapper for the __getCount function.
00421         It allows the user to run __getCount in a separate thread, so that they can
00422         continue working while it runs.
00423         The function will display a progress bar and update with text as it progresses
00424         through its tasks.
00425
00426         Args:
00427             self: Access the class variables and methods
00428
00429         Returns:
00430             The count of the number of records in the database
00431
00432         Doc Author:
00433             Willem van der Schans, Trelent AI
00434
00435         """
00436         if self.__ui_flag:
00437             uiObj = PopupWrapped(text="Batch request running",
00438                                   windowType="progress", error=None)
00439
00440             threadGui = threading.Thread(target=self.__getCount,
00441                                           daemon=False)
00442             threadGui.start()
00443
00444             while threadGui.is_alive():
00445                 uiObj.textUpdate()
00446                 uiObj.windowPush()
00447             else:
00448                 uiObj.stopWindow()
00449
00450         else:
00451             self.__getCount()

```

## Realtor/Core.py

```
00001 import datetime
00002 import threading
00003 import time
00004
00005 import pandas as pd
00006 import requests
00007 from bs4 import *
00008
00009 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00010 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00011 from API_Calls.Functions.Gui.BatchGui import confirmDialog
00012 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00013
00014
00015 class realtorCom:
00016
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
00021         to run before anything else in your class.
00022
00023         Args:
00024             self: Represent the instance of the class
00025
00026         Returns:
00027             A new object
00028
00029         Doc Author:
00030             Willem van der Schans, Trelent AI
00031
00032         """
00033         self.__page_html = None
00034         self.__update_date = None
00035         self.__last_date = None
00036         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00037         self.__linkDict = {}
00038         self.__dfState = None
00039         self.__dfCounty = None
00040         self.__dfZip = None
00041         self.__uiString = "Files Saved to \n"
00042
00043         eventReturn = confirmDialog()
00044         if eventReturn == "Continue":
00045             page_html =
00046             requests.get("https://www.realtor.com/research/data/").text
00047             self.__page_html = BeautifulSoup(page_html, "html.parser")
00048             startTime = datetime.datetime.now().replace(microsecond=0)
00049             self.__linkGetter()
00050             print(
00051                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Link Dictionary = {self.__idDict}")
00052             self.__showUi()
00053             PopupWrapped(text=self.__uiString, windowType="noticeLarge")
00054             print(
00055                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Data retrieved with in {time.strftime('%H:%M:%S',
00056                 time.gmtime((datetime.datetime.now().replace(microsecond=0) -
00057                 startTime).total_seconds()))}")
00058             else:
00059                 print(
00060                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | User Canceled Request")
00061                 pass
00062
00063     def __showUi(self):
00064         """
00065         The __showUi function is a helper function that creates and displays the progress
00066         window.
00067         It also starts the dataUpdater thread, which will update the progress bar as it
00068         runs.
00069
00070         """
```

```

00065     Args:
00066         self: Represent the instance of the class
00067
00068     Returns:
00069         A popupwrapped object
00070
00071     Doc Author:
00072         Willem van der Schans, Trelent AI
00073     """
00074     uiObj = PopupWrapped(text="Request running", windowType="progress",
error=None)
00075
00076     threadGui = threading.Thread(target=self.__dataUpdater,
                                daemon=False)
00077
00078     threadGui.start()
00079
00080     while threadGui.is_alive():
00081         uiObj.textUpdate()
00082         uiObj.windowPush()
00083     else:
00084         uiObj.stopWindow()
00085
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
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"

```



## UtahRealEstate/Core.py

```
00001 import copy
00002 import datetime
00003 import json
00004 import os
00005 import threading
00006 import time
00007 from datetime import date, timedelta
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00018 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00019 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00020 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00021 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00022
00023
00024 class UtahRealEstateInit:
00025
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the initial state of the object.
00031
00032         Args:
00033             self: Represent the instance of the class
00034
00035         Returns:
00036             The __createframe function
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
00042         self.StandardStatus = None
00043         self.ListedOrModified = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.select = None
00047         self.file_name = None
00048         self.append_file = None
00049
00050         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00051
00052     def __ShowGui(self, layout, text):
00053
00054         """
00055         The __ShowGui function is a helper function that creates the GUI window and
00056         displays it to the user.
00057         It takes in two parameters: layout, which is a list of lists containing all the
00058         elements for each row;
00059         and text, which is a string containing what will be displayed as the title of
00060         the window. The __ShowGui
00061         method then uses these parameters to create an instance of sg.Window with all
00062         its attributes set accordingly.
00063
00064         Args:
00065             self: Refer to the current class instance
00066             layout: Pass the layout of the window to be created
00067             text: Set the title of the window
00068
00069         Returns:
00070             A dictionary of values
00071
00072         Doc Author:
00073             Willem van der Schans, Trelent AI
```

```

00070     """
00071     window = sg.Window(text, layout, grab_anywhere=False,
00072         return_keyboard_events=True,
00073         finalize=True,
00074         icon=ImageLoader("taskbar_icon.ico"))
00075     while True:
00076         event, values = window.read()
00077
00078         if event == "Submit":
00079             try:
00080                 self._SetValues(values)
00081                 break
00082             except Exception as e:
00083                 print(e)
00084                 RESTError(993)
00085                 raise SystemExit(993)
00086         elif event == sg.WIN_CLOSED or event == "Quit":
00087             break
00088
00089     window.close()
00090
00091     @staticmethod
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
00096         displayed in the window.
00097         Each element is defined by its type and any additional parameters needed to
00098         define it.
00099
00100         Args:
00101             Returns:
00102                 A list of lists, which is used to create the gui
00103
00104         Doc Author:
00105             Willem van der Schans, Trelent AI
00106         """
00107         sg.theme('Default1')
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,
00113                 "bold"), justification="center"),
00114             sg.Push(),
00115             sg.Push()]
00116
00117         line1 = [sg.HSeparator()]
00118
00119         line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00120             sg.DropDown(default_value="Active", values=["Active", "Closed"],
00121                 key="-status-", size=(31, 1))]
00122
00123         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00124             sg.DropDown(default_value="Listing Date", values=["Listing
00125                 Date", "Modification Date", "Close Date"],
00126                 key="-type-", size=(31, 1))]
00127
00128         line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00129             sg.Input(default_text=(date.today() -
00130                 timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00131                 disabled=False, size=(20, 1)),
00132             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00133                 key="-start_date-", target="-DateStart-")]
00134
00135         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00136             sg.Input(default_text=(date.today()).strftime("%Y-%m-%d"),
00137                 key="-DateEnd-", disabled=False,
00138                 size=(20, 1)),
00139             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00140                 key="-end_date-", target="-DateEnd-")]
00141
00142         line7 = [sg.HSeparator()]

```

```

00137         line8 = [sg.Push(),
00138                     sg.Text("File Settings", font=("Helvetica", 12, "bold"),
justification="center"),
00139                     sg.Push()]
00140
00141         line9 = [sg.HSeparator()]
00142
00143         line10 = [sg.Text("Appending File : ", size=(15, None),
justification="Right"),
00144                     sg.Input(default_text="", key="-AppendingFile-", disabled=True,
size=(20, 1)),
00145                     sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
                                target="-AppendingFile-")]
00146
00147         line11 = [sg.HSeparator()]
00148
00149         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00150
00151         layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8,
line9, line10, line11,
00152                     line12]
00153
00154         return layout
00155
00156     def __SetValues(self, values):
00157
00158         """
00159         The __SetValues function is used to set the values of the variables that are used
in the
00160         __GetData function. The values are passed from a dictionary called 'values'
which is created
00161         by parsing through an XML file using ElementTree. This function also sets
default values for
00162         some of these variables if they were not specified in the XML file.
00163
00164         Args:
00165             self: Represent the instance of the class
00166             values: Pass the values from the gui to this function
00167
00168         Returns:
00169             A dictionary with the following keys:
00170
00171         Doc Author:
00172             Willem van der Schans, Trelent AI
00173
00174         """
00175         self.StandardStatus = values["-status-"]
00176
00177         self.ListedOrModified = values["-type-"]
00178
00179         if values["-DateStart-"] != "":
00180             self.dateStart = values["-DateStart-"]
00181         else:
00182             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00183
00184         if values["-DateEnd-"] != "":
00185             self.dateEnd = values["-DateEnd-"]
00186         else:
00187             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00188
00189         self.select = None
00190
00191         if values["-append_file-"] != "":
00192             self.append_file = str(values["-append_file-"])
00193         else:
00194             self.append_file = None
00195
00196     class UtahRealEstateMain:
00197
00198         def __init__(self, siteClass):
00199
00200             """
00201             The __init__ function is the first function that runs when an object of this class
is created.
00202             It sets up all the variables and functions needed for this class to work properly.

```

```

00205
00206     Args:
00207         self: Represent the instance of the class
00208         siteClass: Determine which site to pull data from
00209
00210     Returns:
00211         Nothing
00212
00213     Doc Author:
00214         Willem van der Schans, Trelent AI
00215     """
00216     self.dataframe = None
00217     self._batches = 0
00218     self._siteClass = siteClass
00219     self._headerDict = None
00220     self._parameterString = ""
00221     self._appendFile = None
00222     self._dateStart = None
00223     self._dateEnd = None
00224     self._restDomain =
'https://resoapi.utahrealstate.com/reso/odata/Property?'
00225     self._keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00226         "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00227     self._filePath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00228         "Security").joinpath("auth.json")
00229     self._key = None
00230     self._record_val = None
00231
00232     try:
00233         self.mainFunc()
00234     except KeyboardInterrupt as e:
00235         # This allows for user cancellation of the program using the quit button
00236         if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00237             RESTError(1101)
00238             print(e)
00239             pass
00240         else:
00241             pass
00242     except Exception as e:
00243         print(e)
00244         RESTError(1001)
00245         raise SystemExit(1001)
00246
00247     def mainFunc(self):
00248
00249         """
00250         The mainFunc function is the main function of this module. It will be called by
the GUI when a user clicks on
00251         the &quot;Run&quot; button in the GUI. The mainFunc function should contain all
of your code for running your program, and it
00252         should return a dataframe that contains all the data you want to display in your
final report.
00253
00254     Args:
00255         self: Reference the object itself
00256
00257     Returns:
00258         A dataframe
00259
00260     Doc Author:
00261         Willem van der Schans, Trelent AI
00262     """
00263     passFlag = False
00264
00265     while not passFlag:
00266         if os.path.isfile(self._keyPath) and os.path.isfile(self._filePath):
00267             try:
00268                 f = open(self._keyPath, "rb")
00269                 key = f.readline()
00270                 f.close()
00271                 f = open(self._filePath, "rb")
00272                 authDict = json.load(f)
00273                 fernet = Fernet(key)
00274                 authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00275                 self._headerDict = {authDict["ure"]["parameter"]: authkey}

```

```

00276         passFlag = True
00277     except Exception as e:
00278         print(
00279             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening
AuthUtil")
00280         AuthUtil()
00281     else:
00282         AuthUtil()
00283
00284     self.__ParameterCreator()
00285
00286     print(
00287         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Param String = {self.__parameterString}")
00288     print(
00289         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Rest Domain = {self.__restDomain}")
00290
00291     self.__getCountUI()
00292
00293     if self.__record_val is None:
00294         self.__record_val = 0
00295
00296     self.__batches = BatchCalculator(self.__record_val, None)
00297
00298     print(
00299         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Batches = {self.__batches} | Rows {self.__record_val}")
00300
00301     if self.__batches != 0:
00302         startTime = datetime.datetime.now().replace(microsecond=0)
00303         eventReturn = BatchInputGui(self.__batches, self.__record_val)
00304         if eventReturn == "Continue":
00305             print(
00306                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} batches sent to server")
00307             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00308         ParameterDict=self.__parameterString,
00309                                     HeaderDict=self.__headerDict,
00310                                     BatchesNum=self.__batches,
00311                                     Type="utah_real_estate")
00312             BatchGuiObject.BatchGuiShow()
00313             self.dataframe = BatchGuiObject.dataframe
00314             print(
00315                 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()))}")
00316             FileSaver("ure", self.dataframe, self.__appendFile)
00317         else:
00318             print(
00319                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} batches canceled by user")
00320         else:
00321             RESTError(994)
00322             raise SystemExit(994)
00323
00324     def __ParameterCreator(self):
00325         """
00326         The __ParameterCreator function is used to create the filter string for the ReST
API call.
00327         The function takes in a siteClass object and extracts all of its parameters into
a dictionary.
00328         It then creates an appropriate filter string based on those parameters.
00329
00330         Args:
00331             self: Bind the object to the class
00332
00333         Returns:
00334             A string to be used as the parameter in the api call
00335
00336         Doc Author:
00337             Willem van der Schans, Trelent AI
00338         """

```

```

00339         filter_string = ""
00340
00341         __Source_dict = {key: value for key, value in
self. siteClass.__dict__.items() if
00342             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"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContr
actDate%20le%20{__Source_dict['dateEnd']}"
00356         elif __Source_dict["ListedOrModified"] == "Modification Date":
00357             filter_string =
f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%2
0ModificationTimestamp%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{__So
urce_dict['dateEnd']}"
00360
00361         filter_string = filter_string +
f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['Standard
Status']}'"
00362
00363         self. parameterString = filter_string
00364
00365     def getCount(self):
00366         """
00367         The getCount function is used to determine the number of records that will be
returned by the query.
00368         This function is called when a user calls the count() method on a ReST object.
The getCount function uses
00369         the $count parameter in OData to return only an integer value representing how
many records would be returned
00370         by the query.
00371
00372         Args:
00373             self: Represent the instance of the class
00374
00375         Returns:
00376             The number of records in the data set
00377
00378         Doc Author:
00379             Willem van der Schans, Trelent AI
00380         """
00381         __count_resp = None
00382
00383         try:
00384             __count_resp =
requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00385                 headers=self. headerDict)
00386
00387         except requests.exceptions.Timeout as e:
00388             print(e)
00389             RESTError(790)
00390             raise SystemExit(790)
00391         except requests.exceptions.TooManyRedirects as e:
00392             print(e)
00393             RESTError(791)
00394             raise SystemExit(791)
00395         except requests.exceptions.MissingSchema as e:
00396             print(e)
00397             RESTError(1101)
00398         except requests.exceptions.RequestException as e:
00399             print(e)
00400             RESTError(405)
00401             raise SystemExit(405)
00402

```

```

00403         self.__record_val = int(__count_resp.json()["@odata.count"])
00404
00405     def __getCountUI(self):
00406
00407         """
00408         The __getCountUI function is a wrapper for the __getCount function.
00409         It creates a progress window and updates it while the __getCount function runs.
00410         The purpose of this is to keep the GUI responsive while running long processes.
00411
00412         Args:
00413             self: Represent the instance of the class
00414
00415         Returns:
00416             A popupwrapped object
00417
00418         Doc Author:
00419             Willem van der Schans, Trelent AI
00420         """
00421         uiObj = PopupWrapped(text="Batch request running", windowType="progress",
error=None)
00422
00423         threadGui = threading.Thread(target=self.__getCount,
00424                                     daemon=False)
00425         threadGui.start()
00426
00427         while threadGui.is_alive():
00428             uiObj.textUpdate()
00429             uiObj.windowPush()
00430         else:
00431             uiObj.stopWindow()

```

# Index

- \_\_appendFile
  - Core.ConstructionMonitorMain, 64
  - Core.UtahRealEstateMain, 112
- \_\_batch\_counter
  - BatchProgressGUI.BatchProgressGUI, 39
- \_\_batches
  - BatchProgressGUI.BatchProgressGUI, 39
  - Core.ConstructionMonitorMain, 64
  - Core.UtahRealEstateMain, 112
- \_\_columnSelection
  - BatchProcessing.BatchProcessorConstructionMonitor, 21
  - BatchProgressGUI.BatchProgressGUI, 39
  - Core.ConstructionMonitorMain, 64
- \_\_counter
  - PopupWrapped.PopupWrapped, 88
- \_\_CreateFrame
  - API\_Calls.Initializer.initializer, 75
  - AuthUtil.AuthUtil, 9
  - Core.ConstructionMonitorInit, 49
  - Core.UtahRealEstateInit, 98
- \_\_createLayout
  - PopupWrapped.PopupWrapped, 81
- \_\_createWindow
  - PopupWrapped.PopupWrapped, 83
- \_\_dataGetter
  - Core.CFBP, 43
- \_\_dataUpdater
  - Core.realtorCom, 91
- \_\_dateEnd
  - Core.UtahRealEstateMain, 112
- \_\_dateStart
  - Core.UtahRealEstateMain, 112
- \_\_dateTracker
  - BatchProcessing.BatchProcessorConstructionMonitor, 22
- \_\_docpath
  - PopupWrapped.PopupWrapped, 88
- \_\_error
  - PopupWrapped.PopupWrapped, 88
- \_\_errorFlag
  - PopupWrapped.PopupWrapped, 88
- \_\_getCount
  - Core.ConstructionMonitorMain, 58
  - Core.UtahRealEstateMain, 106
- \_\_getCountUI
  - Core.ConstructionMonitorMain, 59
  - Core.UtahRealEstateMain, 107
- \_\_headerDict
  - BatchProcessing.BatchProcessorConstructionMonitor, 22
  - BatchProcessing.BatchProcessorUtahRealEstate, 27
  - BatchProgressGUI.BatchProgressGUI, 39
  - Core.ConstructionMonitorMain, 65
  - Core.UtahRealEstateMain, 112
- \_\_idDict
  - Core.realtorCom, 94
- \_\_init\_\_
  - API\_Calls.Initializer.initializer, 74
  - AuthUtil.AuthUtil, 6
  - BatchProcessing.BatchProcessorConstructionMonitor, 17
  - BatchProcessing.BatchProcessorUtahRealEstate, 24
  - BatchProgressGUI.BatchProgressGUI, 30
  - Core.CFBP, 42
  - Core.ConstructionMonitorInit, 47
  - Core.ConstructionMonitorMain, 56
  - Core.realtorCom, 90
  - Core.UtahRealEstateInit, 96
  - Core.UtahRealEstateMain, 104
  - DataTransfer.DataTransfer, 67
  - FileSaver.FileSaver, 70
  - PopupWrapped.PopupWrapped, 80
- \_\_init\_\_.py, 115
- \_\_last\_date
  - Core.realtorCom, 94
- \_\_layout
  - BatchProgressGUI.BatchProgressGUI, 39
  - PopupWrapped.PopupWrapped, 88
- \_\_linkDict
  - Core.realtorCom, 94
- \_\_linkGetter
  - Core.realtorCom, 92
- \_\_maxRequests
  - BatchProcessing.BatchProcessorConstructionMonitor, 22
- \_\_numBatches
  - BatchProcessing.BatchProcessorConstructionMonitor, 22
  - BatchProcessing.BatchProcessorUtahRealEstate, 28
- \_\_page\_html
  - Core.realtorCom, 94
- \_\_ParameterCreator
  - Core.ConstructionMonitorMain, 60
  - Core.UtahRealEstateMain, 108
- \_\_parameterDict
  - BatchProcessing.BatchProcessorConstructionMonitor, 22
  - BatchProgressGUI.BatchProgressGUI, 40
  - Core.ConstructionMonitorMain, 65
- \_\_parameterString
  - BatchProcessing.BatchProcessorUtahRealEstate, 28
  - Core.UtahRealEstateMain, 113
- \_\_record\_val
  - Core.ConstructionMonitorMain, 65
  - Core.UtahRealEstateMain, 113
- \_\_requestCalls



- BatchProcessing.BatchProcessorConstructionMonitor, 22
- requestCount
- BatchProcessing.BatchProcessorConstructionMonitor, 23
- restDomain
- BatchProcessing.BatchProcessorConstructionMonitor, 23
- BatchProcessing.BatchProcessorUtahRealEstate, 28
- BatchProgressGUI.BatchProgressGUI, 40
- Core.ConstructionMonitorMain, 65
- Core.UtahRealEstateMain, 113
- search\_id
- Core.ConstructionMonitorMain, 65
- SetValues
- AuthUtil.AuthUtil, 11
- Core.ConstructionMonitorInit, 51
- Core.UtahRealEstateInit, 99
- ShowGui
- API\_Calls.Initializer.initializer, 77
- AuthUtil.AuthUtil, 13
- Core.ConstructionMonitorInit, 52
- Core.UtahRealEstateInit, 101
- showUi
- Core.CFBP, 44
- Core.realtorCom, 93
- siteClass
- Core.ConstructionMonitorMain, 66
- Core.UtahRealEstateMain, 113
- text
- PopupWrapped.PopupWrapped, 88
- thread
- PopupWrapped.PopupWrapped, 88
- type
- BatchProgressGUI.BatchProgressGUI, 40
- PopupWrapped.PopupWrapped, 89
- ui\_flag
- Core.ConstructionMonitorMain, 66
- update\_date
- Core.realtorCom, 95
- value
- DataTransfer.DataTransfer, 69
- window
- BatchProgressGUI.BatchProgressGUI, 40
- windowObj
- PopupWrapped.PopupWrapped, 89
- \_main\_.c, 116
- \_main\_.py, 164
- API\_Calls.Initializer.initializer, 74
- \_\_CreateFrame, 75
- \_\_init\_\_, 74
- \_\_ShowGui, 77
- classObj, 79
- append\_file
- AuthUtil.AuthUtil, 14
- Core.ConstructionMonitorInit, 54
- Core.UtahRealEstateInit, 102
- appendFlag
- FileSaver.FileSaver, 73
- auth\_key
- Core.ConstructionMonitorInit, 54
- AuthUtil.AuthUtil, 6
- \_\_CreateFrame, 9
- \_\_init\_\_, 6
- \_\_SetValues, 11
- \_\_ShowGui, 13
- append\_file, 14
- file\_name, 14
- filePath, 15
- jsonDict, 15
- k, 15
- keyFlag, 15
- keyPath, 15
- ListedOrModified, 15
- outcomeText, 15
- passFlagCm, 15
- passFlagUre, 15
- popupFlag, 16
- StandardStatus, 16
- AuthUtil.py, 165
- BatchGui.py, 184
- BatchGuiShow
- BatchProgressGUI.BatchProgressGUI, 31
- BatchProcessing.BatchProcessorConstructionMonitor, 17
- \_\_columnSelection, 21
- \_\_dateTracker, 22
- \_\_headerDict, 22
- \_\_init\_\_, 17
- \_\_maxRequests, 22
- \_\_numBatches, 22
- \_\_parameterDict, 22
- \_\_requestCalls, 22
- \_\_requestCount, 23
- \_\_restDomain, 23
- ConstructionMonitorProcessor, 19
- dataframe, 23
- FuncSelector, 21
- valueObject, 23
- BatchProcessing.BatchProcessorUtahRealEstate, 24
- \_\_headerDict, 27
- \_\_init\_\_, 24
- \_\_numBatches, 28
- \_\_parameterString, 28
- \_\_restDomain, 28
- BatchProcessingUtahRealestateCom, 25
- dataframe, 28
- FuncSelector, 27
- valueObject, 28
- BatchProcessing.py, 170
- BatchProcessingUtahRealestateCom
- BatchProcessing.BatchProcessorUtahRealEstate, 25
- BatchProgressGUI.BatchProgressGUI, 30
- \_\_batch\_counter, 39
- \_\_batches, 39
- \_\_columnSelection, 39
- \_\_headerDict, 39

- \_\_init\_\_, 30
- \_\_layout, 39
- \_\_parameterDict, 40
- \_\_restDomain, 40
- \_\_type, 40
- \_\_window, 40
- BatchGuiShow, 31
- createGui, 32
- CreateProgressLayout, 35
- dataframe, 40
- ProgressUpdater, 36
- TimeUpdater, 37
- ValueChecker, 38
- BatchProgressGUI.py, 186
- classObj
  - API\_Calls.Initializer.initializer, 79
- ConstructionMonitorProcessor
  - BatchProcessing.BatchProcessorConstructio
    - nMonitor, 19
- Core.CFBP, 42
  - \_\_dataGetter, 43
  - \_\_init\_\_, 42
  - \_\_showUi, 44
  - link, 45
  - state\_arg, 45
  - uiString, 45
  - year\_arg, 45
- Core.ConstructionMonitorInit, 47
  - \_\_CreateFrame, 49
  - \_\_init\_\_, 47
  - \_\_SetValues, 51
  - \_\_ShowGui, 52
  - append\_file, 54
  - auth\_key, 54
  - dateEnd, 54
  - dateStart, 54
  - rest\_domain, 54
  - size, 54
  - SourceInclude, 55
  - ui\_flag, 55
- Core.ConstructionMonitorMain, 56
  - \_\_appendFile, 64
  - \_\_batches, 64
  - \_\_columnSelection, 64
  - \_\_getCount, 58
  - \_\_getCountUI, 59
  - \_\_headerDict, 65
  - \_\_init\_\_, 56
  - \_\_ParameterCreator, 60
  - \_\_parameterDict, 65
  - \_\_record\_val, 65
  - \_\_restDomain, 65
  - \_\_search\_id, 65
  - \_\_siteClass, 66
  - \_\_ui\_flag, 66
  - dataframe, 66
  - mainFunc, 62
- Core.py, 202, 204, 211, 214
- Core.realtorCom, 90
  - \_\_dataUpdater, 91
  - \_\_idDict, 94
  - \_\_init\_\_, 90
  - \_\_last\_date, 94
  - \_\_linkDict, 94
  - \_\_linkGetter, 92
  - \_\_page\_html, 94
  - \_\_showUi, 93
  - \_\_update\_date, 95
  - dfCounty, 95
  - dfState, 95
  - dfZip, 95
  - uiString, 95
- Core.UtahRealEstateInit, 96
  - \_\_CreateFrame, 98
  - \_\_init\_\_, 96
  - \_\_SetValues, 99
  - \_\_ShowGui, 101
  - append\_file, 102
  - dateEnd, 103
  - dateStart, 103
  - file\_name, 103
  - ListedOrModified, 103
  - select, 103
  - StandardStatus, 103
- Core.UtahRealEstateMain, 104
  - \_\_appendFile, 112
  - \_\_batches, 112
  - \_\_dateEnd, 112
  - \_\_dateStart, 112
  - \_\_getCount, 106
  - \_\_getCountUI, 107
  - \_\_headerDict, 112
  - \_\_init\_\_, 104
  - \_\_ParameterCreator, 108
  - \_\_parameterString, 113
  - \_\_record\_val, 113
  - \_\_restDomain, 113
  - \_\_siteClass, 113
  - dataframe, 113
  - filePath, 113
  - key, 114
  - keyPath, 114
  - mainFunc, 109
- createGui
  - BatchProgressGUI.BatchProgressGUI, 32
- CreateProgressLayout
  - BatchProgressGUI.BatchProgressGUI, 35
- data
  - FileSaver.FileSaver, 73
- dataAppending
  - FileSaver.FileSaver, 73
- dataframe
  - BatchProcessing.BatchProcessorConstructio
    - nMonitor, 23
  - BatchProcessing.BatchProcessorUtahRealEs
    - tate, 28
  - BatchProgressGUI.BatchProgressGUI, 40
  - Core.ConstructionMonitorMain, 66
  - Core.UtahRealEstateMain, 113
- DataSupportFunctions.py, 174

- DataTransfer.DataTransfer, 67
  - \_\_init\_\_, 67
  - \_\_value, 69
  - getValue, 67
  - setValue, 68
  - whileValue, 69
- DataTransfer.py, 191
- dateEnd
  - Core.ConstructionMonitorInit, 54
  - Core.UtahRealEstateInit, 103
- dateStart
  - Core.ConstructionMonitorInit, 54
  - Core.UtahRealEstateInit, 103
- dfCounty
  - Core.realtorCom, 95
- dfState
  - Core.realtorCom, 95
- dfZip
  - Core.realtorCom, 95
- docPath
  - FileSaver.FileSaver, 73
- ErrorPopup.py, 178
- ErrorPrint.py, 179
- file\_name
  - AuthUtil.AuthUtil, 14
  - Core.UtahRealEstateInit, 103
- fileName
  - FileSaver.FileSaver, 73
- filePath
  - AuthUtil.AuthUtil, 15
  - Core.UtahRealEstateMain, 113
- FileSaver.FileSaver, 70
  - \_\_init\_\_, 70
  - appendFlag, 73
  - data, 73
  - dataAppending, 73
  - docPath, 73
  - fileName, 73
  - getPath, 72
  - outputFrame, 73
  - primaryKey, 73
  - uiFlag, 73
- FileSaver.py, 175
- FuncSelector
  - BatchProcessing.BatchProcessorConstructio
 nMonitor, 21
  - BatchProcessing.BatchProcessorUtahRealEs
 tate, 27
- getPath
  - FileSaver.FileSaver, 72
- getValue
  - DataTransfer.DataTransfer, 67
- ImageLoader.py, 193
- Initializer.py, 199
- jsonDict
  - AuthUtil.AuthUtil, 15
- k
  - AuthUtil.AuthUtil, 15
- key
  - Core.UtahRealEstateMain, 114
- keyFlag
  - AuthUtil.AuthUtil, 15
- keyPath
  - AuthUtil.AuthUtil, 15
  - Core.UtahRealEstateMain, 114
- link
  - Core.CFBP, 45
- ListedOrModified
  - AuthUtil.AuthUtil, 15
  - Core.UtahRealEstateInit, 103
- Logger.py, 180
- mainFunc
  - Core.ConstructionMonitorMain, 62
  - Core.UtahRealEstateMain, 109
- openFile
  - PopupWrapped.PopupWrapped, 85
- outcomeText
  - AuthUtil.AuthUtil, 15
- outputFrame
  - FileSaver.FileSaver, 73
- passFlagCm
  - AuthUtil.AuthUtil, 15
- passFlagUre
  - AuthUtil.AuthUtil, 15
- popupFlag
  - AuthUtil.AuthUtil, 16
- PopupWrapped.PopupWrapped, 80
  - \_\_counter, 88
  - \_\_createLayout, 81
  - \_\_createWindow, 83
  - \_\_docpath, 88
  - \_\_error, 88
  - \_\_errorFlag, 88
  - \_\_init\_\_, 80
  - \_\_layout, 88
  - \_\_text, 88
  - \_\_thread, 88
  - \_\_type, 89
  - \_\_windowObj, 89
  - openFile, 85
  - stopWindow, 86
  - textUpdate, 86
  - windowPush, 87
- PopupWrapped.py, 194
- primaryKey
  - FileSaver.FileSaver, 73
- ProgressUpdater
  - BatchProgressGUI.BatchProgressGUI, 36
- rest\_domain
  - Core.ConstructionMonitorInit, 54
- RESError.py, 181
- select
  - Core.UtahRealEstateInit, 103
- setValue
  - DataTransfer.DataTransfer, 68
- size
  - Core.ConstructionMonitorInit, 54
- SourceInclude
  - Core.ConstructionMonitorInit, 55
- StandardStatus

- AuthUtil.AuthUtil, 16
- Core.UtahRealEstateInit, 103
- state\_arg
  - Core.CFBP, 45
- stopWindow
  - PopupWrapped.PopupWrapped, 86
- textUpdate
  - PopupWrapped.PopupWrapped, 86
- TimeUpdater
  - BatchProgressGUI.BatchProgressGUI, 37
- ui\_flag
  - Core.ConstructionMonitorInit, 55
- uiFlag
  - FileSaver.FileSaver, 73
- uiString
  - Core.CFBP, 45
- Core.realtorCom, 95
- ValueChecker
  - BatchProgressGUI.BatchProgressGUI, 38
- valueObject
  - BatchProcessing.BatchProcessorConstructionMonitor, 23
  - BatchProcessing.BatchProcessorUtahRealEstate, 28
- versionChecker.py, 177
- whileValue
  - DataTransfer.DataTransfer, 69
- windowPush
  - PopupWrapped.PopupWrapped, 87
- year\_arg
  - Core.CFBP, 45