PYWINAUTO

INDEX

1. Working on Pywinauto

Pywinauto is a set of libraries for Windows GUI testing automation with Python.

1.1. Installation

You can install the Pywinauto package by running a standard pip command.
pip install Pywinauto

1.2. To Start an Application

Use the Application().start() is the method to launch the application. This returns an application object representing the started application.

```
from pywinauto.application import Application
app = Application().start("notepad.exe")
```

This will open a new file in Notepad.
Always remember to give the .exe path to the start function.

In pywinauto, the backend parameter specifies the technology used to interact with GUI applications on Windows. The two main backends supported are "win32" and "uia":

1.2.1. "win32" Backend

- **Usage**: It is the default backend used in pywinauto and supports older style Windows applications that rely on the traditional Win32 controls.

1.2.2. "uia" Backend

- **Description**: The "uia" (UI Automation) backend uses the UI Automation framework provided by Windows to interact with GUI applications.
- **Usage**: It is used for newer Windows applications that implement their UI using the UI Automation framework, which provides enhanced accessibility and programmability features.

For applications, if we don't mention the backend parameter the default is Win32.

1.3. Connect to already running process

```
app = Application(backend="uia").connect(path=r"C:\Windows\System32\Notepad.exe")
```

To access control elements, you need to let Pywinauto define the application window with these elements. In our case, it's the Notepad window.

Windows Specification is an object that describes either the application window or the control element. In our scenario, the window can be specified in the following way:

```
main_dlg = app.UntitledNotepad
main_dlg = app.top_window()
main_dlg = app.window(title="untitled - Notepad")
main_dlg.Edit.type_keys("Hello Welcome to Notepad",with_spaces=True)
```

with_spaces = True to insert spaces between each word in a sentence.

type_keys() = used to write the text

Edit is optional we can also write into the file using **main_dlg.type_keys("hello sri")**

menu_select() is used to select the menu items in case of notepad the menu items are File,Edit,Format,View,Help

```
main_dlg.menu_select("Format->Font")
```

Print_control_identifiers() is used to list the control identifiers that are present in main_dlg. It tells what type of control is the control button in the notepad for ex; menu item,scroll bar,dialog etc...

```
main_dlg.print_control_identifiers()
```

Example control identifiers for font dialog:

Dialog - 'Untitled - Notepad' (L312, T12, R1152, B664)

['Untitled - NotepadDialog', 'Dialog', 'Untitled - Notepad', 'Dialog0', 'Dialog1']

child_window(title="Untitled - Notepad", control_type="Window")

|

| Dialog - 'Font' (L343, T164, R784, B642)

| ['Font', 'FontDialog', 'Dialog2']

| child_window(title="Font", control_type="Window")

| |

| | Static - 'Font:' (L363, T208, R535, B225)

| | ['Static', 'Font:', 'Font:Static', 'Font:0', 'Font:1', 'Static0', 'Static1']

| | child_window(title="Font:", auto_id="1088", control_type="Text")

| |

| | **ComboBox** - 'Font:' (L363, T225, R535, B365)

| | ['Font:2', 'ComboBox', 'Font:ComboBox', 'ComboBox0', 'ComboBox1']

| | child_window(title="Font:", auto_id="1136", control_type="ComboBox")

| | |

| | | ListBox - 'Font:' (L363, T247, R535, B365)

| | | ['Font:3', 'Font:ListBox', 'ListBox', 'ListBox0', 'ListBox1']

| | | child_window(title="Font:", auto_id="1000", control_type="List")

| | | |

| | | | **ScrollBar** - 'Vertical' (L516, T249, R533, B363)

| | | | ['ScrollBar', 'VerticalScrollBar', 'Vertical', 'ScrollBar0', 'ScrollBar1', 'VerticalScrollBar0', 'VerticalScrollBar1', 'Vertical0', 'Vertical1']

| | | | child_window(title="Vertical", auto_id="NonClientVerticalScrollBar", control_type="ScrollBar")

| | | | |

| | | | | **Button** - 'Line up' (L516, T249, R533, B266)

| | | | | ['Line upButton', 'Line up', 'Button', 'Button0', 'Button1', 'Line upButton0', 'Line upButton1', 'Line up0', 'Line up1']

| | | | | child_window(title="Line up", auto_id="UpButton", control_type="Button")

The hightilighed texts indicates the particular type of control it is for ex font is a dialog

1.3.1. To access a dialog and make use of controls

If backend=”uia” then:

```
font_dialog = main_dlg.child_window(title="Font")
```

If backend=”win32” then:

```
font_dialog = main_dlg.Font
```

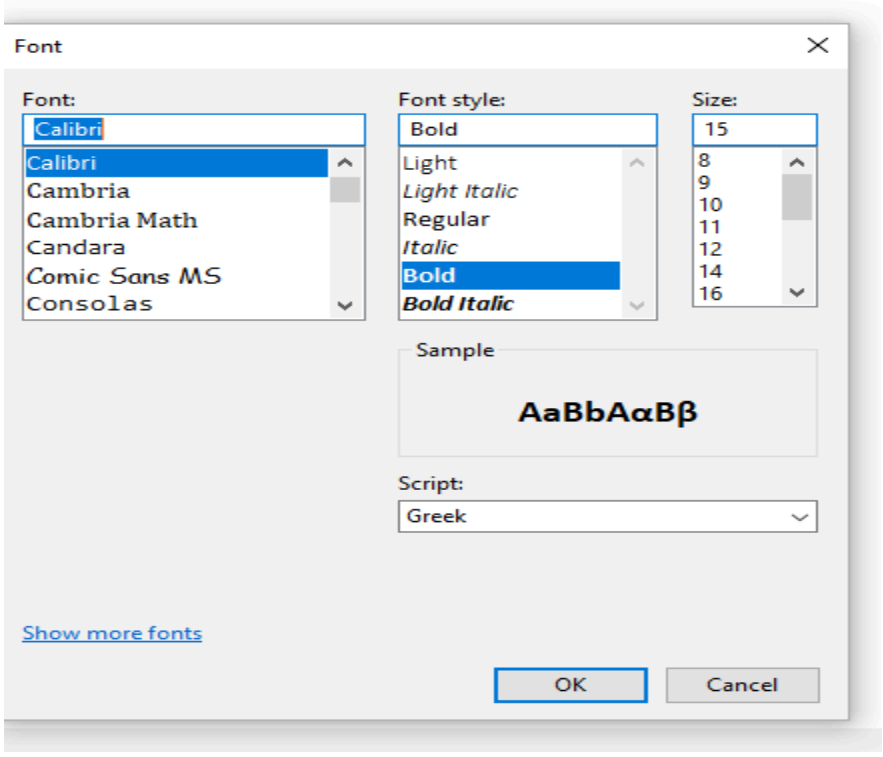
1.3.2. To access different types of controls

- **ComboBox :**

```
combo_box = font_dialog.ComboBox4
combo_box.select("Greek")
```

EditBox:

We can access the edit boxes like this by identifying the number of edit boxes in the dialog For ex ; Font dialog has 3 edit boxes and 1 Combo




```
font_dialog.Edit1.type_keys("Calibri")
font_dialog.Edit2.type_keys("Bold")
font_dialog.Edit3.type_keys("15")
```

1.3.3. To access RadioButtons

In Notepad

1.select the menu item file->pagesetup

```
main_dlg.menu_select("File->Pagesetup")
```

2.Access the Page Setup Dialog

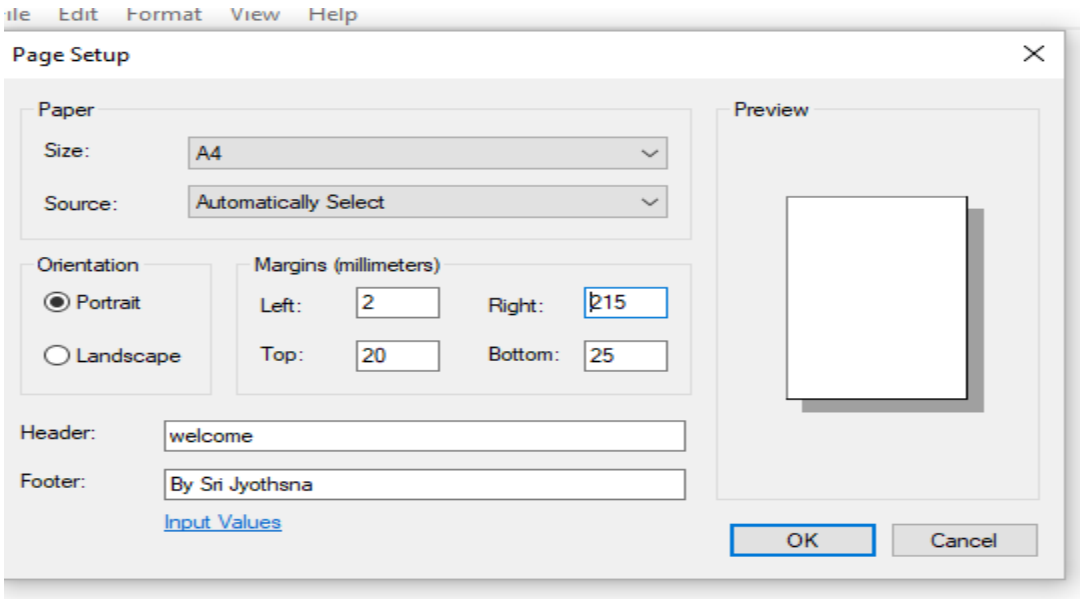
```
pagesetup_dialog = main_dlg.window(title="Page Setup")
```

3.identify the RadioButtons in the dialog and select the button you want to click

```
radio_button = pagesetup_dialog.RadioButton2
radio_button.click()
```

The difference between type_keys and set_text is :

```
#select the menu_item
main_dlg.menu_select("File->Pagesetup")
#access the dialog
pagesetup_dialog = main_dlg.window(title="Page Setup")
pagesetup_dialog.ComboBox1.select("A4")
pagesetup_dialog.Edit1.set_text("2")
pagesetup_dialog.Edit2.type_keys("2")
```



Here both the edit boxes 1 and 2 are given a value 2 but in the output the edit1 box has correct output 2 and the edit2 has 215 means it is appending 2 to the existing 15 value.

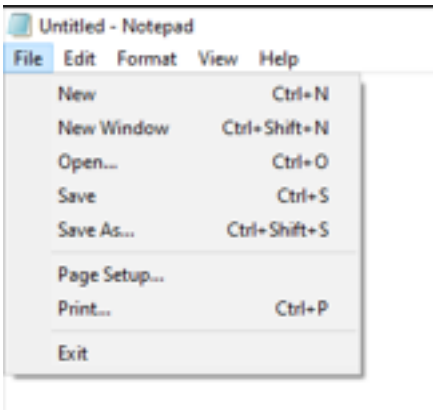
Set_text(): it will clear the edit box and set the value which is given by the user

type_keys(): it will not clear the text and it will append to the existing text

1.3.4. To give shortcuts

For example we have some shortcuts like ctrl+a to select all, ctrl+v to paste like that we can also give these shortcuts using type_keys

Lets understand performing the action using the standard way and by shortcuts.



Here Ctrl+Shift+N is used to open New Window now lets perform this action using two ways.

1.Standard

```
main_dlg = app.window(title="Untitled - Notepad")
main_dlg.menu_select("File->New Window")
```

2.Using Shortcuts

```
main_dlg = app.window(title="Untitled - Notepad")
main_dlg.type_keys("^+N")
```

- **^**: Represents the Ctrl key.
- **+**: Represents the Shift key.
- N: Represents the letter 'N' (or any other key you want to combine with Ctrl+Shift).
- % (percent sign) represents the Alt key.
- # (hash or pound sign) represents the Windows key (Win key).

1.3.5. To read existing text in the window or dialog

get_value() if backend = “uia”

window_text() if backend=”win32”

```
#access the main dialog
main_dlg = app.window(title="Untitled - Notepad")
#access the menu item open
main_dlg.menu_select("File->Open")
#access the dialog open
open_dialog = main_dlg.child_window(title="Open",found_index=0)
#file path to open file in specific location
file_name = r"C:\Users\vlab\Downloads\sri.txt"
open_dialog.type_keys(file_name+"{ENTER}")
print(app.Notepad.Edit.get_value())
```

Here app.Notepad.Edit is used because when we open a file the title will change like sri-Notepad so we cannot use main_dlg which is Untitled – Notepad. So use app.Notepad instead of main_dlg

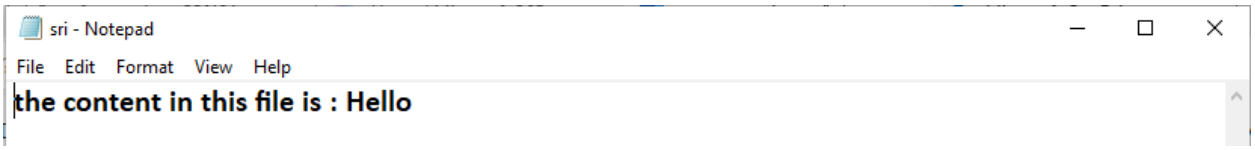
If you use main_dlg which is Untitled – Notepad

```
print(main_dlg.Edit.get_value())
```

raise ElementNotFoundError(kwargs)

pywinauto.findwindows.ElementNotFoundError: {'title': 'Untitled - Notepad', 'backend': 'uia', 'process': 14532}

It will give you an error indicating Untitled – Notepad not found because when you open a file the title will be shown in the picture below.



Instead main_dlg you can use

```
opened_file_dialog = app.window(title="sri - Notepad")
print(opened_file_dialog.Edit.get_value())
```

So these small techniques you have to understand

1.3.6. To send keyboard Actions using type_keys()

```
open_dialog.type_keys(file_name+"{ENTER}")
```

- **Tab Key**: "{TAB}"
- **Backspace Key**: "{BACKSPACE}"
- **Escape Key**: "{ESC}"
- **Ctrl Key**: `"{CTRL}"
- **Shift Key**: "{SHIFT}"

2. Functions

Now lets understand about other functions in pywinauto.

2.1. maximize()

used to maximize the window

```
main_dlg.maximize()
```

2.2. is_maximized()

returns True if Window is Maximized

```
print(main_dlg.is_maximized())
```

2.3. hide_from_taskbar()

hides from taskbar

```
main_dlg.hide_from_taskbar()
```

2.4. show_in_taskbar()

shows in taskbar

```
main_dlg.show_in_taskbar()
```

2.5. Is_in_taskbar()

returns true if it is present in the taskbar otherwise False.
Run this example to understand

```
main_dlg.hide_from_taskbar()
print(main_dlg.is_in_taskbar())
main_dlg.show_in_taskbar()
print(main_dlg.is_in_taskbar())
```

2.6. get_show_state()

returns 1,2,3 based on the window state
1 when it is in normal state
2 when is is minimized
3 when it is maximizes

```
print(main_dlg.get_show_state())
main_dlg.minimize()
```

```
print(main_dlg.get_show_state())
main_dlg.maximize()
print(main_dlg.get_show_state())
```
Run the code for better understanding

2.7. is_dialog()

returns true if it is a dialog

```
print(main_dlg.is_dialog())
```

2.8. menu_items()

returns the menu items for the dialog

```
print(main_dlg.menu_items())
```

2.9. Owner()

returns the owner window for the window if no returns None
Run the code for better understanding

```
print(main_dlg.owner())
main_dlg.menu_select("File->Open")
open_dialog = app.Open
print(open_dialog.owner())
```

2.10. capture_as_image()

The capture_as_image(rect=None) function in pywinauto is used to capture the contents of a specified rectangle (or the entire screen if rect is not provided) as an image. Here's how it works:

- **Parameters**:
  - rect (optional): A tuple specifying the coordinates (left, top, right, bottom) of the rectangle to capture. If rect is None, the function captures the entire screen.
- **Return Value**:
  - Returns an image object (PIL.Image) containing the captured screen content.

To capture screenshot you have to install pillow
(.venv) PS C:\Users\vlab\Desktop\pywinauto> pip install pillow

Run the code for better understanding

```
main_dlg = app.UntitledNotepad
main_dlg.menu_select("File->Open")
```

```
open_dialog = app.Open
image = open_dialog.capture_as_image()
image.save("sri.png")
image.show()
```

2.11. class_name()

returns the class name

```
print(main_dlg.class_name())
```

2.12. click_input ()

To click an element.

2.13. control_count()

returns the no.of children of this control

Run the code for better understanding

```
main_dlg.menu_select("Format->Font")
font_dlg = app.Font
print(font_dlg.control_count())
print(font_dlg.print_control_identifiers())
```

2.14. control_id()

returns the control id for element

```
print(main_dlg.control_id())
```

2.15. double_click_input()

to double click

2.16. element_info

returns the info for the element

Run the code for better understanding.

```
main_dlg.menu_select("Format->Font")
font_dlg = app.Font
```

```
print(font_dlg.element_info)
a = font_dlg.Edit1.type_keys("Calibri")
print(a.element_info
```

2.17. class_name()

returns the class name of an element

```
font_dlg = app.Font
print(font_dlg.class_name())
```

2.18. friendly_class_name ()

Returns the friendly class name of the control

```
print(font_dlg.class_name())
print(font_dlg.friendly_class_name())
```

The difference between class_name() and friendly_class_name() is .. it will make user clear that this is checkbox, radio button and combo box while class_name() gives any type as a button.

2.19. get_properties ()

Returns the properties of the element as a dictionary.

includes all details like class_name(),friendly_class_name(),control_id()

```
main_dlg.menu_select("Format->Font")
font_dlg = app.Font
print(font_dlg.get_properties())
```

Output:

{'class_name': '#32770', 'friendly_class_name': 'Dialog', 'texts': ['Font'], 'control_id': 0, 'rectangle': <RECT L304, T246, R745, B724>, 'is_visible': True, 'is_enabled': True, 'control_count': 27, 'style': -1798831932, 'exstyle': 65793, 'user_data': 0, 'context_help_id': 0, 'fonts': [<LOGFONTW 'Segoe UI' -12>], 'client_rects': [<RECT L0, T0, R425, B439>], 'is_unicode': True, 'menu_items': [], 'automation_id': ''}

2.20. window_text ()

collects the value and prints it in console

Similar to getting value.

Is used to print texts existing in file and text in edit boxes etc...

Run the code for better understanding

```
main_dlg.menu_select("File->Open")
open_dialog = app.Open
file_name = r"C:\Users\vlab\Downloads\sri.txt"
open_dialog.type_keys(file_name+"{ENTER}")
print(app.Notepad.window_text())
print(app.Notepad.Edit.window_text())
```

Output:

sri - Notepad

the content in this file is : Hello

2.21. is_enabled ()

Returns True if the element is enabled

Used to check whether dialog is enabled or not

2.22. is_visible ()

Returns True if the element is visible

2.23. close ()

To close opened dialog

2.24. kill ()

To close the application

2.25. To generate log files

Import logging

logging.basicConfig(filename=”some name”, level = logging.DEBUG,

format = “%(asctime)s - %(levelname)s - %(message)s”)

logging.info(“started”)

logging.info(“ended”)

2.26. get_toggle_state()

When backend = uia it is used to get the state of checkbox

0- unchecked

1-checked

2.27. get_value()

When backend = uia this is used to get the text of edit control

For more reference: https://pywinauto.readthedocs.io/en/latest/code/pywinauto.controls.menuwrapper.html

Example code to automate Notepad using pywinauto:

File_name = uia.py

```
from pywinauto.application import Application
from time import sleep



class note:
def open_npd(self):
        self.app = Application(backend="uia").start("notepad.exe")

        self.main_dlg = self.app.UntitledNotepad

       self.notepad_window = self.app.top_window()

    def new_file(self):

        self.notepad_window.menu_select("File->New")

    def edit(self):

        self.notepad_window.Edit.type_keys(
            "hello guys welcome to notepad you can edit ur text using notepad,you can use font styles in notepad",
            with_spaces=True)

        sleep(1)

    def close_npd(self):

        self.app.kill()
```

```
def font(self):
    self.notepad_window.menu_select("Format->Font")
    font_window = self.notepad_window.child_window(title="Font", control_type="Window")
    font_window.Edit1.set_text("")
    font_window.Edit1.type_keys("calibri")
    sleep(1)
    font_window.Edit2.set_text("")
    sleep(1)
    font_window.Edit2.type_keys("italic")
    font_window.Edit3.set_text("")
    sleep(1)
    font_window.Edit3.set_text("15")
    sleep(1)
    font_window.Ok.click()
    sleep(2)
def find(self):
    self.notepad_window.menu_select("Edit->Find")
    sleep(2)
    find_window = self.notepad_window.child_window(title="Find", control_type="Window")
    find_window.Edit.set_text("")
    find_window.Edit.type_keys("notepad")
    sleep(1)
```

```
print("class -----", find_window.class_name())

checkbox1 = find_window.CheckBox

print("state------", checkbox1.get_toggle_state())

if checkbox1.get_toggle_state() == 1:

    print("checked")

else:

    print("checking the box1")

    checkbox1.click()


checkbox2 = find_window.CheckBox2

print("state------", checkbox1.get_toggle_state())

if checkbox2.get_toggle_state() == 1:

    print("checked")

else:

    print("checking the box2")

checkbox2.click()

radiobutton1 = find_window.RadioButton

print("radio button state", radiobutton1.is_enabled())

radiobutton1.click()

sleep(2)

find_window.FindNext.click()

sleep(1)

find_window.FindNext.click()
```

```
    sleep(1)

    find_window.FindNext.click()

    sleep(1)

    find_window.close()


def replace(self):
    self.notepad_window.menu_select("Edit->Replace")

# self.notepad_window.print_control_identifiers()

    replace_dialog = self.notepad_window.child_window(title="Replace", control_type="Window")

    replace_dialog.Edit1.set_text("")

    sleep(1)

    replace_dialog.Edit1.type_keys("guys")

    sleep(1)

    replace_dialog.Edit2.set_text("")

    sleep(1)

    replace_dialog.Edit2.type_keys("everyone")

    sleep(1)

    replace_dialog.ReplaceAll.click()

    sleep(2)

    replace_dialog.close()


def time(self):
    self.notepad_window.menu_select("Edit->Time/Date")

    sleep(1)
```

```
    def select_all(self):
        self.notepad_window.menu_select("Edit->SelectAll")

        sleep(1)

        self.notepad_window.menu_select("Edit->Delete")

        sleep(1)

        self.notepad_window.menu_select("Edit->undo")

        sleep(1)



    def view(self):
        self.notepad_window.child_window(title="View", control_type="MenuItem").click_input()

        self.notepad_window.child_window(title="Zoom", control_type="MenuItem").click_input()

    # self.notepad_window.print_control_identifiers()

                                    self.notepad_window.child_window(title="Zoom     In     Ctrl+Plus",     auto_id="34",
control_type="MenuItem").click_input()

                                self.notepad_window.child_window(title="Zoom    Out             Ctrl+Minus",    auto_id="35",
control_type="MenuItem")



    def about(self):
        self.notepad_window.menu_select("Help->About Notepad")

        sleep(1)

        # self.notepad_window.print_control_identifiers()

        self.notepad_window.child_window(title="About Notepad", control_type="Window").close()

        # self.app.UntitledNotepad.Ok.click_input()

        sleep(1)

        # self.notepad_window.menu_select("Help->View Help")

        # self.notepad_window.menu_select("Help->Send Feedback")
```

```
def pagesetup(self):
    self.notepad_window.menu_select("File->PageSetup")

    # self.notepad_window.print_control_identifiers()

    pagesetup_window = self.notepad_window.child_window(title="Page Setup", control_type="Window")

    combo_box = pagesetup_window.ComboBox1

    combo_box.select("A4")

    sleep(1)

    radio_button = pagesetup_window.RadioButton2

    radio_button.click()

    sleep(1)

    pagesetup_window.Edit1.set_text("10")

    sleep(1)

    pagesetup_window.Edit2.set_text("15")

    sleep(1)

    pagesetup_window.Edit3.set_text("20")

    sleep(1)

    pagesetup_window.Edit4.set_text("25")

    pagesetup_window.Edit5.set_text("")

    sleep(1)

    pagesetup_window.Edit5.set_text("welcome")

    sleep(1)

    pagesetup_window.Edit6.set_text("")

    sleep(1)
```

```
    pagesetup_window.Edit6.set_text("By Sri Jyothsna")

    sleep(1)

    pagesetup_window.Ok.click()


def print(self):
    self.notepad_window.menu_select("File->Print")

    # self.notepad_window.print_control_identifiers()

    print_window = self.notepad_window.child_window(title="Print", control_type="Window")

    print_window.Edit5.set_text("3")

    print_window.Print.click()


def open(self):
    self.notepad_window.menu_select("File->Open")

    # self.notepad_window.Open.print_control_identifiers()

    open_window = self.notepad_window.child_window(title="Open", control_type="Window")

    file_name = r"C:\Users\vlab\Downloads\jyo.txt"

    open_window.type_keys(file_name + "{ENTER}")

    # open_window.ComboBox2.select(1)

    # open_window.Open.click()

    print("\nthe content in the file is\n", self.app.Notepad.Edit.get_value())
```

Test cases using the Above class

File-name = test_uia.py

```
from uia import note

obj = note()

def test_edit():
  obj.open_npd()
    obj.new_file()
    obj.edit()
    obj.close_npd()



def test_font():
    obj.open_npd()
    obj.edit()
    obj.font()
    obj.close_npd()



def test_find():
    obj.open_npd()
    obj.edit()
    obj.find()
    obj.close_npd()



def test_replace():
    obj.open_npd()
    obj.edit()
    obj.replace()
    obj.close_npd()



def test_time():
    obj.open_npd()
    obj.time()
    obj.close_npd()



def test_select_delete():
    obj.open_npd()
    obj.edit()
    obj.select_all()
    obj.close_npd()



def test_view():
    obj.open_npd()
```

```
    obj.edit()
    obj.view()
    obj.close_npd()



def test_about():
    obj.open_npd()
    obj.about()
    obj.close_npd()



def test_pagesetup():
    obj.open_npd()
    obj.pagesetup()
    obj.close_npd()



def test_prrint():
    obj.open_npd()
    obj.edit()
    obj.print()
    obj.close_npd()
```

Also Automation of Audacity using Pywinauto

Refer to the code for more understanding

https://github.com/srijyothsna18/Testing_team_2024/blob/main/Base/Libraries/audacity.py