

Python to PowerPoint Documentation

CONTENT TABLE

Code Insights / Roundup:.....	2
Program Libraries / Modules:	2
Creation of the Presentation:.....	3
Main Function:	5
Closing statements and things to watch out for:	6

CODE INSIGHTS / ROUNDUP:

The Code “Input.pyw” or “Executable Test.pyw” (depends on which one the reader is checking; they both do principally the same thing) creates an interface, from where the data will be inserted, with the means to create a PowerPoint presentation.

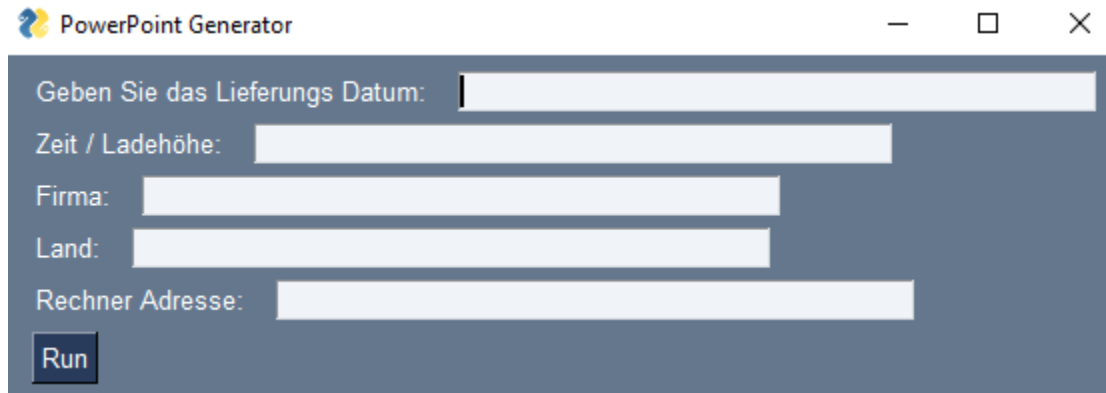
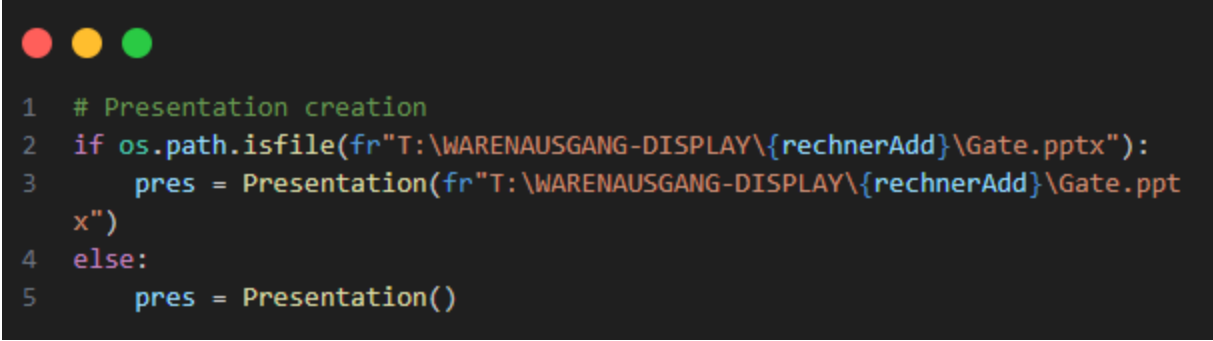


Figure 1 - Program Interface

As seen by Figure 1, the interface takes 5 inputs from the user, the four first are data that will be shown in the PowerPoint presentation, and can be used to test, if the data will be overwritten. The fifth input field “Rechner Adresse” is used to define the HILTI-ID of the PC, this will be used to then create a subfolder in the main folder, where the presentation for said PC will be either created, or just accessed/overwritten. This code will then be connected to a PowerShell Bash script, which will be used to see if the PowerPoint has been edited, and when it is edited, then the application will restart itself and open itself again with the new edits, in Presentation mode. The files are in PYW format, because we wanted to execute them for the testing without needing to open the code every time, later we converted them into .exe files, so that every PC can use them; even without the need of python.

PROGRAM LIBRARIES / MODULES:

The modules used are: **os**, **PySimpleGUI**, **python-pptx** & **pyautogui**. The usage of these is crucial, so that the Presentation and the interface will be created. With **OS**, we do some system checks, these checks contain path checks, to establish that the presentation is created. For example, in Figure 2, we can see the if-else, to check if the path has been created before, we need that to then later decide, with the “Presentation()” constructor, if we want to create a presentation from scratch, or if we want to access the already created Presentation and just edit it. This keeps the code from making redundant data. We also have the same thing with the Directories, where we check if the directory for the presentation is already created or not, same logic as Figure 2.




```

1  # Presentation creation
2  if os.path.isfile(fr"T:\WARENAUSGANG-DISPLAY\{rechnerAdd}\Gate.pptx"):
3      pres = Presentation(fr"T:\WARENAUSGANG-DISPLAY\{rechnerAdd}\Gate.ppt
4  x")
5  else:
6      pres = Presentation()

```

Figure 2 - OS File Path Checking

The library “**PySimpleGUI**” is used to create the Graphical Interface, from which the user adds the data to the presentation. The main module that the program requires is “python-pptx”, where the Presentation, the slides and the textboxes will be created with the required data. The last library needed, is “**pyautogui**”, now this is not used in the main code, but rather in an additional file which is named “F5.pyw”, this pretty much makes it so the key “F5” is automated and this then is connected to the PowerShell script, which then scripts it to be executed when the PowerPoint opens, so that it puts it directly in Presentation mode.



```

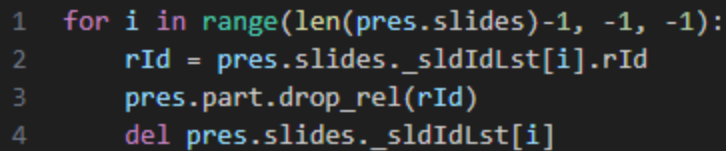
1  import pyautogui
2
3  pyautogui.press('f5')

```

Figure 3 - F5.pyw Automation

CREATION OF THE PRESENTATION:


Besides the explanation here, the whole python script has comments throughout it, so if anything happens to be unclear, then you can also refer to the comments there. To create the presentation, we must first understand how the library works and what the “Presentation()” constructor does. The constructor creates a presentation which will then be filled, but it can be an empty- or a parameter constructor, which respectively means, that it will either create a new file from scratch and save it to the given path later in the end, or that it will access an already existing file. Now for the overwriting and the creation of these slides, we naturally need to access the files, so that we remove redundancy. (just like it’s said before in the module explanation) The thing is, that if we want to run this code, and access the file, it will always create a new slide, that’s why we use a for each loop to iterate and delete all previous slides, to basically always create a new presentation, as seen in Fig. 4

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in a light-colored font and shows a loop that iterates over the slides of a presentation in reverse order, deleting each slide by its ID.

```
1 for i in range(len(pres.slides)-1, -1, -1):
2     rId = pres.slides._sldIdLst[i].rId
3     pres.part.drop_rel(rId)
4     del pres.slides._sldIdLst[i]
```

Figure 4 - For each slide delete

After that we create the slides, a good tip is also to follow the documentation of python-pptx and stack overflow help, because it can get tricky to change certain details. We first create a layout for the slides, so we choose from the default layouts of PowerPoint which are all numbered according to their order. We then create a slide, which should follow said layout, and, in the slide, we also place the background image which is hard coded as a raw string in the Folder, if it should ever be deleted you should refer to the old viewGate.pptx files, and just screenshot the background image there. We then create the title and subtitle which hold the most important info of the slide, that being the company, the place and the dates. Throughout the whole texts, f-Strings, r-Strings or fr-Strings will be used, whenever you see r before the strings, means that you are dealing with a raw string, so no string commands can be executed in strings (e.g., “\n” creates a new line, but r“\n” outputs the string as it is). F-Strings are then used to add the input data in one string instead of dividing it in a lot of strings with “+” etc. Fr-Strings are a combination of both, which we use for the file paths. The other PowerPoint based functions are the textbox creation and the stylistic changes. The textbox is the most difficult tool to coordinate, because it requires the user to input the position, as well as the width and height of the element in inches, and to try and coordinate it properly in the presentation is a challenge. The creation is then made by adding the aforementioned properties to the “add_textbox()” function, as parameters. All of these elements besides the subtitle, are a part of the class “slide.shapes” which, just as the name suggests, create these so-called shapes in the slide. Lastly, we change the font name, size and/or color of the textboxes created, with an additional variable which just helps to shorten the commands that will be executed below. In the chance that the directory for the requested PC is not created, the directory will be then created before the file will be saved in said directory. This all is in a function named “datenSammel()”, where the parameters are the variables that will be in the input fields, these variables are then used throughout the whole script, for the creation of info in the textboxes.



```

1  # Slide Creation, Insert a Layout in the Array
2      slide_layout = pres.slide_layouts[0]
3      slide = pres.slides.add_slide(slide_layout)
4      pic = slide.shapes.add_picture(r"T:\WARENAUSGANG-DISPLAY\Scripte\bg.png", 0, 0,
width=pres.slide_width, height=pres.slide_height)
5      slide.shapes._spTree.remove(pic._element)
6      slide.shapes._spTree.insert(2, pic._element)
7
8
9      # Title and subtitle are added with the text
10     title = slide.shapes.title
11     subtitle = slide.placeholders[1]
12     title.text = f"{liefertFirma} {landFirma}"
13     subtitle.text = f"Abholung am {lieferungDatum} von {ladeZeit}"
14
15     # Textbox creation with the cardinal directions set in inches
16     left = Inches(10)
17     height = Inches(5.5)
18     top = Inches(8.5)
19     width = Inches(5)
20     textBox = slide.shapes.add_textbox(left, top, width, height)
21     tf = textBox.text_frame
22     tf.text = f"{rechnerAdd}"
23
24     # Font and color changes in title and subtitle
25     title_para = slide.shapes.title.text_frame.paragraphs[0]
26     subtitle_para = slide.placeholders[1].text_frame.paragraphs[0]
27     title_para.font.name = "Arial Black"
28     title_para.font.size = Pt(76)
29     subtitle_para.font.name = "Arial"
30     subtitle_para.font.size = Pt(50)
31     subtitle_para.font.color.rgb = RGBColor(255, 0, 0)
32     tf.paragraphs[0].font.name = "Arial Black"
33     tf.paragraphs[0].font.size = Pt(24)

```

Figure 5 - Creation of the PowerPoint

MAIN FUNCTION:

The main function holds the GUI created by PySimpleGUI. We create the Interface with a layout, which is nothing more than a list, which holds the text to request the user something, the input fields and the button to run the code. The Window will then be created with the aforementioned layout. A While loop will run to keep the loop alive, until either the button "run" or the X-Button are pressed. When we "run" the interface, data

from the input fields will be inserted into the PowerPoint and the window will then be closed.

```
1  sg.theme("DarkBlue3")
2  layout = [
3      [sg.Text("Geben Sie das Lieferungs Datum: "), sg.Input(key="lieferungDatum")],
4      [sg.Text("Zeit / Ladehöhe: "), sg.Input(key="ladeZeit")],
5      [sg.Text("Firma: "), sg.Input(key="lieferteFirma")],
6      [sg.Text("Land: "), sg.Input(key="landFirma")],
7      [sg.Text("Rechner Adresse: "), sg.Input(key="rechnerAdd")],
8      [sg.Button("Run")]
9  ]
10
11 window = sg.Window("PowerPoint Generator", layout, resizable=True)
12
13 while True:
14     event, values = window.read()
15     if event == sg.WIN_CLOSED:
16         break
17     elif event == "Run":
18         datenSammel(
19             values["lieferungDatum"],
20             values["ladeZeit"],
21             values["lieferteFirma"],
22             values["landFirma"],
23             values["rechnerAdd"]
24         )
25     sg.popup("Presentation generated successfully!", title="Success")
26     window.close()
```

Figure 6 - Main Function

CLOSING STATEMENTS AND THINGS TO WATCH OUT FOR:

The code will then be converted into an .exe which is able to be executed by each user, it's foolproof and simple to use (to convert it, the pc that does the conversion should have all the libraries installed). The presentation will be newly-opened every time something changes, due to PowerPoint not allowing the user to change information in the slide while it's running. The PowerShell code, will always be correlated with these codes, should anything in the paths change, the PowerShell script shall also be updated. Just as a heads' up, what in the PowerShell script happens, is the checking of the existence of the file and then of the "date modified", so that it can acknowledge when to restart the program and alter the information.