

Workbooks and Worksheets

Working with Excel in Python

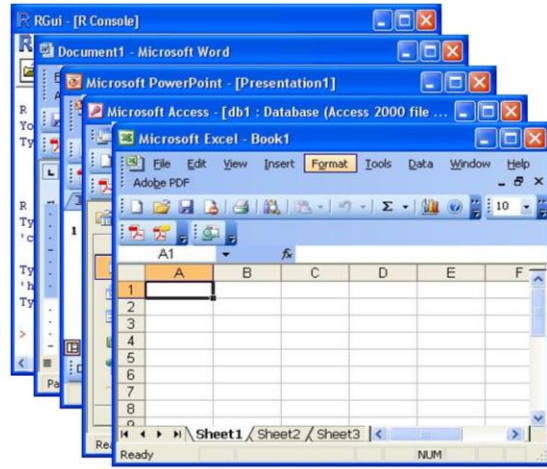
1

This video will discuss how to use the win32com module to work with Excel workbooks and worksheets.

Scripting is not just for ArcGIS

- Python can be used with other programs including:

- R statistical software
- Word
- PowerPoint
- Access
- Excel



Python can access many other software applications besides ArcGIS.

These applications include R and Microsoft Office products.

Accessing other software

- The `win32com.client` module provides access to other software applications:

```
import win32com.client
```

- This module does not automatically install with ArcGIS. It can be downloaded from:
<http://sourceforge.net/projects/pywin32/files/pywin32/Build216/>
- Select pywin version that matches your version of Python
 - i.e. download `pywin32-216.win32-py2.7.exe` for Python 2.7

3

Modules, like `arcpy`, give Python to access other software applications. We'll use the **`win32com.client`** module for working with Microsoft Excel.

`win32com` is not an out-of-the-box module and must be downloaded and installed. The link provided in the notes is one location from which it can be downloaded.

The version must match your version of Python (i.e. python 2.7 for ArcGIS 10.3). If your Python installation is 32-bit, then you will need the win32 version, even if your computer is 64-bit.

Creating Microsoft Applications

- Create the application object with the appropriate statement below...

```
xlApp = win32com.client.Dispatch("Excel.Application")
```

```
access_App = win32com.client.Dispatch("Access.Application")
```

```
word_App = win32com.client.Dispatch("Word.Application")
```

```
ppt_App = win32com.client.Dispatch("Powerpoint.Application")
```

- Make application visible...

```
xlApp.Visible = 1
```

4

Applications can be accessed using win32com's **Dispatch** method and specifying the appropriate name of the application. These example statements show how to create application objects for several Microsoft applications including Excel, Access, Word, and Powerpoint. This lecture will focus on the Excel application.

Setting the application's **visible** property to 1 (i.e. True) will make it visible while the script works with it. Setting the visible property to zero will make the application run in the background.

Workbooks

- Open existing workbook...

```
workbook = xlApp.Workbooks.Open(r"C:\Data\TextFile.xls")
```

- Close and save an existing workbook...

```
workbook.Close(SaveChanges = True)
```

set as False to close
without saving

- Create a new workbook...

```
workbook = xlApp.Workbooks.Add()
```

- Save workbook as...

```
workbook.SaveAs(r"C:\Data\TextFile.xls")
```

5

To create a workbook object by opening an existing Excel file, use the **Workbooks.Open** method of the application object.

To close a workbook, use the workbook object's **Close** method. The workbook can be saved by specifying the `SaveChanges = True`.

To create a new workbook, use the `Workbooks.Add` method of the application object.

You can save the workbook, with a new file name, by using the workbook's **SaveAs** method.

Example: creating a new workbook

```
# import module...
import win32com.client

# create application object...
xlApp = win32com.client.Dispatch("Excel.Application")

# make application visible...
xlApp.Visible = 1

# create new workbook...
workbook = xlApp.Workbooks.Add()
```

6

This slide will show an example of the code for creating a new workbook. The first statement imports the win32com.client module.

The **Dispatch** method creates the Excel application object.

The application is made visible.

And a new workbook is created using the **Workbooks.Add** method.

Worksheets

- Get existing worksheet...

```
worksheet = workbook.Worksheets("Sheet1")
```

- Create a new worksheet ...

```
worksheet = workbook.Sheets.Add()
```

- Change worksheet name...

```
worksheet.Name = "Example sheet"
```

7

This statement shows how to get the "Sheet1" worksheet from the workbook.

To create a new worksheet, use the workbook's **Sheets.Add** method.

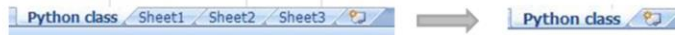
You can change the name of a worksheet by modifying the **Name** property.

Copy worksheet

- Create copy of worksheet in new workbook...

```
worksheet.Copy()
```

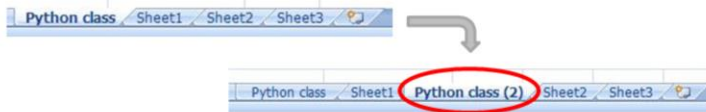
Note: this creates
a new Excel file



- Create copy of worksheet in same workbook...

```
worksheet.Copy(workbook.Worksheets("Sheet2"))
```

Place copy before this sheet



8

A worksheet can be copied to a new Excel file.

Or a worksheet can be copied within its current workbook. The copy is placed before the specified worksheet.

Delete worksheet

- Delete worksheet...

```
workbook.Worksheets("Sheet1").Delete()
```

- Excel will prompt to confirm deletion

- To disable alerts...

```
xlApp.DisplayAlerts = False
```

9

A worksheet can be deleted from a workbook. By default, Excel will prompt you to confirm the deletion of a worksheet – the script would pause until you respond to the prompt.

The **DisplayAlerts** can be set to False which will disable any prompts. Deletions or saving over existing files will simply be performed without needing confirmation.