35 Working with Excel files

35.1 About

Microsoft Excel is one of the most widely used software applications in the world and it is common to find data files in Excel format.
There are two native Excel formats commonly used:

- **.xls**, known as Excel binary file format. This is a proprietary format which was the main format used in versions of Excel up to Excel 2003.
- **.xlsx**, known as Office open XML format. This is an open format which was the main format used in Excel 2007 and is the main format used up to the present day.

Although Microsoft encourages users to use the .xlsx format, it is common to find files in the older .xls format, including files provided by public dataset providers. If you have Excel on your machine it is easy to open a .xls file and re-save it as .xlsx. There are also several other spreadsheet packages such as Numbers (on Mac) or OpenOffice that can convert between the two. If you don't have any of these package available, then there are online services such as Zamzar (https://www.zamzar.com/convert/xls-to-xlsx/) that can convert between them (as well as between several other file types).

In this activity we will use the Python library **Openpyxl** to work with .xlsx files.

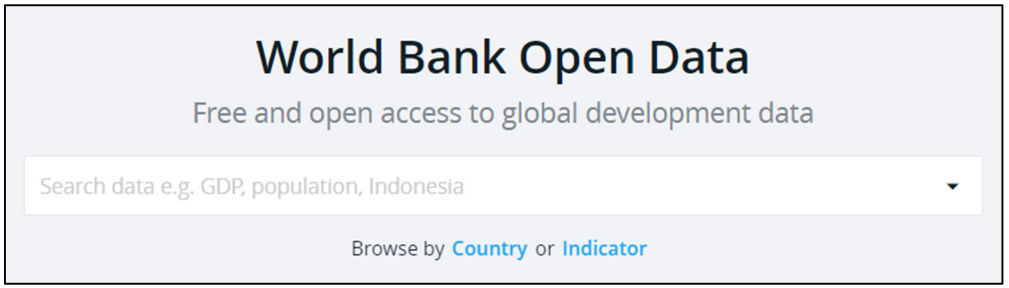
Note- Openpyxl can only handle .xlsx files. If you do have files in .xls format, and have no way of converting between them, then there is another Python library, **Xlrd**, that can be used to handle .xls files. At the time of writing it has no active maintainers, so it is not recommended to use it.

35.2 Get a dataset

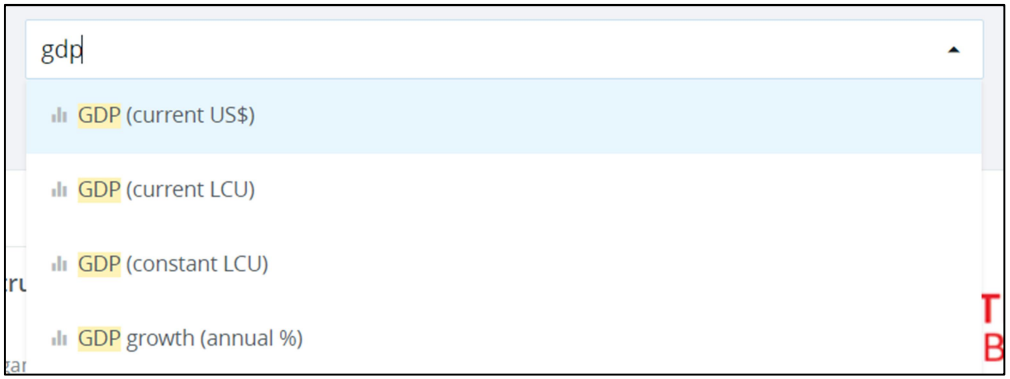
35.2.1 Download a file

For this activity we will require a spreadsheet file to use. There are a number of different public sources of datasets that can be used- the World Health Organisation (WHO), International Monetary Fund (IMF) and World Bank all have a catalogue of datasets that are available to download. In this example we will use a file from the World Bank.

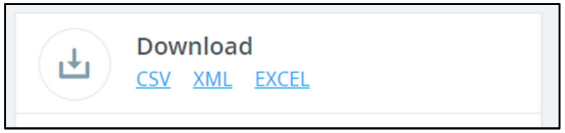
Start by opening a browser and visiting https://data.worldbank.org/. The World Bank Open Data site opens.



In the search field, type **GDP**. There will be several results.



Pick **GDP (current US$)** which will probably be the first result.
When the page appears find the **Download** panel.

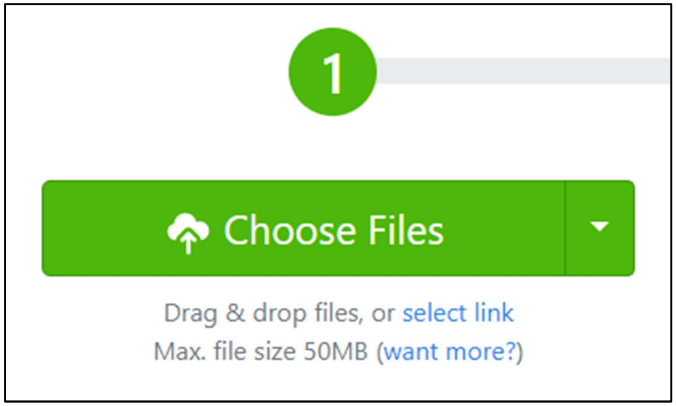


Click **EXCEL**. The file is downloaded. Depending on which browser you are using, it will probably be saved into your **Downloads** folder.

35.2.2 Convert the file

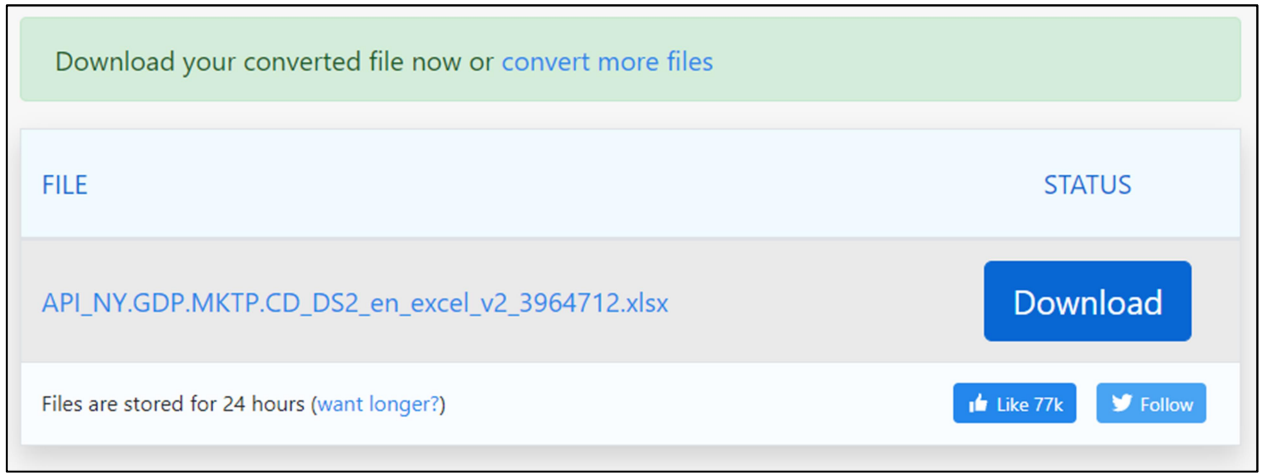
Note that the file downloaded from the World Bank website is a .xls file.

If you have Excel or another spreadsheet package on your computer, you can open the file in that application and resave it in .xlsx format.
If there is no suitable application installed, use Zamzar to convert the file. To do this, in the browser visit https://www.zamzar.com/convert/xls-to-xlsx/.



Click **Choose Files…** and navigate to the file that was just downloaded.
Leave the target file type as **.xlsx**. Click **Convert Now**.

When the conversion process is complete click **Download**.



The .xlsx file is downloaded.
Use the Windows file explorer to copy the file to somewhere convenient such as your **Documents** folder and rename it to **gdp.xlsx**.

35.2.3 Examine the file

If you have Excel or another spreadsheet application on your computer that can read .xlsx files, open the file and have a look at it. The file contains historical GDP (gross domestic product) data for all the countries in the world from 1960 onwards.
Note that there are sheets called **Data**, **Metadata - Countries** and **Metadata - Indicators**. The main GDP data is on the sheet called **Data**.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Data Source	World Development Indicators										
2	Last Updated Date	15/10/2020										
3												
4	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	1966	1967
5	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD								
6	Afghanistan	AFG	GDP (current US$)	NY.GDP.MKTP.CD	537777811.1	548888895.6	546666677.8	751111191.1	800000044.4	1006666638	1399999967	1673333418
7	Angola	AGO	GDP (current US$)	NY.GDP.MKTP.CD								
8	Albania	ALB	GDP (current US$)	NY.GDP.MKTP.CD								
9	Andorra	AND	GDP (current US$)	NY.GDP.MKTP.CD								
10	Arab World	ARB	GDP (current US$)	NY.GDP.MKTP.CD								
11	United Arab Emirates	ARE	GDP (current US$)	NY.GDP.MKTP.CD								
12	Argentina	ARG	GDP (current US$)	NY.GDP.MKTP.CD			24450604878	18272123664	25605249382	28344705967	28630474728	24256667553
13	Armenia	ARM	GDP (current US$)	NY.GDP.MKTP.CD								
14	American Samoa	ASM	GDP (current US$)	NY.GDP.MKTP.CD								
15	Antigua and Barbuda	ATG	GDP (current US$)	NY.GDP.MKTP.CD								
16	Australia	AUS	GDP (current US$)	NY.GDP.MKTP.CD	18577668272	19652816665	19892485161	21507447643	23764139321	25936835032	27268451114	30397580916
17	Austria	AUT	GDP (current US$)	NY.GDP.MKTP.CD	6592693841	7311749633	7756110210	8374175258	9169983886	9994070616	10887682273	11579431669
18	Azerbaijan	AZE	GDP (current US$)	NY.GDP.MKTP.CD								
19	Burundi	BDI	GDP (current US$)	NY.GDP.MKTP.CD	195999990	202999992	213500006	232749998	260750008	158994963	165444571.4	178297142.9
20	Belgium	BEL	GDP (current US$)	NY.GDP.MKTP.CD	11658722591	12400145222	13264015675	14260017387	15960106681	17371457608	18651883472	19992040788
21	Benin	BEN	GDP (current US$)	NY.GDP.MKTP.CD	226195579.4	235668222.4	236434906.8	253927646.5	269818988.3	289908720.6	302925280.8	306222000.4
22	Burkina Faso	BFA	GDP (current US$)	NY.GDP.MKTP.CD	330442817.2	350247237.1	379567178.3	394040749.1	410321785.6	422916848.4	433889831.6	450753993.2
23	Bangladesh	BGD	GDP (current US$)	NY.GDP.MKTP.CD	4274893913	4817580184	5081413340	5319458351	5386054619	5906636557	6439687598	7253575399
24	Bulgaria	BGR	GDP (current US$)	NY.GDP.MKTP.CD								
25	Bahrain	BHR	GDP (current US$)	NY.GDP.MKTP.CD								
26	Bahamas, The	BHS	GDP (current US$)	NY.GDP.MKTP.CD	169803921.6	190098039.2	212254902	237745098	266666666.7	300392156.9	340000000	390196078.4
27	Bosnia and Herzegovina	BIH	GDP (current US$)	NY.GDP.MKTP.CD								
28	Belarus	BLR	GDP (current US$)	NY.GDP.MKTP.CD								
29	Belize	BLZ	GDP (current US$)	NY.GDP.MKTP.CD	28071888.56	29964370.71	31856922.86	33749405.01	36193826.12	40069930.07	44405594.41	47379310.34
30	Bermuda	BMU	GDP (current US$)	NY.GDP.MKTP.CD	84466654.08	89249986.7	94149985.97	96366652.31	107566650.6	114339049	134173373.8	155102984.6
31	Bolivia	BOL	GDP (current US$)	NY.GDP.MKTP.CD	373879363.6	406684585.7	444665186.4	478805990.1	539491477.3	604377104.4	669191919.2	755808080.8
32	Brazil	BRA	GDP (current US$)	NY.GDP.MKTP.CD	15165569913	15236854859	19926293839	23021477292	21211892260	21790035117	27062716578	30591834054
33	Barbados	BRB	GDP (current US$)	NY.GDP.MKTP.CD								

35.3 Install Openpyxl

Openpyxl is the library we will use to read and write Excel files in Python.

Open a command prompt and enter the command:

```
pip install openpyxl
```

Wait for the setup to complete. Use **pip list** to check that the library appears in the list.

```
jdcal          1.4.1
openpyxl       3.0.5
pip            20.2.3
```

35.4 Create a project

Open PyCharm and create a new project. Call it **ExcelProject** or similar.

At the top of the main.py file, add the import for Openpyxl:

```
import openpyxl
```

35.5 Open and close the workbook

In this section we will need the .xlsx file that you downloaded earlier. Make sure you know where it is located. In the following commands, substitute your own folder name.
Open the workbook and display the sheet names. Don't forget that on Windows we need to use the **** notation to indicate a single **** in the folder path.
Close the workbook after using it.

```
workbook = openpyxl.load_workbook("C:\\Users\\User\\Documents\\gdp.xlsx")
print(workbook.sheetnames)

workbook.close()
```

Save and run and check the output matches the sheet names in the file.

```
['Data', 'Metadata - Countries', 'Metadata - Indicators']

Process finished with exit code 0
```

Note, closing a workbook is the last thing we can do with it. All the following lines of code go before the **close** line.

35.6 Get data from a sheet

By default, when a workbook is opened, the first sheet is active. We can also activate a sheet either using its number (index position) or sheet name.
Activate a sheet then print its title:

```
sheet = workbook["Data"]
print(sheet.title)
```

Save and run and check the output again:

```
['Data', 'Metadata - Countries', 'Metadata - Indicators']
Data

Process finished with exit code 0
```

Recall that the worksheet looks like this:

	A	B	
1	Data Source	World Development Indicators	
2	Last Updated Date	15/10/2020	
3			
4	Country Name	Country Code	Inc
5	Aruba	ABW	GD
6	Afghanistan	AFG	GD
7	Angola	AGO	GD
8	Albania	ALB	GD
9	Andorra	AND	GD

The actual country names are in column A and start on row 5.

To read a country name from an individual cell, add this:

```
print(sheet["A5"].value)
```

This should give us the name of the country in that cell:

```
Aruba

Process finished with exit code 0
```

Note that the sheet also has a cell method that lets us specify the row and column separately, which is useful for iterating. Add this:

```
print(sheet.cell(5, 1).value)
```

In this example **5** represents the row, **1** represents the column (i.e. column A).

To get data from multiple cells, such as a range or a whole column or row, the selection can be specified just as it is in Excel itself. This returns a tuple of cell objects, which can then be iterated using Python's normal iteration methods.
Add this code:

```
country_names = sheet["A"]
for country_name in country_names:
    print(country_name.value)
```

This produces an output that starts like this:

```
Data Source
Last Updated Date
None
Country Name
Aruba
Afghanistan
Angola
Albania
Andorra
Arab World
United Arab Emirates
Argentina
Armenia
American Samoa
Antigua and Barbuda
Australia
```

Note that this also gives us some values that we are not interested in, from the first few rows of the table (**Data Source** and so on).
Although the countries start at row 5, the last row might change in different copies of the spreadsheet (for example, if a disputed territory is claimed by a country) so it might not be a good idea to rely on the spreadsheet ending at a certain row. Instead, change the specification of the data range to use the **iter_rows** method like this:

```
country_names = sheet.iter_rows(min_col = 1, max_col = 1, min_row = 5, values_only = True)
for country_name in country_names:
```

```
    print(country_name[0])
```

This time the output only contains the values we want:

```
Aruba
Afghanistan
Angola
Albania
Andorra
Arab World
United Arab Emirates
Argentina
Armenia
American Samoa
```

Scroll down to the end to make sure the last value is actually the last value from the spreadsheet:

```
Kosovo
Yemen, Rep.
South Africa
Zambia
Zimbabwe

Process finished with exit code 0
```

Now let's change the range to include the GDP data. The value for 2020 (at the time of writing, the last column which contains data) is in column BM which is column 65.
Change the code to get all the data and display the whole tuple:

```
country_names = sheet.iter_rows(min_col = 1, max_col = 64, min_row = 5, values_only = True)
for country_name in country_names:
    print(country_name)
```

This should produce an output like this:

Let's change the output to show only the country name and the GDP for 1990 (column AI which is column 35, index 34 in the tuple) and 2019 (column BL which is column 64, index 63 in the tuple)

```
country_names = sheet.iter_rows(min_col = 1, max_col = 64, min_row = 5, values_only = True)
for country_name in country_names:
    print("{} : {} : {}".format(country_name[0], country_name[34], country_name[63]))
```

This time the output should be like this:

```
Aruba : 764887117.194486 : None
Afghanistan : None : 19101353832.737125
Angola : 11228764963.161764 : 94635415869.98508
Albania : 2028553750 : 15278077446.864292
Andorra : 1029048481.8805093 : 3154057987.23833
Arab World : 447404218571.6186 : 2815410447182.2505
United Arab Emirates : 50701443748.29747 : 421142267937.65015
Argentina : 141352368714.6913 : 449663446954.07275
Armenia : 2256838858.42714 : 13672802157.832392
American Samoa : None : None
Antigua and Barbuda : 459469058.88888884 : 1727759259.2592592
```

35.7 Using data from multiple sheets

Let's say that we want to output data only for countries in the Latin America and Caribbean region.

The sheet called **Metadata - Countries** contains a list of countries and their regions:

	A	B	C	D	E
1	Country Code	Region	IncomeGroup	SpecialNotes	TableName
2	ABW	Latin America & Caribbean	High income		Aruba
3	AFG	South Asia	Low income		Afghanistan
4	AGO	Sub-Saharan Africa	Lower middle income		Angola
5	ALB	Europe & Central Asia	Upper middle income		Albania
6	AND	Europe & Central Asia	High income		Andorra
7	ARB			Arab World aggregate	Arab World
8	ARE	Middle East & North Africa	High income		United Arab Emirates
9	ARG	Latin America & Caribbean	Upper middle income		Argentina
10	ARM	Europe & Central Asia	Upper middle income		Armenia
11	ASM	East Asia & Pacific	Upper middle income		American Samoa
12	ATG	Latin America & Caribbean	High income		Antigua and Barbuda
13	AUS	East Asia & Pacific	High income	Fiscal year end: June	Australia

Note that the actual country data starts on row 2.

Each country has a code (column A) and a region (column B). This looks like a dictionary. Recall that a dictionary is a set of key/value pairs. We can use the country code as the key and the region as the value.

First let's read the values from the sheet and display them. Add this code after the lines that open the workbook.

```
sheet = workbook["Metadata - Countries"]
country_regions = sheet.iter_rows(min_col = 1, max_col = 2, min_row = 2, values_only = True)

for country_region in country_regions:
    print("{} : {}".format(country_region[0], country_region[1]))
```

Check that we get the output we want:

```
ABW : Latin America & Caribbean
AFG : South Asia
AGO : Sub-Saharan Africa
ALB : Europe & Central Asia
AND : Europe & Central Asia
ARB : None
ARE : Middle East & North Africa
ARG : Latin America & Caribbean
ARM : Europe & Central Asia
ASM : East Asia & Pacific
ATG : Latin America & Caribbean
AUS : East Asia & Pacific
AUT : Europe & Central Asia
AZE : Europe & Central Asia
BDI : Sub-Saharan Africa
BEL : Europe & Central Asia
BEN : Sub-Saharan Africa
```

Now add the lines that convert these items to a dictionary:

```
dict_regions = dict(country_regions)
print(dict_regions)
```

The output:

```
{'ABW': 'Latin America & Caribbean', 'AFG': 'South Asia', 'AGO': 'Sub-Saharan Africa', 'ALB': 'Europe & Central Asia', 'AND': 'Europe & Centr

Process finished with exit code 0
```

Change the code that reads the GDP values from the Data sheet to the following. Note that on the Data sheet, the country code is in column B (column 2, index 1). For each row, the code finds the region in the dictionary we just created.

```
sheet = workbook["Data"]
country_names = sheet.iter_rows(min_col = 1, max_col = 64, min_row = 5, values_only = True)
for country_name in country_names:
    region_code = dict_regions[country_name[1]]
    print("{} : {} : {} : {}".format(country_name[0], region_code, country_name[34], country_name[63]))
```

This runs OK:

```
Aruba : Latin America & Caribbean : 764887117.194486 : None
Afghanistan : South Asia : None : 19101353832.737125
Angola : Sub-Saharan Africa : 11228764963.161764 : 94635415869.98508
Albania : Europe & Central Asia : 2028553750 : 15278077446.864292
Andorra : Europe & Central Asia : 1029048481.8805093 : 3154057987.23833
Arab World : None : 447404218571.6186 : 2815410447182.2505
United Arab Emirates : Middle East & North Africa : 50701443748.29747 : 421142267937.65015
Argentina : Latin America & Caribbean : 141352368714.6913 : 449663446954.07275
Armenia : Europe & Central Asia : 2256838858.42714 : 13672802157.832392
American Samoa : East Asia & Pacific : None : None
Antigua and Barbuda : Latin America & Caribbean : 459469058.88888884 : 1727759259.2592592
Australia : East Asia & Pacific : 310777222008.4648 : 1392680589329.1375
```

But encounters a problem when a country code from the Data sheet is not found in the dictionary:

```
Dominica : Latin America & Caribbean : 201428730 : 596033333.3333333
Denmark : Europe & Central Asia : 138247285815.85495 : 348078018463.9052
Dominican Republic : Latin America & Caribbean : 7073675544.808465 : 88941298257.72153
Traceback (most recent call last):
  File "C:\python-workspace\ExcelProject\main.py", line 17, in <module>
    region_code = dict_regions[country_name[1]]
KeyError: 'INX'
```

Modify the loop to only display the output if the country code is actually in the dictionary:

```
for country_name in country_names:
    if country_name[1] in dict_regions:
        region_code = dict_regions[country_name[1]]
        print("{} : {} : {} : {}".format(country_name[0], region_code, country_name[34], country_name[63]))
```

This time the code executes with no problems:

```
Samoa : East Asia & Pacific : 125766269.75535831 : 850655017.2204882
Kosovo : Europe & Central Asia : None : 7926108374.384236
Yemen, Rep. : Middle East & North Africa : 5647119229.007634 : None
South Africa : Sub-Saharan Africa : 115552349035.44061 : 351431649241.43854
Zambia : Sub-Saharan Africa : 3285217391.3043475 : 23064722446.351265
Zimbabwe : Sub-Saharan Africa : 8783816700 : 21440758800

Process finished with exit code 0
```

Since we only want countries in the Latin America and Caribbean region, modify the loop one more time:

```
for country_name in country_names:
    if (country_name[1] in dict_regions) and (dict_regions[country_name[1]] == "Latin America & Caribbean"):
        region_code = dict_regions[country_name[1]]
        print("{} : {} : {} : {}".format(country_name[0], region_code, country_name[34], country_name[63]))
```

This time we get the output we want:

```
Aruba : Latin America & Caribbean : 764887117.194486 : None
Argentina : Latin America & Caribbean : 141352368714.6913 : 449663446954.07275
Antigua and Barbuda : Latin America & Caribbean : 459469058.88888884 : 1727759259.2592592
Bahamas, The : Latin America & Caribbean : 3166000000 : 12827000000
Belize : Latin America & Caribbean : 412086445.49217653 : 1879613600
Bolivia : Latin America & Caribbean : 4867582620.207083 : 40895322865.41244
Brazil : Latin America & Caribbean : 461951781999.99994 : 1839758040765.623
Barbados : Latin America & Caribbean : 2012131457.2664447 : 5209000000
Chile : Latin America & Caribbean : 33113887817.97311 : 282318159744.6496
Colombia : Latin America & Caribbean : 47844090709.990845 : 323802808108.246
Costa Rica : Latin America & Caribbean : 5711687786.759886 : 61773944173.673645
Cuba : Latin America & Caribbean : 28645436569.148937 : None
Curacao : Latin America & Caribbean : None : None
```

35.8 Checkpoint

At this stage your script should look like this:

```
import openpyxl

workbook = openpyxl.load_workbook("C:\\Users\\User\\Documents\\gdp.xlsx")

sheet = workbook["Metadata - Countries"]
country_regions = sheet.iter_rows(min_col = 1, max_col = 2, min_row = 2, values_only = True)
dict_regions = dict(country_regions)

sheet = workbook["Data"]
country_names = sheet.iter_rows(min_col = 1, max_col = 64, min_row = 5, values_only = True)
for country_name in country_names:
    if (country_name[1] in dict_regions) and (dict_regions[country_name[1]] == "Latin America & Caribbean"):
        region_code = dict_regions[country_name[1]]
        print("{} : {} : {} : {}".format(country_name[0], region_code, country_name[34], country_name[63]))
```

```
workbook.close()
```

35.9 Write data to the spreadsheet

Now that we have extracted the data we want, let's write it to another sheet in the workbook.

Start by creating a new sheet in the workbook and adding some titles. To do this, put this code after the workbook is opened.

```
workbook.create_sheet("LA&C")

new_sheet = workbook["LA&C"]
new_sheet["A1"] = "Latin America and Caribbean GDP"
new_sheet["A3"] = "Country code"
new_sheet["B3"] = "Country name"
new_sheet["C3"] = "1990"
new_sheet["D3"] = "2019"
```

Change the code in the loop to add the data to the new sheet.

```
for country_name in country_names:
    if (country_name[1] in dict_regions) and (dict_regions[country_name[1]] == "Latin America & Caribbean"):
        region_code = dict_regions[country_name[1]]
        print("{} : {} : {} : {}".format(country_name[0], region_code, country_name[34], country_name[63]))
        new_sheet.append((country_name[1], country_name[0], country_name[34], country_name[63]))
```

Add a line to save the workbook to a new file. Put this directly before the workbook is closed. Change the folder name to your own.

```
workbook.save("C:\\Users\\User\\Documents\\gdp_modified.xlsx")
```

Run the script. A new file will be created. Open it in Excel (or another application that supports .xlsx files), and check that the data has been added to the new sheet.

	A	B	C	D	E
1	Latin America and Caribbean GDP				
2					
3	Country code	Country name	1990	2019	
4	ABW	Aruba	764887117.2		
5	ARG	Argentina	1.41352E+11	4.49663E+11	
6	ATG	Antigua and Barbuda	459469058.9	1727759259	
7	BHS	Bahamas, The	3166000000	12827000000	
8	BLZ	Belize	412086445.5	1879613600	
9	BOL	Bolivia	4867582620	40895322865	
10	BRA	Brazil	4.61952E+11	1.83976E+12	
11	BRB	Barbados	2012131457	5209000000	
12	CHL	Chile	33113887818	2.82318E+11	
13	COL	Colombia	47844090710	3.23803E+11	
14	CRI	Costa Rica	5711687787	61773944174	
15	CUB	Cuba	28645436569		
16	CUW	Curacao			
17	CYM	Cayman Islands			
18	DMA	Dominica	201428730	596033333.3	
19	DOM	Dominican Republic	7073675545	88941298258	
20	ECU	Ecuador	15239278100	1.07436E+11	
21	GRD	Grenada	278098763	1228170370	
22	GTM	Guatemala	7650125217	76710385880	

35.10 Challenge

35.10.1 High earners

The metadata tab also has a column indicating if the country is a high, middle or low income country. Add another tab to the spreadsheet that includes the GDP for all high income countries for the years 2000, 2010 and 2019.

35.10.2 And clause order

In the code in this section, there is an **if** statement that looks like this

```
if (country_name[1] in dict_regions) and (dict_regions[country_name[1]] == "Latin America & Caribbean"):
```

Switch the statement around so that the parts of the statement are the opposite way around, like this:

```
if (dict_regions[country_name[1]] == "Latin America & Caribbean") and (country_name[1] in dict_regions):
```

Run the script again. What happens? Why?