Module 09 – External App Conectivity(Excel)

Session Topics

- Create Spreadsheet using openpyxl
- Reading Excel Spreadsheets with openpyxl
- Writing Excel Spreadsheets with openpyxl

Python dictionary

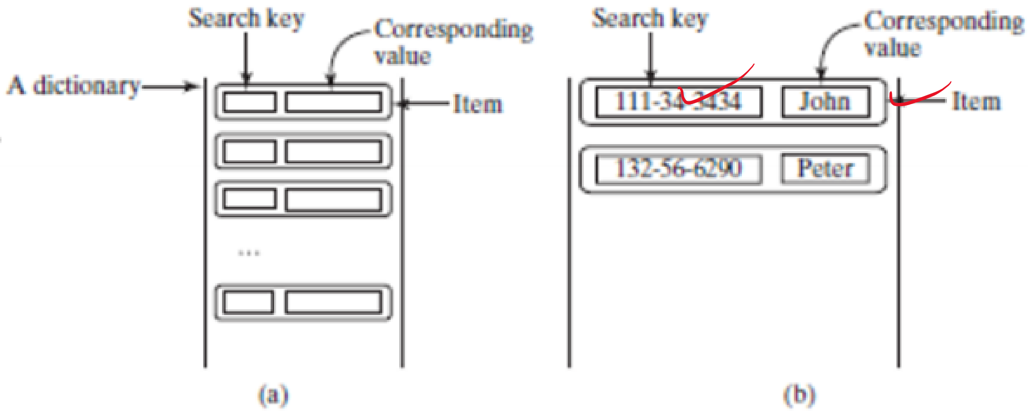
A dictionary is a collection that stores the values along with the keys.

The keys are like an index operator.

In a dictionary, the key must be a hashable object.

A dictionary cannot contain duplicate keys.

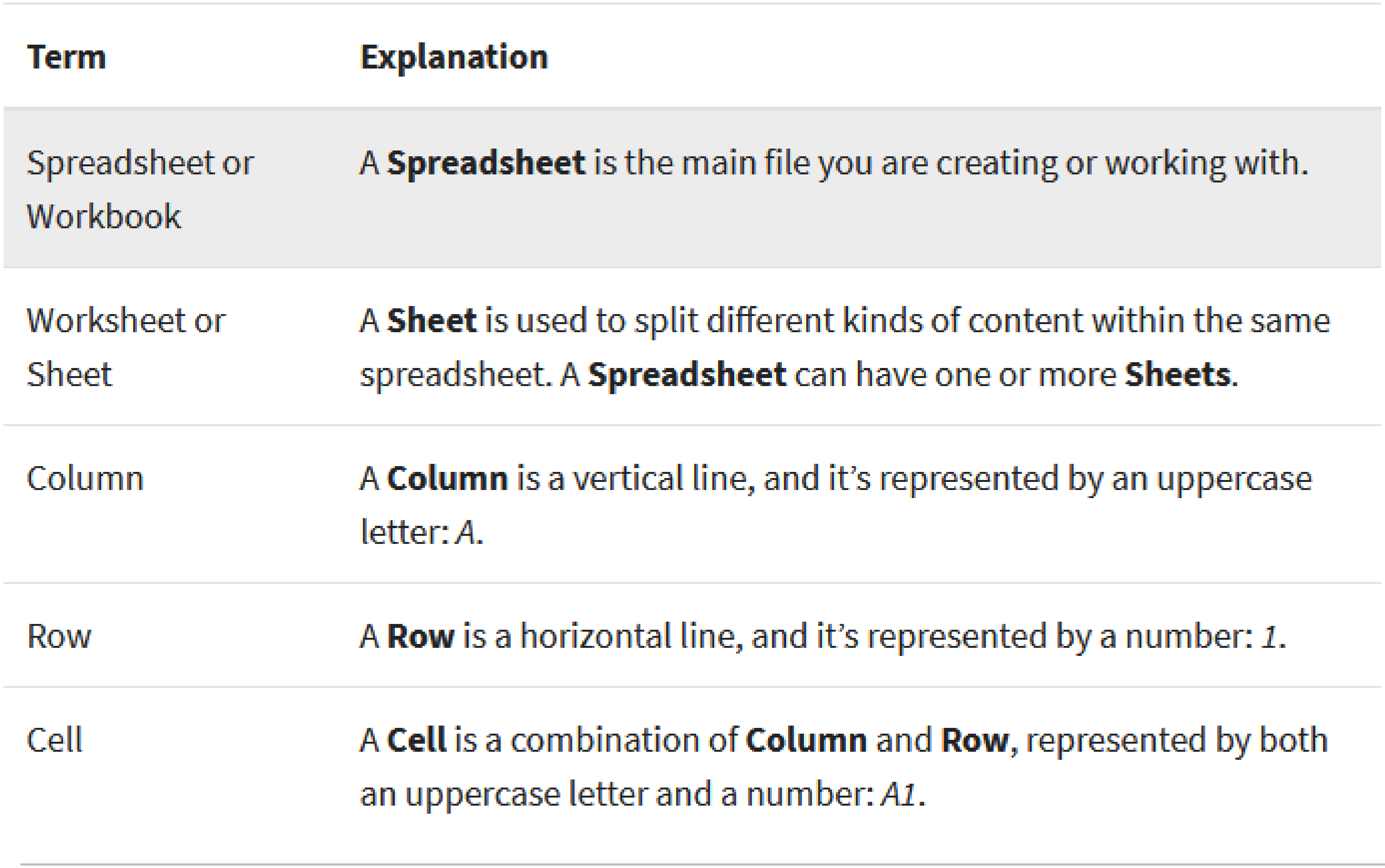
Each key maps to one value.



A key and its corresponding value form an *item* (or *entry*) stored in a dictionary, as shown in below:

```
Students[‘00011123’] = “John Smith”
```

Basic Excel Terminology



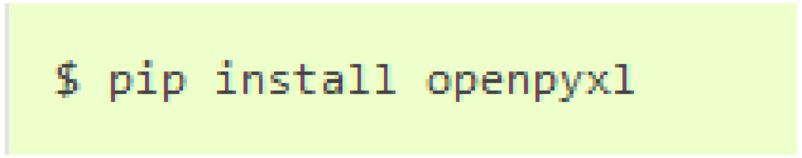
Term	Explanation
Spreadsheet or Workbook	A **Spreadsheet** is the main file you are creating or working with.
Worksheet or Sheet	A **Sheet** is used to split different kinds of content within the same spreadsheet. A **Spreadsheet** can have one or more **Sheets**.
Column	A **Column** is a vertical line, and it's represented by an uppercase letter: *A*.
Row	A **Row** is a horizontal line, and it's represented by a number: *1*.
Cell	A **Cell** is a combination of **Column** and **Row**, represented by both an uppercase letter and a number: *A1*.

openpyxl Library

Documentation found:

https://openpyxl.readthedocs.io/en/stable/

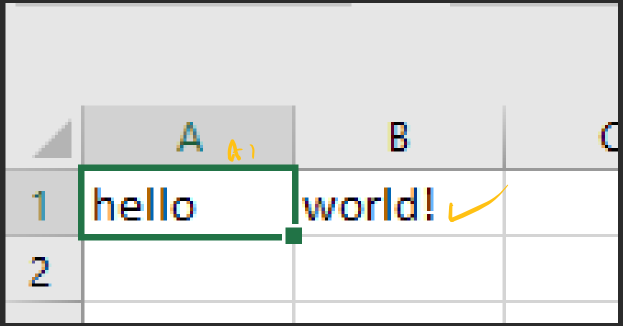
Installation:



```
$ pip install openpyxl
```

Create Spreadsheet using openpyxl

```
from openpyxl import Workbook

workbook = Workbook()
sheet = workbook.active

sheet["A1"] = "hello"
sheet["B1"] = "world!"

workbook.save(filename="hello_world.xlsx")
```

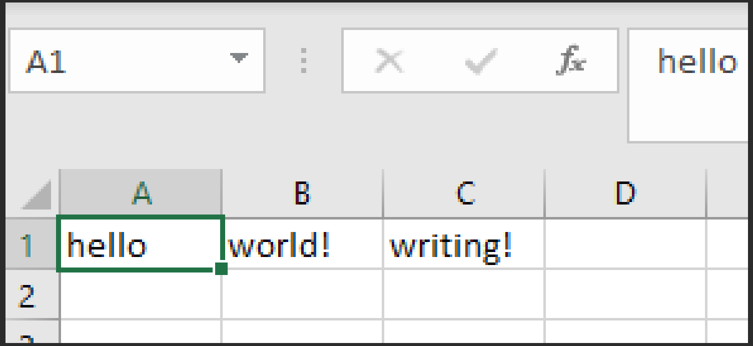


	A	B	C
1	hello	world!	
2			

hello_openpyxl.py

Appending to Excel Cell

```
from openpyxl import load_workbook

workbook = load_workbook(filename="hello_world.xlsx")
sheet = workbook.active

sheet["C1"] = "writing!"

workbook.save(filename="hello_world_append.xlsx")
```

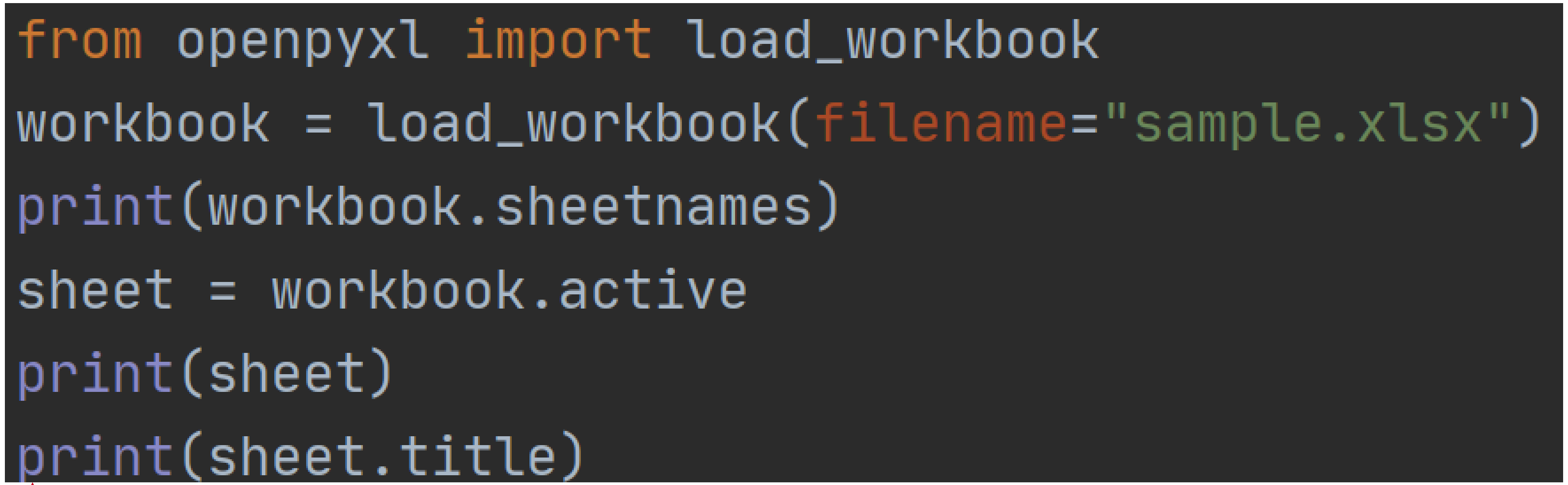


	A	B	C	D
1	hello	world!	writing!	
2				

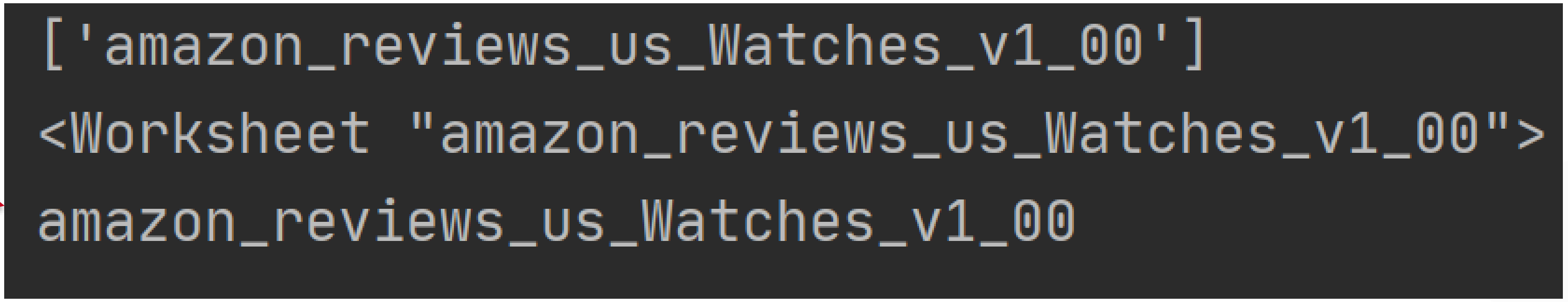
hello_append.py

Reading Excel Spreadsheets with openpyxl

- Exploring the sheet class:



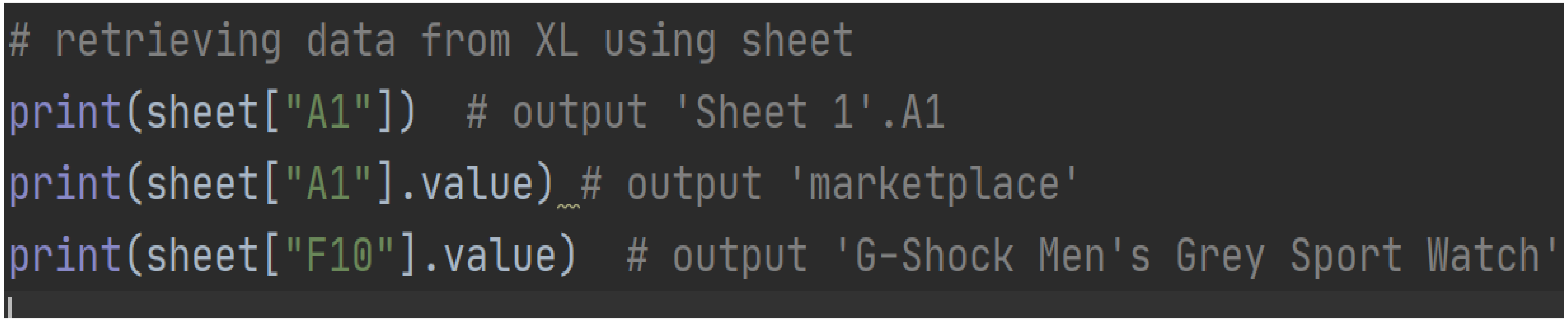
```
from openpyxl import load_workbook
workbook = load_workbook(filename="sample.xlsx")
print(workbook.sheetnames)
sheet = workbook.active
print(sheet)
print(sheet.title)
```



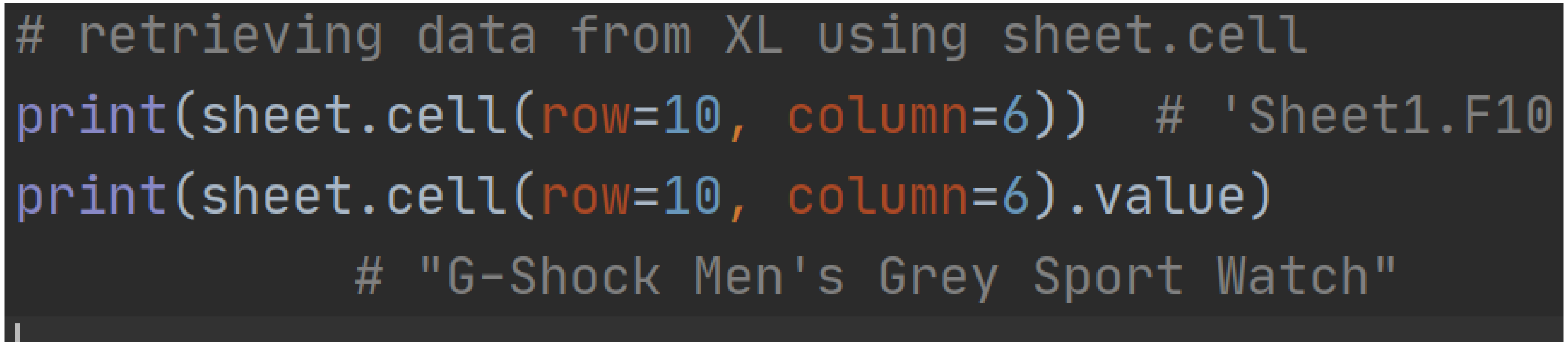
```
['amazon_reviews_us_Watches_v1_00']
<Worksheet "amazon_reviews_us_Watches_v1_00">
amazon_reviews_us_Watches_v1_00
```

reading_XL_1.py

Reading Excel Spreadsheets with openpyxl (cont’d)



```python
# retrieving data from XL using sheet
print(sheet["A1"])  # output 'Sheet 1'.A1
print(sheet["A1"].value) # output 'marketplace'
print(sheet["F10"].value)  # output 'G-Shock Men's Grey Sport Watch'
```



```python
# retrieving data from XL using sheet.cell
print(sheet.cell(row=10, column=6))  # 'Sheet1.F10
print(sheet.cell(row=10, column=6).value)
             # "G-Shock Men's Grey Sport Watch"
```

reading_XL_1.py

Additional Reading Operations

There are a few arguments you can pass to **load_workbook()** that change the way a spreadsheet is loaded.

The most important ones are the following two Booleans:

1. **read_only** loads a spreadsheet in read-only mode allowing you to open very large Excel files.

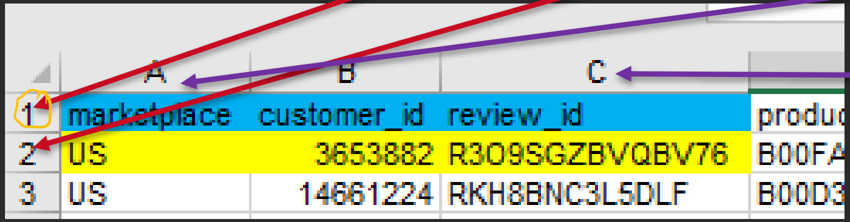
2. **data_only** ignores loading formulas and instead loads only the resulting values.

Importing Data from a Spreadsheet

- Iterating through the Data
- Manipulating Data using Python’s Default Data Structures
- Convert Data into Python Classes

Iterating Through the Data

```
for value in sheet.iter_rows(min_row=1,
                             max_row=2,
                             min_col=1,
                             max_col=3,
                             values_only=True):
    print(value)
```



	A	B	C	
1	marketplace	customer_id	review_id	produc
2	US	3653882	R3O9SGZBVQBV76	B00FA
3	US	14661224	RKH8BNC3L5DLF	B00D3

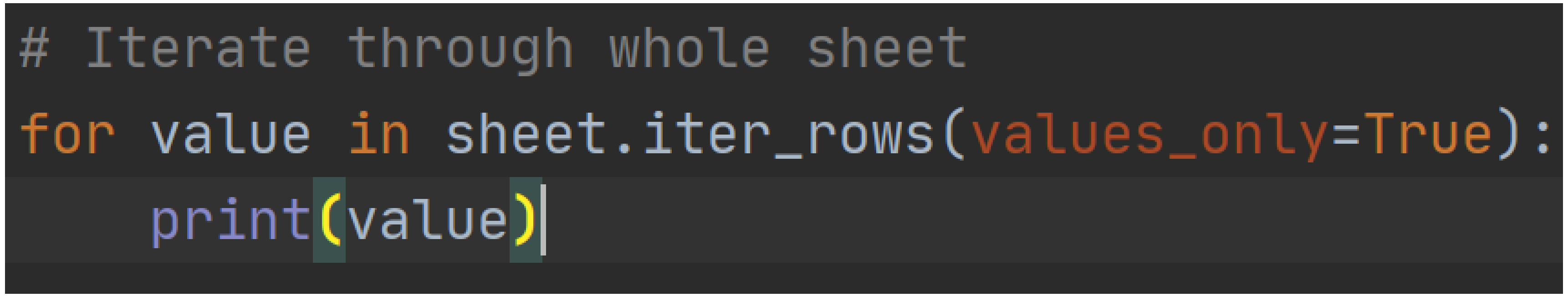
```
('marketplace', 'customer_id', 'review_id')
('US', 3653882, 'R3O9SGZBVQBV76')
```

```
print("+++++++ Iterate Over columns ++++++")
for value in sheet.iter_cols(min_row=1,
                             max_row=2,
                             min_col=1,
                             max_col=3,
                             values_only=True):
    print(value)
```

```
('marketplace', 'US')
('customer_id', 3653882)
('review_id', 'R3O9SGZBVQBV76')
```

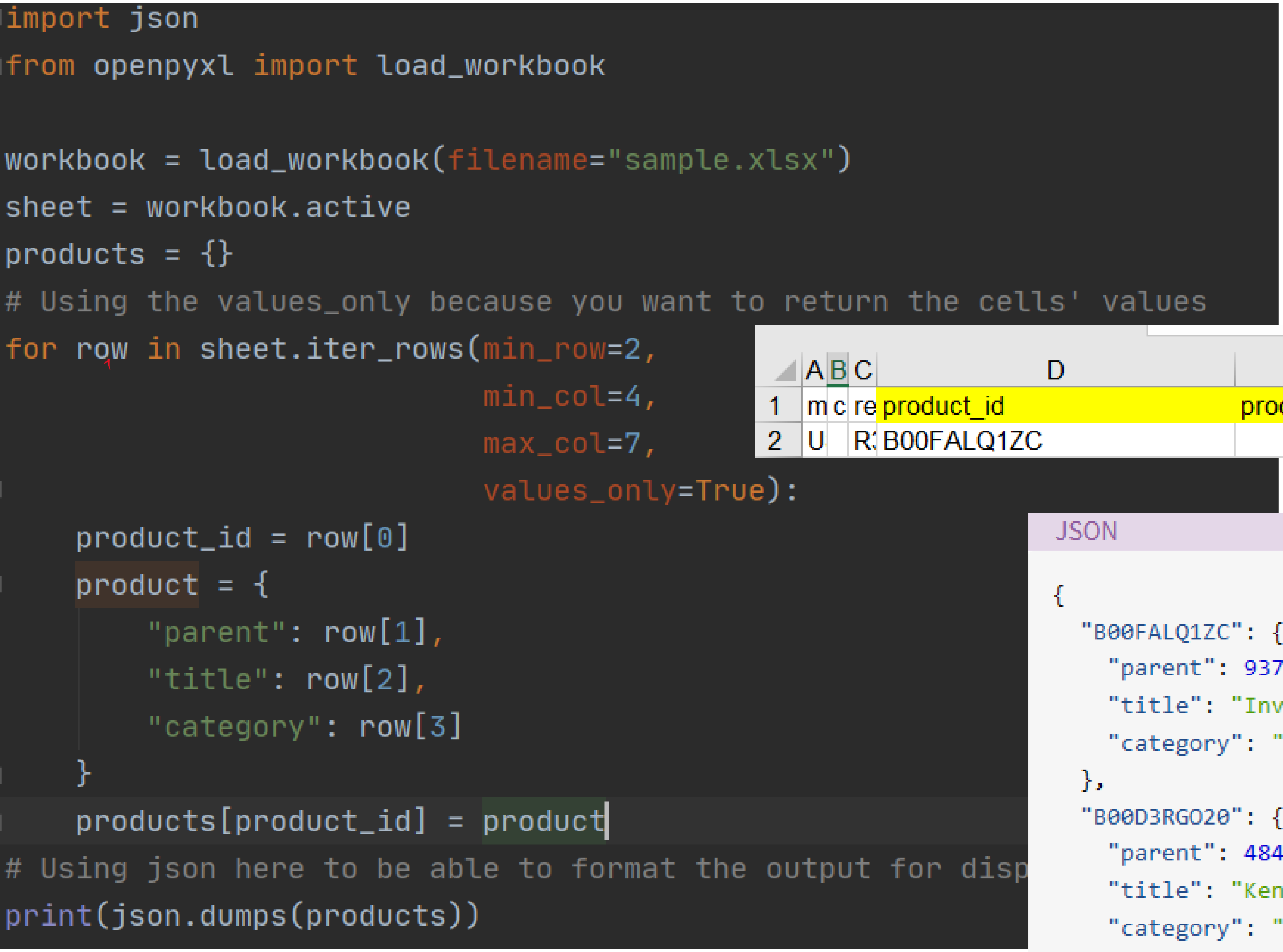
iteration_demo1.py

Iterating Through the Data (cont’d)

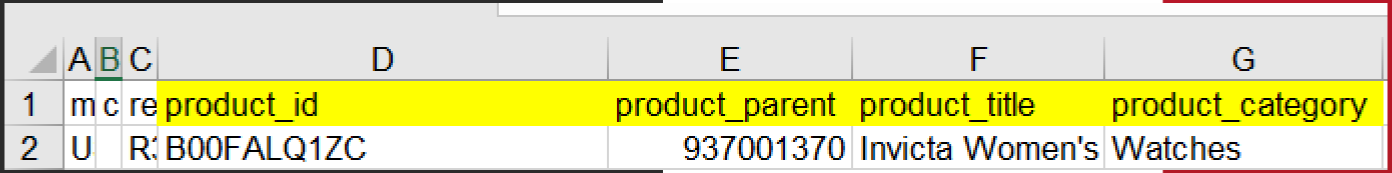


```python
# Iterate through whole sheet
for value in sheet.iter_rows(values_only=True):
    print(value)
```

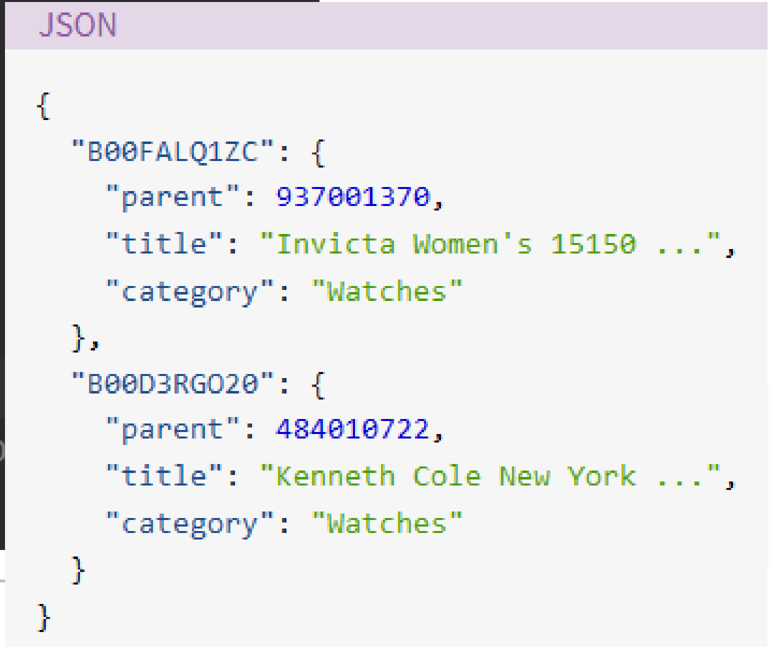
Manipulating Data Using Python Data Structures



```python
import json
from openpyxl import load_workbook

workbook = load_workbook(filename="sample.xlsx")
sheet = workbook.active
products = {}
# Using the values_only because you want to return the cells' values
for row in sheet.iter_rows(min_row=2,
                           min_col=4,
                           max_col=7,
                           values_only=True):
    product_id = row[0]
    product = {
        "parent": row[1],
        "title": row[2],
        "category": row[3]
    }
    products[product_id] = product
# Using json here to be able to format the output for disp
print(json.dumps(products))
```



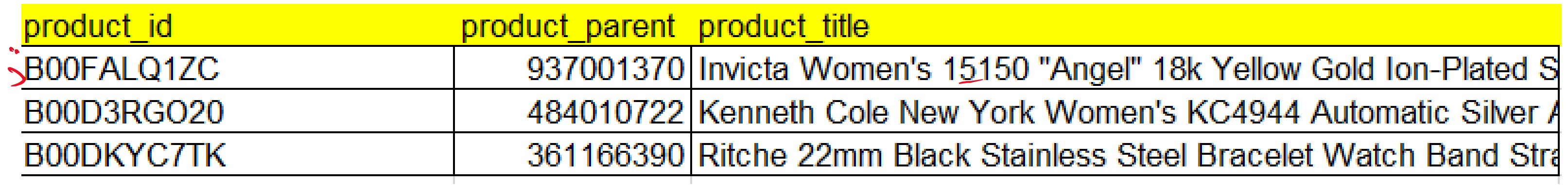
	A	B	C	D	E	F	G
1	m	c	re	product_id	product_parent	product_title	product_category
2	U		R:	B00FALQ1ZC	937001370	Invicta Women's	Watches



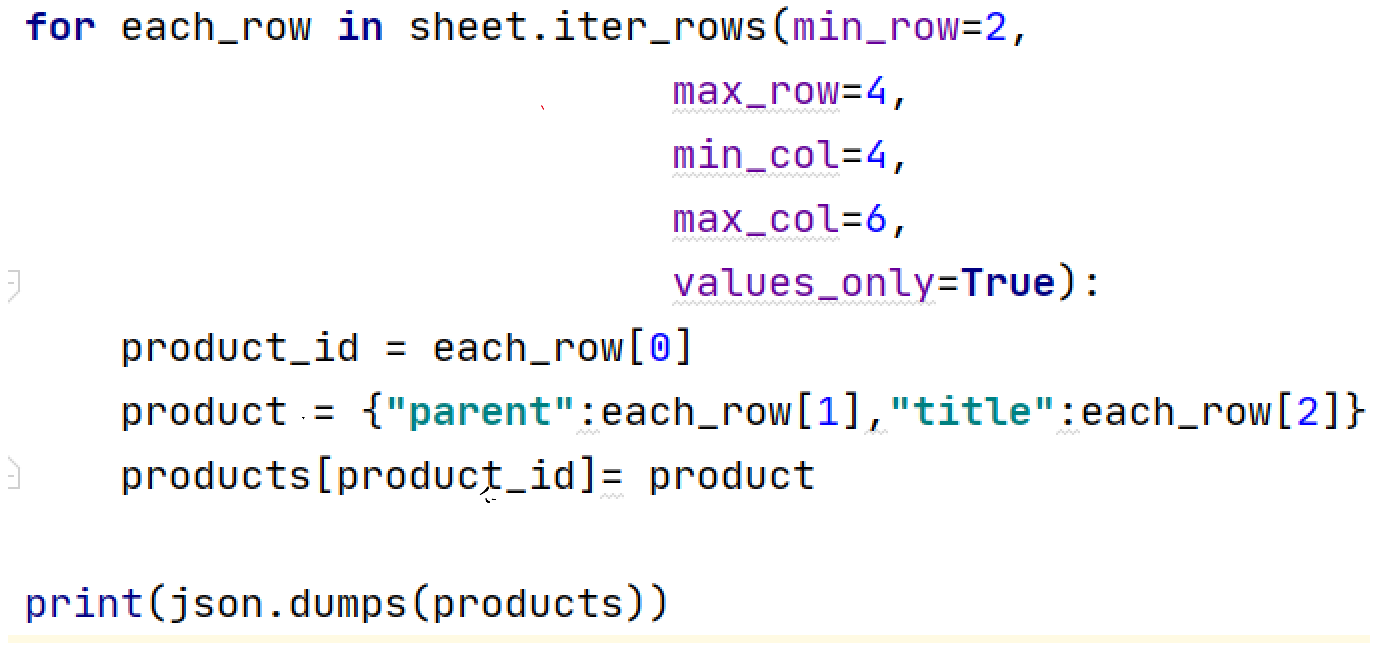
JSON

```json
{
  "B00FALQ1ZC": {
    "parent": 937001370,
    "title": "Invicta Women's 15150 ...",
    "category": "Watches"
  },
  "B00D3RGO20": {
    "parent": 484010722,
    "title": "Kenneth Cole New York ...",
    "category": "Watches"
  }
}
```

parse_products_to_dict.py

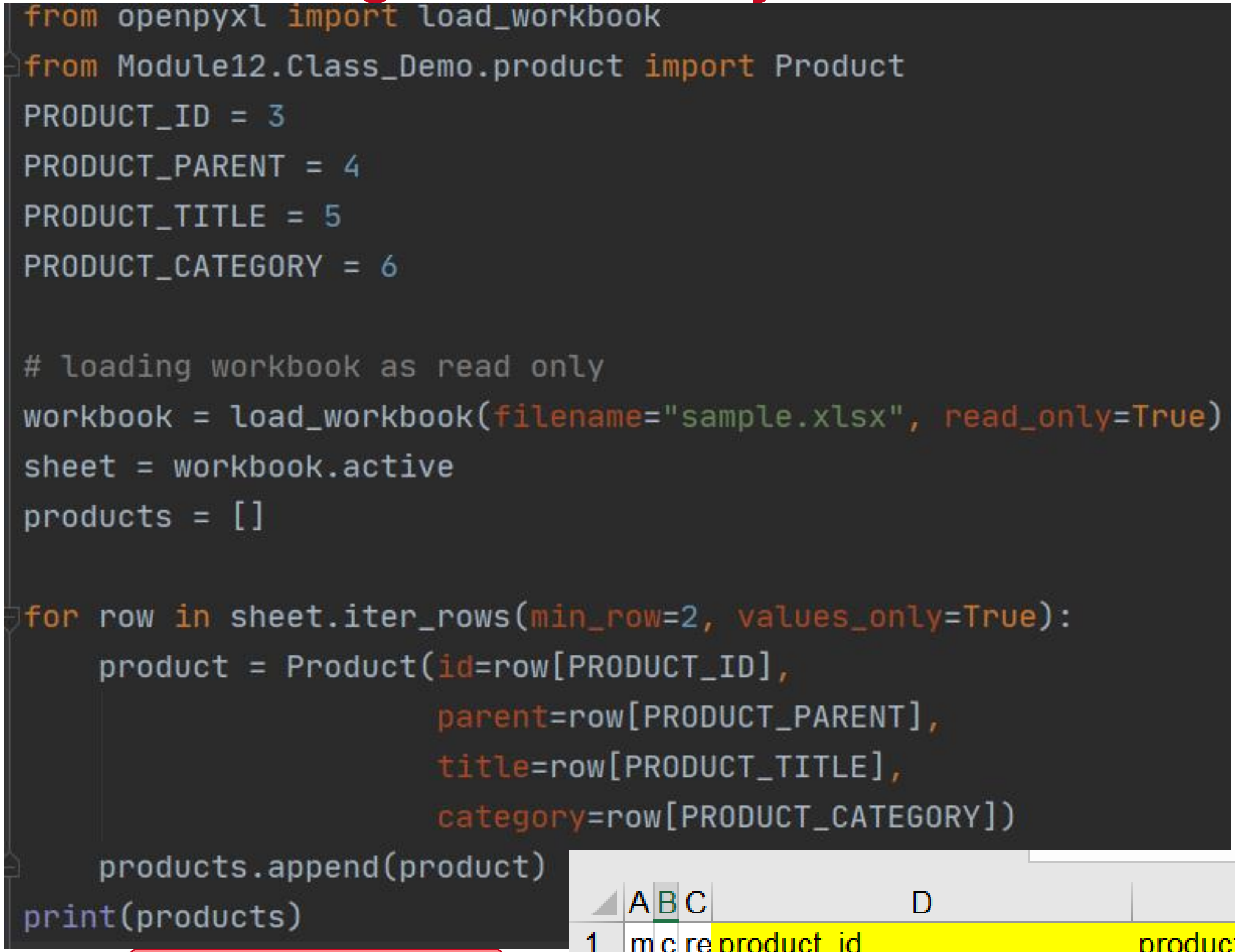


product_id	product_parent	product_title
B00FALQ1ZC	937001370	Invicta Women's 15150 "Angel" 18k Yellow Gold Ion-Plated S
B00D3RGO20	484010722	Kenneth Cole New York Women's KC4944 Automatic Silver A
B00DKYC7TK	361166390	Ritche 22mm Black Stainless Steel Bracelet Watch Band Stra



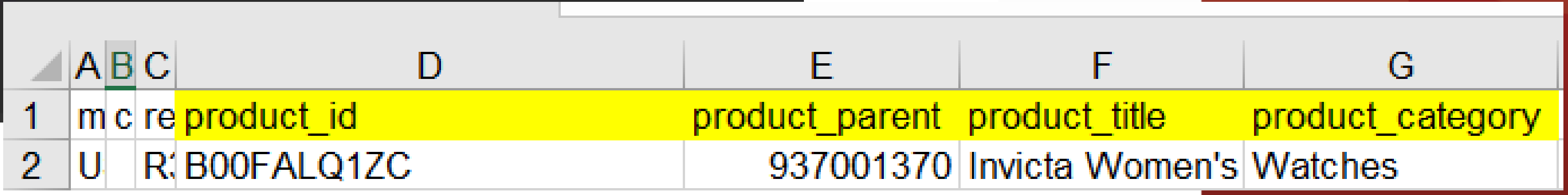
```python
for each_row in sheet.iter_rows(min_row=2,
                                max_row=4,
                                min_col=4,
                                max_col=6,
                                values_only=True):
    product_id = each_row[0]
    product = {"parent":each_row[1],"title":each_row[2]}
    products[product_id]= product

print(json.dumps(products))
```

Converting Data Into Python Classes



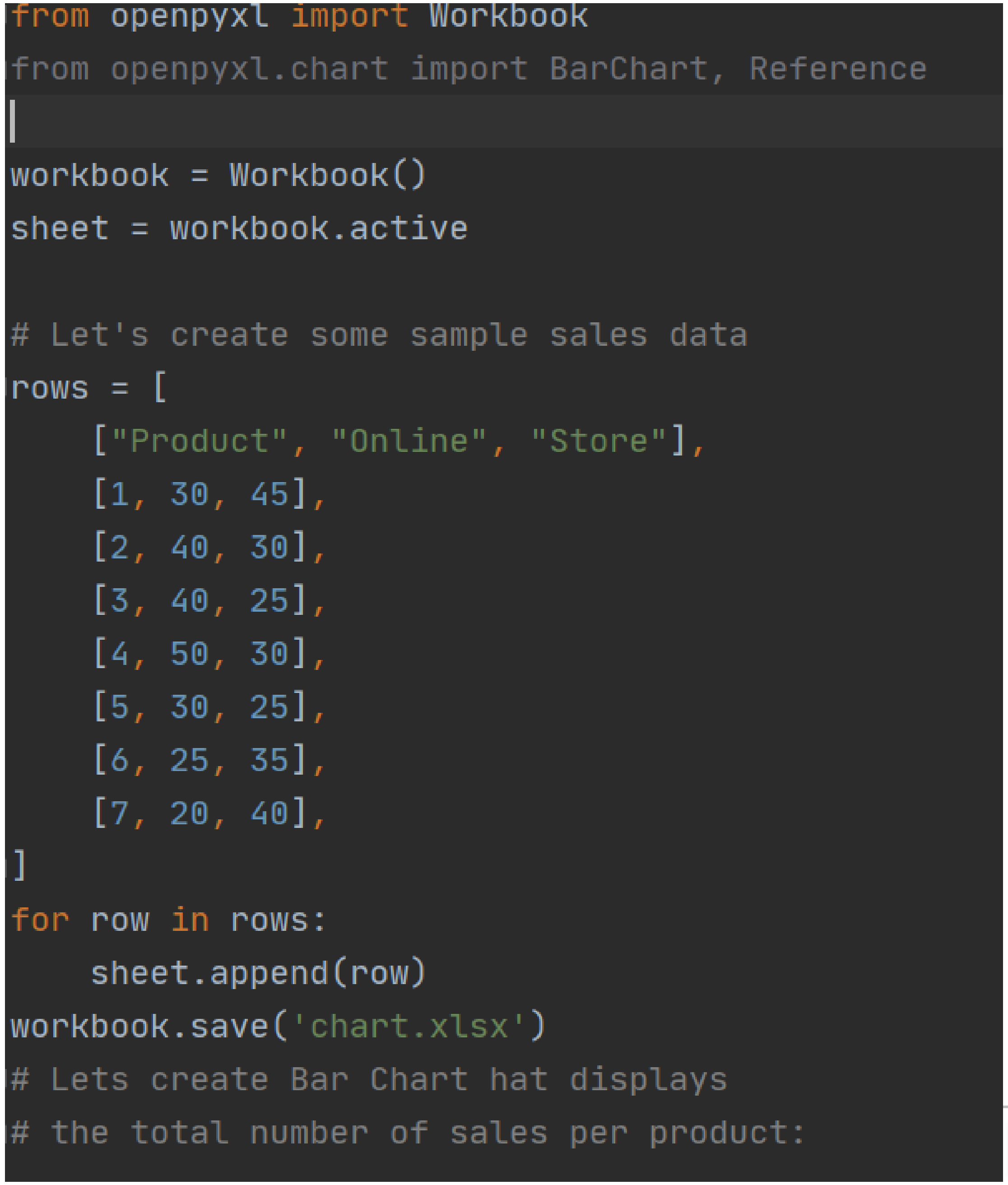
```
from openpyxl import load_workbook
from Module12.Class_Demo.product import Product
PRODUCT_ID = 3
PRODUCT_PARENT = 4
PRODUCT_TITLE = 5
PRODUCT_CATEGORY = 6

# loading workbook as read only
workbook = load_workbook(filename="sample.xlsx", read_only=True)
sheet = workbook.active
products = []

for row in sheet.iter_rows(min_row=2, values_only=True):
    product = Product(id=row[PRODUCT_ID],
                      parent=row[PRODUCT_PARENT],
                      title=row[PRODUCT_TITLE],
                      category=row[PRODUCT_CATEGORY])
    products.append(product)
print(products)
```

parse_products.py

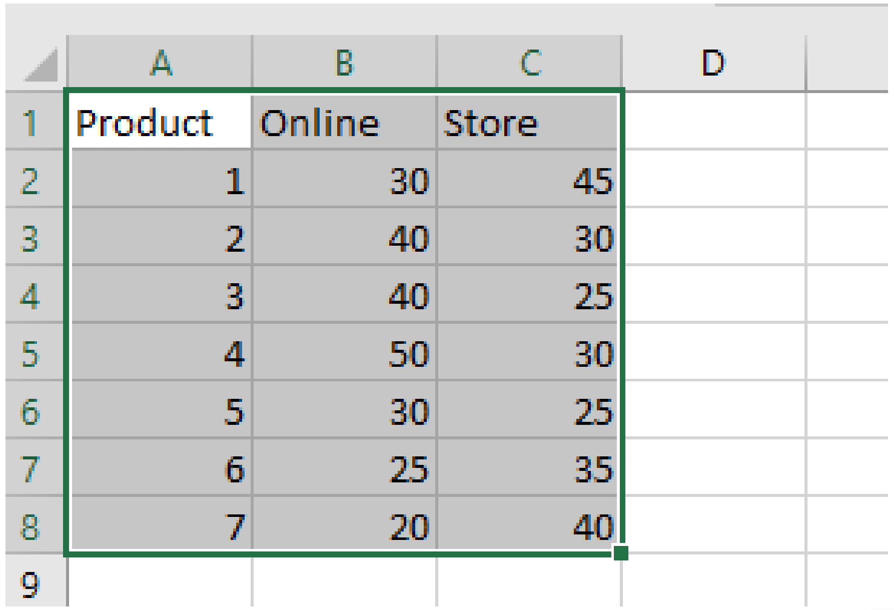


	A	B	C	D	E	F	G
1	m	c	re	product_id	product_parent	product_title	product_category
2	U		R:	B00FALQ1ZC	937001370	Invicta Women's	Watches

Creating a BarChart from Excel sheet



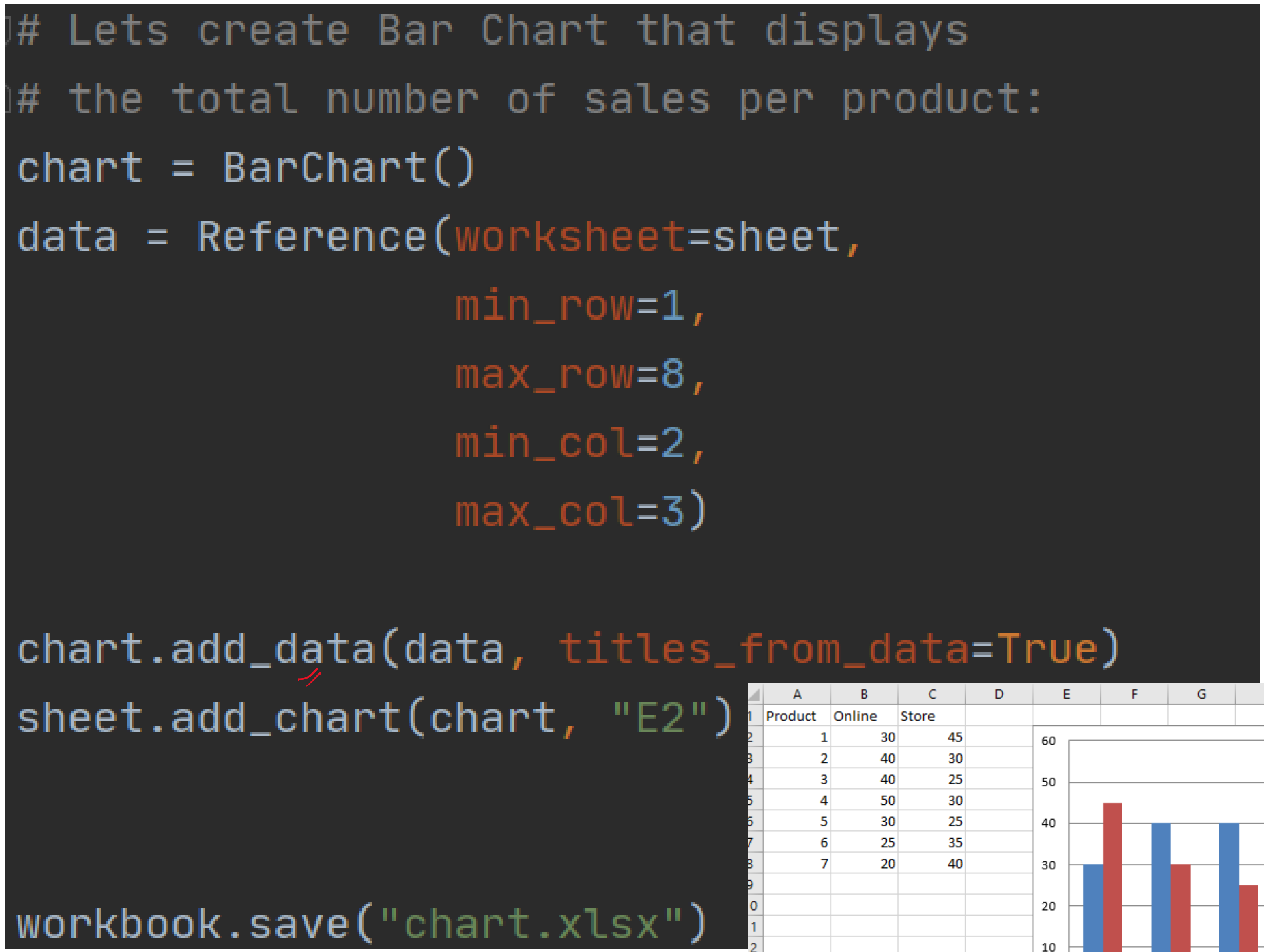
```
from openpyxl import Workbook
from openpyxl.chart import BarChart, Reference

workbook = Workbook()
sheet = workbook.active

# Let's create some sample sales data
rows = [
    ["Product", "Online", "Store"],
    [1, 30, 45],
    [2, 40, 30],
    [3, 40, 25],
    [4, 50, 30],
    [5, 30, 25],
    [6, 25, 35],
    [7, 20, 40],
]
for row in rows:
    sheet.append(row)
workbook.save('chart.xlsx')
# Lets create Bar Chart hat displays
# the total number of sales per product:
```



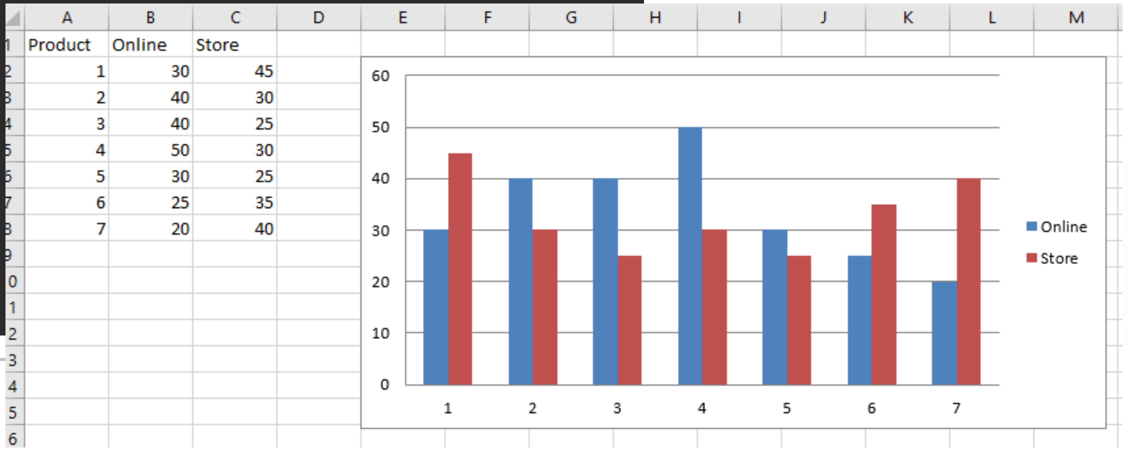
	A	B	C	D
1	Product	Online	Store	
2	1	30	45	
3	2	40	30	
4	3	40	25	
5	4	50	30	
6	5	30	25	
7	6	25	35	
8	7	20	40	
9				

barchart_demo.py

Creating a BarChart from Excel sheet (cont'd)



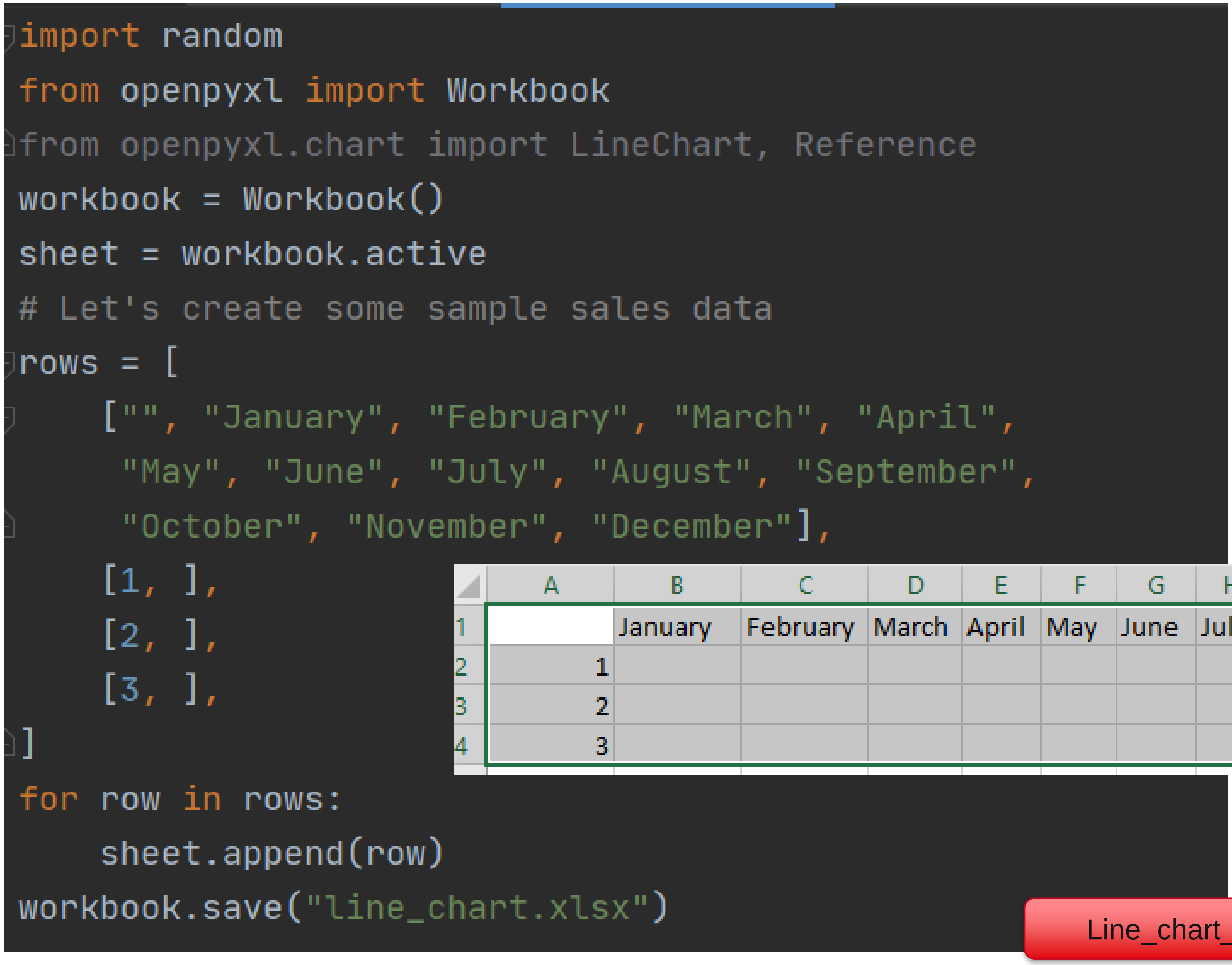
```
# Lets create Bar Chart that displays
# the total number of sales per product:
chart = BarChart()
data = Reference(worksheet=sheet,
                 min_row=1,
                 max_row=8,
                 min_col=2,
                 max_col=3)

chart.add_data(data, titles_from_data=True)
sheet.add_chart(chart, "E2")


workbook.save("chart.xlsx")
```



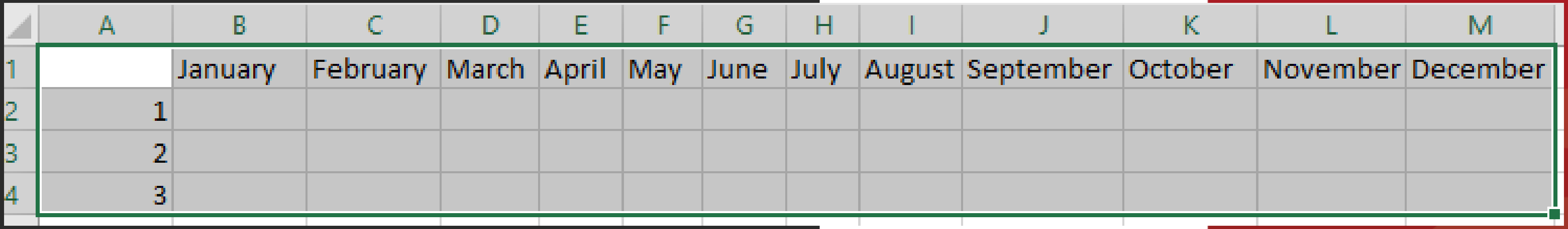
Product	Online	Store
1	30	45
2	40	30
3	40	25
4	50	30
5	30	25
6	25	35
7	20	40

barchart_demo.py

Creating a Line Chart



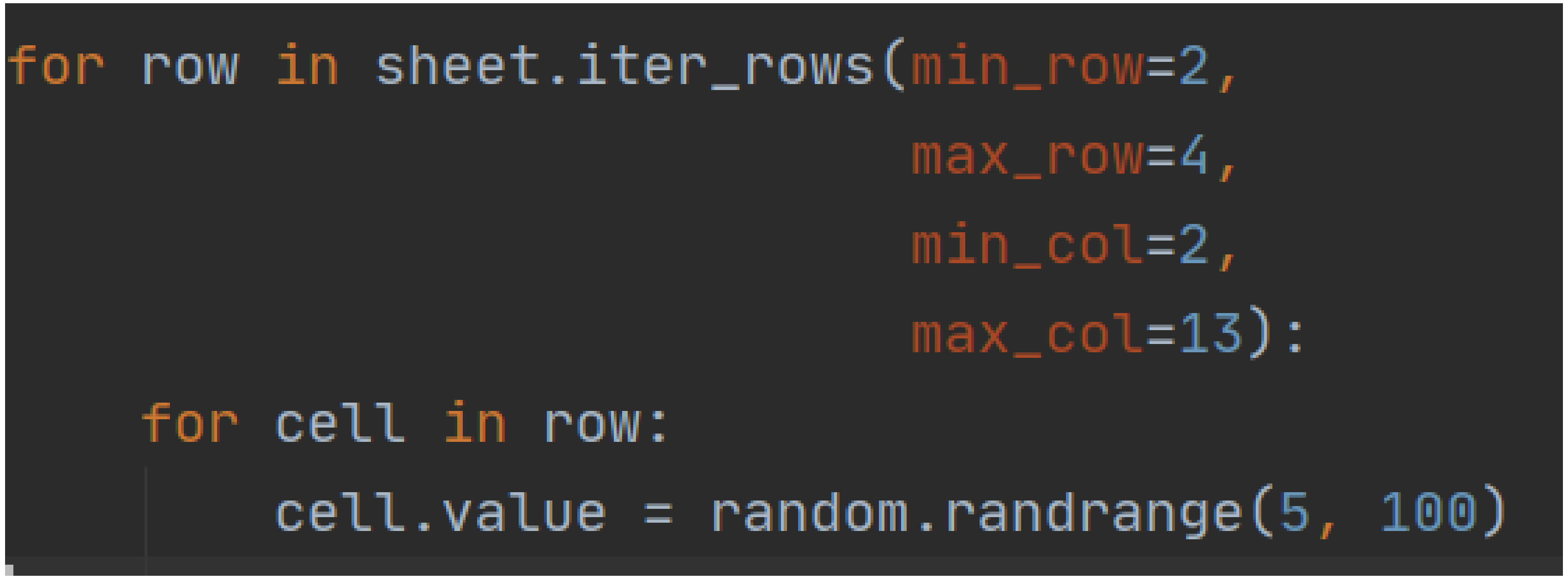
```
import random
from openpyxl import Workbook
from openpyxl.chart import LineChart, Reference
workbook = Workbook()
sheet = workbook.active
# Let's create some sample sales data
rows = [
    ["", "January", "February", "March", "April",
     "May", "June", "July", "August", "September",
     "October", "November", "December"],
    [1, ],
    [2, ],
    [3, ],
]
for row in rows:
    sheet.append(row)
workbook.save("line_chart.xlsx")
```



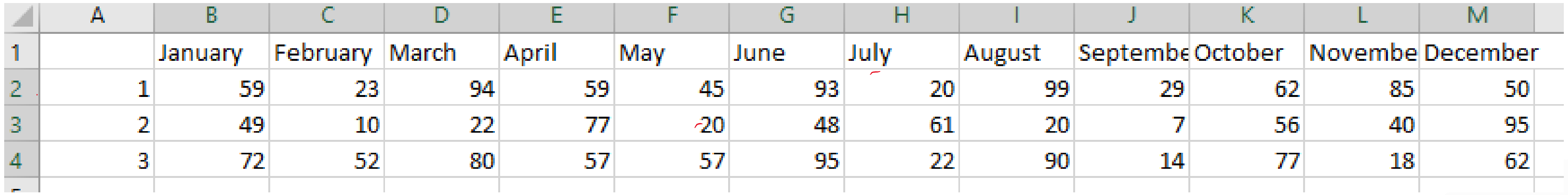
	A	B	C	D	E	F	G	H	I	J	K	L	M
1		January	February	March	April	May	June	July	August	September	October	November	December
2	1												
3	2												
4	3												

Line_chart_demo.py

Creating a Line Chart (Cont'd)



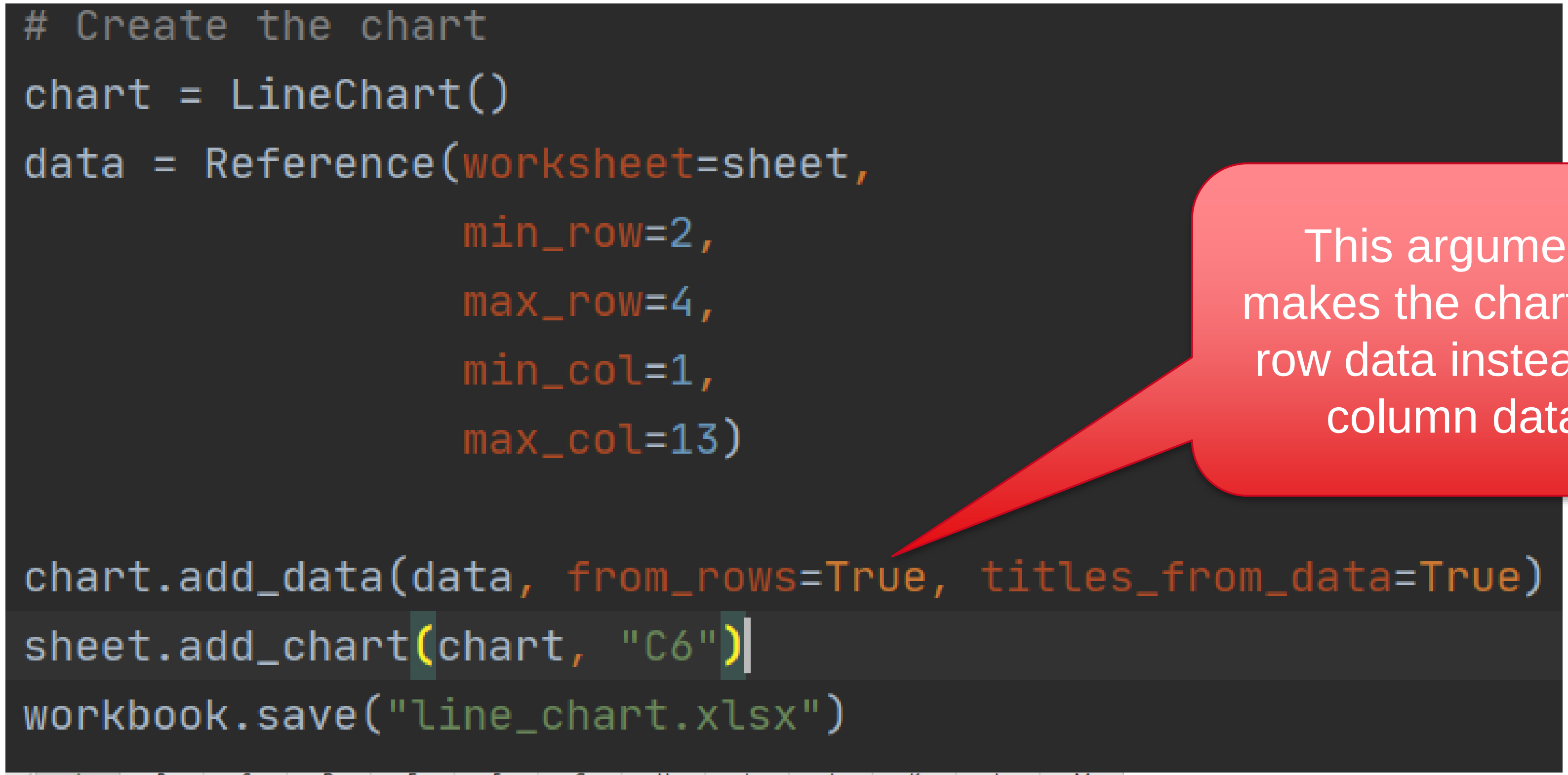
```
for row in sheet.iter_rows(min_row=2,
                           max_row=4,
                           min_col=2,
                           max_col=13):
    for cell in row:
        cell.value = random.randrange(5, 100)
```



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		January	February	March	April	May	June	July	August	Septembe	October	Novembe	December
2	1	59	23	94	59	45	93	20	99	29	62	85	50
3	2	49	10	22	77	20	48	61	20	7	56	40	95
4	3	72	52	80	57	57	95	22	90	14	77	18	62

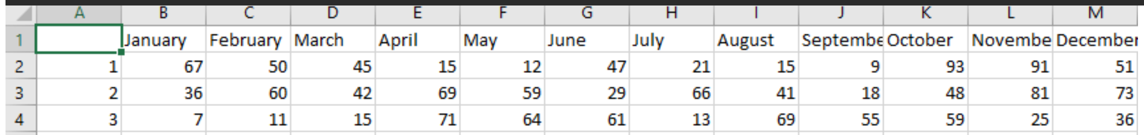
line_chart_demo.py

Creating a Line Chart (Cont'd)

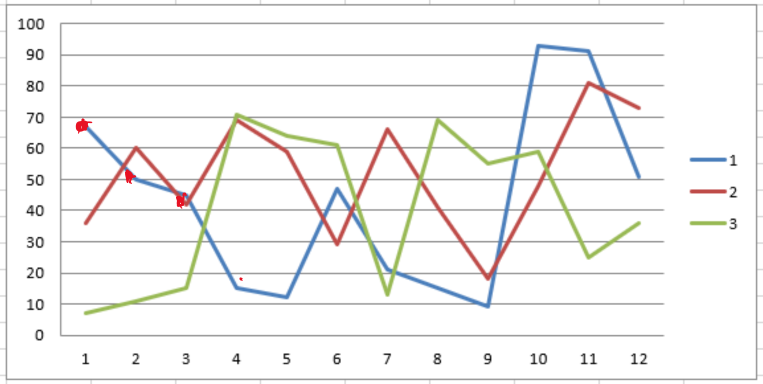


```
# Create the chart
chart = LineChart()
data = Reference(worksheet=sheet,
                 min_row=2,
                 max_row=4,
                 min_col=1,
                 max_col=13)

chart.add_data(data, from_rows=True, titles_from_data=True)
sheet.add_chart(chart, "C6")
workbook.save("line_chart.xlsx")
```

This argument makes the chart plot row data instead of column data



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		January	February	March	April	May	June	July	August	Septembe	October	Novembe	December
2	1	67	50	45	15	12	47	21	15	9	93	91	51
3	2	36	60	42	15	69	59	29	66	41	18	48	81
4	3	7	11	15	71	64	61	13	69	55	59	25	36

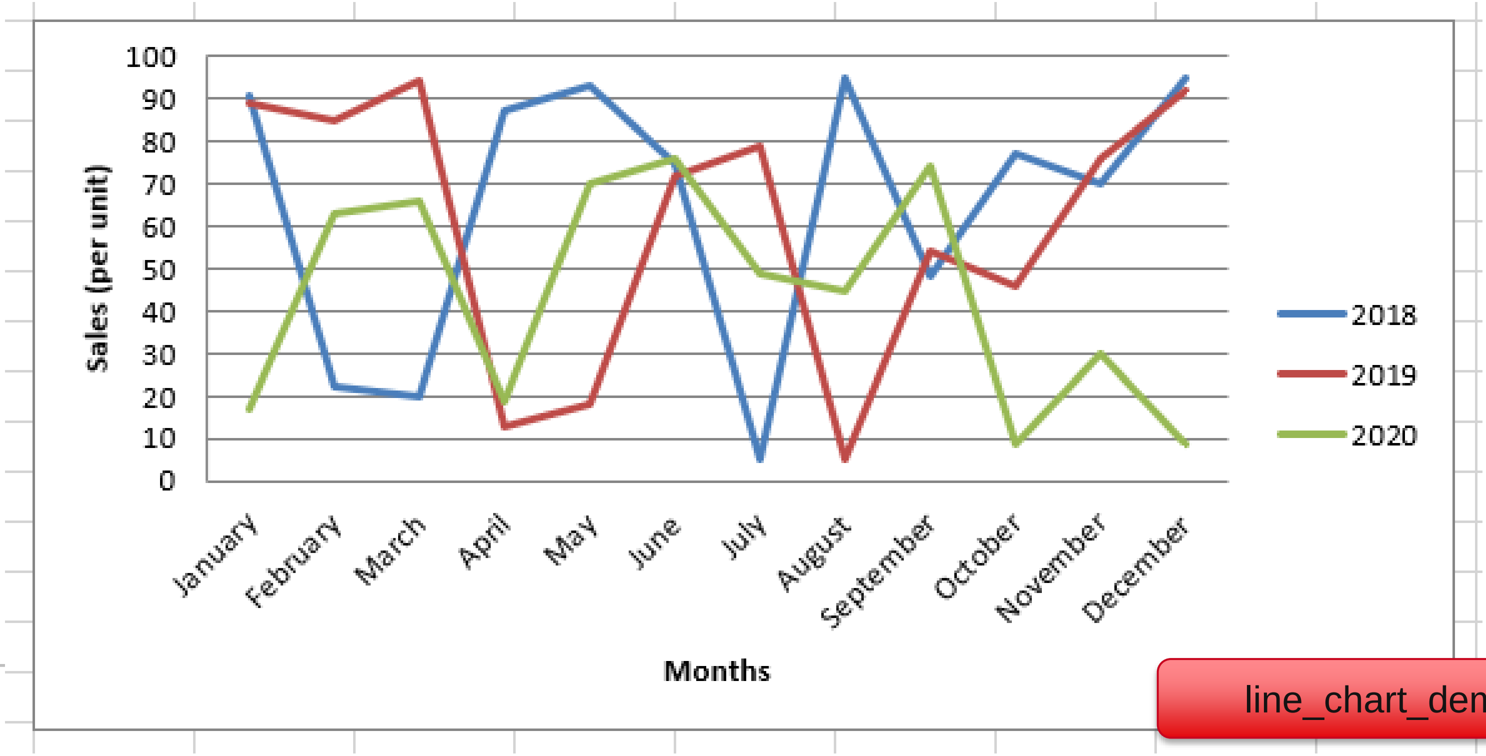


line_chart_demo.py

Chart Formatting

```
cats = Reference(worksheet=sheet,
                 min_row=1,
                 max_row=1,
                 min_col=2,
                 max_col=13)
chart.set_categories(cats)
```

```
# Add axes to chart to improve readability
chart.x_axis.title = "Months"
chart.y_axis.title = "Sales (per unit)"
```



line_chart_demo.py