Excel, CSV, JSON, XML with Python

Hexavarsity

Objective

- Excel
- CSV
- JSON
- XML

Handling Excel

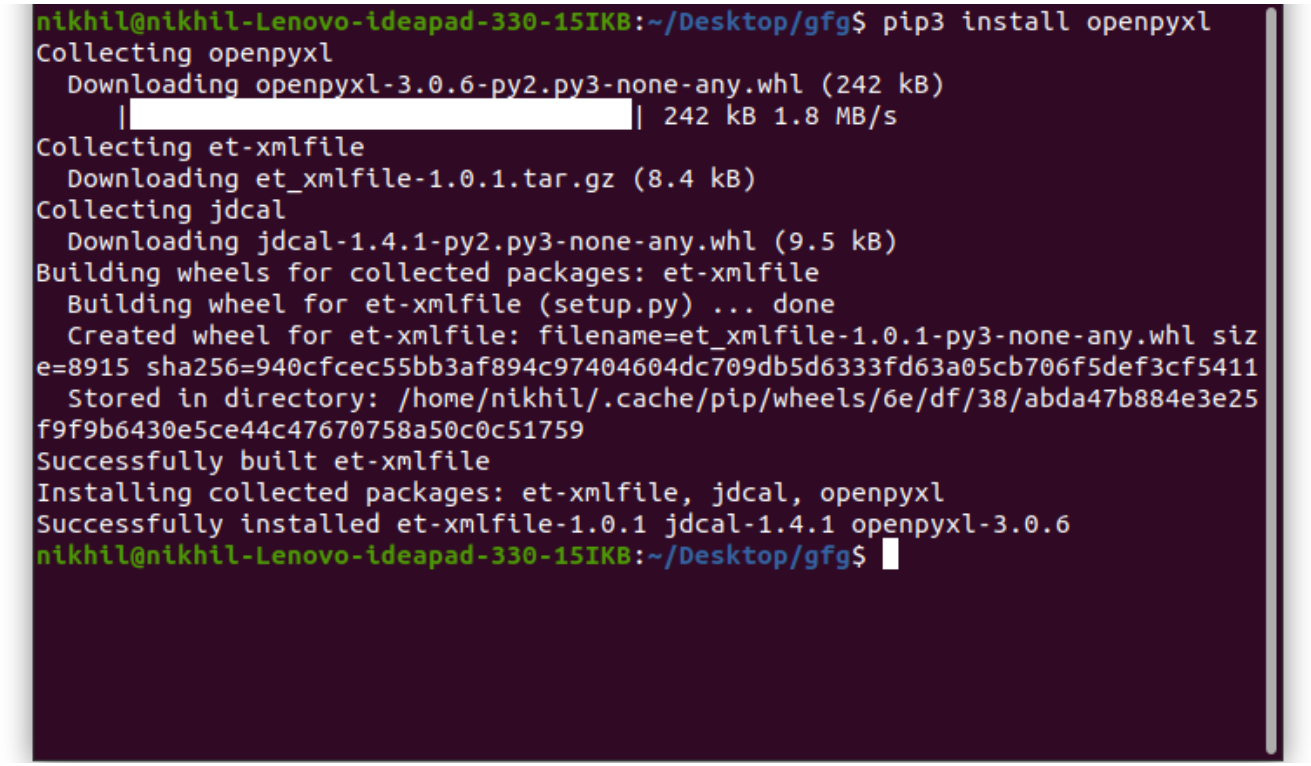
Term	Explanation
Spreadsheet or Workbook	A **Spreadsheet** is the main file you are creating or working with.
Worksheet or Sheet	A **Sheet** is used to split different kinds of content within the same spreadsheet. A **Spreadsheet** can have one or more **Sheets**.
Column	A **Column** is a vertical line, 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.

Getting Started

- Openpyxl is a Python library that provides various methods to interact with Excel Files using Python. It allows operations like reading, writing, arithmetic operations, plotting graphs, etc.

- This module does not come in-built with Python. To install this type the below command in the terminal.

pip install openpyxl



Reading from Spreadsheets

- To read an Excel file open the spreadsheet using the **load_workbook()** method.
- Use the **active** to select the first sheet available and the **cell** attribute to select the cell by passing the row and column parameter.
- The **value** attribute prints the value of the cell.

	A	B	C	D
1	Name	Course	Branch	Semester
2	Ankit	B.Tech	CSE	4
3	Rahul	M.Tech	CSE	2
4	Priya	MBA	HR	3
5	Nikhil	B.Tech	CSE	4
6	Nisha	B.Tech	Biotech	5
7				

Example:

```
import openpyxl

# Give the location of the file
path = "gfg.xlsx"

# To open the workbook workbook object is created
wb_obj = openpyxl.load_workbook(path)

# Get workbook active sheet object from the active attribute
sheet_obj = wb_obj.active

# Cell object is created by using sheet object's cell() method.
cell_obj = sheet_obj.cell(row = 1, column = 1)

print(cell_obj.value)
```

Reading from Multiple Cells

- We can get the count of the total rows and columns using the max_row and max_column
  - row = sheet_obj.max_row
  - column = sheet_obj.max_column
- We can also read from multiple cells using the cell name. This can be seen as the list slicing of Python.
- cell_obj = sheet_obj['A1': 'B6']

Example:

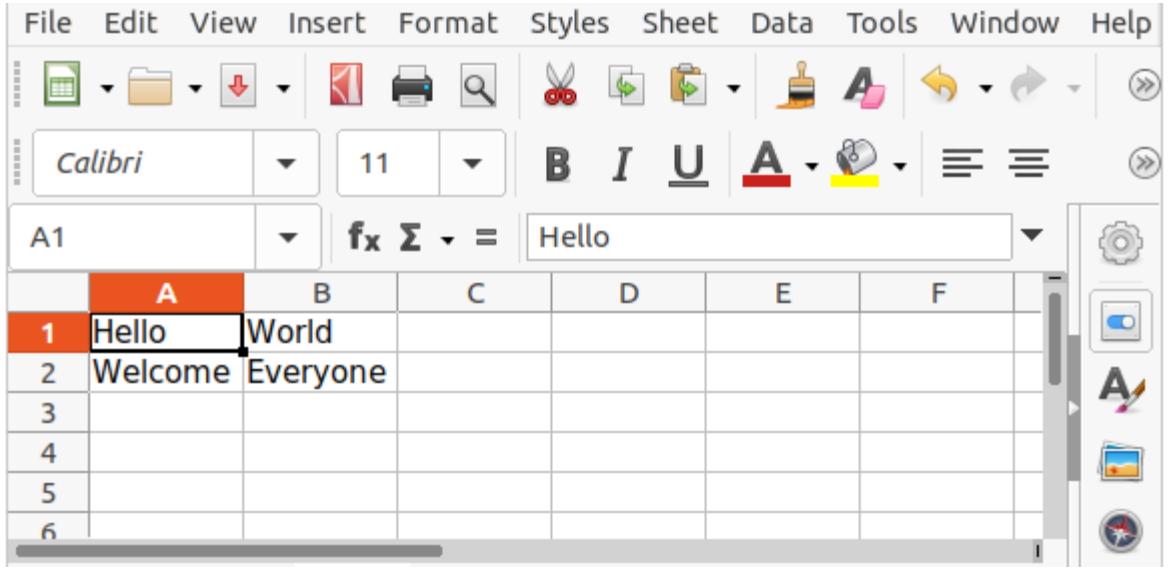
```
import openpyxl

path = "gfg.xlsx"
wb_obj = openpyxl.load_workbook(path)

sheet_obj = wb_obj.active
row = sheet_obj.max_row
column = sheet_obj.max_column
for i in range(1, row + 1):
   cell_obj = sheet_obj.cell(row = i, column = 1)
   print(cell_obj.value)
for i in range(1, column + 1):
   cell_obj = sheet_obj.cell(row = 2, column = i)
   print(cell_obj.value, end = " ")
```

Writing to Spreadsheets

- create a new spreadsheet, and then we will write some data to the newly created file. An empty spreadsheet can be created using the **Workbook()** method.

```
# import openpyxl module
import openpyxl
wb = openpyxl.Workbook()

sheet = wb.active
c1 = sheet.cell(row = 1, column = 1)

c1.value = "Hello"

c2 = sheet.cell(row= 1 , column = 2)
c2.value = "World"

c3 = sheet['A2']
c3.value = "Welcome"

c4 = sheet['B2']
c4.value = "Everyone"

wb.save("sample.xlsx")
```

Example:

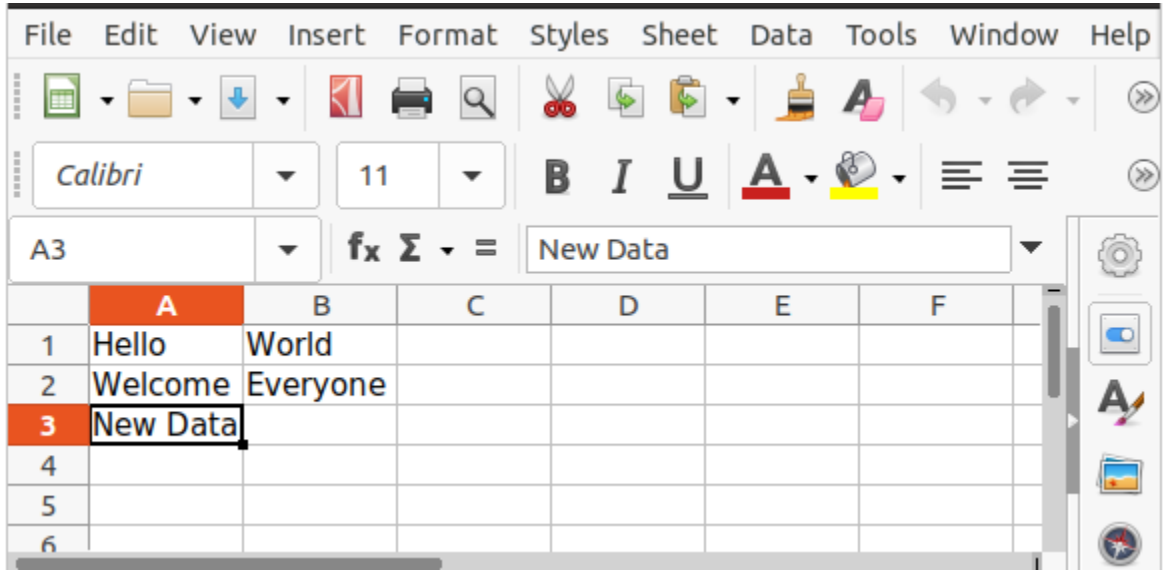


Appending to the Spreadsheet

- every time you try to write to a spreadsheet the existing data gets overwritten, and the file is saved as a new file. This happens because the **Workbook()** method always creates a new workbook file object. To write to an existing workbook you must open the file with the **load_workbook()** method.

```
# import openpyxl module
import openpyxl

wb = openpyxl.load_workbook("sample.xlsx")

sheet = wb.active

c = sheet['A3']
c.value = "New Data"

wb.save("sample.xlsx")
```

Example:



Demo

Reading a CSV file

- **CSV** (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database.
- A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record.
- Each record consists of one or more fields, separated by commas.

```
import csv
filename = "aapl.csv"
rows, fields = [], []

with open(filename, 'r') as csvfile:
  csvreader = csv.reader(csvfile)
  fields = next(csvreader)
  for row in csvreader:
    rows.append(row)
  print("Total no. of rows: %d"%(csvreader.line_num))

print('Field names are:' + ', '.join(field for field in fields))
print('\nFirst 5 rows are:\n')
for row in rows[:5]:
  for col in row:
    print("%10s"%col,end=" "),
  print('\n')
```

Example:

Writing a CSV file

- **CSV** (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database.
- A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record.
- Each record consists of one or more fields, separated by commas.

```
import csv

fields = ['Name', 'Branch', 'Year', 'CGPA']
rows = [ ['Nikhil', 'COE', '2', '9.0'],
        ['Sanchit', 'COE', '2', '9.1'],
        ['Aditya', 'IT', '2', '9.3'],
        ['Sagar', 'SE', '1', '9.5'],
        ['Prateek', 'MCE', '3', '7.8'],
        ['Sahil', 'EP', '2', '9.1']]
filename = "university_records.csv"

with open(filename, 'w') as csvfile:
    csvwriter = csv.writer(csvfile)

    csvwriter.writerow(fields)
    csvwriter.writerows(rows)
```

Example:

	A	B	C	D	E
1	Name	Branch	Year	CGPA	
2	Nikhil	COE	2	9	
3	Sanchit	COE	2	9.1	
4	Aditya	IT	2	9.3	
5	Sagar	SE	1	9.5	
6	Prateek	MCE	3	7.8	
7	Sahil	EP	2	9.1	
8					

Demo

Read JSON file using Python

- The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data.

- Python supports JSON through a built-in package called json. To use this feature, we import the json package in Python script.

- The text in JSON is done through quoted-string which contains the value in key-value mapping within { }.

Deserialization of JSON

- The Deserialization of JSON means the conversion of JSON objects into their respective Python objects.

- The load()/loads() method is used for it. If you have used JSON data from another program or obtained as a string format of JSON, then it can easily be deserialized with load()/loads(), which is usually used to load from string, otherwise, the root object is in list or dict.

- json.load(): json.load() accepts file object, parses the JSON data, populates a Python dictionary with the data and returns it back to you.

```
{
    "emp_details": [
        {
            "emp_name": "Shubham",
            "email": "ksingh.shubh@gmail.com",
            "job_profile": "intern"
        },
        {
            "emp_name": "Gaurav",
            "email": "gaurav.singh@gmail.com",
            "job_profile": "developer"
        },
        {
            "emp_name": "Nikhil",
            "email": "nikhil@geeksforgeeks.org'
            "job_profile": "Full Time"
        }
    ]
}
```

Read from JSON

- The Deserialization of JSON means the conversion of JSON objects into their respective Python objects.

```
import json

a = '{"name": "Bob", "languages": "English"}'

y = json.loads(a)

print("JSON string = ", y)
print()

f = open ('data.json', "r")

data = json.loads(f.read())

for i in data['emp_details']:
    print(i)
```

Example:

```
{'emp_name': 'Shubham', 'email': 'ksingh.shubh@gmail.com', 'job_profile': 'intern'}
{'emp_name': 'Gaurav', 'email': 'gaurav.singh@gmail.com', 'job_profile': 'developer'}
{'emp_name': 'Nikhil', 'email': 'nikhil@geeksforgeeks.org', 'job_profile': 'Full Time'}
```


Write from JSON

```
import json

a = '{"name": "Bob", "languages": "English"}'

y = json.loads(a)

print("JSON string = ", y)
print()

f = open ('data.json', "r")

data = json.loads(f.read())

for i in data['emp_details']:
   print(i)

f.close()
```

Example:

```
1  {"name": "sathiyajith", "rollno": 56, "cgpa": 8.6, "phonenumber": "9976770500"}
```


Demo

Reading and Writing XML Files in Python

- Extensible Markup Language, commonly known as XML is a language designed specifically to be easy to interpret by both humans and computers altogether.
- The language defines a set of rules used to encode a document in a specific format.

```
<?xml version="1.0" encoding="utf-8"?>
<saranghe>
 <child name="Frank" test="0">
  FRANK likes EVERYONE
 </child>
 <unique>
  Add a video URL in here
 </unique>
 <child name="Texas" test="1">
  TEXAS is a PLACE
 </child>
 <child name="Frank" test="2">
  Exclusively
 </child>
 <unique>
  Add a workbook URL here
 </unique>
 <data>
  Add the content of your article here
  <family>
   Add the font family of your text here
  </family>
  <size>
   Add the font size of your text here
  </size>
 </data>
</saranghe>
```

BeautifulSoup xml parser read XML

- reading and writing the xml file we would be using a Python library named BeautifulSoup. In order to install the library
  - pip install beautifulsoup4
  - pip install lxml

```
from bs4 import BeautifulSoup

with open('dict.xml', 'r') as f:
   data = f.read()

Bs_data = BeautifulSoup(data, "xml")

b_unique = Bs_data.find_all('unique')

print(b_unique)

b_name = Bs_data.find('child', {'name':'Frank'})
print(b_name)
value = b_name.get('test')
print(value)
```

Example:

```
[<unique>
  Add a video URL in here
 </unique>, <unique>
  Add a workbook URL here
 </unique>]
<child name="Frank" test="0">
  FRANK likes EVERYONE
 </child>
0
```

BeautifulSoup xml parser Write XML

```
from bs4 import BeautifulSoup

with open('dict.xml', 'r') as f:
   data = f.read()

bs_data = BeautifulSoup(data, 'xml')

for tag in bs_data.find_all('child', {'name':'Frank'}):
   tag['test'] = "WHAT !!"

print(bs_data.prettify())
```

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<saranghe>
 <child name="Frank" test="WHAT !!">
  FRANK likes EVERYONE
 </child>
 <unique>
  Add a video URL in here
 </unique>
 <child name="Texas" test="1">
  TEXAS is a PLACE
 </child>
 <child name="Frank" test="WHAT !!">
  Exclusively
 </child>
 <unique>
  Add a workbook URL here
 </unique>
 <data>
  Add the content of your article here
  <family>
   Add the font family of your text here
  </family>
  <size>
   Add the font size of your text here
  </size>
 </data>
</saranghe>
```

Queries

References

- https://www.geeksforgeeks.org/read-json-file-using-python/
- https://realpython.com/openpyxl-excel-spreadsheets-python/

Thank you

Innovative Services

Passionate Employees

Delighted Customers