```
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from openpyxl) (1.1.0)
```

```
# required packages '!pip install openpyxl'
import openpyxl


# name of the excel workbook
xlsx_file_path = 'unicef_sowc.xlsx'

# load the excel spreadsheet (workbook)
workbook = openpyxl.load_workbook (xlsx_file_path)

# print to make sure it loaded - 'sanity' test or 'debug' test
print (workbook)
```

```
<openpyxl.workbook.workbook.Workbook object at 0x7c4aa74c3b50>
```

```
# variable to hold the names of the sheets
sheet_names = workbook.sheetnames

# iterate through the sheet names and print them
print ("Names of the sheets in the workbook:")
for sheet_name in sheet_names:
    print(sheet_name)
```

```
Names of the sheets in the workbook:
Data Notes
Table 9
```

```
# name of the sheet you want to access
sheet_name = 'Table 9' #expect an error

# access the specific sheet by name
sheet = workbook[sheet_name]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-5-0345df96678b> in <cell line: 5>()
      3
      4 # access the specific sheet by name
----> 5 sheet = workbook[sheet_name]

/usr/local/lib/python3.10/dist-packages/openpyxl/workbook/workbook.py in
__getitem__(self, key)
    285             if sheet.title == key:
    286                 return sheet
--> 287         raise KeyError("Worksheet {0} does not exist.".format(key))
    288
    289     def __delitem__(self, key):

KeyError: 'Worksheet Table 9 does not exist.'
```

Next steps: Explain error

```
# name of the sheet you want to access
sheet_name = 'Table 9 ' # fixed spacing

# access the specific sheet by name
sheet = workbook[sheet_name]

# print to make sure it loaded - 'sanity' test or 'debug' test
print(sheet)
```

```
<Worksheet "Table 9 ">
```

```
# show what methods are available
print(dir(sheet))
```

```
['BREAK_COLUMN', 'BREAK_NONE', 'BREAK_ROW', 'HeaderFooter', 'ORIENTATION_LANDSCAPE', 'ORIENTATION_PORTRAIT', 'PAPERSIZE_A3', 'PAPERSIZE
```

```
# show it is iterable (we can use a for loop)
print(sheet.rows)
```

```
<generator object Worksheet._cells_by_row at 0x7c4a991bfd80>
```

```
# documentation on the 'rows' method
help(sheet.rows)
```

```
Help on generator object:

_cells_by_row = class generator(object)
 |  Methods defined here:
 |
 |  __del__(...)
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __next__(self, /)
 |      Implement next(self).
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  close(...)
 |      close() -> raise GeneratorExit inside generator.
 |
 |  send(...)
 |      send(arg) -> send 'arg' into generator,
 |      return next yielded value or raise StopIteration.
 |
 |  throw(...)
 |      throw(value)
 |      throw(type[,value[,tb]])
 |
 |      Raise exception in generator, return next yielded value or raise
 |      StopIteration.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  gi_code
 |
 |  gi_frame
 |
 |  gi_running
 |
 |  gi_yieldfrom
 |      object being iterated by yield from, or None
```

```
# row data from the worksheet

# iterate over each row and cell, then print the values
for row in sheet.rows:
    for cell in row:
        print(cell.value, end='\t')
print()
```

```
None    TABLE 9. CHILD PROTECTION       None    None    None    None    None    None    None    None    None    None    None    None
2005–2012*      None    None    None    None    None    Child marriage (%)
2005–2012*      None    None    None    Birth registration (%)+
2005–2012*      None    Female genital mutilation/cutting (%)+
2002–2012*      None    None    None    None    None    Justification of wife beating (%)
 2005–2012*     None    None    None    Violent discipline (%)+
2005–2012*      None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
2005–2012*      None    None    None    None    None    Mariage d’enfants (%)
2005–2012*      None    None    None    Enregistrement
des naissances
(%)+
2005–2012*
        None    Mutilations génitales féminines/excision (%)+
2002–2012*      None    None    None    None    None    Justification de la
violence conjugale (%)
2005–2012*      None    None    None    Discipline imposée par la violence (%)+
2005–2012*      None    None    None    None    None    None    None    None    None    None    None    None    None    None    None
        None    marié à  18 ans None    total   None    femmes a        None    filles b        None    soutien à cette pratique c
nacimiento (%) + 2005–2012*     None    Mutilación/excisión genital (%)+
```

```
2002–2012*      None    None    None    None    None    Justificación de golpear
a la mujer (%)
2005–2012*      None    None    None    Disciplina violenta (%)+  2005–2012*    None    None    None    None    None    None    None
```

```
# Print the contents of each row and cells, also improve readability

# iterate over each row
for row_index, row_values in enumerate(sheet.iter_rows(min_row=1, values_only=True), start=1):
    row_name = f"Row {row_index}"

    print(row_name)

    #iterate through each cell in the row
    for cell_index, cell_value in enumerate(row_values, start=1):
        print(f"  Cell {cell_index}: {cell_value}")

    # improve readability by adding a separator between each row
    print("-" * 20)
```

```
  Cell 45: None
  Cell 46: None
  Cell 47: None
  Cell 48: None
  Cell 49: None
--------------------
Row 349
  Cell 1: None
  Cell 2: None
  Cell 3: None
  Cell 4: None
  Cell 5: None
  Cell 6: None
  Cell 7: None
  Cell 8: None
  Cell 9: None
  Cell 10: None
  Cell 11: None
  Cell 12: None
  Cell 13: None
  Cell 14: None
  Cell 15: None
  Cell 16: None
  Cell 17: None
  Cell 18: None
  Cell 19: None
  Cell 20: None
  Cell 21: None
  Cell 22: None
  Cell 23: None
  Cell 24: None
  Cell 25: None
  Cell 26: None
  Cell 27: None
  Cell 28: None
  Cell 29: None
  Cell 30: None
  Cell 31: None
  Cell 32: None
  Cell 33: None
  Cell 34: None
  Cell 35: None
  Cell 36: None
  Cell 37: None
  Cell 38: None
  Cell 39: None
  Cell 40: None
  Cell 41: None
  Cell 42: None
  Cell 43: None
  Cell 44: None
  Cell 45: None
  Cell 46: None
  Cell 47: None
  Cell 48: None
  Cell 49: None
--------------------
```

```
# skip to the header string "Countries and areas"
start_row = None
# iterate over the data
for row_index, row_values in enumerate(sheet.iter_rows(min_row=1, values_only=True), start=1):
    # check if the row contains the header string
  if "Countries and areas" in row_values:
      # if found, go to the next row
      start_row = row_index + 1
      break

# dictionary to store extracted data
extracted_data = {}

# loop through the rows starting with start_row
if start_row is not None:
    # extract the data from each row (i.e country, child labor, and other data)
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, values_only=True), start=start_row):
        country_name = row_values[1]
        child_labor_data = {
            'total': row_values[4],
            'male': row_values[6],
            'female': row_values[8]
        }
        other_data = row_values[10:]

        # store data in the dictionary
        extracted_data[country_name] = {'child_labor': child_labor_data, 'other_data': other_data}

        # print the extracted data and associated a row number
        print(f"Row {row_index}: {row_values[1:4]}")
        print(f"    Child Labor (%): {row_values[4]} (total), {row_values[6]} (male), {row_values[8]} (female)")
        print(f"    Other Data: {row_values[10:]}")
        print("-" * 50)

else:
    print("'Countries and areas' not found")
```

```
Row 6: (None, None, None)
    Child Labor (%): None (total), None (male), None (female)
    Other Data: (None, None, None, None, None, None, 'prevalence', None, None, None, 'attitudes', None, None, None, None, None, None,
--------------------------------------------------
Row 7: (None, None, None)
    Child Labor (%): total (total), male (male), female (female)
    Other Data: ('married by 15', None, 'married by 18', None, 'total', None, 'womena  ', None, 'girlsb ', None, 'support for the pra
--------------------------------------------------
Row 8: ('FRENCH HEADINGS', 'Pays et zones', None)
    Child Labor (%): Travail des enfants (%)+
2005–2012* (total), None (male), None (female)
    Other Data: ('Mariage d’enfants (%)\n2005–2012*', None, None, None, 'Enregistrement\ndes naissances\n(%)+\n2005–2012*\n', None,
--------------------------------------------------
Row 9: (None, None, None)
    Child Labor (%): None (total), None (male), None (female)
    Other Data: (None, None, None, None, None, None, 'prévalence', None, None, None, 'attitudes', None, None, None, None, None, None,
--------------------------------------------------
Row 10: (None, None, None)
    Child Labor (%): total (total), garçons (male), filles (female)
    Other Data: ('marié à  15 ans\n', None, 'marié à  18 ans', None, 'total', None, 'femmes a  ', None, 'filles b ', None, 'soutien à
--------------------------------------------------
Row 11: ('SPANISH HEADINGS', None, 'Países y zonas')
    Child Labor (%): Trabajo infantil (%)+ 2005–2012* (total), None (male), None (female)
    Other Data: ('Matrimonio precoz (%) 2005–2012*', None, None, None, 'Inscripción del\nnacimiento (%) + 2005–2012*', None, 'Mutilac
--------------------------------------------------
Row 12: (None, None, None)
    Child Labor (%): None (total), None (male), None (female)
    Other Data: (None, None, None, None, None, None, 'prevalencia', None, None, None, 'actitudes', None, None, None, None, None, None
--------------------------------------------------
Row 13: (None, None, None)
    Child Labor (%): total (total), hombre (male), mujer (female)
    Other Data: ('casados a los 15 años', None, 'casados a los 18 años', None, 'total', None, 'Mujeresa  ', None, 'hijasb ', None, 'a
--------------------------------------------------
Row 14: (None, 'FRENCH COUNTRY NAMES', 'SPANISH COUNTRY NAMES')
    Child Labor (%): None (total), None (male), None (female)
    Other Data: (None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, No
--------------------------------------------------
Row 15: ('Afghanistan', 'Afghanistan', 'Afganistán')
    Child Labor (%): 10.3 (total), 11 (male), 9.6 (female)
    Other Data: (15, None, 40.4, None, 37.4, None, '–', None, '–', None, '–', None, '–', None, 90.2, None, 74.4, None, 74.8, None, 74
--------------------------------------------------
Row 16: ('Albania', 'Albanie', 'Albania')
    Child Labor (%): 12 (total), 14.4 (male), 9.4 (female)
    Other Data: (0.2, None, 9.6, None, 98.6, None, '–', None, '–', None, '–', None, 36.4, None, 29.8, None, 75.1, None, 78.3, None, 7
```

```
--------------------------------------------------
Row 17: ('Algeria', 'Algérie', 'Argelia')
    Child Labor (%): 4.7 (total), 5.5 (male), 3.9 (female)
    Other Data: (0.1, None, 1.8, None, 99.3, None, '-', None, '-', None, '-', None, '-', None, 67.9, None, 87.7, None, 88.8, None, 86
--------------------------------------------------
Row 18: ('Andorra', 'Andorre', 'Andorra')
    Child Labor (%): -  (total), - (male), - (female)
    Other Data: ('-', None, '-', None, 100, 'v', '-', None, '-', None, '-', None, '-', None, '-', None, '-', None, '-', None, '-', No
--------------------------------------------------
Row 19: ('Angola', 'Angola', 'Angola')
    Child Labor (%): 23.5 (total), 22.1 (male), 24.8 (female)
    Other Data: ('-', None, '-', None, 36, 'x', '-', None, '-', None, '-', None, '-', None, '-', None, '-', None, '-', None, '-', Nor
--------------------------------------------------
```

```
# start from row 15, the first country
start_row = 15

# stop at row 212, the last country
stop_row = 212

# make sure when have are extracting data based on the countries
if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and start_row <= stop_row:
    extracted_data = {}
    # extract the data from each row
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, max_row=stop_row, values_only=True), start=start_row):
        country_name = row_values[1]

        # skip rows where country_name is None
        if country_name is None:
            continue

        child_labor_data = {
            'total': row_values[4],
            'male': row_values[6],
            'female': row_values[8]
        }
        other_data = row_values[10:]

        # store data in the dictionary
        extracted_data[country_name] = {'child_labor': child_labor_data, 'other_data': other_data}

        # print the names of the country only
        print("\nExtracted Country Names:")
        for i, name in enumerate(extracted_data.keys(), start=1):
            print(f"{i}. {name}")

else:
    print("Error with start or stop row values")
```

```
Streaming output truncated to the last 5000 lines.
21. Bolivia (Plurinational State of)
22. Bosnia and Herzegovina
23. Botswana
24. Brazil
25. Brunei Darussalam
26. Bulgaria
27. Burkina Faso
28. Burundi
29. Cabo Verde
30. Cambodia
31. Cameroon
32. Canada
33. Central African Republic
34. Chad
35. Chile
36. China
37. Colombia
38. Comoros
39. Congo
40. Cook Islands
41. Costa Rica
42. Côte d'Ivoire
43. Croatia
44. Cuba
45. Cyprus
46. Czech Republic
47. Democratic People's Republic of Korea
48. Democratic Republic of the Congo
49. Denmark
50. Djibouti
```

```
51. Dominica
52. Dominican Republic
53. Ecuador
54. Egypt
55. El Salvador
56. Equatorial Guinea
57. Eritrea
58. Estonia
59. Ethiopia
60. Fiji
61. Finland
62. France
63. Gabon
64. Gambia
65. Georgia
66. Germany
67. Ghana
68. Greece
69. Grenada
70. Guatemala
71. Guinea
72. Guinea-Bissau
73. Guyana
74. Haiti
75. Holy See
76. Honduras
77. Hungary
```

```
# now that we are extracting the data from the countries

# iterate the data starting with the first country and stop processing on the last country

if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and start_row <= stop_row:
    extracted_data = {}

    # get the headers
    headers_row = next(sheet.iter_rows(min_row=1, max_row=1, values_only=True))
    headers = headers_row[1:]

    # extract the data from each row
    for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row, max_row=stop_row, values_only=True), start=start_row):
        country_name = row_values[1]

        # skip rows where country_name is None
        if country_name is None:
          continue

        # create a dictionary to store data for the current country
        country_data = {}

        # process child labor data
        child_labor_labels = ['total', 'male', 'female']
        child_labor_values = [None if value in ('-', ' ', None) or not isinstance(value, (int, float)) else float(value) if isinstance(value,

        country_data['child_labor'] = dict(zip(child_labor_labels, child_labor_values))

        #process other data
        other_data_labels = ['married_by_15', 'married_by_18']
        other_data_values = [None if value in ('-', ' ', None) or not isinstance(value, (int, float)) else float(value) if isinstance(value,
        country_data['other_data'] = dict(zip(other_data_labels, other_data_values))

        # add the country to dictionary
        extracted_data[country_name] = country_data

        # print the extracted or pulled data that we are interested in
        for country, data in extracted_data.items():
            print(f"\nCountry: {country}")
            print("Data:")
            for category, values in data.items():
                print(f"  {category}: {values}")
            print("-" * 50)

else:
    print("Error with start or stop row values")
```

```
Streaming output truncated to the last 5000 lines.
  other_data: {'married_by_15': None, 'married_by_18': 18.9}
--------------------------------------------------

Country: Romania
```

```
Data:
  child_labor: {'total': 0.9, 'male': None, 'female': 1.2}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Russian Federation
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Rwanda
Data:
  child_labor: {'total': 28.5, 'male': None, 'female': 26.7}
  other_data: {'married_by_15': None, 'married_by_18': 8.1}
--------------------------------------------------

Country: Saint Kitts and Nevis
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Saint Lucia
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Saint Vincent and the Grenadines
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Samoa
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: San Marino
Data:
  child_labor: {'total': None, 'male': None, 'female': None}
  other_data: {'married_by_15': None, 'married_by_18': None}
--------------------------------------------------

Country: Sao Tome and Principe
Data:
  child_labor: {'total': 7.5, 'male': None, 'female': 7.7}
  other_data: {'married_by_15': None, 'married_by_18': 34.4}
--------------------------------------------------
```