```
!pip install openpyxl
```

```
    Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/d
    Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10
```

```
#requred packages `!pip install openpxyl`
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` tes
print(workbook)
```

```
    <openpyxl.workbook.workbook.Workbook object at 0x7fa499e104f0>
```

```
# variable to hold the anmes 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_names = 'Table 9' # expect an error

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

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
<ipython-input-8-a3a4a4b0384c> 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):
```

```
# 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', 'ORIENTATI
```

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

```
<generator object Worksheet._cells_by_row at 0x7fa499ec8430>
```

```
# 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 rai
 |      StopIteration.
 |
 |  ------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  gi_code
 |
 |  gi_frame
 |
 |  gi_running
 |
 |  gi_yieldfrom
 |      object being iterated by yield from, or None
```

```
# raw 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    TABLEAU 9. PROTECTION DE L'ENFANT       None    None
None    None    None    TABLA 9. PROTECCIÓN INFANTIL    None    None
None    None    None    None    None    None    None    None    None
None    Countries and areas     None    None    Child labour (%)+
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
 2005–2012*     None    None    None    Violent discipline (%)+
2005–2012*      None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None
None    None    None    None    total   None    male    None    female
None    FRENCH HEADINGS Pays et zones   None    Travail des enfants (%
2005–2012*      None    None    None    None    None    Mariage d'enfa
2005–2012*      None    None    None    Enregistrement
des naissances
(%)+
2005–2012*
        None    Mutilations génitales féminines/excision (%)+
2002–2012*      None    None    None    None    None    Justification
violence conjugale (%)
2005–2012*      None    None    None    Discipline imposée par la viol
2005–2012*      None    None    None    None    None    None    None
None    None    None    None    None    None    None    None    None
None    None    None    None    total   None    garçons None    filles
        None    marié à  18 ans None    total   None    femmes a
None    SPANISH HEADINGS        None    Países y zonas  Trabajo infant
nacimiento (%) + 2005–2012*     None    Mutilación/excisión genital (%
2002–2012*      None    None    None    None    None    Justificación
a la mujer (%)
2005–2012*      None    None    None    Disciplina violenta (%)+  2005
None    None    None    None    None    None    None    None    None
None    None    None    None    total   None    hombre  None    mujer
None    None    FRENCH COUNTRY NAMES    SPANISH COUNTRY NAMES   None
None    Afghanistan     Afghanistan     Afganistán      10.3    None
None    Albania Albanie Albania 12              14.4            9.4
None    Algeria Algérie Argelia 4.7     y       5.5     y       3.9
None    Andorra Andorre Andorra –               –               –
None    Angola  Angola  Angola  23.5    x       22.1    x       24.8
None    Antigua and Barbuda     Antigua-et-Barbuda      Antigua y Barb
None    Argentina       Argentine       Argentina       6.5     y
None    Armenia Arménie Armenia 3.9     None    4.7     None    2.9
None    Australia       Australie       Australia       –
None    Austria Autriche        Austria –               –
None    Azerbaijan      Azerbaïdjan     Azerbaiyán      6.5     y
None    Bahamas Bahamas Bahamas –               –               –
None    Bahrain Bahreïn Bahrein 4.6     x       6.3     x       3
None    Bangladesh      Bangladesh      Bangladesh      12.8
None    Barbados        Barbade Barbados        –               –
None    Belarus Bélarus Belarús 1.4             1.3             1.5
None    Belgium Belgique        Bélgica –               –
None    Belize  Belize  Belice  5.8             6.7             5
None    Benin   Bénin   Benin   45.6            46.5            44.6
None    Bhutan  Bhoutan Bhután  2.9     None    2.6     None    3.1
None    Bolivia (Plurinational State of)        Bolivie (État plurinat
None    Bosnia and Herzegovina  Bosnie-Herzégovine      Bosnia y Herze
None    Botswana        Botswana        Botswana        9       y
None    Brazil  Brésil  Brasil  8.6     y       11.2    y       5.9
```

```
# 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_or
  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)
```

```
Streaming output truncated to the last 5000 lines.
  Cell 49: None
--------------------
Row 252
  Cell 1: None
  Cell 2: None
  Cell 3: None
  Cell 4: + En las Notas generales a los datos se puede encontrar una
  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
--------------------
Row 253
  Cell 1: None
  Cell 2: None
  Cell 3: None
  Cell 4: * Datos referidos al año disponible más reciente durante el
```

```
# skip to the header string "countries and areas"
start_row = None
# iterate over each row
for row_index, row_values in enumerate(sheet.iter_rows(min_row=1,values_or
  # 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
      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,'oth

        # print the extracted data and associated a row numberprint(f"Row
else:
    print ("'Countries and areas' not found")
```

```
# 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
  extracted_data = {}
    # extract the data from each row
  for row_index, row_values in enumerate(sheet.iter_rows(min_row=start_row
      country_name = row_values

    # 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")
```

```
Extracted Country Names:
1.(None, None, None, None, None, None, None, None, None, None, None, N
```

```
# now that we are extracting the data from the countries
# iterate the data starting with the first country and stop processing on
if 1 <= start_row <= sheet.max_row and 1 <= stop_row <= sheet.max_row and
  extracted_data = {}

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

  # extract the data from each row
  for row_index, row_values in enumerate(sheet.iter_rows(min_row=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 isins
      country_data[' child labor'] = dict(zip(child_labor_labels, child_la

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

      # 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")
```

```
      nCountry: Afghanistan
```

```
Data:
  child labor: {'total': 10.3, 'male': None, 'female': 11.0}
 other_data: {}
--------------------------------------------------
 nCountry: Albania
Data:
  child labor: {'total': 12.0, 'male': None, 'female': 14.4}
 other_data: {}
--------------------------------------------------
 nCountry: Algeria
Data:
  child labor: {'total': 4.7, 'male': None, 'female': 5.5}
 other_data: {}
--------------------------------------------------
 nCountry: Andorra
Data:
  child labor: {'total': None, 'male': None, 'female': None}
 other_data: {}
--------------------------------------------------
 nCountry: Angola
Data:
  child labor: {'total': 23.5, 'male': None, 'female': 22.1}
 other_data: {}
--------------------------------------------------
 nCountry: Antigua and Barbuda
Data:
  child labor: {'total': None, 'male': None, 'female': None}
 other_data: {}
--------------------------------------------------
 nCountry: Argentina
Data:
  child labor: {'total': 6.5, 'male': None, 'female': 7.6}
 other_data: {}
--------------------------------------------------
 nCountry: Armenia
Data:
  child labor: {'total': 3.9, 'male': None, 'female': 4.7}
 other_data: {}
--------------------------------------------------
 nCountry: Australia
Data:
  child labor: {'total': None, 'male': None, 'female': None}
 other_data: {}
--------------------------------------------------
 nCountry: Austria
Data:
  child labor: {'total': None, 'male': None, 'female': None}
 other_data: {}
--------------------------------------------------
 nCountry: Azerbaijan
Data:
  child labor: {'total': 6.5, 'male': None, 'female': 7.5}
 other_data: {}
--------------------------------------------------
 nCountry: Bahamas
Data:
  child labor: {'total': None, 'male': None, 'female': None}
 other_data: {}
```