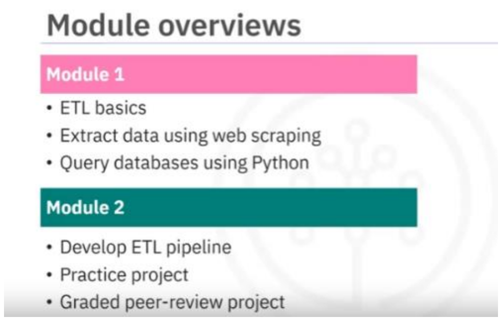
# Data Engineering Day 03

**The credit for this course goes to Coursera. Click More**

**Another link : Azure data Engineer**

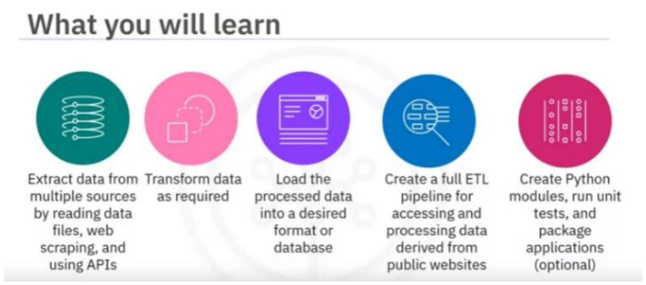
**Python Project Extract, Transform and Load (ETL) for Data Engineers.**



Module overviews

Module 1
- ETL basics
- Extract data using web scraping
- Query databases using Python

Module 2
- Develop ETL pipeline
- Practice project
- Graded peer-review project

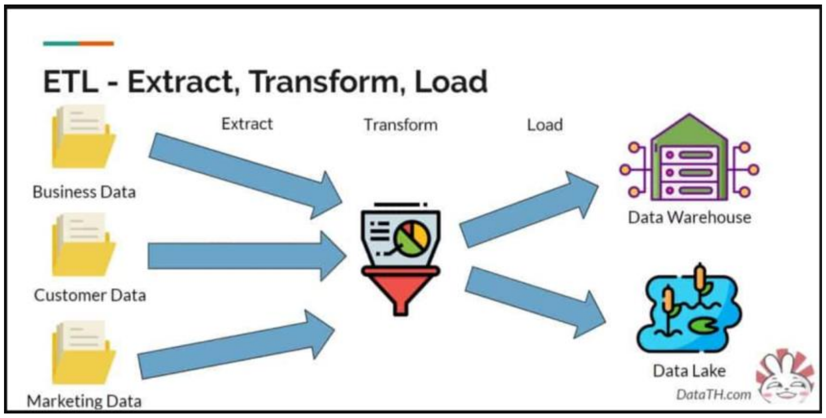


What you will learn

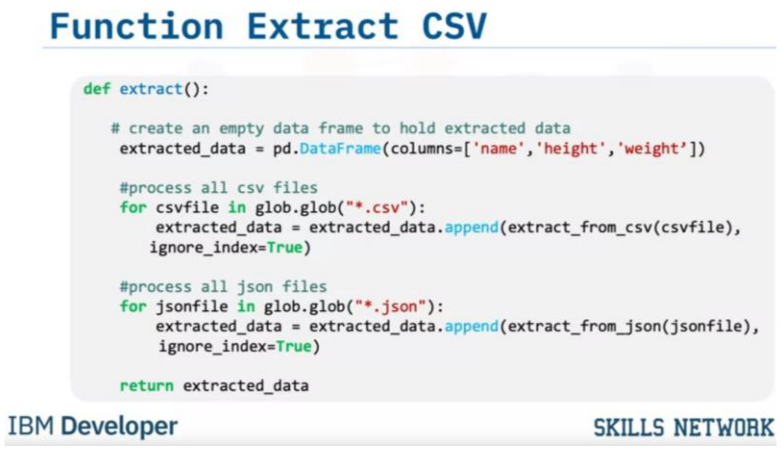
- Extract data from multiple sources by reading data files, web scraping, and using APIs
- Transform data as required
- Load the processed data into a desired format or database
- Create a full ETL pipeline for accessing and processing data derived from public websites
- Create Python modules, run unit tests, and package applications (optional)

## Extract, Transform and Load (ETL) in practice:

- In this method, we gather data from different sources that might have different formats. We then convert this data into a single, consistent format. After that, we store this formatted data in CSV files, which are commonly used for storing tabular data. This process makes the data easier to read and work with for data scientists.



**Extracts the data and its process:**

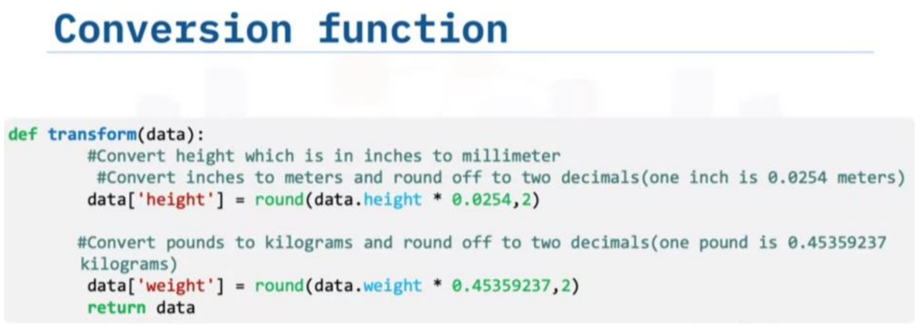


## Function Extract CSV

```
def extract():

    # create an empty data frame to hold extracted data
     extracted_data = pd.DataFrame(columns=['name','height','weight'])

     #process all csv files
     for csvfile in glob.glob("*.csv"):
         extracted_data = extracted_data.append(extract_from_csv(csvfile),
        ignore_index=True)

     #process all json files
     for jsonfile in glob.glob("*.json"):
         extracted_data = extracted_data.append(extract_from_json(jsonfile),
          ignore_index=True)

     return extracted_data
```

IBM Developer SKILLS NETWORK

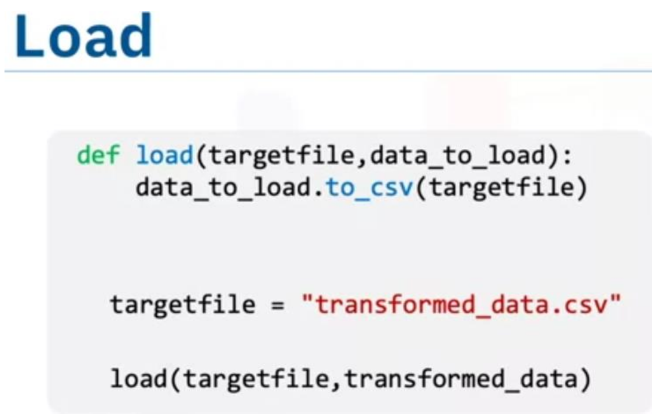
**Transforms the data and its process:**



## Conversion function

```
def transform(data):
        #Convert height which is in inches to millimeter
         #Convert inches to meters and round off to two decimals(one inch is 0.0254 meters)
        data['height'] = round(data.height * 0.0254,2)

       #Convert pounds to kilograms and round off to two decimals(one pound is 0.45359237
       kilograms)
        data['weight'] = round(data.weight * 0.45359237,2)
        return data
```

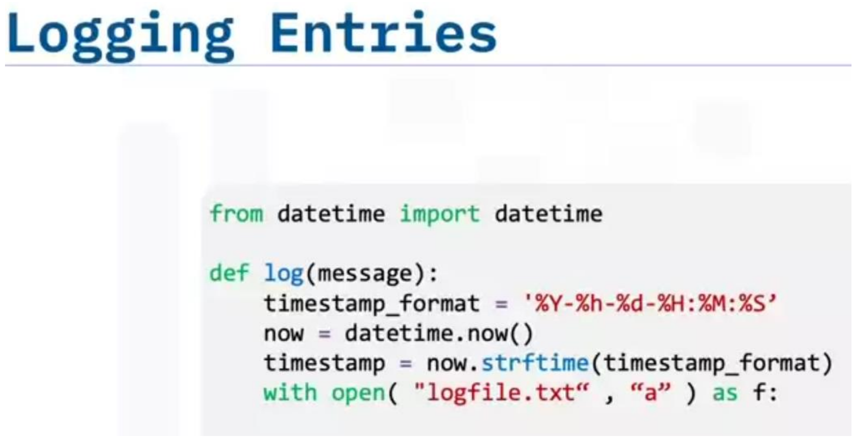
**Loading and its process:**



## Load

```
def load(targetfile,data_to_load):
    data_to_load.to_csv(targetfile)


  targetfile = "transformed_data.csv"

  load(targetfile,transformed_data)
```

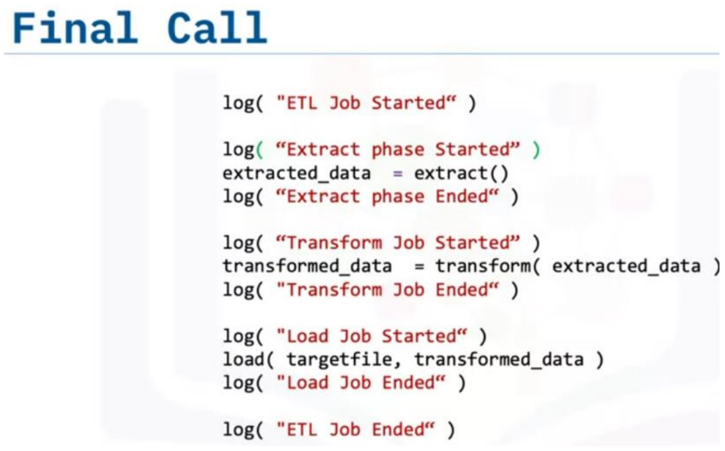
**Logging functions:**



## Logging Entries

```
from datetime import datetime

def log(message):
    timestamp_format = '%Y-%h-%d-%H:%M:%S'
    now = datetime.now()
    timestamp = now.strftime(timestamp_format)
    with open( "logfile.txt“ , “a” ) as f:
```

**Functions call:**



## Final Call

```
log( "ETL Job Started“ )

log( “Extract phase Started” )
extracted_data  = extract()
log( “Extract phase Ended“ )

log( “Transform Job Started” )
transformed_data  = transform( extracted_data )
log( "Transform Job Ended“ )

log( "Load Job Started“ )
load( targetfile, transformed_data )
log( "Load Job Ended“ )

log( "ETL Job Ended“ )
```

*Question 01* **Hands-on Lab: Extract, Transform, Load (ETL):**

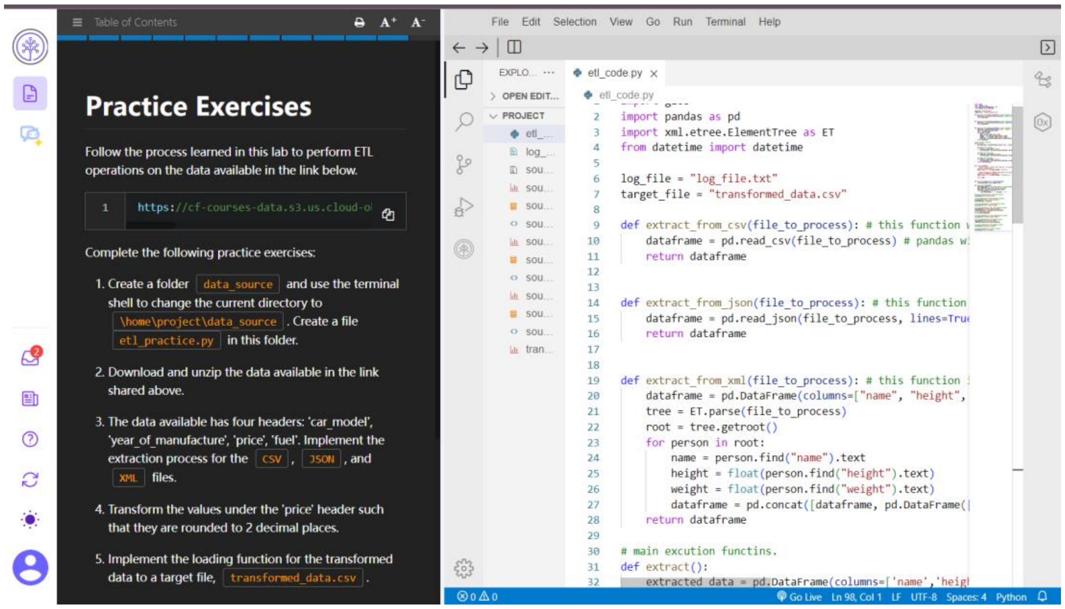


Table of Contents
File Edit Selection View Go Run Terminal Help
EXPLO...
OPEN EDIT...
PROJECT
etl_code.py
Go Live Ln 98, Col 1 LF UTF-8 Spaces: 4 Python

**Practice Exercises**

Follow the process learned in this lab to perform ETL operations on the data available in the link below.

```
1  https://cf-courses-data.s3.us.cloud-o
```

Complete the following practice exercises:

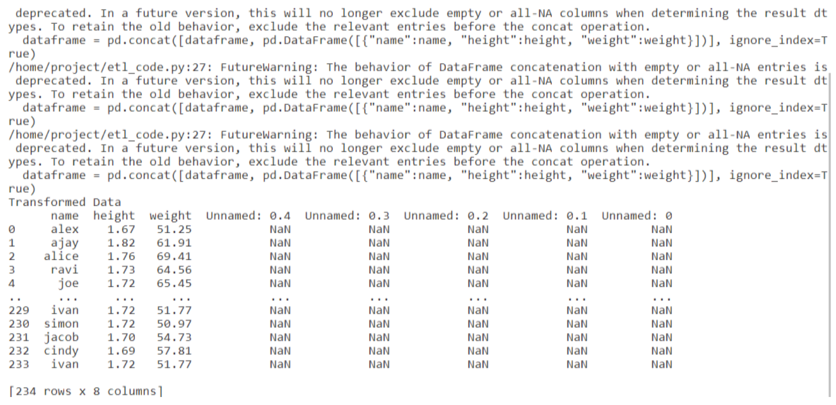
1. Create a folder `data_source` and use the terminal shell to change the current directory to `\home\project\data_source`. Create a file `etl_practice.py` in this folder.
2. Download and unzip the data available in the link shared above.
3. The data available has four headers: 'car_model', 'year_of_manufacture', 'price', 'fuel'. Implement the extraction process for the `CSV`, `JSON`, and `XML` files.
4. Transform the values under the 'price' header such that they are rounded to 2 decimal places.
5. Implement the loading function for the transformed data to a target file, `transformed_data.csv`.

```python
import pandas as pd
import xml.etree.ElementTree as ET
from datetime import datetime

log_file = "log_file.txt"
target_file = "transformed_data.csv"

def extract_from_csv(file_to_process): # this function w
    dataframe = pd.read_csv(file_to_process) # pandas wi
    return dataframe


def extract_from_json(file_to_process): # this function
    dataframe = pd.read_json(file_to_process, lines=True
    return dataframe


def extract_from_xml(file_to_process): # this function i
    dataframe = pd.DataFrame(columns=["name", "height",
    tree = ET.parse(file_to_process)
    root = tree.getroot()
    for person in root:
        name = person.find("name").text
        height = float(person.find("height").text)
        weight = float(person.find("weight").text)
        dataframe = pd.concat([dataframe, pd.DataFrame([
    return dataframe

# main excution functins.
def extract():
    extracted data = pd.DataFrame(columns=['name','heig
```

```python
import glob
import pandas as pd
import xml.etree.ElementTree as ET
from datetime import datetime
log_file = "log_file.txt"
target_file = "transformed_data.csv"
def extract_from_csv(file_to_process): # this function will extract the data from csv file.
dataframe = pd.read_csv(file_to_process) # pandas will read csv file and transfer data into
dataframe
return dataframe
def extract_from_json(file_to_process): # this function will extract the data from json file.
dataframe = pd.read_json(file_to_process, lines=True)
return dataframe
def extract_from_xml(file_to_process): # this function is to ectract the  data from the xml file.
        dataframe = pd.DataFrame(columns=["name", "height", "weight"])
tree = ET.parse(file_to_process)
    root = tree.getroot()
    for person in root:
        name = person.find("name").text
        height = float(person.find("height").text)
        weight = float(person.find("weight").text)
        dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height,
"weight":weight}])], ignore_index=True)
    return dataframe

# main excution functins.
def extract():
    extracted_data = pd.DataFrame(columns=['name','height','weight']) # create an empty data frame 
to hold extracted data

    # process all csv files
    for csvfile in glob.glob("*.csv"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_csv(csvfile))], 
ignore_index=True)

    # process all json files
    for jsonfile in glob.glob("*.json"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_json(jsonfile))], 
ignore_index=True)

    # process all xml files
    for xmlfile in glob.glob("*.xml"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_xml(xmlfile))], 
ignore_index=True)
    return extracted_data
def transform(data):
    '''Convert inches to meters and round off to two decimals
    1 inch is 0.0254 meters '''
    data['height'] = round(data.height * 0.0254,2)

    '''Convert pounds to kilograms and round off to two decimals
    1 pound is 0.45359237 kilograms '''
    data['weight'] = round(data.weight * 0.45359237,2)

    return data
def load_data(target_file, transformed_data):
    transformed_data.to_csv(target_file)

def log_progress(message):
    timestamp_format = '%Y-%h-%d-%H:%M:%S' # Year-Monthname-Day-Hour-Minute-Second
now = datetime.now() # get current timestamp
    timestamp = now.strftime(timestamp_format)
    with open(log_file,"a") as f:
        f.write(timestamp + ',' + message + '\n')

# Log the initialization of the ETL process
log_progress("ETL Job Started")

# Log the beginning of the Extraction process
log_progress("Extract phase Started")
extracted_data = extract()
```

```
        # Log the completion of the Extraction process
        log_progress("Extract phase Ended")

        # Log the beginning of the Transformation process
        log_progress("Transform phase Started")
        transformed_data = transform(extracted_data)
        print("Transformed Data")
        print(transformed_data)

        # Log the completion of the Transformation process
        log_progress("Transform phase Ended")

        # Log the beginning of the Loading process
        log_progress("Load phase Started")
        load_data(target_file,transformed_data)

        # Log the completion of the Loading process
        log_progress("Load phase Ended")

        # Log the completion of the ETL process
        log_progress("ETL Job Ended")
```

*the figure below shows output obtained from the code editor.*



```
 deprecated. In a future version, this will no longer exclude empty or all-NA columns when determining the result dt
ypes. To retain the old behavior, exclude the relevant entries before the concat operation.
  dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height, "weight":weight}])], ignore_index=T
rue)
/home/project/etl_code.py:27: FutureWarning: The behavior of DataFrame concatenation with empty or all-NA entries is
 deprecated. In a future version, this will no longer exclude empty or all-NA columns when determining the result dt
ypes. To retain the old behavior, exclude the relevant entries before the concat operation.
  dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height, "weight":weight}])], ignore_index=T
rue)
/home/project/etl_code.py:27: FutureWarning: The behavior of DataFrame concatenation with empty or all-NA entries is
 deprecated. In a future version, this will no longer exclude empty or all-NA columns when determining the result dt
ypes. To retain the old behavior, exclude the relevant entries before the concat operation.
  dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height, "weight":weight}])], ignore_index=T
rue)
Transformed Data
      name  height  weight  Unnamed: 0.4  Unnamed: 0.3  Unnamed: 0.2  Unnamed: 0.1  Unnamed: 0
0     alex    1.67   51.25           NaN           NaN           NaN           NaN         NaN
1     ajay    1.82   61.91           NaN           NaN           NaN           NaN         NaN
2    alice    1.76   69.41           NaN           NaN           NaN           NaN         NaN
3     ravi    1.73   64.56           NaN           NaN           NaN           NaN         NaN
4      joe    1.72   65.45           NaN           NaN           NaN           NaN         NaN
..     ...     ...     ...           ...           ...           ...           ...         ...
229   ivan    1.72   51.77           NaN           NaN           NaN           NaN         NaN
230  simon    1.72   50.97           NaN           NaN           NaN           NaN         NaN
231  jacob    1.70   54.73           NaN           NaN           NaN           NaN         NaN
232  cindy    1.69   57.81           NaN           NaN           NaN           NaN         NaN
233   ivan    1.72   51.77           NaN           NaN           NaN           NaN         NaN

[234 rows x 8 columns]
```