Liionsden raw data file parsing workflow

1) Fields of the **Parser** associated with the **ExperimentDataFile** are used to specify arguments for the **parse_data_file()** function. This results in an **engine** class with properties including **data** (**engine.data**).

battDB.models.py

Create an **ExperimentDataFile**.
ExperimentDataFile.clean():
  parsed_file = **parse_data_file(file_obj, file_format, columns)**

- **file_obj** is the associated **raw_data_file.file**
- **file_format** is the associated **Parser.file_format** and is the name of an available parsing engine
- **columns** is the column *names* as they appear in the **Parser** and should match the column headings in the file. E.g. ["time/s", "Ecell/V"]

parsing_engines.parsing_engines_base.py

parse_data_file(file_obj, file_format, columns):
  engine = get_parsing_engine(file_format).factory(file_obj)

- **engine** is initialized using functions in the appropriate *<cycler>_engine.py* module

parsing_engines.maccor_engine.py

e.g. **MaccorParsingEngine.factory(file_obj):**
  load_maccor_data(file_obj)

load_maccor_data(file_obj):
  data = pd.read_excel(…)

- The **load_<cycler>_data()** function is used to read the file into a pandas dataframe
- Methods of the **<Cycler>ParsingEngine** class are used to read the file when initializing the **engine** e.g. **get_header_size()** etc.

---

2) Methods and properties of **engine** are used to set attributes of the **ExperimentDataFile**.

battDB.models.py

ExperimentDataFile.clean():
  parsed_file = parse_data_file(file_obj, file_format, columns)

  self.attributes["file_columns"] = parsed_file["file_columns"]
  …
  self.**ts_headers** = self.attributes.get("**parsed_columns**")
  self.**ts_data** = **parsed_file["data"]**

- Keys of the **parsed_file** dict are used to set fields of the **ExperimentDataFile** object
- This includes **data** and **parsed_columns**, which populate **ts_data** and **ts_headers**, respectively (ts = timeseries)

parsing_engines.parsing_engines_base.py

parse_data_file(file_obj, file_format, columns):
  engine = get_parsing_engine(file_format).factory(file_obj)
  …
  parsed_columns, parsed_header_columns =
  get_parsed_columns(file_columns, columns, col_mapping)
  …
  data =
  engine.get_data_generator_for_columns(parsed_columns)
  …
  return { … ,
  "parsed_columns": **parsed_columns**,
  "parsed_header_columns": **parsed_header_columns**,
  "**data**": list(**data**), … }

engine.get_data_generator_for_columns(parsed_columns):
  for row in self.**data**[**cols**].itertuples():
  yield list(row)[1:]

- **parsed_columns** is a list of the column names from the **Parser** used, which were found in the file
- **col_mapping** is a property of the **engine**; a dict of column headings to handle subtle deviations between the column name in **Parser** and that found in the file e.g.: {
  "time": "time/s",
  "Time": "time/s",
  "time/s": "time/s", … }
- **parsed_header_columns** is a list the same length as **parsed_columns** but with the original column headings as they appear in the file
- **parsed_header_columns** is used as input to **get_data_generator_for_columns()** to get the required data from the pandas DataFrame (self.**data**)

N.B. for producing plots in **battDB.views.ExperimentView**, **.ts_headers** is used, which is set to **parsed_columns**