PulseHeatPipe

PyPulseHeatPipe is a Python Library for data analysis and for data plotting/visualisation specifically for PHP experimental data.

pkg installation

PulseHeatPipe is a Python library designed to perform thermodynamic data analysis on experimental data from pulsating heat pipes (PHP).

This library helps estimate optimal working conditions based on the change in Gibbs free energy within the PHP system.

Key Features:

- Generate a blank template file for experimental data entry.
- Perform data extraction, transformation, and loading (ETL) from multiple experimental files.
- Convert units to the SI system.
- Calculate thermal resistance.
- Calculate Gibbs free energy at given temperatures and pressures.
- Select data within a specific temperature range.
- Compute and save statistical properties (mean, standard deviation) of the data.
- Calculate average values of thermal properties.

- Identify optimal thermal properties based on the minimum Gibbs free energy change.

Attributes:

```
T_k (float): Conversion constant to Kelvin (default is 273.15).

P_const (float): Conversion constant to absolute bar (default is 1.013).

R_const (float): Real gas constant (default is 8.314 J/Kmol).

dG_standard (float): Standard Gibbs free energy change of water (default is
30.9 KJ/mol).

P_standard (float): Standard atmospheric pressure (default is 1.013 bar).

verbose (bool): Verbosity flag for printing detailed messages (default is
True).

dir_path (str): Directory path for the working directory (default is '.').

sample (str): Sample name for the data file (default is 'default').
```

Methods:

```
blank_file(blank='blank.xlsx'): Generate a blank sample Excel file for
experimental data entry.

data_etl(name='*', ext='.xlsx'): Load, filter, and combine experimental data
from multiple Excel files.

convert_to_si(df: pd.DataFrame): Convert data columns to SI units based on
their units in the column names.

compute_thermal_resistance(data: pd.DataFrame, T_evaporator_col: str,
T_condenser_col: str, Q_heat_col: str,TR_output_col: str)

compute_gibbs_free_energy(data: pd.DataFrame, T_evaporator_col: str,
T_condenser_col: str, P_bar: str, to_csv: bool): Calculate the change in
Gibbs free energy at given
```

temperatures and pressures.

```
data_chop(data: pd.DataFrame, Tmin: int, Tmax: int, T_col: str,
chop_suggested: bool): Select data within a specified temperature range.

compute_data_stat(data: pd.DataFrame, property: str, to_csv: bool, method:
str, decimals: int): Calculate and save statistical properties (mean,
standard deviation) of the data
```

grouped by a specified property.

```
thermal_property_avg(data: pd.DataFrame, decimals: int): Calculate average
values of measured thermal properties and return them as a list of strings.

best_TP(data: pd.DataFrame, decimals: int, gfe_col: str): Identify optimal
thermal properties based on the minimum change in Gibbs free energy and
return them as a list of strings.
```

Usage:

```
    from PyPulseHeatPipe import PulseHeatPipe

    # Setting up the analysis

    analysis = PulseHeatPipe("dir_path", "sample")

    # Generating a blank sample file

    analysis.blank_file()

    # Performing data ETL

    df, df_conv = analysis.data_etl()

    # Converting data to SI units

    df_si = analysis.convert_to_si(df)

    # Calculating thermal resistance
    df_tr = analysis.compute_thermal_resistance(data: pd.DataFrame,
T_evaporator_col: str, T_condenser_col: str,Q_heat_col: str,TR_output_col:
str)

    # Calculating Gibbs free energy

    gfe_data = analysis.compute_gibbs_free_energy(df)
```

```
# Selecting data within a temperature range

selected_data = analysis.data_chop(df, Tmin=300, Tmax=400)

# Computing statistical properties of the data

df_mean, df_std = analysis.compute_data_stat(df)

# Calculating average values of thermal properties

avg_properties = analysis.thermal_property_avg(df)

# Identifying optimal thermal properties

optimal_properties = analysis.best_TP(df)


# Data Visualization

dv = DataVisualization(dir_path='data/', sample='DI_Water')

# get a interactive dashboard

dv.get_dashboard(data=gfe_data)

# get a custom plots

dv.get_plots(data=df_gfe,
          x_col='Te_mean[K]',
          y_col='dG[KJ/mol]',
          auto_data_chop=True,
          plot_method='combined',
          figsize=(15, 7))
```

Please find more detail at https://github.com/nirmalparmarphd/PyPulseHeatPipe