## PreProcessCAE.py

- **get_pre_processing_input(str, bool)** -> tuple

==========

- create_nodes_dataframe(str) -> pd.DataFrame
- create_elements_dataframe(str) -> pd.DataFrame
- create_materials_dataframe(str) -> pd.DataFrame
- create_properties_dataframe(str) -> pd.DataFrame
- create_loads_dataframe(str) -> pd.DataFrame
- create_boundary_conditions_dataframe(str) -> pd.DataFrame

==========

- print_all_dataframes(pd.DataFrame, ...) -> None
- print_dataframe_configuration(pd.DataFrame, str) -> None
- fix_scientific_notation(str) -> str

## class InterfaceCAE

- create_preprocess_frame(*self*) -> None
- create_solver_frame(*self*) -> None
- create_postprocess_frame(*self*) -> None
- select_file_button(*self*) -> None
- solve_button(*self*) -> None
- plot_mesh_button(*self*) -> None
- plot_displacement_button(*self*) -> None
- plot_stress_button(*self*) -> None

## PostProcessCAE.py

- **get_post_processing_output**(np.array, np.array, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame) -> pd.DataFrame
- create_displacement_dataframe(np.array, pd.DataFrame, pd.DataFraeme) -> pd.DataFrame
- create_forces_dataframe(np.array, pd.DataFrame, pd.DataFrame) -> pd.DataFrame
- create_stress_dataframe(np.array, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame) -> pd.DataFrame
- create_strain_dataframe(np.array, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame) -> pd.DataFrame
- compute_element_stress(FiniteElement, np.array) -> tuple
- compute_element_strain(FiniteElement, np.array, pd.DataFrame, tuple) -> np.array
- compute_von_mises_stress(np.array) -> float
- save_results_to_csv(str, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame) -> None
- create_mesh(pd.DataFrame, pd.DataFrame) -> None
- plot_mesh(pd.DataFrame, pd.DataFrame) -> None
- plot_results_displacement(pd.DataFrame, pd.DataFrame, pd.DataFrame) -> None
- plot_results_stress(pd.DataFrame, pd.DataFrame, pd.DataFrame) -> None

## SolverCAE.py

- **solve_system**(np.array, np.array, np.array) -> tuple
- **assemble_global_stiffness_matrix**(pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame) -> np.array
- **assemble_global_load_vector**(str, pd.DataFrame, pd.DataFrame) -> np.array

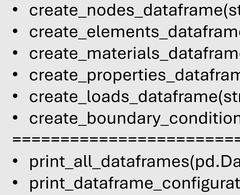
==========

- **impose_boundary_conditions_penalty_method**(str, pd.DataFrame, pd.DataFrame, np.array, np.array) -> tuple
- get_boundary_condition_dofs(str, pd.DataFrame, pd.DataFrame) -> np.array
- set_fixed_boundary_conditions(np.array, np.array, np.array) -> tuple

## class FiniteElement

- __init__(*self*, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- get_node_coordinates(*self*, pd.DataFrame) -> np.array
- get_global_index_dofs(*self*, pd.DataFrame) -> np.array

# 1D Elements




- **2 nodes**
- **6 DOFs per node**

### class CBAR(FiniteElement)

- __init__(*self*, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix(*self*) -> np.array**
- compute_local_stiffness_matrix_cbar(*self*) -> np.array
- compute_transformation_matrix(*self*) -> np.array




- **2 nodes**
- **2 DOFs per node**

### class Beam_Axial

- __init__(*self*, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix(*self*) -> np.array**
- compute_local_stiffness_matrix_cbar(*self*) -> np.array
- compute_transformation_matrix(*self*) -> np.array




- **2 nodes**
- **2 DOFs per node**

### class Beam_Torsional

- __init__(*self*, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix(*self*) -> np.array**
- compute_local_stiffness_matrix_cbar(*self*) -> np.array
- compute_transformation_matrix(*self*) -> np.array


- **2 nodes**
- **3 DOFs per node**

### class Beam_Flexural

- __init__(*self*, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix(*self*) -> np.array**
- compute_local_stiffness_matrix_cbar(*self*) -> np.array
- compute_transformation_matrix(*self*) -> np.array

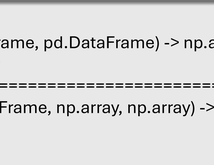
# 2D Elements

### class Element2D(FiniteElement)

- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- assemble_jacobian_matrix(*self*, float, float) -> np.array
- assemble_B_matrix(*self*, float, float) -> np.array
- assemble_D_matrix(*self*, str) -> np.array
- assemble_G_matrix(*self*, np.array, float) -> np.array
- assemble_P_matrix(*self*, np.array, np.array) -> np.array

### class CTRIA3(Element2D)

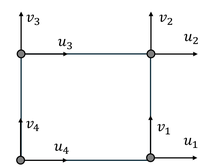
- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix**(*self*) -> np.array
- compute_quadrature(*self*) -> tuple
- compute_shape_functions(*self*, float, float) -> np.array
- compute_shape_functions_derivatives(*self*, float, float) -> tuple




- **3 nodes**
- **2 DOFs per node**

### class CQUAD4(Element2D)

- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix**(*self*) -> np.array
- compute_quadrature(*self*) -> tuple
- compute_shape_functions(*self*, float, float) -> np.array
- compute_shape_functions_derivatives(*self*, float, float) -> tuple




- **4 nodes**
- **2 DOFs per node**

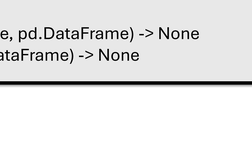
# 3D Elements

### Class Element3D(FiniteElement)

- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- *assemble_jacobian_matrix*(*self*, float, float, float) -> np.array
- assemble_B_matrix(*self*, float, float, float) -> np.array
- assemble_D_matrix(*self*) -> np.array
- assemble_G_matrix(*self*, np.array) -> np.array
- assemble_P_matrix(*self*, np.array, np.array, np.array) -> np.array

### class CTETRA

- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix**(*self*) -> np.array
- quadrature(*self*) -> tuple
- compute_shape_functions(*self*, float, float, float) -> np.array
- compute_shape_functions_derivatives(*self*, float, float, float) -> tuple




- **4 nodes**
- **3 DOFs per node**

### class CHEXA

- __init__(self, pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame)
- **assemble_stiffness_matrix**(*self*) -> np.array
- quadrature(*self*) -> tuple
- compute_shape_functions(*self*, float, float, float) -> np.array
- compute_shape_functions_derivatives(*self*, float, float, float) -> tuple




- **8 nodes**
- **3 DOFs per node**