**Imports:**

**from ..misc.report import get\_kwargs**

**from .eda\_clean import eda\_clean**

**from .eda\_peaks import eda\_peaks**

**from .eda\_phasic import eda\_phasic**

* from ..misc.report import get\_kwargs: Imports a specific helper function called get\_kwargs. Its job is to figure out which settings belong to which function.
* from .eda\_clean import eda\_clean, from .eda\_peaks import eda\_peaks, from .eda\_phasic import eda\_phasic: Imports the "worker" functions.

**Function Definition:**

**def eda\_methods(sampling\_rate=1000, method="default",**

**method\_cleaning="default", method\_peaks="default",**

**method\_phasic="default", \*\*kwargs):**

This defines the eda\_methods function and its parameters.

**Parameters:**

* **method="default"**: A general, high-level choice. It's a shortcut. If you don't specify the details, this method will be used for everything.
* **method\_cleaning="default", method\_peaks="default", method\_phasic="default"**: These allow you to be more specific. You can use the "default" method for most things but choose a special method for cleaning, for example.
* **\*\*kwargs**: A "catch-all" for any other settings you want to provide. For instance, you might want to specify a threshold=0.1 for peak detection. This \*\*kwargs will catch it.

“”” EDA Processing Methods”””

This is the docstring, a detailed comment explaining what the function does, its parameters, and what it returns. It is the function's instruction manual.

# Sanitize Inputs

method\_cleaning = str(method).lower() if method\_cleaning == "default" \

else str(method\_cleaning).lower()

method\_phasic = str(method).lower() if method\_phasic == "default" \

else str(method\_phasic).lower()

method\_peaks = str(method).lower() if method\_peaks == "default" \

else str(method\_peaks).lower()

* **if method\_cleaning == "default":**: This checks if you provided a specific cleaning method.
* **...str(method).lower()**: If you don't, it takes the general method choice (e.g., "biosppy"), converts it to a string, and makes it all lowercase.
* **else str(method\_cleaning).lower()**: If you did provide a specific method\_cleaning, it simply takes that choice and makes it lowercase.
* This ensures that no matter how you capitalize your input (e.g., "Biosppy", "biosppy", "BIOSPPY"), the program will treat it the same.

# Get arguments to be passed to underlying functions

kwargs\_cleaning, report\_info = get\_kwargs(report\_info, eda\_clean)

kwargs\_phasic, report\_info = get\_kwargs(report\_info, eda\_phasic)

kwargs\_peaks, report\_info = get\_kwargs(report\_info, eda\_peaks)

* **get\_kwargs(report\_info, eda\_clean)**: This helper function looks at the eda\_clean function and asks, "what settings do you accept?". It then looks in the report\_info["kwargs"] bag and pulls out only those settings.
* **kwargs\_cleaning...**: It returns a new, smaller dictionary (kwargs\_cleaning) containing only the settings relevant to eda\_clean.

# Save keyword arguments in dictionary

report\_info["kwargs\_cleaning"] = kwargs\_cleaning

report\_info["kwargs\_phasic"] = kwargs\_phasic

report\_info["kwargs\_peaks"] = kwargs\_peaks

# References

report\_info[“references”] = list(np.unique(refs))

* **np.unique(refs)**: If you used two methods that both cited the same paper, it would be in the refs list twice. np.unique removes any duplicates.
* **list(...)**: Converts the result back to a standard list.
* The unique list of references is saved in the report\_info dictionary.

my\_plan = eda\_methods(sampling\_rate=500, method\_cleaning="biosppy",

method\_peaks="gamboa2008", method\_phasic="neurokit",

window\_size=1000) # This is an extra setting for the cleaning method