**eda\_fixpeaks.py**

This function serves two main purposes:

1. **Standardizing the inputs (Paperwork):** It takes the result from findpeaks (the previous step), which could be in a few different formats, and makes sure it is organized into a standard, predictable dictionary format.
2. **Being a placeholder:** It sits in the processing pipeline: findpeaks -> fixpeaks -> getfeatures.

In short, it takes the identified peaks and onsets, ensures they are in the right format, and passes them on without (currently) changing any of the values.

**Import:**

**from ..signal.signal\_formatpeaks import \_signal\_formatpeaks\_sanitize```**

**\* This function "sanitizes" the input, meaning it cleans it up and puts it into a standard format.**

**---**

**### \*\*The main `eda\_fixpeaks` function\*\***

**```python**

**def eda\_fixpeaks(peaks, onsets=None, height=None):**

* **peaks**: It can be a dictionary or DataFrame containing the full output from the eda\_findpeaks function.

**"""Correct SCR peaks"""**

* This is a docstring (user manual). It manages the user's expectations and explains the function's current role.

# Format input

peaks, onsets, height = \_eda\_fixpeaks\_retrieve(peaks, onsets, height)

* Calls the helper function \_eda\_fixpeaks\_retrieve to do the "paperwork standardization". The cleaned-up lists are then assigned to the peaks, onsets, and height variables.

# Prepare output

info = {"SCR\_Onsets": onsets, "SCR\_Peaks": peaks, "SCR\_Height": height}

return info

* It returns the standardized dictionary.

**\_eda\_fixpeaks\_retrieve(peaks, onsets=None, height=None)**

* This defines the helper function that eda\_fixpeaks calls.

original\_input = peaks

* This is important because it will search inside this variable if onsets or height aren't provided separately.

peaks = \_signal\_formatpeaks\_sanitize(original\_input, key="Peaks")

* This is the first call to the "sanitizer". It tells it, "look inside original\_input for something with a key related to 'Peaks' (e.g., 'SCR\_Peaks') and give me just the numbers." The result overwrites the peaks variable.

if onsets is None:

onsets = \_signal\_formatpeaks\_sanitize(original\_input, key="Onsets")

* This checks if you provided the onsets list separately.
* If you did not (onsets is None), it assumes the onset information must be inside the original\_input and uses the sanitizer to find and extract it.

if height is None:

height = \_signal\_formatpeaks\_sanitize(original\_input, key="Height")

Same as above

return peaks, onsets, height

* It returns the three cleaned-up, standardized lists of numbers.

**Example**

**Raw output from eda\_findpeaks:**

initial\_info = {

"SCR\_Peaks": [150, 450, 800],

"SCR\_Onsets": [100, 400, 750],

"SCR\_Height": [1.08, 1.2, 0.97],

"Some\_other\_info": "blah"

}

**Call to eda\_fixpeaks:**

**fixed\_info = eda\_fixpeaks(initial\_info)**

1. \_eda\_fixpeaks\_retrieve is called with initial\_info.
2. The sanitizer looks inside initial\_info with key="Peaks" and extracts [150, 450, 800]. This becomes the peaks variable.
3. Since onsets was None, the sanitizer looks inside initial\_info with key="Onsets" and extracts [100, 400, 750]. This becomes the onsets variable.
4. Since height was None, the sanitizer looks inside initial\_info with key="Height" and extracts [1.08, 1.2, 0.97]. This becomes the height variable.
5. The main eda\_fixpeaks function receives these three clean lists.
6. It does no corrections.
7. It packages them into a new dictionary:

{

"SCR\_Onsets": [100, 400, 750],

"SCR\_Peaks": [150, 450, 800],

"SCR\_Height": [1.08, 1.2, 0.97]

}

**eda\_findpeaks.py**

**Purpose:** The purpose is to look at the phasic EDA signal (waves) and find the locations of potential SCRs.