Copilot

Yes, you're absolutely right. It's a good practice to avoid repeating the same string or error message. You can define a custom exception class for this purpose. Here's how you can do it:

```
class InvalidKeyError(ValueError):
    """Exception raised for errors in the input.

    Attributes:
        message -- explanation of the error
    """

    def __init__(self):
        self.message = "Invalid key in params. Only 'T' and 'D' are allowed."
        super().__init__(self.message)

class MyClass:
    def __init__(self, params: dict = {"T": 1.0, "D": 0.5}) -> None:
        self._params: dict = params

    @property
    def params(self) -> dict:
        """ Get or set the parameters """
        return self._params

    @params.setter
    def params(self, params) -> None:
        if isinstance(params, dict):
            if set(params.keys()).issubset({"T", "D"}):
                self._params = params
            else:
                raise InvalidKeyError()
        elif isinstance(params, tuple) and len(params) == 2:
            key, value = params
            if key in {"T", "D"}:
                self._params[key] = value
            else:
                raise InvalidKeyError()
        else:
            raise ValueError("params must be a dict or a tuple of length 2")
```

Now, `InvalidKeyError` will be raised whenever an invalid key is used, and the error message is defined in one place, making it easier to manage and modify if needed. This is a more elegant and maintainable approach. Good catch! 👍