Copilot

You can make your exception more flexible by passing the allowed keys as a parameter to your `InvalidKeyError` class. 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, allowed_keys: set):
        self._message = f"Invalid key in params. Only {allowed_keys} are allowed."
        super().__init__(self.message)

    @property
    def message(self):
        return self._message
```

And in your `MyClass`, you can define the allowed keys as a class attribute:

```
from typing import Dict, Tuple, Union

class MyClass:
    ALLOWED_KEYS = {"T", "D"}

    def __init__(self, params: Dict[str, float] = {"T": 1.0, "D": 0.5}) -> None:
        self._params: Dict[str, float] = params

    @property
    def params(self) -> Dict[str, float]:
        """ Get or set the parameters """
        return self._params

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

Now, if you want to change the allowed keys, you just need to update the `ALLOWED_KEYS` attribute in `MyClass`. The error message in `InvalidKeyError` will automatically adapt to reflect the new allowed keys. This makes your code more flexible and easier to maintain. Good question! 👍