Copilot

In Python, type annotations and type hints are essentially the same thing. They both serve the purpose of indicating the expected type of a variable, function argument, or return value. The `typing` module in Python 3.5 and onwards provides a standard way to use these type hints.

Here's an example of how you might use the `typing` module for type hints:

```python
from typing import Dict, Tuple

class MyClass:
    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({"T", "D"}):
                self._params = params
            else:
                raise ValueError("Invalid keys in params. Only 'T' and 'D' are allowed.")
        elif isinstance(params, tuple) and len(params) == 2:
            key, value = params
            if key in {"T", "D"}:
                self._params[key] = value
            else:
                raise ValueError("Invalid key in params. Only 'T' and 'D' are allowed.")
        else:
            raise ValueError("params must be a dict or a tuple of length 2")
```

In this code, `Dict[str, float]` is a type hint that indicates a dictionary with string keys and float values, and `Union[Dict[str, float], Tuple[str, float]]` is a type hint that indicates a value that can be either a dictionary with string keys and float values, or a tuple containing a string and a float.

Using the `typing` module for type hints can make your code more self-documenting and can help catch certain types of errors earlier in the development process. However, it's worth noting that Python's type hints are optional and are not enforced at runtime. They are mainly intended as a form of documentation and are most beneficial in large codebases and when working with a team.