python注释风格

VsCode插件

[autoDocstring - Python Docstring Generator - Visual Studio Marketplace]

vscode中的autoDocstring插件提供了这几种风格

> docblockr、pep257、google、google-notypes、sphinx、sphinx-notpyes、numpy、numpy-notypes、one-line-sphinx

安装完成后进入设置搜索Auto Docstring

Docstring Format选项选择注释风格

PyCharm

PyCharm中提供了这几种风格

> Epytext、reStructuredText、Numpy、Google

设置位置

> File -> Settings -> Tools -> Python Integrated Tools -> Docstrings -> Docstring format

PyCharm默认不开启参数类型注释，开启类型的设置位置

> File -> Settings -> Editor -> General -> Smart Keys -> Insert type placeholders in the documentation comment stub

其中，autoDocstring的sphinx 和 pycharm的reStructuredText有些像

注释的自动生成

直接在类、方法内的第一行敲三个双引号Enter即可

注释风格的选择

考虑到算法开发常用的IDE：VsCode和PyCharm，建议使用numpy、sphinx/reStructuredText、google这三种中的一种

常见的注释风格

docblockr-autoDocstring

```python
class PyClass(object):
    """_summary_

    Arguments:
        object {_type_} -- _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Arguments:
            parm1 {_type_} -- _description_

        Keyword Arguments:
            parm2 {bool} -- _description_ (default: {True})

        Returns:
            str -- _description_
        """
        pass
```

pep257-autoDocstring

```python
class PyClass(object):
    """_summary_

    Arguments:
        object -- _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Arguments:
            parm1 -- _description_

        Keyword Arguments:
            parm2 -- _description_ (default: {True})

        Returns:
            _description_
        """
        pass
```

google-autoDocstring

```
class PyClass(object):
    """_summary_

    Args:
        object (_type_): _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Args:
            parm1 (_type_): _description_
            parm2 (bool, optional): _description_. Defaults to True.

        Returns:
            str: _description_
        """
        pass
```

google-notypes-autoDocstring

```
class PyClass(object):
    """_summary_

    Args:
        object: _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Args:
            parm1: _description_
            parm2: _description_. Defaults to True.

        Returns:
            _description_
        """
        pass
```

sphinx-autoDocstring

```
class PyClass(object):
    """_summary_

    :param object: _description_
    :type object: _type_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        :param parm1: _description_
        :type parm1: _type_
        :param parm2: _description_, defaults to True
        :type parm2: bool, optional
        :return: _description_
        :rtype: str

        """
        pass
```

sphinx-notpyes-autoDocstring

```
class PyClass(object):
    """_summary_

    :param object: _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        :param parm1: _description_
        :param parm2: _description_, defaults to True
        :return: _description_
        """
        pass
```

numpy-autoDocstring

```
class PyClass(object):
    """_summary_

    Parameters
    ----------
    object : _type_
        _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Parameters
        ----------
        parm1 : _type_
            _description_
        parm2 : bool, optional
            _description_, by default True

        Returns
        -------
        str
            _description_
        """
        pass
```

numpy-notypes-autoDocstring

```
class PyClass(object):
    """_summary_

    Parameters
    ----------
    object
        _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        Parameters
        ----------
        parm1
            _description_
        parm2, optional
            _description_, by default True

        Returns
        -------
            _description_
        """
        pass
```

one-line-sphinx-autoDocstring

```
class PyClass(object):
    """_summary_

    :param _type_ object: _description_
    """

    def __init__(self) -> None:
        """_summary_
        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """_summary_

        :param _type_ parm1: _description_
        :param bool parm2: _description_, defaults to True
        :return str: _description_
        """
        pass
```

epytext-PyCharm

```
class PyClass(object):
    """

    """

    def __init__(self) -> None:
        """

        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """

        @param parm1:
        @type parm1:
        @param parm2:
        @type parm2:
```

```
        @return:
        @rtype:
        """
        pass
```

reStructuredText-PyCharm

```
class PyClass(object):
    """

    """

    def __init__(self) -> None:
        """


        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """

        :param parm1:
        :type parm1:
        :param parm2:
        :type parm2:
        :return:
        :rtype:
        """
        pass
```

numpy-PyCharm

```
class PyClass(object):
    """

    """

    def __init__(self) -> None:
        """


        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """

        Parameters
        ----------
        parm1 :
        parm2 :

        Returns
        -------

        """
        pass
```

google- PyCharm

```
class PyClass(object):
    """

    """

    def __init__(self) -> None:
        """

        """
        pass

    def PyMethod(self, parm1, parm2=True) -> str:
        """

        Args:
            parm1 ():
            parm2 ():

        Returns:

        """
        pass
```