🐰 Python 5大記法完全ガイド

ふわふわ大福店のうさうさ店長で学ぶ、実務で使えるコメント・docstringの書き方

📚 目次

1. 5大記法とは
2. Googleスタイル
3. NumPyスタイル
4. reStructuredTextスタイル
5. Epytextスタイル
6. 型ヒント方式
7. 完全比較表
8. チートシート
9. 同じコードを5つの記法で

---

1. 5大記法とは

📖 Python docstringの5つの主要スタイル

#	記法	読み方	主な使用者	推奨度
1	**Google**	グーグル	Google、多くの企業	★★★★★
2	**NumPy**	ナムパイ	NumPy、SciPy、科学計算	★★★★☆
3	**reStructuredText**	リストラクチャードテキスト	Python公式、Sphinx	★★★☆☆
4	**Epytext**	エピテキスト	Epydoc（古い）	★☆☆☆☆
5	**型ヒント**	かたヒント	最新のPython	★★★★★

🎯 どれを使うべき？

新しいプロジェクト → Googleスタイル + 型ヒント
科学計算・データ分析 → NumPyスタイル
Sphinxドキュメント → reStructuredText
既存プロジェクト → プロジェクトの規約に従う

---

2. Googleスタイル

📖 特徴

- **最も読みやすい**
- **シンプルで覚えやすい**
- **業界標準**
- **VSCodeなどIDEのサポートが充実**

💻 基本構文

```python
def function_name(arg1, arg2, arg3=None):
    """
    1行要約（動詞で始める）

    詳細な説明（複数行可）

    Args:
        arg1 (型): 説明
        arg2 (型): 説明
        arg3 (型, optional): 説明。デフォルトはNone

    Returns:
        型: 説明

    Raises:
        例外名: 条件

    Example:
        >>> function_name(1, 2)
        結果
    """
    pass
```

🐰 ふわふわ大福店の例

```python
```

```python
class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes:
        owner_name (str): 店長の名前
        stock (int): 在庫数
        sold (int): 累計販売数

    Example:
        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        True
    """

    def __init__(self, owner_name, stock):
        """
        大福店を初期化する

        Args:
            owner_name (str): 店長の名前
            stock (int): 初期在庫数

        Raises:
            ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0

    def sell(self, quantity, price=150, discount=0):
        """
        大福を販売する

        指定された個数の大福を販売し、売上を計算します。

        Args:
            quantity (int): 販売個数
            price (int, optional): 単価。デフォルトは150円
            discount (float, optional): 割引率（0.0-1.0）。デフォルトは0
```

```
    Returns:
      int: 売上金額（割引後）
      None: 在庫不足の場合

    Raises:
      ValueError: quantityが0以下の場合

    Example:
      >>> shop = DaifukuShop("うさうさ", 20)
      >>> shop.sell(5)
      750
      >>> shop.sell(3, discount=0.1)
      405

    Note:
      在庫不足の場合はNoneを返し、在庫は変更されません。
    """
    if quantity <= 0:
      raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
      return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity

    return total

  def restock(self, quantity):
    """
    在庫を補充する

    Args:
      quantity (int): 補充個数

    Raises:
      ValueError: quantityが0以下の場合
    """
    if quantity <= 0:
      raise ValueError("補充個数は1以上にしてください")

    self.stock += quantity
```

✅ Googleスタイル チートシート

```
セクション名　使い方
────────────────────────────────────────────────────────────

Args:        引数の説明
  arg (型): 説明
  arg (型, optional): 説明。デフォルトは値

Returns:     返り値の説明
  型: 説明

Yields:      yield文の説明（ジェネレーター）
  型: 説明

Raises:      例外の説明
  例外名: 条件

Example:     使用例
  >>> コード
  結果

Note:        注意事項
  説明

Warning:     警告
  説明

See Also:    関連項目
  関数名: 説明
```

---

3. NumPyスタイル

📖 特徴

- **科学計算向け**
- **詳細な説明に適している**
- **NumPy、SciPy、pandasで使用**
- **アンダーラインで区切る**

💻 基本構文

```
python
```

```
def function_name(arg1, arg2, arg3=None):
    """
    1行要約

    詳細な説明

    Parameters
    ----------
    arg1 : 型
        説明
    arg2 : 型
        説明
    arg3 : 型, optional
        説明（デフォルト: None）

    Returns
    -------
    型
        説明

    Raises
    ------
    例外名
        条件

    Examples
    --------
    >>> function_name(1, 2)
    結果
    """
    pass
```

🐰 ふわふわ大福店の例

python

```
class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes
    ----------
    owner_name : str
        店長の名前
    stock : int
        在庫数
    sold : int
        累計販売数

    Examples
    --------
    >>> shop = DaifukuShop("うさうさ", 20)
    >>> shop.sell(5)
    True
    """

    def __init__(self, owner_name, stock):
        """
        大福店を初期化する

        Parameters
        ----------
        owner_name : str
            店長の名前
        stock : int
            初期在庫数

        Raises
        ------
        ValueError
            stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0

    def sell(self, quantity, price=150, discount=0):
```

```
    """
    大福を販売する

    指定された個数の大福を販売し、売上を計算します。

    Parameters
    ----------
    quantity : int
        販売個数
    price : int, optional
        単価（デフォルト: 150円）
    discount : float, optional
        割引率 0.0-1.0（デフォルト: 0）

    Returns
    -------
    int or None
        売上金額（割引後）。在庫不足の場合はNone

    Raises
    ------
    ValueError
        quantityが0以下の場合

    Examples
    --------
    >>> shop = DaifukuShop("うさうさ", 20)
    >>> shop.sell(5)
    750
    >>> shop.sell(3, discount=0.1)
    405

    Notes
    -----
    在庫不足の場合はNoneを返し、在庫は変更されません。

    See Also
    --------
    restock : 在庫を補充する
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
```

```
        self.stock -= quantity
        self.sold += quantity

        return total

    def restock(self, quantity):
        """
        在庫を補充する

        Parameters
        ----------
        quantity : int
            補充個数

        Raises
        ------
        ValueError
            quantityが0以下の場合
        """
        if quantity <= 0:
            raise ValueError("補充個数は1以上にしてください")

        self.stock += quantity
```

✅ NumPyスタイル チートシート

```
セクション名　書き方
---------------------------------------------------------

Parameters　引数の説明
----------
arg : 型
    説明
arg : 型, optional
    説明（デフォルト: 値）

Returns　返り値の説明
-------
型
    説明

Yields　yield文の説明
------
型
    説明

Raises　例外の説明
```

```
------
例外名
    条件

Examples      使用例
--------
>>> コード
結果

Notes         注意事項
-----
説明

Warnings      警告
--------
説明

See Also      関連項目
--------
関数名 : 説明
```

---

4. reStructuredTextスタイル

📖 特徴

- **Sphinx標準**
- **Python公式ドキュメントで使用**
- **`:param:`, `:type:`, `:return:` などのフィールド**
- **詳細だが冗長**

💻 基本構文

```python
```

```
def function_name(arg1, arg2, arg3=None):
    """
    1行要約

    詳細な説明

    :param arg1: 説明
    :type arg1: 型
    :param arg2: 説明
    :type arg2: 型
    :param arg3: 説明（デフォルト: None）
    :type arg3: 型
    :return: 説明
    :rtype: 型
    :raises 例外名: 条件

    .. code-block:: python

        >>> function_name(1, 2)
        結果
    """
    pass
```

ふわふわ大福店の例

```
python
```

```
class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    :ivar owner_name: 店長の名前
    :vartype owner_name: str
    :ivar stock: 在庫数
    :vartype stock: int
    :ivar sold: 累計販売数
    :vartype sold: int

    .. code-block:: python

        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        True
    """

    def __init__(self, owner_name, stock):
        """
        大福店を初期化する

        :param owner_name: 店長の名前
        :type owner_name: str
        :param stock: 初期在庫数
        :type stock: int
        :raises ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0

    def sell(self, quantity, price=150, discount=0):
        """
        大福を販売する

        指定された個数の大福を販売し、売上を計算します。

        :param quantity: 販売個数
        :type quantity: int
        :param price: 単価（デフォルト: 150円）
```

```
    :type price: int
    :param discount: 割引率 0.0-1.0（デフォルト: 0）
    :type discount: float
    :return: 売上金額（割引後）。在庫不足の場合はNone
    :rtype: int or None
    :raises ValueError: quantityが0以下の場合

    .. code-block:: python

       >>> shop = DaifukuShop("うさうさ", 20)
       >>> shop.sell(5)
       750
       >>> shop.sell(3, discount=0.1)
       405

    .. note::
       在庫不足の場合はNoneを返し、在庫は変更されません。

    .. seealso::
       :func:`restock` - 在庫を補充する
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity

    return total

def restock(self, quantity):
    """
    在庫を補充する

    :param quantity: 補充個数
    :type quantity: int
    :raises ValueError: quantityが0以下の場合
    """
    if quantity <= 0:
        raise ValueError("補充個数は1以上にしてください")

    self.stock += quantity
```

✅ reStructuredTextスタイル チートシート

フィールド　　書き方

:param 名前:　　引数の説明
:type 名前:　　引数の型
:return:　　返り値の説明
:rtype:　　返り値の型
:raises 例外名:　例外の説明
:ivar 名前:　　インスタンス変数
:vartype 名前:　変数の型
:cvar 名前:　　クラス変数

ディレクティブ　書き方

.. note::　　注意事項
.. warning::　　警告
.. code-block::　コードブロック
.. seealso::　　関連項目

---

5. Epytextスタイル

📖 特徴

- **Epydoc専用（古い）**
- **@記号を使う**
- **現在はあまり使われない**
- **レガシーコードで見かける**

💻 基本構文

python

```
def function_name(arg1, arg2, arg3=None):
    """
    1行要約

    詳細な説明

    @param arg1: 説明
    @type arg1: 型
    @param arg2: 説明
    @type arg2: 型
    @param arg3: 説明（デフォルト: None）
    @type arg3: 型
    @return: 説明
    @rtype: 型
    @raise 例外名: 条件
    """
    pass
```

ふわふわ大福店の例

python

```
class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    @ivar owner_name: 店長の名前
    @type owner_name: str
    @ivar stock: 在庫数
    @type stock: int
    @ivar sold: 累計販売数
    @type sold: int
    """

    def __init__(self, owner_name, stock):
        """
        大福店を初期化する

        @param owner_name: 店長の名前
        @type owner_name: str
        @param stock: 初期在庫数
        @type stock: int
        @raise ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0

    def sell(self, quantity, price=150, discount=0):
        """
        大福を販売する

        指定された個数の大福を販売し、売上を計算します。

        @param quantity: 販売個数
        @type quantity: int
        @param price: 単価（デフォルト: 150円）
        @type price: int
        @param discount: 割引率 0.0-1.0（デフォルト: 0）
        @type discount: float
        @return: 売上金額（割引後）。在庫不足の場合はNone
        @rtype: int or None
        @raise ValueError: quantityが0以下の場合
```

```
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity

    return total
```

✅ Epytextスタイル チートシート

```
タグ          書き方
──────────────────────────────────────────────────────────
@param 名前:     引数の説明
@type 名前:      引数の型
@return:         返り値の説明
@rtype:          返り値の型
@raise 例外名:    例外の説明
@ivar 名前:      インスタンス変数
@cvar 名前:      クラス変数
@note:           注意事項
@warning:        警告
@see:            関連項目
```

---

6. 型ヒント方式

📖 特徴

- **Python 3.5以降の標準**
- **コード自体に型情報を記述**
- **最も現代的**
- **IDEの補完が最強**
- **mypy等の型チェッカーで検証可能**

💻 基本構文

```
python
```

```python
from typing import Optional, List, Dict

def function_name(
    arg1: int,
    arg2: str,
    arg3: Optional[float] = None
) -> int:
    """
    1行要約

    詳細な説明（型情報は関数定義に書くので、docstringは簡潔に）

    Args:
        arg1: 説明（型は省略可）
        arg2: 説明
        arg3: 説明。デフォルトはNone

    Returns:
        説明

    Raises:
        ValueError: 条件
    """
    pass
```

🐰 ふわふわ大福店の例（完全版）

```python
```

```python
from typing import Optional, Dict, List

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes:
        owner_name: 店長の名前
        stock: 在庫数
        sold: 累計販売数
    """

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        Args:
            owner_name: 店長の名前
            stock: 初期在庫数

        Raises:
            ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name: str = owner_name
        self.stock: int = stock
        self.sold: int = 0

    def sell(
        self,
        quantity: int,
        price: int = 150,
        discount: float = 0
    ) -> Optional[int]:
        """
        大福を販売する

        指定された個数の大福を販売し、売上を計算します。

        Args:
            quantity: 販売個数
            price: 単価。デフォルトは150円
```

```
        discount: 割引率（0.0-1.0）。デフォルトは0

    Returns:
        売上金額（割引後）。在庫不足の場合はNone

    Raises:
        ValueError: quantityが0以下の場合

    Example:
        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total: int = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity

    return total

def restock(self, quantity: int) -> None:
    """
    在庫を補充する

    Args:
        quantity: 補充個数

    Raises:
        ValueError: quantityが0以下の場合
    """
    if quantity <= 0:
        raise ValueError("補充個数は1以上にしてください")

    self.stock += quantity

def get_stats(self) -> Dict[str, any]:
    """
    統計情報を取得する

    Returns:
        店舗の統計情報を含む辞書
    """
```

```
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold
    }

def get_sales_history(self) -> List[Dict[str, int]]:
    """
    販売履歴を取得する（サンプル）

    Returns:
        販売履歴のリスト
    """
    return [
        {'quantity': 5, 'revenue': 750},
        {'quantity': 3, 'revenue': 450}
    ]
```

✅ 型ヒント チートシート

python

```python
from typing import (
    List, Dict, Set, Tuple,
    Optional, Union, Any,
    Callable, Iterator, Generator
)

# 基本型
def func(x: int, y: str, z: float) -> bool:
    pass

# リスト・辞書
def func(items: List[str]) -> Dict[str, int]:
    pass

# Optional（Noneの可能性）
def func(x: Optional[int] = None) -> Optional[str]:
    pass

# Union（複数の型）
def func(x: Union[int, str]) -> Union[bool, None]:
    pass

# タプル
def func() -> Tuple[int, str, float]:
    return 1, "a", 1.0

# 関数型
def func(callback: Callable[[int, str], bool]) -> None:
    pass

# Any（任意の型）
def func(x: Any) -> Any:
    pass
```

---

7. 完全比較表

📊 5大記法の比較

項目	Google	NumPy	reST	Epytext	型ヒント
読みやすさ	★★★★★	★★★★☆	★★★☆☆	★★☆☆☆	★★★★★
書きやすさ	★★★★★	★★★☆☆	★★☆☆☆	★★☆☆☆	★★★★☆
IDE補完	★★★★☆	★★★☆☆	★★★☆☆	★★☆☆☆	★★★★★
型チェック	❌	❌	❌	❌	✅

項目	Google	NumPy	reST	Epytext	型ヒント
学習コスト	低	中	中	中	中
ドキュメント生成	✅	✅	✅	✅	✅
推奨度	★★★★★	★★★★☆	★★★☆☆	★☆☆☆☆	★★★★★

📝 同じ関数を5つのスタイルで比較

python

```python
# ==========================================
# 1. Googleスタイル
# ==========================================
def calculate_total(quantity: int, price: int = 150) -> int:
    """
    合計金額を計算する

    Args:
        quantity (int): 個数
        price (int, optional): 単価。デフォルトは150円

    Returns:
        int: 合計金額
    """
    return quantity * price


# ==========================================
# 2. NumPyスタイル
# ==========================================
def calculate_total(quantity: int, price: int = 150) -> int:
    """
    合計金額を計算する

    Parameters
    ----------
    quantity : int
        個数
    price : int, optional
        単価（デフォルト: 150円）

    Returns
    -------
    int
        合計金額
    """
    return quantity * price


# ==========================================
# 3. reStructuredTextスタイル
# ==========================================
def calculate_total(quantity: int, price: int = 150) -> int:
    """
    合計金額を計算する
```

```
    :param quantity: 個数
    :type quantity: int
    :param price: 単価（デフォルト: 150円）
    :type price: int
    :return: 合計金額
    :rtype: int
    """
    return quantity * price


# ========================================
# 4. Epytextスタイル
# ========================================
def calculate_total(quantity: int, price: int = 150) -> int:
    """
    合計金額を計算する

    @param quantity: 個数
    @type quantity: int
    @param price: 単価（デフォルト: 150円）
    @type price: int
    @return: 合計金額
    @rtype: int
    """
    return quantity * price


# ========================================
# 5. 型ヒント方式（docstringは最小限）
# ========================================
def calculate_total(quantity: int, price: int = 150) -> int:
    """
    合計金額を計算する

    Args:
        quantity: 個数
        price: 単価。デフォルトは150円

    Returns:
        合計金額
    """
    return quantity * price
```

8. チートシート

📋 セクション名対応表

セクション	Google	NumPy	reST	Epytext
引数	`Args:`	`Parameters` `----------`	`:param:` `:type:`	`@param:` `@type:`
返り値	`Returns:`	`Returns` `-------`	`:return:` `:rtype:`	`@return:` `@rtype:`
例外	`Raises:`	`Raises` `------`	`:raises:`	`@raise:`
使用例	`Example:`	`Examples` `--------`	`.. code-block::`	（記法なし）
注意	`Note:`	`Notes` `-----`	`.. note::`	`@note:`
警告	`Warning:`	`Warnings` `--------`	`.. warning::`	`@warning:`
参照	`See Also:`	`See Also` `--------`	`.. seealso::`	`@see:`

📋 推奨の組み合わせ

```python
```

```python
# ✅ 最推奨: Googleスタイル + 型ヒント
from typing import Optional

def sell(quantity: int, price: int = 150) -> Optional[int]:
    """
    大福を販売する

    Args:
        quantity: 販売個数
        price: 単価。デフォルトは150円

    Returns:
        売上金額。在庫不足の場合はNone

    Raises:
        ValueError: quantityが0以下の場合
    """
    pass


# ✅ 科学計算なら: NumPyスタイル + 型ヒント
import numpy as np
from typing import Union

def analyze_data(data: np.ndarray) -> Union[float, None]:
    """
    データを分析する

    Parameters
    ----------
    data : np.ndarray
        分析対象のデータ

    Returns
    -------
    float or None
        分析結果。データが不正な場合はNone
    """
    pass
```

---

9. 同じコードを5つの記法で

🐰 完全実装：ふわふわ大福店クラス

📝 記法1: Googleスタイル

```python
```

```python
from typing import Optional, Dict

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes:
        owner_name (str): 店長の名前
        stock (int): 在庫数
        sold (int): 累計販売数
        revenue (int): 累計売上

    Example:
        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
    """

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        Args:
            owner_name (str): 店長の名前
            stock (int): 初期在庫数

        Raises:
            ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0
        self.revenue = 0

    def sell(
        self,
        quantity: int,
        price: int = 150,
        discount: float = 0
    ) -> Optional[int]:
        """
```

```
    大福を販売する

    指定された個数の大福を販売し、売上を計算します。

    Args:
        quantity (int): 販売個数
        price (int, optional): 単価。デフォルトは150円
        discount (float, optional): 割引率（0.0-1.0）。デフォルトは0

    Returns:
        int: 売上金額（割引後）
        None: 在庫不足の場合

    Raises:
        ValueError: quantityが0以下の場合

    Example:
        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
        >>> shop.sell(3, discount=0.1)
        405

    Note:
        在庫不足の場合はNoneを返し、在庫は変更されません。
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity
    self.revenue += total

    return total

def get_stats(self) -> Dict[str, any]:
    """
    統計情報を取得する

    Returns:
        Dict[str, any]: 以下のキーを持つ辞書
            - owner (str): 店長名
            - stock (int): 在庫数
```

```
        - sold (int): 累計販売数
        - revenue (int): 累計売上
    """
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold,
        'revenue': self.revenue
    }
```

📝 記法2: NumPyスタイル

python

```python
from typing import Optional, Dict

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes
    ----------
    owner_name : str
        店長の名前
    stock : int
        在庫数
    sold : int
        累計販売数
    revenue : int
        累計売上

    Examples
    --------
    >>> shop = DaifukuShop("うさうさ", 20)
    >>> shop.sell(5)
    750
    """

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        Parameters
        ----------
        owner_name : str
            店長の名前
        stock : int
            初期在庫数

        Raises
        ------
        ValueError
            stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
```

```
    self.stock = stock
    self.sold = 0
    self.revenue = 0

def sell(
    self,
    quantity: int,
    price: int = 150,
    discount: float = 0
) -> Optional[int]:
    """
    大福を販売する

    指定された個数の大福を販売し、売上を計算します。

    Parameters
    ----------
    quantity : int
        販売個数
    price : int, optional
        単価（デフォルト: 150円）
    discount : float, optional
        割引率 0.0-1.0（デフォルト: 0）

    Returns
    -------
    int or None
        売上金額（割引後）。在庫不足の場合はNone

    Raises
    ------
    ValueError
        quantityが0以下の場合

    Examples
    --------
    >>> shop = DaifukuShop("うさうさ", 20)
    >>> shop.sell(5)
    750
    >>> shop.sell(3, discount=0.1)
    405

    Notes
    -----
    在庫不足の場合はNoneを返し、在庫は変更されません。
    """
    if quantity <= 0:
```

```
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity
    self.revenue += total

    return total

def get_stats(self) -> Dict[str, any]:
    """
    統計情報を取得する

    Returns
    -------
    dict
        店舗の統計情報を含む辞書

        - owner : str
            店長名
        - stock : int
            在庫数
        - sold : int
            累計販売数
        - revenue : int
            累計売上
    """
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold,
        'revenue': self.revenue
    }
```

📝 記法3: reStructuredTextスタイル

```
python
```

```
from typing import Optional, Dict

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    :ivar owner_name: 店長の名前
    :vartype owner_name: str
    :ivar stock: 在庫数
    :vartype stock: int
    :ivar sold: 累計販売数
    :vartype sold: int
    :ivar revenue: 累計売上
    :vartype revenue: int

    .. code-block:: python

        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
    """

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        :param owner_name: 店長の名前
        :type owner_name: str
        :param stock: 初期在庫数
        :type stock: int
        :raises ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0
        self.revenue = 0

    def sell(
        self,
        quantity: int,
        price: int = 150,
```

```
    discount: float = 0
) -> Optional[int]:
    """
    大福を販売する

    指定された個数の大福を販売し、売上を計算します。

    :param quantity: 販売個数
    :type quantity: int
    :param price: 単価（デフォルト: 150円）
    :type price: int
    :param discount: 割引率 0.0-1.0（デフォルト: 0）
    :type discount: float
    :return: 売上金額（割引後）。在庫不足の場合はNone
    :rtype: int or None
    :raises ValueError: quantityが0以下の場合

    .. code-block:: python

        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
        >>> shop.sell(3, discount=0.1)
        405

    .. note::
       在庫不足の場合はNoneを返し、在庫は変更されません。
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity
    self.revenue += total

    return total

def get_stats(self) -> Dict[str, any]:
    """
    統計情報を取得する

    :return: 店舗の統計情報を含む辞書
    :rtype: dict
```

```
    """
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold,
        'revenue': self.revenue
    }
```

📝 記法4: Epytextスタイル

python

```python
from typing import Optional, Dict

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    @ivar owner_name: 店長の名前
    @type owner_name: str
    @ivar stock: 在庫数
    @type stock: int
    @ivar sold: 累計販売数
    @type sold: int
    @ivar revenue: 累計売上
    @type revenue: int
    """

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        @param owner_name: 店長の名前
        @type owner_name: str
        @param stock: 初期在庫数
        @type stock: int
        @raise ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0
        self.revenue = 0

    def sell(
        self,
        quantity: int,
        price: int = 150,
        discount: float = 0
    ) -> Optional[int]:
        """
        大福を販売する

        指定された個数の大福を販売し、売上を計算します。
```

```
    @param quantity: 販売個数
    @type quantity: int
    @param price: 単価（デフォルト: 150円）
    @type price: int
    @param discount: 割引率 0.0-1.0（デフォルト: 0）
    @type discount: float
    @return: 売上金額（割引後）。在庫不足の場合はNone
    @rtype: int or None
    @raise ValueError: quantityが0以下の場合
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity
    self.revenue += total

    return total

def get_stats(self) -> Dict[str, any]:
    """
    統計情報を取得する

    @return: 店舗の統計情報を含む辞書
    @rtype: dict
    """
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold,
        'revenue': self.revenue
    }
```

📝 記法5: 型ヒント方式（docstringは最小限）

```
python
```

```python
from typing import Optional, Dict, Any

class DaifukuShop:
    """
    大福店クラス

    大福の在庫管理と販売を行うクラスです。

    Attributes:
        owner_name: 店長の名前
        stock: 在庫数
        sold: 累計販売数
        revenue: 累計売上
    """

    owner_name: str
    stock: int
    sold: int
    revenue: int

    def __init__(self, owner_name: str, stock: int) -> None:
        """
        大福店を初期化する

        Args:
            owner_name: 店長の名前
            stock: 初期在庫数

        Raises:
            ValueError: stockが負の数の場合
        """
        if stock < 0:
            raise ValueError("在庫数は0以上にしてください")

        self.owner_name = owner_name
        self.stock = stock
        self.sold = 0
        self.revenue = 0

    def sell(
        self,
        quantity: int,
        price: int = 150,
        discount: float = 0
    ) -> Optional[int]:
        """
```

```
    大福を販売する

    指定された個数の大福を販売し、売上を計算します。

    Args:
        quantity: 販売個数
        price: 単価。デフォルトは150円
        discount: 割引率（0.0-1.0）。デフォルトは0

    Returns:
        売上金額（割引後）。在庫不足の場合はNone

    Raises:
        ValueError: quantityが0以下の場合

    Example:
        >>> shop = DaifukuShop("うさうさ", 20)
        >>> shop.sell(5)
        750
    """
    if quantity <= 0:
        raise ValueError("個数は1以上にしてください")

    if quantity > self.stock:
        return None

    total: int = int(quantity * price * (1 - discount))
    self.stock -= quantity
    self.sold += quantity
    self.revenue += total

    return total

def get_stats(self) -> Dict[str, Any]:
    """
    統計情報を取得する

    Returns:
        店舗の統計情報を含む辞書
    """
    return {
        'owner': self.owner_name,
        'stock': self.stock,
        'sold': self.sold,
        'revenue': self.revenue
    }
```

📝 最終チェックリスト

✅ docstring品質チェック

【基本】
☐ 三連引用符（"""）を使っている
☐ 関数/クラスの直後に配置
☐ 1行目に要約がある
☐ 要約は動詞で始まっている

【スタイル選択】
☐ プロジェクトの規約を確認した
☐ 一貫したスタイルを使用
☐ 型ヒントと組み合わせている（推奨）

【内容】
☐ すべての引数を説明している
☐ 返り値を説明している
☐ 例外を説明している（必要に応じて）
☐ 使用例がある（複雑な場合）

【推奨スタイル別】
Googleスタイル:
☐ Args: セクションがある
☐ Returns: セクションがある
☐ 読みやすい形式

NumPyスタイル:
☐ Parameters セクションがある
☐ アンダーラインで区切っている
☐ 科学計算に適した説明

型ヒント:
☐ 関数定義に型情報を記載
☐ docstringは簡潔
☐ mypy等でチェック可能

🎓 実践演習

演習1: この関数を5つの記法で書き換えてください

python

```
def calculate_discount(price, rate):
    if rate < 0 or rate > 1:
        raise ValueError("割引率は0-1の範囲")
    return int(price * (1 - rate))
```

<details> <summary>解答例（Googleスタイル）</summary> ```python def calculate_discount(price: int, rate: float) -> int: """ 割引後の価格を計算する

```
Args:
    price (int): 元の価格
    rate (float): 割引率（0.0-1.0）

Returns:
    int: 割引後の価格

Raises:
    ValueError: rateが0-1の範囲外の場合

Example:
    >>> calculate_discount(1000, 0.1)
    900
"""
if rate < 0 or rate > 1:
    raise ValueError("割引率は0-1の範囲")
return int(price * (1 - rate))
```

```


</details>

---

## 📚 まとめ

### 🎯 推奨事項

#### 新規プロジェクト

✅ Googleスタイル + 型ヒント

- 最も読みやすい
- 業界標準
- IDE補完が強力
```

科学計算プロジェクト

✅ NumPyスタイル + 型ヒント

- NumPy/SciPy/pandasとの親和性
- 詳細な説明に向いている

Sphinxドキュメント

✅ reStructuredTextスタイル

- Sphinx標準
- 自動ドキュメント生成が容易

レガシーコード

⚠️ プロジェクトの既存スタイルに従う

- 一貫性が最重要

💡 ベストプラクティス

```python
# ✅ 推奨: Googleスタイル + 型ヒント
from typing import Optional

def sell(quantity: int, price: int = 150) -> Optional[int]:
    """
    大福を販売する

    Args:
        quantity: 販売個数
        price: 単価。デフォルトは150円

    Returns:
        売上金額。在庫不足の場合はNone
    """
    pass
```

📋 クイックリファレンス

やりたいこと	使うべきスタイル
新規プロジェクト	Google + 型ヒント
データ分析	NumPy + 型ヒント
公式ドキュメント	reStructuredText
シンプルに	型ヒントのみ

🐰 うさうさ店長からのメッセージ:

「5つの記法を紹介しましたが、迷ったら**Googleスタイル + 型ヒント**を使ってください！

重要なのは:

1. 一貫したスタイルを使う
2. すべての引数と返り値を説明する
3. 型情報を明記する
4. チーム全体で同じスタイルを使う

最初は完璧を目指さず、まずは書くことから始めましょう。
徐々に詳しく書けるようになります！」

📎 関連記事:

- docstring完全ガイド
- 型ヒント徹底解説
- Sphinx自動ドキュメント生成
- VSCodeでのdocstring活用