docstring書き方 完全ゼロからスタート 🔰

プログラミング初心者でも5分で理解できる！

🎯 この記事のゴール

- docstringが**何なのか**わかる
- docstringが**書ける**ようになる
- docstringを**確認できる**ようになる

Step 1: docstringって何？

超簡単に言うと...

関数の説明書です！

```python
def add(x, y):
    """2つの数を足し算する。""" ← これがdocstring！
    return x + y
```

なぜ必要？

❌ docstringがないと...

```python
def calc(a, b):
    return a * 1.1 + b
```

誰も何をする関数かわからない！

- aって何？
- bって何？
- 1.1って何？

✅ docstringがあると...

```python
```

```python
def calculate_total(price, shipping):
    """商品代金と送料から合計金額を計算する。

    Args:
        price (int): 商品代金（税込）
        shipping (int): 送料

    Returns:
        float: 合計金額（10%の手数料込み）
    """
    return price * 1.1 + shipping
```

一目でわかる！

- priceは商品代金
- shippingは送料
- 10%の手数料がかかる

Step 2: 最初の一歩【超基本】

ルール1: 三重引用符を使う

```python
def hello():
    """あいさつする。"""  # """を3つ
    print("こんにちは")
```

ルール2: defの直後に書く

```python
def hello():
    """あいさつする。"""  # ← defの次の行
    print("こんにちは")
```

ルール3: 動詞で終わる

```python
```

```
# ❌ 悪い
def hello():
    """あいさつ"""  # 名詞だけ

# ✅ 良い
def hello():
    """あいさつする。"""  # 動詞
```

Step 3: レベルアップ【引数を説明】

引数がない場合

```
python

def get_today():
    """今日の日付を取得する。"""
    import datetime
    return datetime.date.today()
```

これだけでOK！

引数がある場合

```
python

def greet(name):
    """名前を使ってあいさつする。

    Args:
        name (str): あいさつする相手の名前
    """
    return f"こんにちは、{name}さん！"
```

解説:

- `Args:` ← 引数の説明を始めるキーワード
- `name (str):` ← 引数名と型
- `あいさつする相手の名前` ← 何の引数か説明

Step 4: さらにレベルアップ【戻り値を説明】

```
python
```

```python
def add(x, y):
    """2つの数を足し算する。

    Args:
        x (int): 1つ目の数
        y (int): 2つ目の数

    Returns:
        int: 足し算の結果
    """
    return x + y
```

解説:

- `Returns:` ← 戻り値の説明を始めるキーワード
- `int:` ← 返ってくる値の型
- `足し算の結果` ← 何を返すか説明

Step 5: 実践練習【簡単な例】

例1: 掛け算

```python
def multiply(a, b):
    """2つの数を掛け算する。

    Args:
        a (int): 1つ目の数
        b (int): 2つ目の数

    Returns:
        int: 掛け算の結果
    """
    return a * b
```

例2: あいさつ

```python
```

```python
def say_hello(name, age):
    """名前と年齢を使って自己紹介する。

    Args:
        name (str): 名前
        age (int): 年齢

    Returns:
        str: 自己紹介の文章
    """
    return f"私は{name}です。{age}歳です。"
```

例3: 消費税計算

```python
def calc_tax(price, tax_rate=0.1):
    """税込価格を計算する。

    Args:
        price (int): 税抜価格
        tax_rate (float, optional): 消費税率。デフォルトは0.1（10%）

    Returns:
        int: 税込価格
    """
    return int(price * (1 + tax_rate))
```

ポイント:

- `optional` ← 省略可能な引数
- `デフォルトは0.1` ← デフォルト値を説明

Step 6: 確認方法【help()を使う】

確認の仕方

```python
```

```python
# 1. 関数を定義
def add(x, y):
    """2つの数を足し算する。

    Args:
        x (int): 1つ目の数
        y (int): 2つ目の数

    Returns:
        int: 足し算の結果
    """
    return x + y

# 2. helpで確認
help(add)
```

表示結果

```
Help on function add in module __main__:

add(x, y)
    2つの数を足し算する。

    Args:
        x (int): 1つ目の数
        y (int): 2つ目の数

    Returns:
        int: 足し算の結果
```

書いたdocstringがキレイに表示される！

Step 7: よくある間違い【初心者編】

間違い1: 引用符の数

```
python
```

```python
# ❌ 間違い
def wrong():
    "これは間違い"  # "が1個だけ

# ✅ 正しい
def correct():
    """これは正しい"""  # """が3個
```

間違い2: 位置

```python
# ❌ 間違い
"""これは間違い"""  # 関数の外
def wrong():
    pass

# ✅ 正しい
def correct():
    """これは正しい"""  # defの直後
    pass
```

間違い3: 説明が不十分

```python
# ❌ 悪い
def process(data):
    """処理する。"""
    # データって何？何を処理？

# ✅ 良い
def process(user_list):
    """ユーザーリストから重複を削除する。

    Args:
        user_list (list): ユーザー名のリスト

    Returns:
        list: 重複を削除したユーザー名のリスト
    """
```

Step 8: 型の種類【よく使うもの】

基本の型

型	説明	例
`int`	整数	`10`, `-5`, `0`
`float`	小数	`3.14`, `1.5`
`str`	文字列	`"こんにちは"`, `"太郎"`
`bool`	真偽値	`True`, `False`
`list`	リスト	`[1, 2, 3]`, `["a", "b"]`
`dict`	辞書	`{"name": "太郎"}`

使用例

```python
def example(number, text, flag, items):
    """型の例。

    Args:
        number (int): 整数
        text (str): 文字列
        flag (bool): 真偽値
        items (list): リスト
    """
    pass
```

Step 9: テンプレート【コピペ用】

パターンA: 引数なし

```python
def 関数名():
    """何をする関数か1行で説明。"""
    # 処理
    pass
```

パターンB: 引数あり、戻り値なし

```python
```

```python
def 関数名(引数1, 引数2):
    """何をする関数か1行で説明。

    Args:
        引数1 (型): 説明
        引数2 (型): 説明
    """
    # 処理
    pass
```

パターンC: 引数あり、戻り値あり（最もよく使う）

```python
def 関数名(引数1, 引数2):
    """何をする関数か1行で説明。

    Args:
        引数1 (型): 説明
        引数2 (型): 説明

    Returns:
        型: 説明
    """
    # 処理
    return 結果
```

パターンD: オプション引数あり

```python
def 関数名(引数1, 引数2=デフォルト値):
    """何をする関数か1行で説明。

    Args:
        引数1 (型): 説明
        引数2 (型, optional): 説明。デフォルトはデフォルト値

    Returns:
        型: 説明
    """
    # 処理
    return 結果
```

Step 10: 実践問題【やってみよう】

問題1: 割り算

以下の関数にdocstringを書いてください。

```python
def divide(a, b):
    return a / b
```

<details> <summary>答えを見る</summary> ```python def divide(a, b): """2つの数を割り算する。

```
Args:
    a (float): 割られる数
    b (float): 割る数

Returns:
    float: 割り算の結果
"""
return a / b
```

</details>

問題2: リストの合計

以下の関数にdocstringを書いてください。
```python
def sum_list(numbers):
    return sum(numbers)
```

<details> <summary>答えを見る</summary> ```python def sum_list(numbers): """数値リストの合計を計算する。

```
Args:
    numbers (list): 数値のリスト

Returns:
    float: 合計値
"""
return sum(numbers)
```

</details>

問題3: 割引計算

以下の関数にdocstringを書いてください。
```python
def apply_discount(price, discount=0.1):
    return price * (1 - discount)
```

<details> <summary>答えを見る</summary> ```python def apply_discount(price, discount=0.1): """割引後の価格を計算する。

```
Args:
    price (int): 元の価格
    discount (float, optional): 割引率。デフォルトは0.1（10%）

Returns:
    float: 割引後の価格
"""
return price * (1 - discount)
```

</details>

Step 11: チェックリスト【書く前に確認】

docstringを書いたら、これをチェック！

- [ ] `"""` 三重引用符で囲んだ？
- [ ] defの直後に書いた？
- [ ] 1行目に概要を書いた？
- [ ] 動詞で終わってる？（〜する）
- [ ] ピリオド（。）で終わってる？
- [ ] 引数を全部説明した？
- [ ] 型を書いた？ `(int)`, `(str)` など
- [ ] 戻り値を説明した？（あれば）
- [ ] デフォルト値を説明した？（あれば）

Step 12: よくある質問

Q1: 英語で書くべき？日本語でいい？

A: プロジェクトによる
- **日本のチーム** → 日本語OK
- **国際的なプロジェクト** → 英語推奨
- **迷ったら** → 周りのコードに合わせる

Q2: どこまで詳しく書けばいい？

A: 相手が理解できるレベル
- **個人開発** → 簡潔でOK
- **チーム開発** → 詳しく書く
- **オープンソース** → かなり詳しく書く

Q3: 全部の関数に書かないとダメ？

A: 理想は全部、でも優先順位をつける
- **優先度高**: 他の人が使う関数
- **優先度中**: 自分しか使わない関数
- **優先度低**: 超簡単な関数（1行とか）

Q4: 書くのに時間がかかる...

A: 慣れです！

- 最初は時間かかる → 普通
- テンプレートを使う → 楽になる
- 繰り返す → 自然に書けるようになる

Step 13: まとめ【3つのポイント】

ポイント1: 最低限これだけは書く
```python
def 関数名(引数):
    """何をする関数か1行で説明。"""
    pass
```

ポイント2: 引数と戻り値も書く

```python
def 関数名(引数):
    """何をする関数か1行で説明。

    Args:
        引数 (型): 説明

    Returns:
        型: 説明
    """
    pass
```

ポイント3: help()で確認する癖をつける

```python
help(関数名)  # 必ず確認！
```

🎓 卒業課題【実際に書いてみよう】

以下の関数を完成させてください。

```python

```

```python
def calculate_bmi(height, weight):
    # TODO: docstringを書く

    bmi = weight / (height ** 2)
    return bmi
```

ヒント

- heightは身長（メートル）
- weightは体重（キログラム）
- BMIを計算して返す

<details> <summary>模範解答</summary> ```python def calculate_bmi(height, weight): """BMI（体格指数）を計算する。

```
    Args:
        height (float): 身長（メートル）
        weight (float): 体重（キログラム）

    Returns:
        float: BMI値
    """
    bmi = weight / (height ** 2)
    return bmi
```

</details>

🚀 次のステップ

初級クリア後は...

1. **エラー処理を追加** (`Raises:` を書く)
2. **使用例を追加** (`Examples:` を書く)
3. **複雑な型を扱う** (`list of dict` など)

もっと学びたい人へ

- Google Style Guide を読む
- 有名なライブラリのdocstringを見る
- チームのコーディング規約を確認

✨ おめでとうございます！

これであなたもdocstringが書けます！

大切なのは:
- ✅ 完璧を求めない
- ✅ まずは書いてみる
- ✅ 継続する

頑張ってください！ 🎉

プログラミング初心者から、一歩前進！