08 Python基礎

関数の定義

- 教科書 P.28「関数の定義」〜P.34「キーワード引数」までを熟読してください。

定義の仕方

```
def 関数名(引数,...):
    """
    関数の説明文
    """
    処理内容
      :
```

- 戻り値は、`return`で返す
  - returnを省略した場合や返す値を指定しない場合は`None`が返る。
- 正しくdocstringを記述することで、vscodeではHintを表示してくれる。
  - `"""`もしくは`'''`で囲む（トリプルクォートで囲む）
  - 極力、記述すべき。
- docstringの正しい書き方を学ぼう
  https://note.nkmk.me/python-docstring/
  https://qiita.com/11ohina017/items/118b3b42b612e527dc1d


ファイル 編集 選択 表示 移動 実行 ターミナル ヘルプ
08
08_func_1.py 2
08_func_1.py > ...
1 def func1(a, b, c):
2 """
3 3つの引数の和を返す
4 """
5 return (a+b+c)
6
7
8 x = 10
9 y = fu
10
func1
function
FutureWarning
FileNotFoundError
ModuleNotFoundError
def func1(
a: Any,
b: Any,
c: Any
) -> Any
3つの引数の和を返す

型ヒント

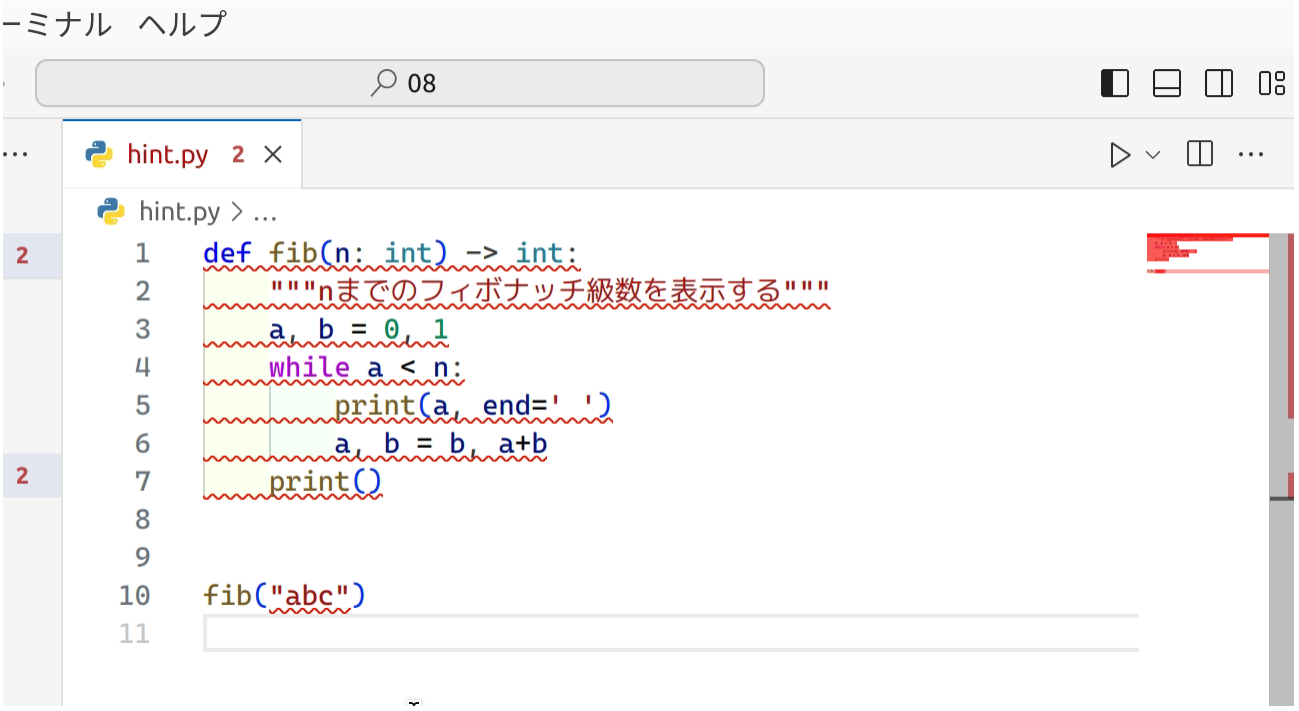
> https://docs.python.org/ja/3.9/library/typing.html
>
> 型ヒント以外にも関連する機能が記載されているので、目を通すことをお勧めする。

- Pythonは動的型付け言語であるため、型を指定せずに変数宣言できる。また関数の引数や戻り値に型を宣言する必要はない。
- 他言語使用者からの要望や、コードの可読性向上を目的に `型ヒント` 機能が盛り込まれた。（Python3.5以降）
    - Python自身には、これを使用する機能はない。（エラーチェック等を行なわないが、情報としては保持している）
    - サードパーティ製のツールに委ねられている。
        - mypyというツールが一般的（インストールが必要）
        - vscode用の拡張機能を利用するのが簡単

- フィボナッチ級数を求める関数に型ヒントを付けてみる

```python
def fib(n: int) -> int:
    """nまでのフィボナッチ級数を表示する"""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()


fib("abc")
```



- 引数の型が合わないこと
- 戻り値の型が合わないことを指摘される。

関数の引数

- 関数の引数は2種類ある
  - 位置引数（通常の引数）
  - キーワード引数

> https://docs.python.org/ja/3.9/glossary.html
>
> **argument**
>
> (実引数) 関数を呼び出す際に、 関数 (または メソッド) に渡す値です。実引数には２種類あります:
>
> - *キーワード引数*: 関数呼び出しの際に引数の前に識別子がついたもの (例: `name=`) や、 `**` に続けた辞書の中の値として渡された引数。例えば、次の `complex()` の呼び出しでは、 `3` と `5` がキーワード引数です:
>
> ```
> complex(real=3, imag=5)
> complex(**{'real': 3, 'imag': 5})
> ```
>
> - *位置引数*: キーワード引数以外の引数。位置引数は引数リストの先頭に書くことができ、また `*` に続けた iterable の要素として渡すことができます。例えば、次の例では `3` と `5` は両方共位置引数です:
>
> ```
> complex(3, 5)
> complex(*(3, 5))
> ```

> 実引数は関数の実体において名前付きのローカル変数に割り当てられます。割り当てを行う規則については 呼び出し (call) を参照してください。シンタックスにおいて実引数を表すためにあらゆる式を使うことが出来ます。評価された値はローカル変数に割り当てられます。

位置引数

- 記述の順番通りに受け取る。
    - 並び順が重要

キーワード引数

- 仮引数をキーワードとして使用して受け取る
    - 記述の順番は関係ない

【arg_1.py】

```
def sample(a, b, c):
    print(f'a => {a}')
    print(f'b => {b}')
    print(f'c => {c}')
    print("-="*10)


# 通常の呼び出し（位置引数）
sample(10, "abc", 20)

sample("xyz", "123", 45)

# キーワード引数での呼び出し
sample(a=10, c="abc", b=20)

sample(c=10, a="abc", b=20)
```

位置引数とキーワード引数の混在

- 両方の引数を同時に使用することは可能
- 位置引数を先に指定する必要がある

【arg_2.py】

```
def sample(a, b, c):
    print(f'a => {a}')
    print(f'b => {b}')
```

```
    print(f'c => {c}')
    print("-="*10)


# 混在での呼び出し
sample(10, c=20, b=30)  # OK

sample(10, 20, c=30)  # OK

sample(10, a=20, b=30)  # NG

sample(a=20, b=30, 40)  # NG
```

標準関数での利用

出力後に改行しない

```
print('012')
print('abc', end='')
print('xyz')
```

【実行結果】

```
012
abcxyz
```

区切り文字を設定

```
print(10, 20, 30, sep=':')
```

【実行結果】

```
10:20:30
```

> https://docs.python.org/ja/3.9/library/functions.html#print

- すべてのキーワード引数を調査・確認してください。
    - 具体的な使用方法を確認してください。

練習1

- 以下の動作を行う関数を作成し、呼び出した結果を出力するメイン部分も作成してください。

    1. `str2int( )` 関数を定義する(string2int.py)

        - 数字文字列を受け取り、数値（整数）に変換して返す。
        - 文字列でない場合は、そのまま値を返す。
        - 数値に変換できない場合は0を返す
        - ヒント

        > https://docs.python.org/ja/3.9/library/functions.html#type
        >
        > https://note.nkmk.me/python-str-num-determine/

    【実行結果】

```
$ python string2int.py
0
10
100
```

    【string2int.py】

```
def str2int(s):
     """文字列を数値に変換した値を返す"""




# main
print(str2int('a'))
print(str2int('10'))
print(str2int(100))
```

2. 先に作成した `str2int( )` を改良もしくは利用し、 `list2int( )` 関数を作成しよう。(strlist2int.py)

    - リストを渡した場合、全ての要素の文字列を数値（整数）に変換して返す。
        - 要素が文字列でない場合は、0とする。

- リストでなはなく文字列の場合は、受け取った値を数値に変換して返す。
- リストでもなく、文字列でもない場合は、Noneを返す。
  【実行結果】

```
$ python strlist2int2.py
[5, 0, 100, 10, 1]
100
0
```

【strlist2int.py】

```
def list2int(s):
    """文字列を数値に変換した値を返す（List対応）"""




# main
print(list2int(['5','ab','100',10,1]))
print(list2int('100'))
print(list2int('xyz'))
```

3. ２つのリストを受け取り、受け取ったリストを結合して返す関数 `combine_list( )` を作成する。(combine.py)
    - リスト以外が渡された場合は、エラー出力し、受け取った引数をリストにして返すものとする。
      【実行結果】

```
$ python combine.py
[1, 2, 3, 4, 5, 6]
[4, 5, 6, 1, 2, 3]
引数がリストではありません
['a', [1, 2, 3]]
引数がリストではありません
[[1, 2, 3], 'xyz']
引数がリストではありません
[10, 'abc']
```

【combine.py】

```
def combine_list(list1, list2):
    '''2つのリストを結合し、リストで返す'''




# main
# 正常な引数
print(combine_list([1, 2, 3], [4, 5, 6]))
print(combine_list(list2=[1, 2, 3], list1=[4, 5, 6]))
```

```
# 誤った引数
print(combine_list('a', [1, 2, 3]))
print(combine_list([1, 2, 3], 'xyz'))
print(combine_list(10, 'abc'))
```

練習2

- 以下のプログラムを作成してください。関数を作成し、それを検証・実行するmain部分も作成してください。

1. **テキストをリストに変換する関数**　(ren2_1.py)

   - 文字列をリストに変換する関数 `str2list()` を作成してください。
   - 各要素は空白で区切るものとします。
   - 【実行結果】

   ```
   $ python ren2_1.py
   空白で区切られた文字列を入力してください：ab 12 cdef 34
   ['ab', '12', 'cdef', '34']
   ```

2. **文字列内の単語の数を数える関数**　(ren2_2.py)

   - ユーザーに文章を入力してもらい、その文章内に含まれる単語の数を数える関数 `word_count()` を作成してください。
     単語は空白で区切られているものとします。
   - 【実行結果】

   ```
   $ python ren2_2.py
   空白で区切られた文字列を入力してください：abc 123 44 55
   4
   ```

3. **偶数番目の要素を取り出す関数**　(ren2_3.py)

   - ユーザーに整数のリストを入力してもらい、そのリスト内の偶数番目の要素だけを取り出す関数 `even_list()` を作成してください。
   - その関数を使って偶数番目の要素（先頭を1番目と数えて）だけを出力してください。
   - リストは空白で区切って入力するものとします。
   - 【実行結果】

```
$ python ren2_3.py
空白で区切られた整数を入力してください：1 2 3 4 5 6 7 8
[2, 4, 6, 8]
```

4. **リスト内の偶数だけを取り出す関数**　(ren2_4.py)

- ユーザーに整数のリストを入力してもらい、そのリスト内の偶数要素だけを取り出す関数'`get_even()`'を作成してください。
- その関数を使って偶数だけを出力してください。
- リストは空白で区切って入力するものとします。
- 【実行結果】

```
$ python ren2_4.py
空白で区切られた整数を入力してください：1 1 2 2 3 3 4
[2, 2, 4]
```