📖 Effective Python読書メモ 🐍

この資料について

- 『Effective Python』を読んで自分のためになったTipsをアウトプットとして残すことが目的

目次

Sec1. Pythonic思考

- そもそもpythonicってなんだ?
  - Pythonコミュニティにおける**特定のスタイルに沿った**コードを表す形容詞
  - **Pythonらしい**、シンプルで読みやすい**コードの書き方**のこと
- (関連)「PEP20 - The Zen of Python」
  - Pythonの設計について記述されたイディオム

代表的なPythonic思考な記述方法①

- PEP8スタイルガイドに従う
  - Pythonは正しい構文であればどんな書き方でも動くが、一貫したスタイルに従うと、コードはより扱いやすく、読みやすくなる
  - チーム、大きなコミュニティで他のPythonプログラマと共通のスタイルを分かち合うことで、プロジェクトの共同作業がより捗る。
  - 他人(将来の自分)のためにもスタイルガイドに従うべし。

代表的なPythonic思考な記述方法②

- f文字列による埋め込み
  - Cスタイルフォーマット、str.formatを避け、f文字列による埋め込みを使うことで可読性が向上する

```
key = 'my_var'
value = 1.234
c_tuple = '%-10s = %.2f' % (key, value)   # Cスタイルフォーマット
f_string = f'{key:<10} = {valeu:.2f}'   # f文字列
assert c_tuple == f_string
```

代表的なPythonic思考な記述方法③

- 複雑な式の代わりにヘルパー関数を書く
  - orやandのような論理演算子は安易に使わず、意図がわかる名前を付けた関数を別に定義する
  - 特に、同じロジックを繰り返す必要がある場合は有効的

```
def get_first_int(values, key, default=0):
    found = values.get(key, [""])
    if found[0]:
        return int(found[0])
    else:
        return default

hoge = get_first_int(my_values, 'fuga')
```

代表的なPythonic思考な記述方法④

- indexではなくアンパック、rangeではなくenumerateを使う
  - アンパックを使うことでひとつの代入文で複数の値を代入できる
  - enmuerateを使うことでイテレータでループしながら、要素のインデックスを取り出せる

```
flavor_list = ['vanilla', 'chocolate', 'strawberry']
for i, flavor in enumerate(flavor_list, 1):  # 第二引数でカウンタを開始する数も指定可能
    print(f'{i}: {flavor}')
```

代表的なPythonic思考な記述方法⑤

- 代入式で繰り返しを防ぐ
  - セイウチ演算子 `:=` を使うことで変数名への値代入と評価を1つの式で行うことができる
  - Pythonにはswitch/case文やdo/whileループはないものの、代入式を用いることで明確に記述できる

```
def pick_fruit():
    ...
def make_juice(fruit, count):
    ...

bottles = []
while fresh_fruit := pick_furuit():
    for fruit, count in fresh_fruit.items():
        batch = make_juice(fruit, count)
        bottles.extend(batch)
```