Pythonブームの裏で――

Python入門書の執筆&監修 現場レポート

ODC2021Online用

プロフィール

執筆編集サイド:
編集プロダクション リブロワークス取締役
大津雄一郎

1990年代に印刷所のDTPオペレーター

2000年前後にIT系出版社の社員ライター

そのあと6年ほどIT系フリーライター

2008年に代表の小山とリブロワークスを設立して現在に至る

https://libroworks.co.jp/

書籍監修サイド:
株式会社ビープラウド取締役 / Python Climber
鈴木たかのり(@takanory)

一般社団法人PyCon JP Association副代表理事、Python mini Hack-a-thon主催などPythonコミュニティでも活動

Python関連の書籍を個人、業務で多数執筆、監修、翻訳

Amazon著者ページ

これまでに監修をお願いした本

2018

2020

2020

2021

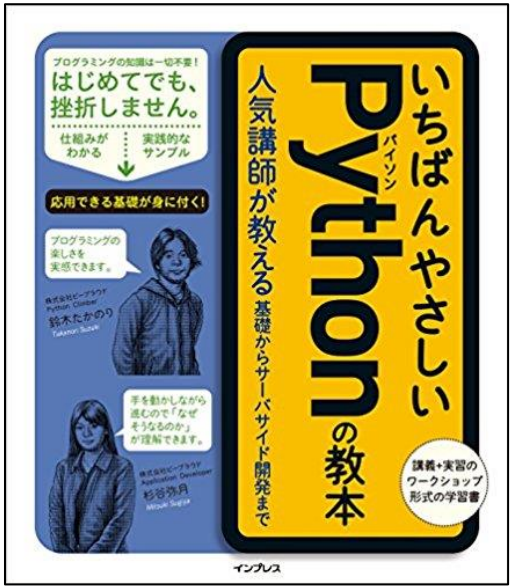
2021

2021

最初の監修依頼～「Pythonふりがなプログラミング」初版

プログラムにふりがなという形で、説明を入れている本

前年にビープラウドさんに執筆をお願いして、編集・制作を担当していたつながりで



2017

2018

監修を依頼した理由

執筆と編集を並行でやるようになり、他人の原稿を読む機会が増えると、

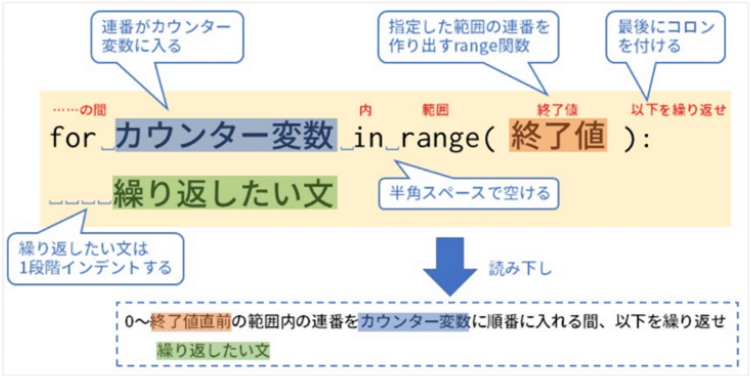
知らないことが結構あることに気付いて

書くのが怖くなる

指摘「リストを使ったforのほうがPython的」

●for文の書き方を覚えよう

for文は「for カウンター変数 in 〜:」で1セットです。inの後のrange関数が範囲内の連番を作り出すので、その連番の数だけ繰り返されます。for文で繰り返し文は、その次から1段階インデントして書きます。



forの後のカウンター変数には、range回数が作り出す連番が1つずつ順番に入ります。カウンター変数は繰り返したい文の中で利用できるのですが、まずは単純に同じことを繰り返し文を書いてみましょう。

matoba tatsuya 2018年4月23日

例えば、以下のようなforでも3回実行されるので、連番より要素の数が繰り返しに関係します。

```
for _ in [1, 1, 1]:
    print('tes't')
```

Suzuki Takanori 2018年4月26日

ちょっとどう書けばいいかという話になりますが、python的にinの右側にある繰り返し可能オブジェクトを1つずつとり出して変数に設定、なくなるまでそれを繰り返す。という動作です。ここでは、range()の話とforの話の2つをまぜて説明しているので、混乱の元になると思います。

Suzuki Takanori 2018年4月26日

06のリストの説明をしてから、そのリストを'使って for で繰り返しの説明をする方がPython的だよなーと思うんですよねー(大幅な書き直しになりますが

YO 返信　↓ 18 件の未読

Suzuki Takanori 2018年4月26日

ちょっとどう書けばいいかという話になりますが、python的にinの右側にある繰り返し可能オブジェクトを1つずつとり出して変数に設定、なくなるまでそれを繰り返す。という動作です。ここでは、range()の話とforの話の2つをまぜて説明しているので、混乱の元になると思います。

Suzuki Takanori 2018年4月26日

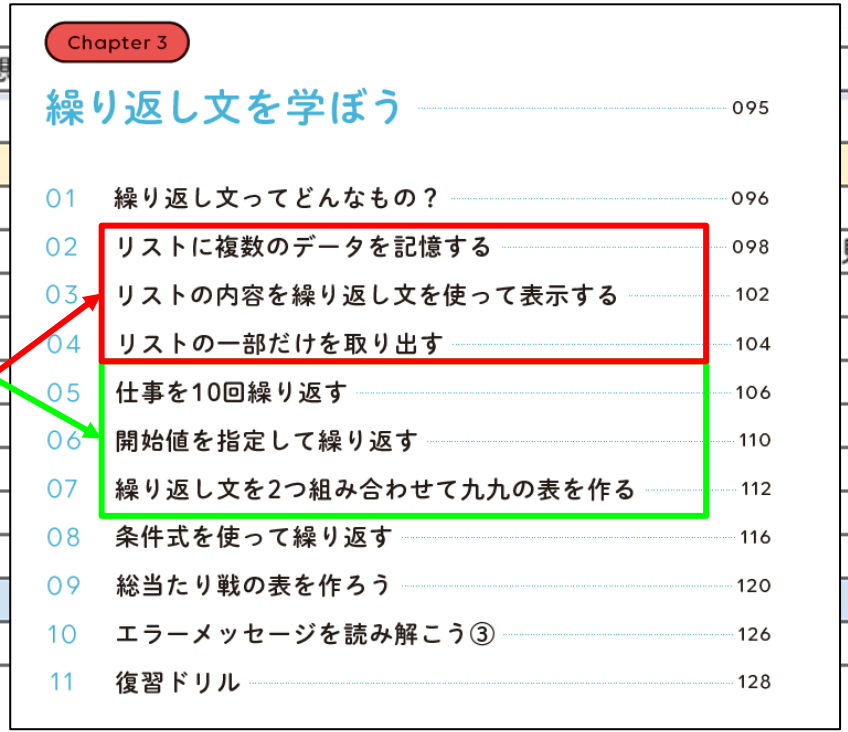
06のリストの説明をしてから、そのリストを'使って for で繰り返しの説明をする方がPython的だよなーと思うんですよねー(大幅な書き直しになりますが

構成をひっくり返して……

初期構成案

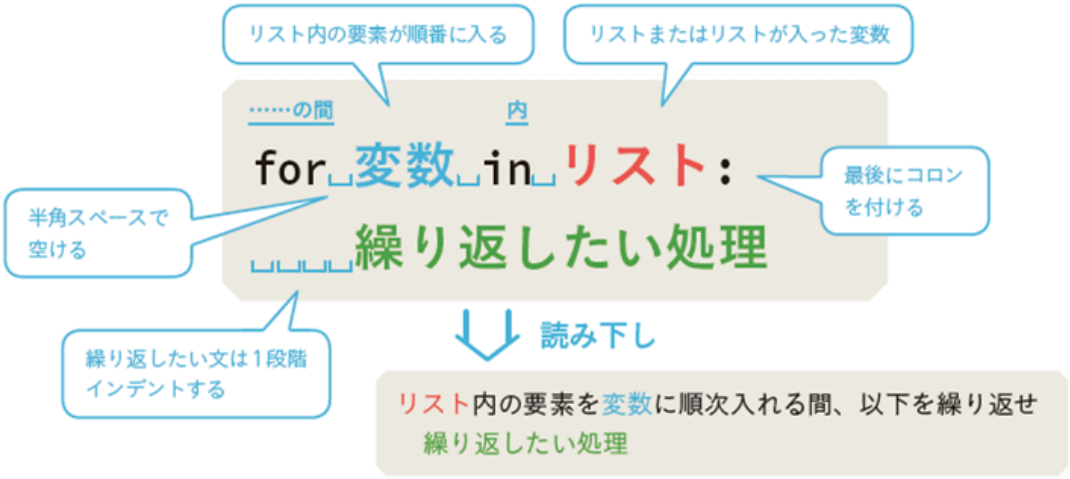
章	節	ページ想
第3章	繰り返す文	
	繰り返す文ってどんなもの?	
	仕事を10回繰り返す	
	1〜10を繰り返す	
	10〜1を繰り返す	
	繰り返し文を2つ組み合わせて九九の表を作る	
	リストに複数のデータを記憶する	
	リストのデータを繰り返し文で表示する	
	条件式を使って繰り返す	
	総当たり戦の表を作ろう	
	エラーメッセージを読み解こう③	
	復習ドリル	

最終目次



for文の書き方を覚えよう

for文は「for 変数 in リスト:」で1セットです。リストの要素が順番に変数に入り、リストの要素の数だけインデントされた文が繰り返されます。



技術評論社「やさしくわかるPythonの教室」

特徴としては会話メインなので、通称「会話Python」

フォーマット面で特殊なギミックはないけど、最初に出版社さんから提示された叩き台となる構成案があったので、そこで違いが出てきている

監修期間長め

指摘「演算子ではない」

順位	演算子	説明
1	(式)、[式]、{キー：値}、{式}	式結合またはタプル表示、リスト表示、辞書表示、集合表示
2	リスト[添字]、リスト[添字:添字]、関数(引数)、オブジェクト.属性	添字指定、スライス操作、関数呼び出し、属性参照
3	await ○○	Await 式
4	**	べき乗
5	+○○、-○○、~○○	正数、負数、ビット単位のNOT
6	*、@、/、//、%	掛け算、行列計算、割り算、切り捨て割り算、余り
7	+、-	加算および減算
8	<<、>>	シフト演算
9	&	ビット単位のAND
10	^	ビット単位のXOR
11	|	ビット単位のOR
12	in、not in、is、is not、<、<=、>、>=、!=、==	帰属テスト、同一性テスト、比較
13	not ○○	論理演算のNOT
14	and	論理演算のAND
15	or	論理演算のOR
16	if～else	条件式

128 件の注釈

清水川 貴之 1月30日

これは演算子ではなく文法のデリミタです。
https://docs.python.org/ja/3/reference/lexical_analysis.html#operators

返信を追加...

清水川 貴之 1月30日

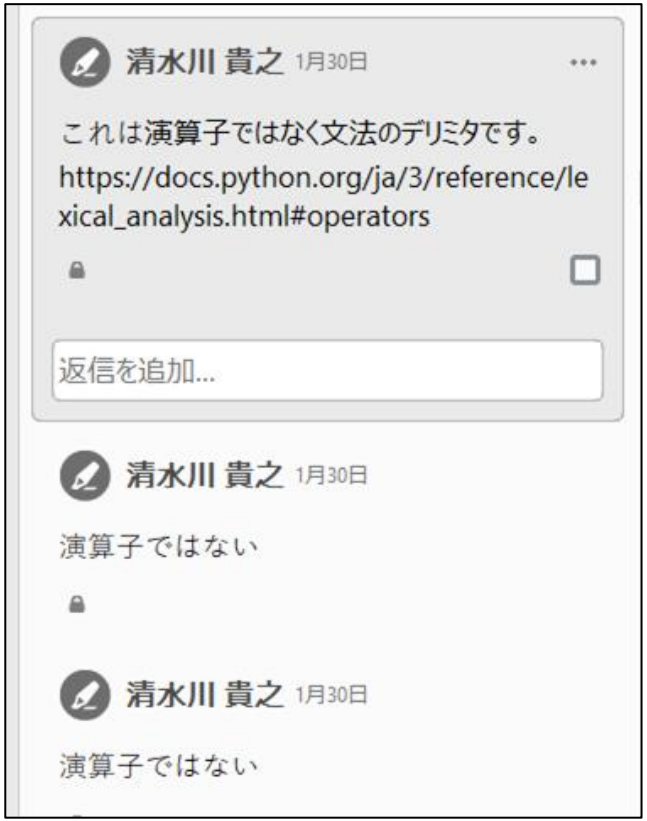
演算子ではない

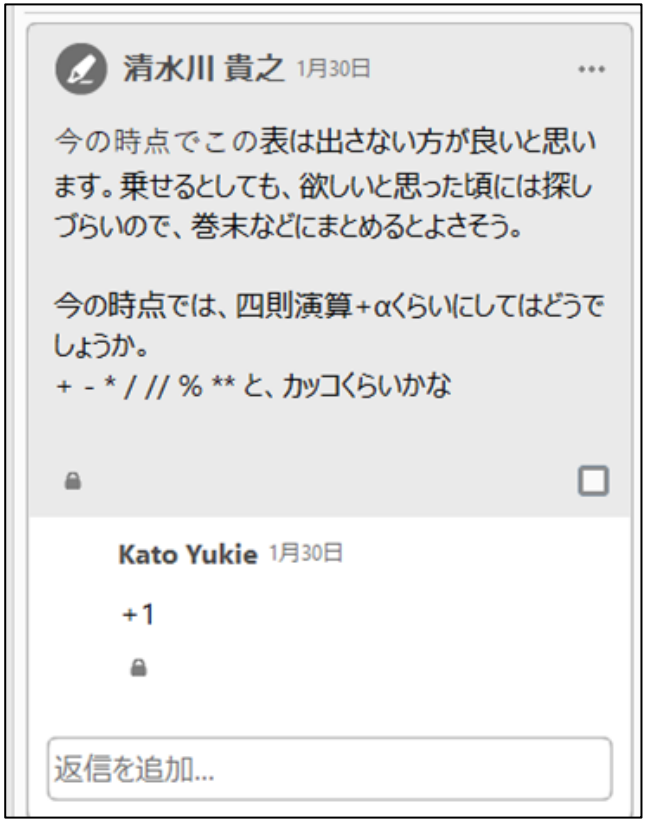
清水川 貴之 1月30日

演算子ではない

清水川 貴之 1月30日

名称的には、単項演算子です。


清水川 貴之 1月30日
これは演算子ではなく文法のデリミタです。
https://docs.python.org/ja/3/reference/lexical_analysis.html#operators
返信を追加...
清水川 貴之 1月30日
演算子ではない
清水川 貴之 1月30日
演算子ではない


清水川 貴之 1月30日
今の時点でこの表は出さない方が良いと思います。乗せるとしても、欲しいと思った頃には探しづらいので、巻末などにまとめるとよさそう。
今の時点では、四則演算+αくらいにしてはどうでしょうか。
+ - * / // % ** と、カッコくらいかな
Kato Yukie 1月30日
+1
返信を追加...

表が小さくなりました

●計算などに使う記号の優先順位

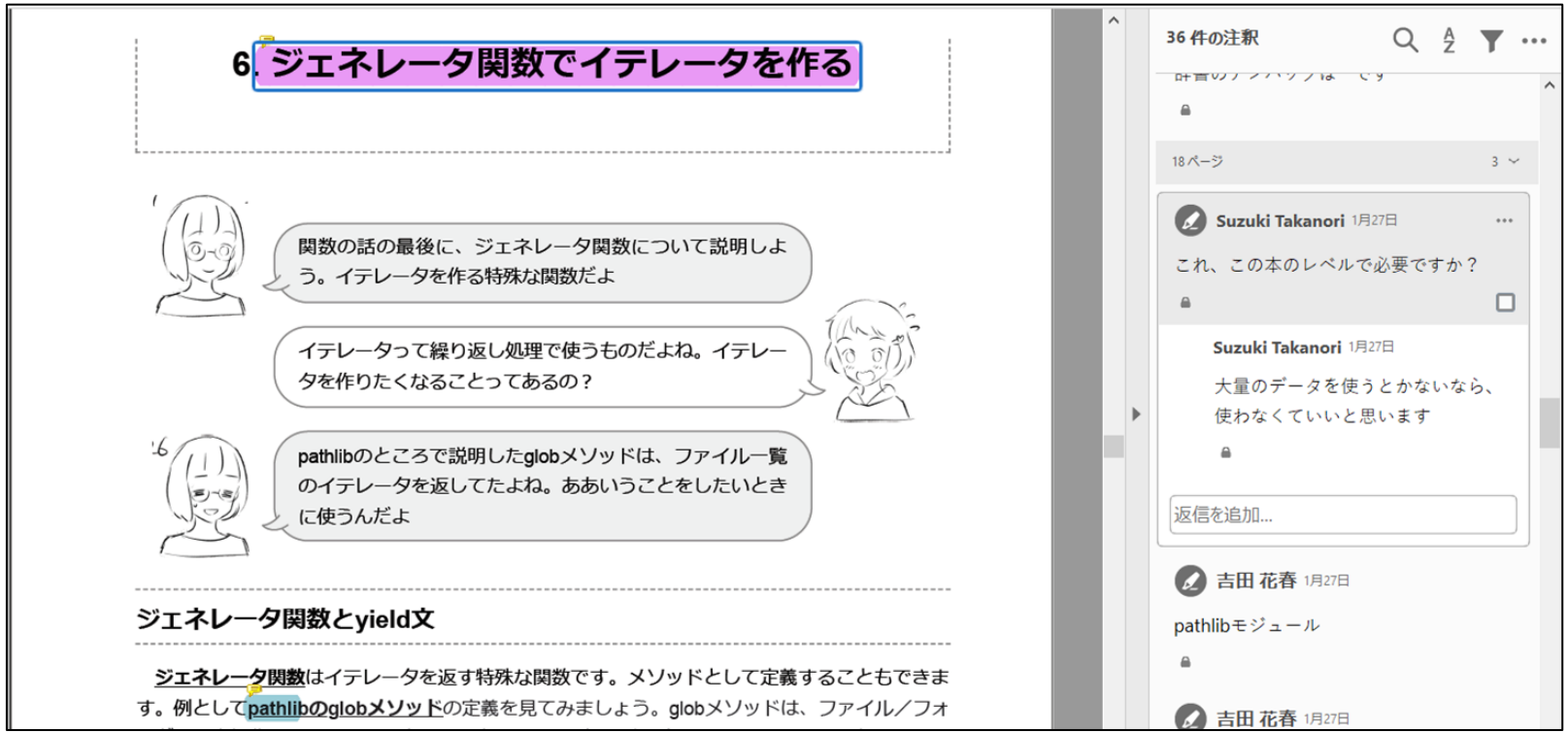
順位	記号	働き
1	(式)、[式]、{式}	式の優先順位アップや、リスト、タプルのためのカッコ類
2	関数(引数)、オブジェクト.メソッド、リスト[添え字]	関数やメソッドの呼び出しに使う記号や、リストやタプルなどの要素を参照する記号
4	**	べき乗
5	+○○、-○○	正数、負数
6	*、@、/、//、%	掛け算、行列計算、割り算、切り捨て割り算、余り
7	+、-	足し算、引き算
8～18	条件などに使う比較演算子や論理演算子など(Chapter 3以降で解説)	

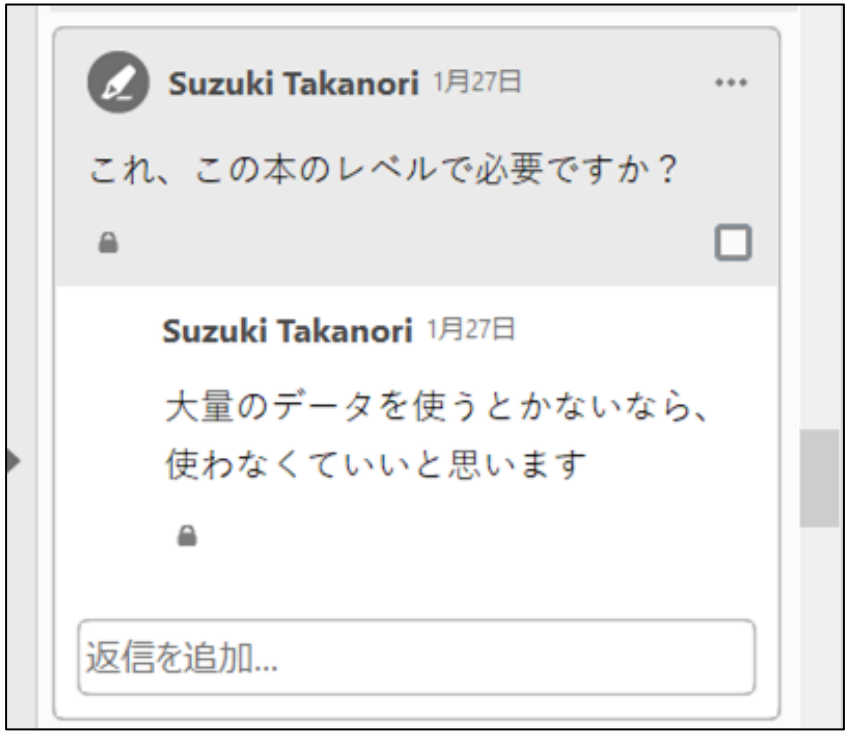
※正確にいうと、この表には演算子だけでなく、デリミタ(区切り文字)に分類される記号なども入っています。式に使われる記号の処理順を示したものと考えてください。

何か途中がところどころ抜けている表だね

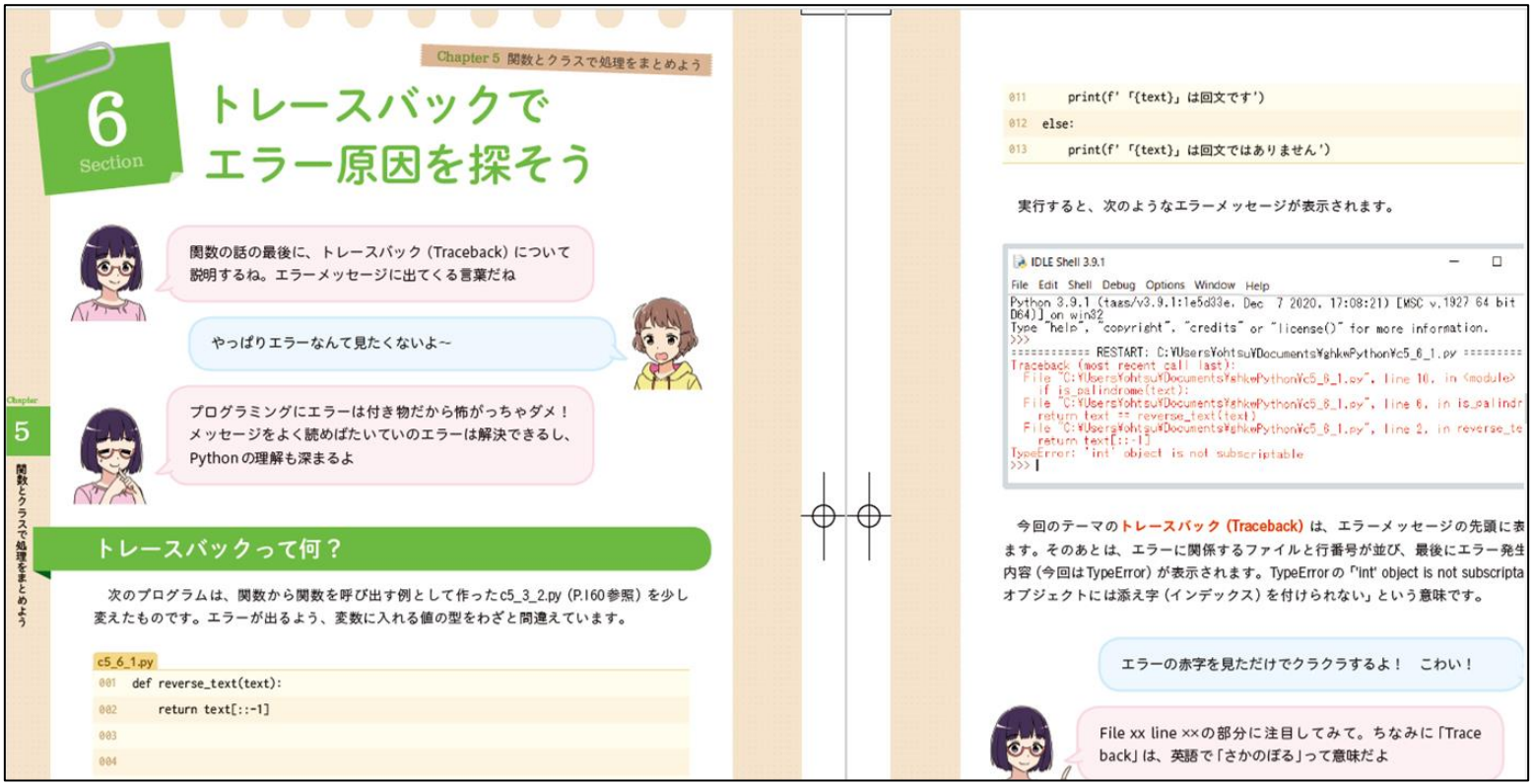
記号類は計算用以外にもたくさんあるんだけど、多すぎて混乱するからChapter 2で出てくるものだけをまとめたよ

指摘「これ、必要ですか？」


6. ジェネレータ関数でイテレータを作る
関数の話の最後に、ジェネレータ関数について説明しよう。イテレータを作る特殊な関数だよ
イテレータって繰り返し処理で使うものだよね。イテレータを作りたくなることってあるの？
pathlibのところで説明したglobメソッドは、ファイル一覧のイテレータを返してたよね。ああいうことをしたいときに使うんだよ
ジェネレータ関数とyield文
ジェネレータ関数はイテレータを返す特殊な関数です。メソッドとして定義することもできます。例としてpathlibのglobメソッドの定義を見てみましょう。globメソッドは、ファイル／フォ
36 件の注釈
18 ページ
3
Suzuki Takanori 1月27日
これ、この本のレベルで必要ですか？
Suzuki Takanori 1月27日
大量のデータを使うとかないなら、使わなくていいと思います
返信を追加...
吉田 花春 1月27日
pathlibモジュール
吉田 花春 1月27日


Suzuki Takanori 1月27日
これ、この本のレベルで必要ですか？
Suzuki Takanori 1月27日
大量のデータを使うとかないなら、
使わなくていいと思います
返信を追加...

全然違う話にしました



Chapter 5 関数とクラスで処理をまとめよう

6 Section

トレースバックで エラー原因を探そう

関数の話の最後に、トレースバック（Traceback）について説明するね。エラーメッセージに出てくる言葉だね

やっぱりエラーなんて見たくないよ～

プログラミングにエラーは付き物だから怖がっちゃダメ！メッセージをよく読めばたいていのエラーは解決できるし、Pythonの理解も深まるよ

トレースバックって何？

次のプログラムは、関数から関数を呼び出す例として作ったc5_3_2.py（P.160参照）を少し変えたものです。エラーが出るよう、変数に入れる値の型をわざと間違えています。

c5_6_1.py

```
001 def reverse_text(text):
002     return text[::-1]
003
004
```

```
011     print(f'「{text}」は回文です')
012 else:
013     print(f'「{text}」は回文ではありません')
```

実行すると、次のようなエラーメッセージが表示されます。

```
IDLE Shell 3.9.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:¥Users¥ohtsu¥Documents¥ghkw¥Python¥c5_6_1.py ========
Traceback (most recent call last):
  File "C:¥Users¥ohtsu¥Documents¥ghkw¥Python¥c5_6_1.py", line 10, in <module>
    if is_palindrome(text):
  File "C:¥Users¥ohtsu¥Documents¥ghkw¥Python¥c5_6_1.py", line 6, in is_palindr
    return text == reverse_text(text)
  File "C:¥Users¥ohtsu¥Documents¥ghkw¥Python¥c5_6_1.py", line 2, in reverse_te
    return text[::-1]
TypeError: 'int' object is not subscriptable
>>>
```

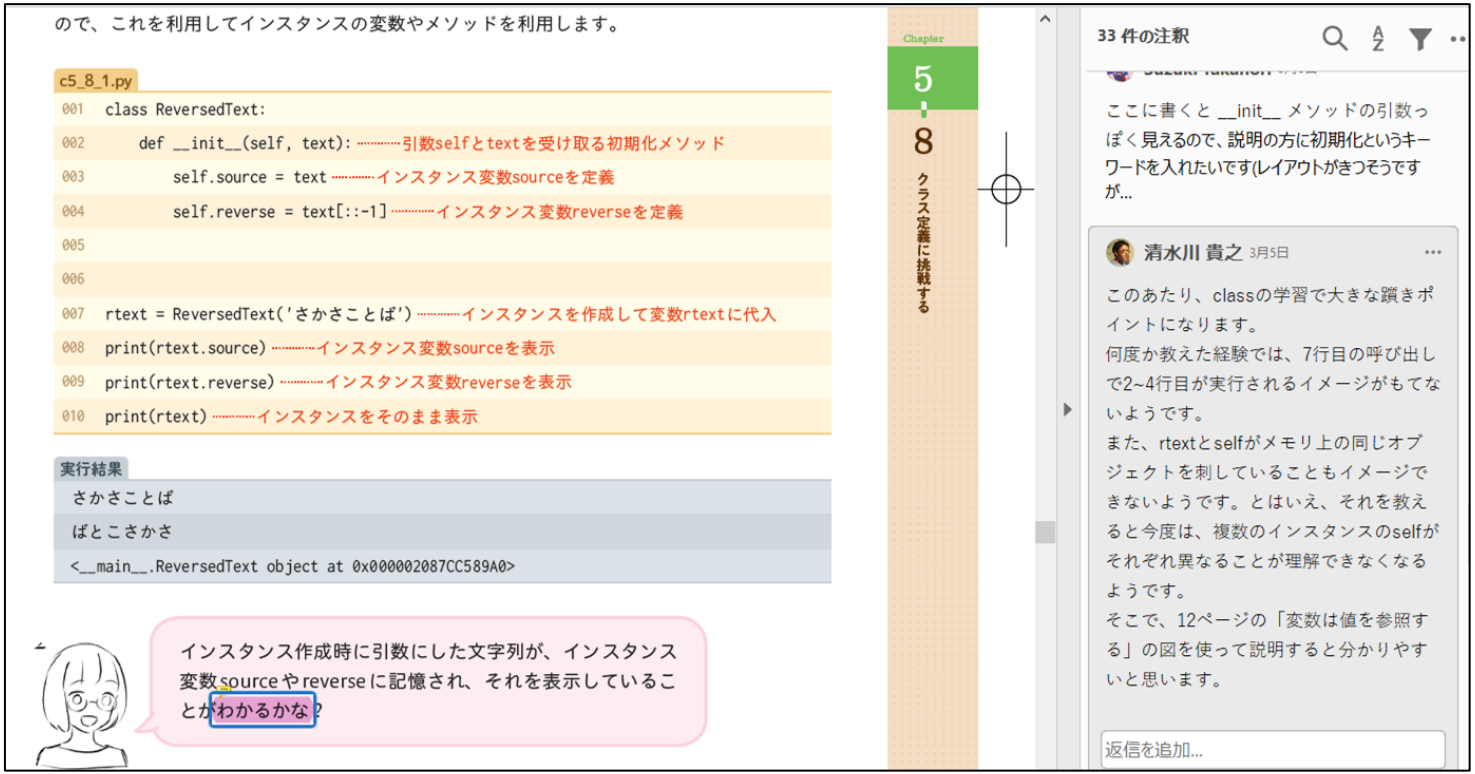
今回のテーマの**トレースバック（Traceback）**は、エラーメッセージの先頭に表ます。そのあとは、エラーに関係するファイルと行番号が並び、最後にエラー発生内容（今回はTypeError）が表示されます。TypeErrorの "'int' object is not subscripta オブジェクトには添え字（インデックス）を付けられない」という意味です。

エラーの赤字を見ただけでクラクラするよ！　こわい！

File xx line ××の部分に注目してみて。ちなみに「Traceback」は、英語で「さかのぼる」って意味だよ

Chapter 5 関数とクラスで処理をまとめよう

指摘「classの学習で大きなつまずきポイントですよ」

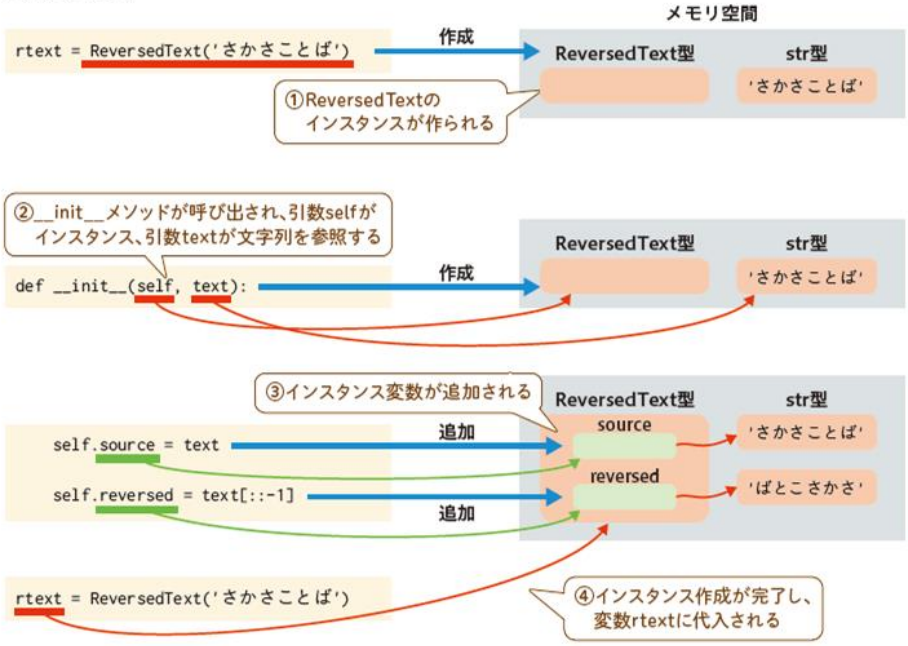


清水川 貴之 3月5日 ・・・

このあたり、classの学習で大きな躓きポイントになります。
何度か教えた経験では、7行目の呼び出しで2~4行目が実行されるイメージがもてないようです。
また、rtextとselfがメモリ上の同じオブジェクトを刺していることもイメージできないようです。とはいえ、それを教えると今度は、複数のインスタンスのselfがそれぞれ異なることが理解できなくなるようです。
そこで、12ページの「変数は値を参照する」の図を使って説明すると分かりやすいと思います。

図を追加しました

ReversedTextクラスのインスタンス作成は一瞬にすぎません。その作成プロセスを図解で見てみましょう。



マイナビ出版「Pythonつみあげトレーニングブック」

通称「忍者Python」

「少林寺36房」の特訓シーンをヒントにした

「ミッション」で復習しつつ、無意識で読解できる学習法

mission 2-01

達成目標 80 秒

式を見て処理順を示せ①

2章 ▼ 基本的なデータと計算

式に含まれる演算子の処理順を書き込んでください。

1 ＋ 2 ＊ 3 ┈┈▶ 1 ②＋ ①2 ＊ 3

1. `1 + 2 * 3`
2. `(1 + 2) * 3 * 4`
3. `1 * 2 * 3`
4. `1 + (2 * 3) * 4`

mission 3-03

達成目標 120 秒

式を見て処理順を示せ③

3章 ▼ 命令と条件分岐

式に含まれる演算子の処理順を書き込んでください（カッコは演算子ではありません）。

`i ①+ 1 ②< 10 ④and i ③< 100`

1. `not True and True`
2. `True or not False and True`
3. `(True or not False) and True`
4. `12 < a + i and a + i < 20`
5. `text != password or text == ''`

指摘「正解が読み取りにくい」

6

class datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

全ての引数がオプションで、デフォルト値は 0 です。引数は整数、浮動小数点数でもよく、正でも負でもかまいません。

days, seconds, microseconds だけが内部的に保持されます。引数は以下のようにして変換されます:

1 ミリ秒は 1000 マイクロ秒に変換されます。

1 分は 60 秒に変換されます。

1 時間は 3600 秒に変換されます。

1 週間は 7 日に変換されます。

https://docs.python.org/ja/3.9/library/datetime.html#timedelta-objects

①days、seconds、microseconds以外の引数は無視される

②1分は60000000マイクロ秒に変換される

③0.5秒は500000マイクロ秒に変換される

65 件の注釈

Suzuki Takanori 4月28日

これが正解だと思うけど、読み取るの難しい...

返信を追加...

10 ページ 1

Suzuki Takanori 4月28日

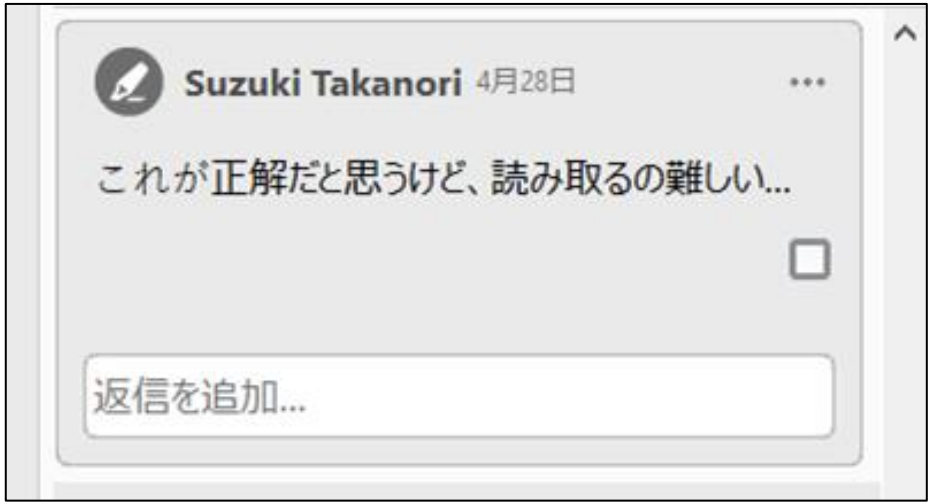
見出しが2つしかないので、ちょっとわかりにくいです

11 ページ 5

Suzuki Takanori 4月28日

②1分は60000000マイクロ秒に変換される

③0.5秒は500000マイクロ秒に変換される



0.5秒→500ミリ秒にして少し簡単に

6

1 ミリ秒は 1000 マイクロ秒に変換されます。
1 分は 60 秒に変換されます。
1 時間は 3600 秒に変換されます。
1 週間は 7 日に変換されます。
https://docs.python.org/ja/3.9/library/datetime.html#timedelta-objec

① days、seconds、microseconds以外の引数は無視される
② 500ミリ秒は500,000マイクロ秒に変換される
③ 1分は600,000,000マイクロ秒に変換される

指摘「関数にする意味がない」

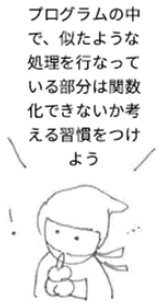
関数から戻り値を返すようにしておくと、複数の関数を組み合わせて使う時に便利です。shift_order関数とget_job関数を組み合わせて使うよう、shift_order関数の呼び出し部分を書き換えてみましょう

・c7_1_5.py

```
def get_job(key):
    job_dic = {
        'keibi': '城内と庭の警備',
        'teire': '手裏剣の手入れ',
        'teisatsu': '敵地への偵察',
    }
    return job_dic[key]


def shift_order(name, job):
    print(f'''{name}様
シフト管理係の忍者です。
本日、{job}をお願いします。''')


shift_order('服部', get_job('keibi'))  ----引数を変更
shift_order('河村', get_job('teisatsu'))  ----引数を変更
```



20 件の注釈

dict にしたい

6ページ 1

Suzuki Takanori 4月27日

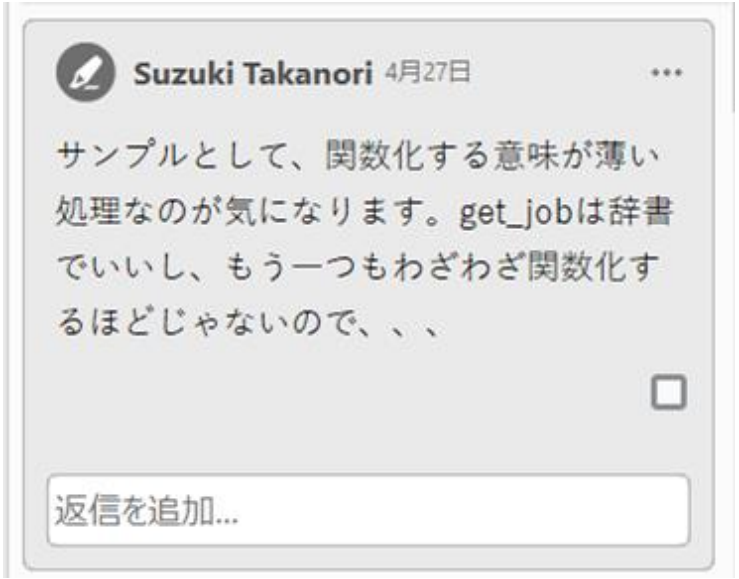
サンプルとして、関数化する意味が薄い処理なのが気になります。get_jobは辞書でいいし、もう一つもわざわざ関数化するほどじゃないので、、、

返信を追加...

7ページ 2

蔵本 俊郎 4月23日

PEP8的にはdocstring """""" (トリプルのダブルクオート)での記述が必要です。こ


Suzuki Takanori 4月27日
サンプルとして、関数化する意味が薄い処理なのが気になります。get_jobは辞書でいいし、もう一つもわざわざ関数化するほどじゃないので、、、
返信を追加...

関数内で少し加工する形に

関数から戻り値を返すようにしておくと、複数の関数を組み合わせて使う時に便利です。manuf_order関数とget_order関数を組み合わせて使うよう、manuf_order関数の呼び出し部分を書き換えてみましょう。

c7_1_5.py

```
001 def get_order(shuriken_num, makibishi_num):
002     order_text = ''
003     if shuriken_num > 0:
004         order_text += f'手裏剣{shuriken_num}個'
005     if makibishi_num > 0:
006         order_text += f'まきびし{makibishi_num}個'
007     return order_text
008
009
010 def manuf_order(name, order):
011     print(f'''{name}様
012 忍者です。
013 本日、{order}の製造をお願いします。''')
014
015
016 manuf_order('服部', get_order(3, 0)) ……引数を変更
017 manuf_order('河村', get_order(1, 4)) ……引数を変更
```

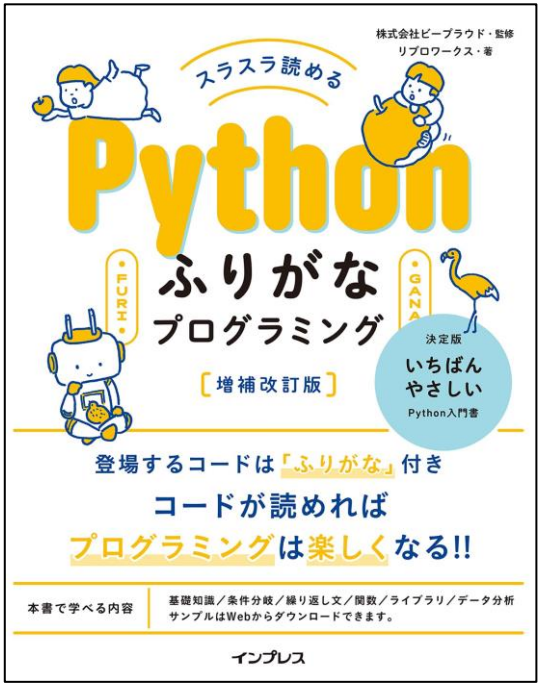
プログラムの中で、似たような処理を行っている部分は関数化できないか考える習慣を付けよう

インプレス「Pythonふりがな改訂」

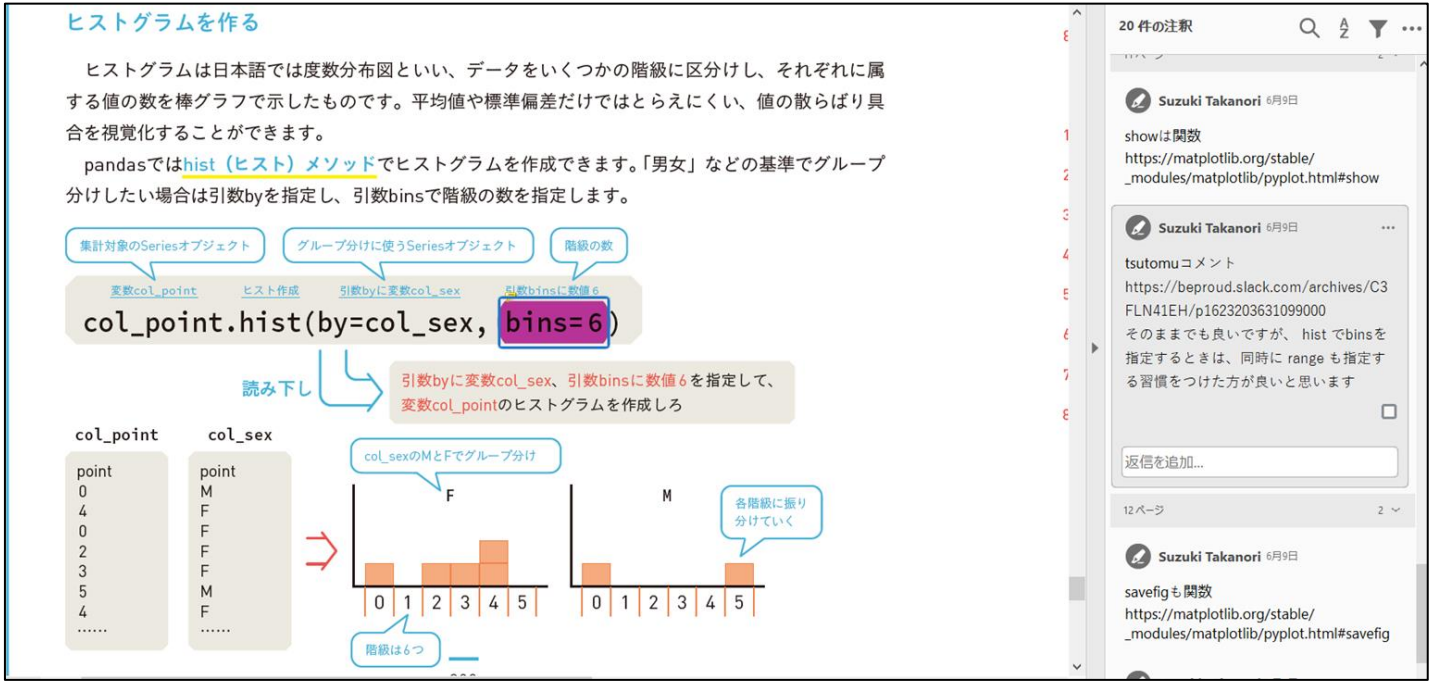
版型を大きくして、長いコードを読みやすく紙面調整

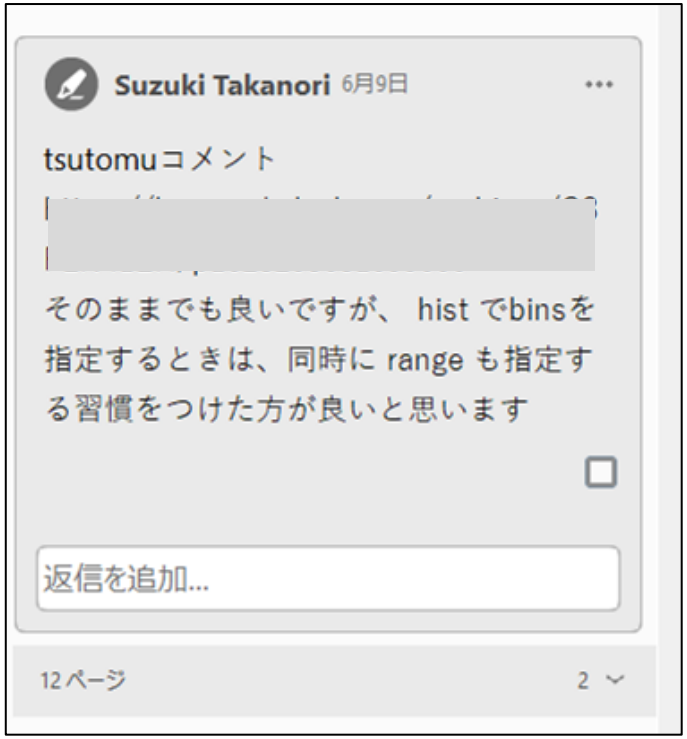
それに伴って応用的な内容の6章「データ分析」を追加

改訂版なので内容の変更度は小さめ



「binsと同時にrangeも指定したほうがよい」

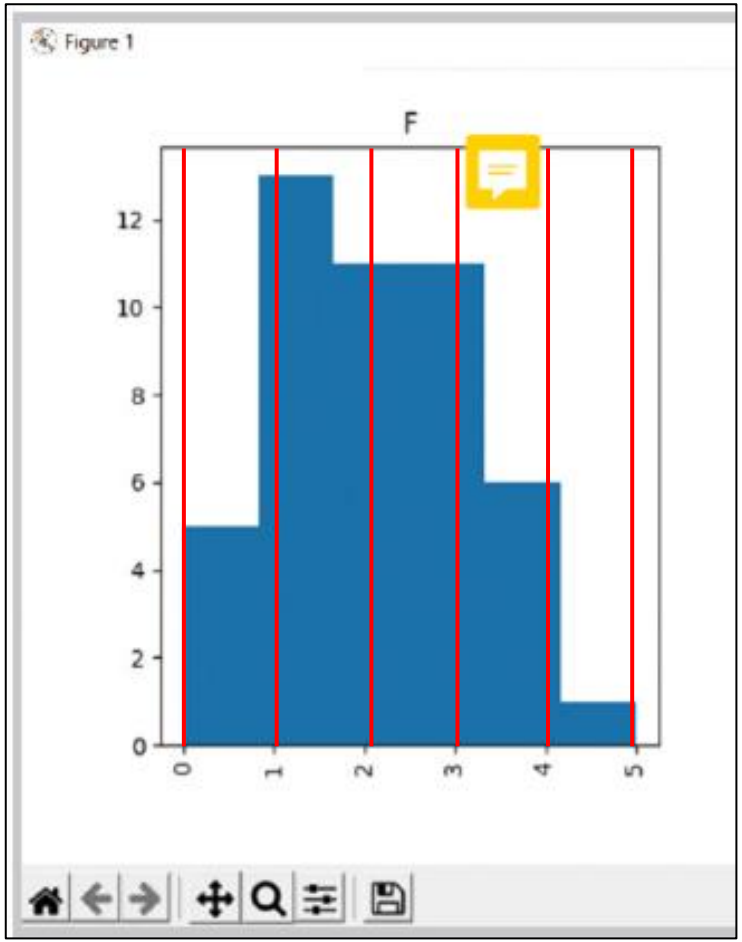



Suzuki Takanori 6月9日
tsutomuコメント
そのままでも良いですが、 hist でbinsを指定するときは、同時に range も指定する習慣をつけた方が良いと思います
返信を追加...
12 ページ
2

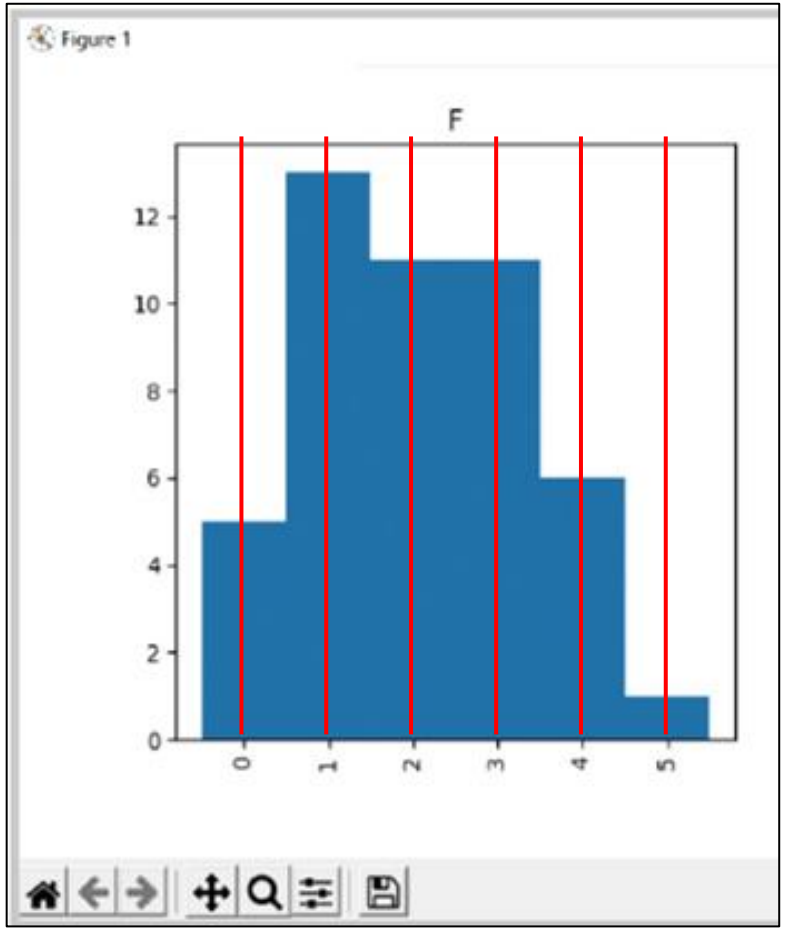
そのまま採用

```
    変数col_point    入れろ変数df0    文字列「point」
6   col_point = df0['point']
    変数col_point    ヒスト作成    引数byに変数col_sex    引数binsに数値6    引数rangeにタプル(-0.5, 5.5)
7   col_point.hist(by=col_sex, bins=6, range=(-0.5, 5.5))
pltモジュール  表示しろ
8   plt.show()
```

range引数なし



range引数あり



余談：こんな感じで監修してもらっています

- AtomのMarkdown用パッケージ（MDBP）とVivliostyleを使って、紙面イメージ（仮組みPDF）を作成
  - 実際にDTPを始める前に
  - https://vivliostyle.org/ja/
  - https://atom.io/packages/mdbp-markdown-book-preview
- Acrobat Proの共有レビュー機能を使って監修
  - リンク送信が可能。ブラウザでもAcrobat DC（無料版）でもレビューに参加できる
  - 他人のコメントに対してコメントできる
  - レビュー込みの状態でPDF形式でダウンロードできる
- 監修反映後にDTP

chap2.md — C:¥Users¥ohtsu¥Documents¥GitHub¥GH_kaiwaPython — Atom

ファイル(F) 編集(E) 表示(V) 選択(S) 検索(I) パッケージ(P) ヘルプ(H) File

untitled | chap2.md

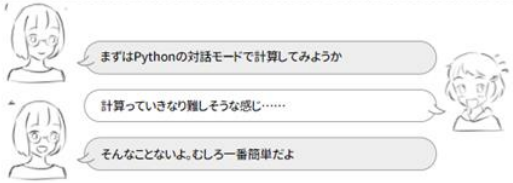
```
自分が使っている型を意識することが必要です。
40
41 ### 命令の種類—演算子、関数、メソッド
42
43 【先生f4】データ中心に説明するといっても、命令も少しは必要。この章では命令の基本的な使い方も説明するよ
44
45 Pythonの「命令」にあたるものには、**演算子、関数、メソッド**などがあります。次の第3章以降にもさまざまな命令が登場しますが、基本的な使い方は共通しています。この章ではそれらの基本的な使い方を覚えていきましょう。
46
47 @negative:4
48
49 【生徒f5】要するに、この章でやるのは全部基本の話ってことね
50
51 【先生f6】基本なくして応用なしよ。まぁ、あまり身構えずに一緒に体験していきましょう
52
53 ## 電卓のように計算してみよう
54
55 【先生f1】まずはPythonの対話モードで計算してみようか
56
57 【生徒f5】計算っていきなり難しそうな感じ……
58
59
60 【先生f2】そんなことないよ。むしろ一番簡単だよ
61
62 ### 対話モードで「値」を入力する
63 IDLEを起動してデータを入力してみましょう。プログラム内で扱うデータは、**値**と呼ぶことが多いので、以降は「値」とします。まずは**数値**です。何でもいいので、半角の数字だけを入力して〖Enter〗キーを入力してみてください。原則的にプログラムは**半角英数で入力してください**。
64
65 ###### 対話モード
66 ```py
```

30_genkou¥chap2.md 1:1 CRLF 3969 W | 31298 C CRLF UTF-8 MDBP Markdown main Fetch GitHub

Vivlios | librow | 進行中 | インプレ | chap2 | 127.0.0.1:8085/30_genkou/viewer/#src=../chap2.html&f=e... | 更新

4 / 49 Vivliostyle

2. 電卓のように計算してみよう



対話モードで「値」を入力する

IDLEを起動してデータを入力してみましょう。プログラム内で扱うデータは、**値**と呼ぶことが多いので、以降は「値」とします。まずは**数値**です。何でもいいので、半角の数字だけを入力して Enter キーを入力してみてください。原則的にプログラムは**半角英数で入力してください**。

対話モード

```
>>> 128 …… 128と入力
128
```

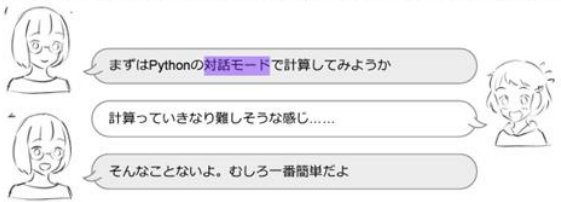
このような小数点を含まない数値のことを**整数(integer)**といいます。
次は小数点を含む数値を入力してみましょう。数字の途中に「.(ピリオド)」を入れてください。このような数値のことを**浮動小数点数(floating value)**ともいいます。

対話モード

```
>>> 45.3 …… 45.3と入力
45.3
```

今度は**文字列(string)**を入力しましょう。文字列とは、文字が集まってできた値のことです。前後を「'(シングルクォート)」か「"(ダブルクォート)」で囲むと文字列になります。

2. 電卓のように計算してみよう



対話モードで「値」を入力する

IDLEを起動して、値（Value）を入力してみましょう。値とはデータのことです。まずは**数値**です。何でもいいので、半角の数字だけを入力して Enter キーを入力してみてください。原則的にプログラムは**半角文字で入力してください**。

対話モード

```
>>>128 ----128と入力
128
```

次は小数点を含む数値を入力してみましょう。数字の途中に「.（ピリオド）」を入れてください。このような数値のことを**浮動小数点数（floating value）**ともいいます。

対話モード

```
>>>45.3
45.3
```

締め：実際のところ、今プロになっている人はどうやってプログラミング入門しましたか？

今、仕事なり趣味なりでプログラミングを書いている人、「わからなくても、とりあえず手を動かしてやってみる」ができた人が大半のような気がします。

書籍を読んでもよくわからなくて、実際にプログラム作ったり公式のドキュメントを読んでみたりいろいろやっているうちにふとわかる。その段階で書籍を読み返すと、ちゃんと説明書いてあったりする……なんて体験もシバシバ。

大昔の本に「習うより慣れろ」とあったが、今でも入門の最短ルートはそれなのかもしれません。「納得するまで手を動かさない」という人は、挫折することが多い……