pf-5. 式の抽象化と関数

(Python プログラミング入門演習，全６回)

https://www.kkaneko.jp/cc/pf/index.html

金子邦彦

アウトライン

5-1 式の抽象化

5-2 関数

5-1. 式の抽象化

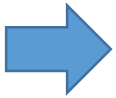
式の抽象化

100 * 1.1

150 * 1.1

400 * 1.1

類似した複数の**式**

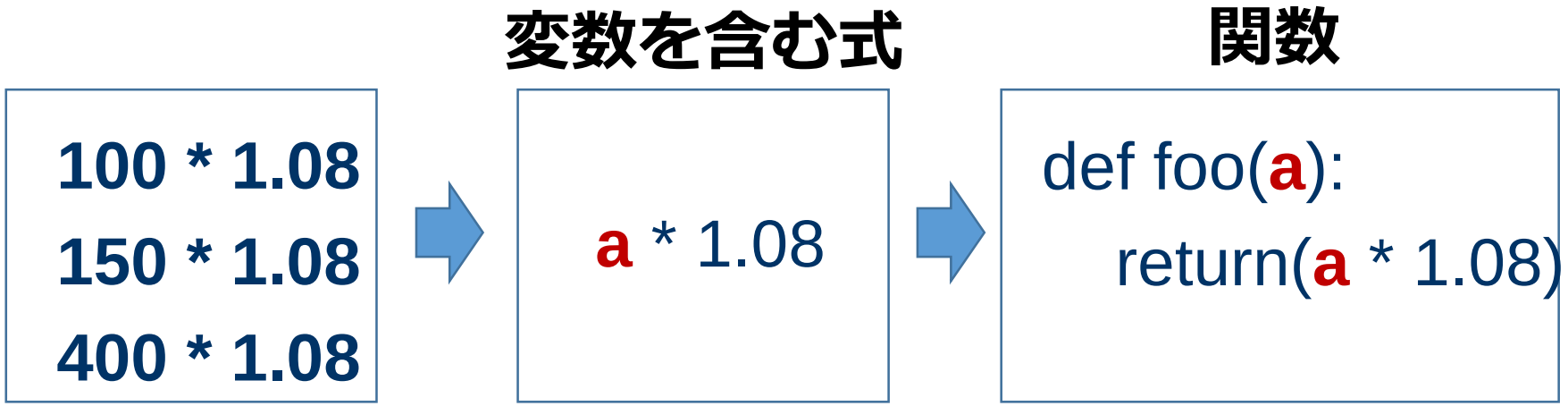


a * 1.1

変数 a を使って，複数の**式**を１つにまとめる（**抽象化**）

5-2. 関数

・式


変数を含む式
関数
100 * 1.08
150 * 1.08
400 * 1.08
a * 1.08
def foo(a):
return(a * 1.08)
抽象化
「変数を含む式に
名前（関数名）を
付けたものが
『関数』である」
と見立てることも

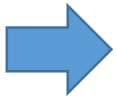
関数

100 * 1.1

150 * 1.1

400 * 1.1

類似した複数の**式**



a * 1.1

変数 a を使って，複数の**式**を１つにまとめる（**抽象化**）

```
1  def foo(a):
2      return a * 1.1
3  print(foo(100))
4  print(foo(150))
5  print(foo(400))
```

式「**a * 1.1**」を含む**関数 foo** を**定義**

関数 foo を使用.
100, 150, 400 は**引数**

Print output (drag lower righ

```
110.00000000000001
165.0
440.00000000000006
```

関数

```
def foo(a):
    return a * 1.1
```

- この**関数**の**本体**は
「**return a * 1.1**」

- この**関数**は，式「**a * 1.1**」に，**名前 foo** を付けたものと考えることもできる

式の抽象化と関数

抽象化前

```
1  print(100 * 1.1)
2  print(150 * 1.1)
3  print(400 * 1.1)
```

類似した複数の**式**

Print output (drag lower righ

```
110.00000000000001
165.0
440.00000000000006
```

実行結果

抽象化後

```
1  def foo(a):
2      return a * 1.1
3  print(foo(100))
4  print(foo(150))
5  print(foo(400))
```

関数の定義と使用

Print output (drag lower righ

```
110.00000000000001
165.0
440.00000000000006
```

同じ
実行結果になる

私の意見

- プログラミングでの根本問題は何でしょうか？
  - 誤り（バグ）の無いプログラムの作成
- プログラミングの一番の基礎は何でしょうか？
  - 抽象化を行うこと.
  - 抽象化により，繰り返し同じことを書くことが減り，バグを防げる.
  - プログラムの変更（消費税率10%変更）も簡単に.

実習の指示

- 資料：１２～１９
- 次のことを理解しマスターする
  - 式の抽象化と関数

実習

① ウェブブラウザを起動する

② Python Tutor を使いたいので，次の URL を開く

http://www.pythontutor.com/

※ Internet Explorer でうまく動かない場合がある

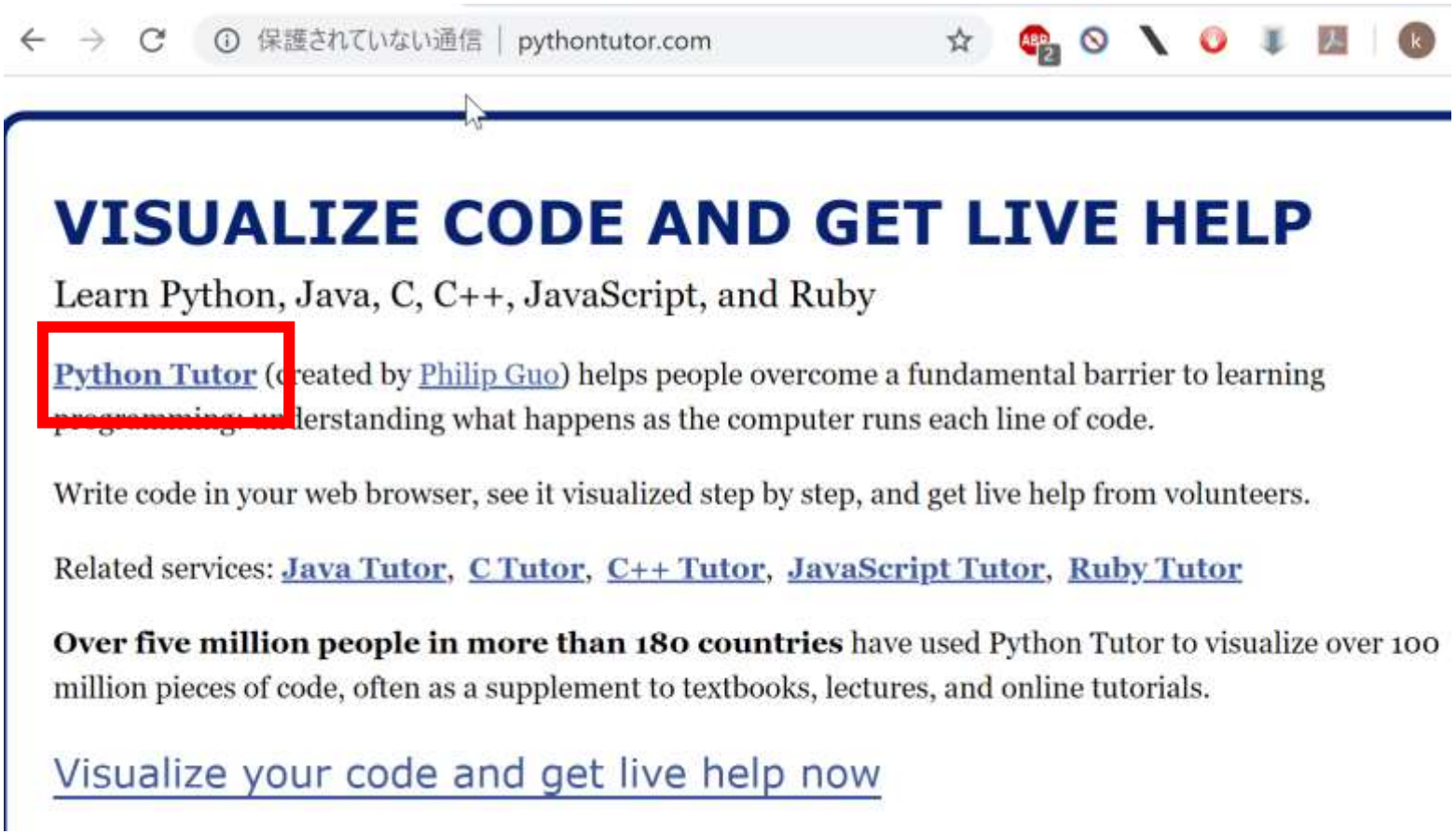
→ うまく動かないときは Google Chrome を試してください

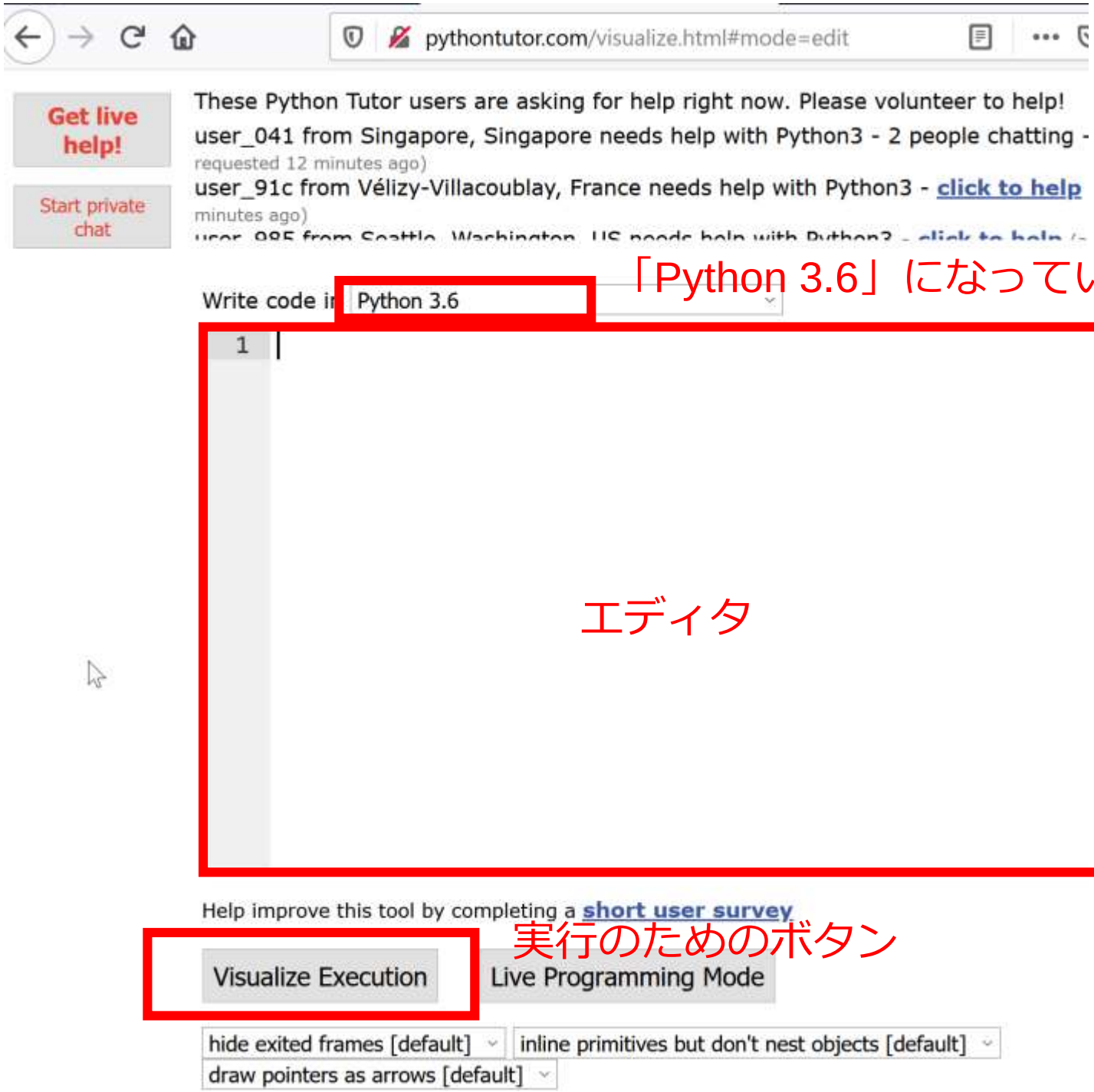
※ 途中で「Server Busy・・・」というメッセージが出ることがある.

→ 混雑している．少し（数秒から数十秒）待つと自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

※ 日本語モードはない．英語で使う

③ 「Python Tutor」をクリック

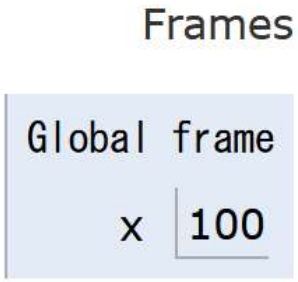



pythontutor.com/visualize.html#mode=edit
Get live help!
These Python Tutor users are asking for help right now. Please volunteer to help!
user_041 from Singapore, Singapore needs help with Python3 - 2 people chatting -
requested 12 minutes ago)
Start private chat
user_91c from Vélizy-Villacoublay, France needs help with Python3 - click to help
minutes ago)
「Python 3.6」になっている
Write code in
Python 3.6
1
エディタ
Help improve this tool by completing a short user survey
実行のためのボタン
Visualize Execution
Live Programming Mode
hide exited frames [default]
inline primitives but don't nest objects [default]
draw pointers as arrows [default]

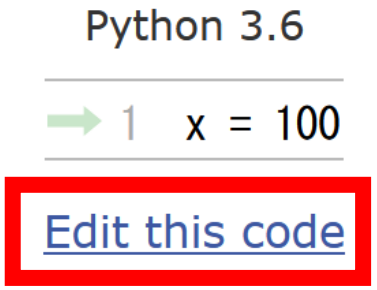
Python Tutor でのプログラム実行手順

(1)「**Visualize Execution**」をクリック.

(2)「**Last**」をクリック.



(3) **結果を確認**する.



(4)「**Edit this code**」をクリックして戻る

④ エディタで，次のプログラムを入れなさい

```
1  def foo(a):
2      return a * 1.1
3  print(foo(100))
4  print(foo(150))
5  print(foo(400))
```

「def foo(a)」の直後に「:」

字下げも正確に！「return a * 1.1」だけを字下げ

⑤ プログラムを実行し，結果を確認しなさい

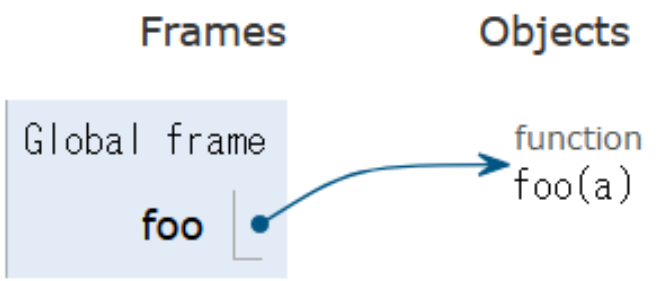
Python 3.6

```
1  def foo(a):
2      return a * 1.1
3  print(foo(100))
4  print(foo(150))
5  print(foo(400))
```

Edit this code

Print output (drag lower right corner to r

```
110.00000000000001
165.0
440.00000000000006
```



表示を確認

```
1  def foo(a):
2      return a * 1.1
3      print(foo(100))
4      print(foo(150))
5      print(foo(400))
```

これでは動かない

```
1  def foo(a):
2      return a * 1.1
3  print(foo(100))
4  print(foo(150))
5  print(foo(400))
```

これは**動く**

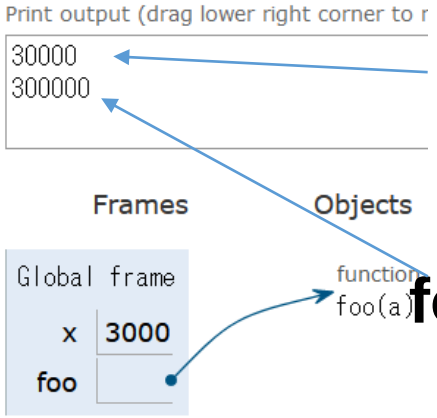
「delキー」などを使いながら書き換えてください

関数の中の式の評価のタイミング

Python 3.6

```
1  x = 30
2
3  def foo(a):
4      return a * x
5
6  x = 300
7  print(foo(100))
8
9  x = 3000
10 print(foo(100))
```

Edit this code



foo(100) の値は **30000**

a 100 **x** 300

foo(100) の値は **300000**

a 100 **x** 3000

関数の中の**式**「a * x」の評価では,
最新の a の**値**, **最新**の x の**値**が用いられる