

# pf-9. 関数呼び出し

(Python 入門)

URL: <https://www.kkaneko.jp/pro/pf/index.html>

金子邦彦



# 関数の中の式の評価のタイミング



```
trinket Run ? Modules  
main.py  
1 x = 50  
2  
3 def foo(a):  
4     return a * x  
5  
6 x = 400  
7 print(foo(100))  
8  
9 x = 3000  
10 print(foo(100))
```

Powered b  
40000  
300000

foo(100) の値は 40000  
a 100 x 400

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

- Trinket は**オンライン**の Python、HTML 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）
- **Python の標準機能**を登載、その他、次のモジュールやパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

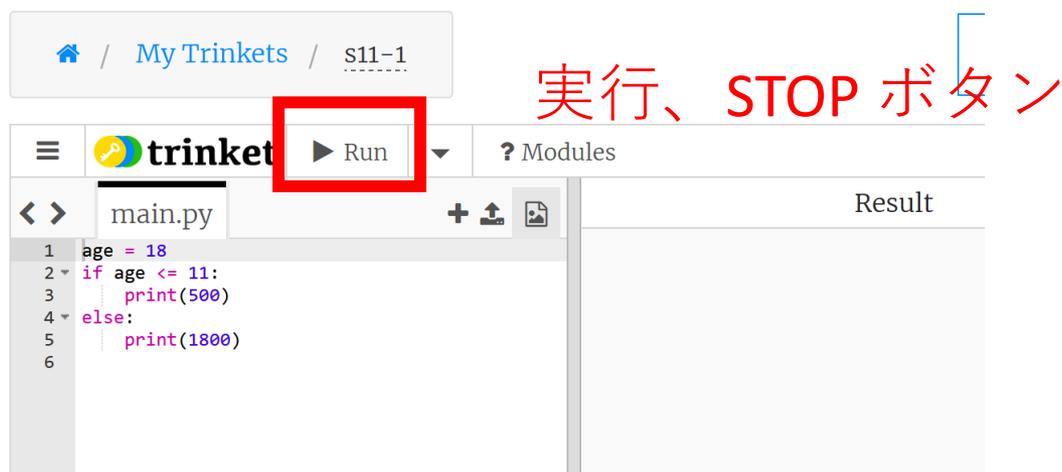


# trinket でのプログラム実行

- trinket は Python, HTML などのプログラムを書き実行できるサイト

- <https://trinket.io/python/0fd59392c8>

のように、違うプログラムには違う URL が割り当てられる



ソースコードの  
メイン画面

実行結果

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを書き替えて再度実行することも可能

# 演習

資料：6

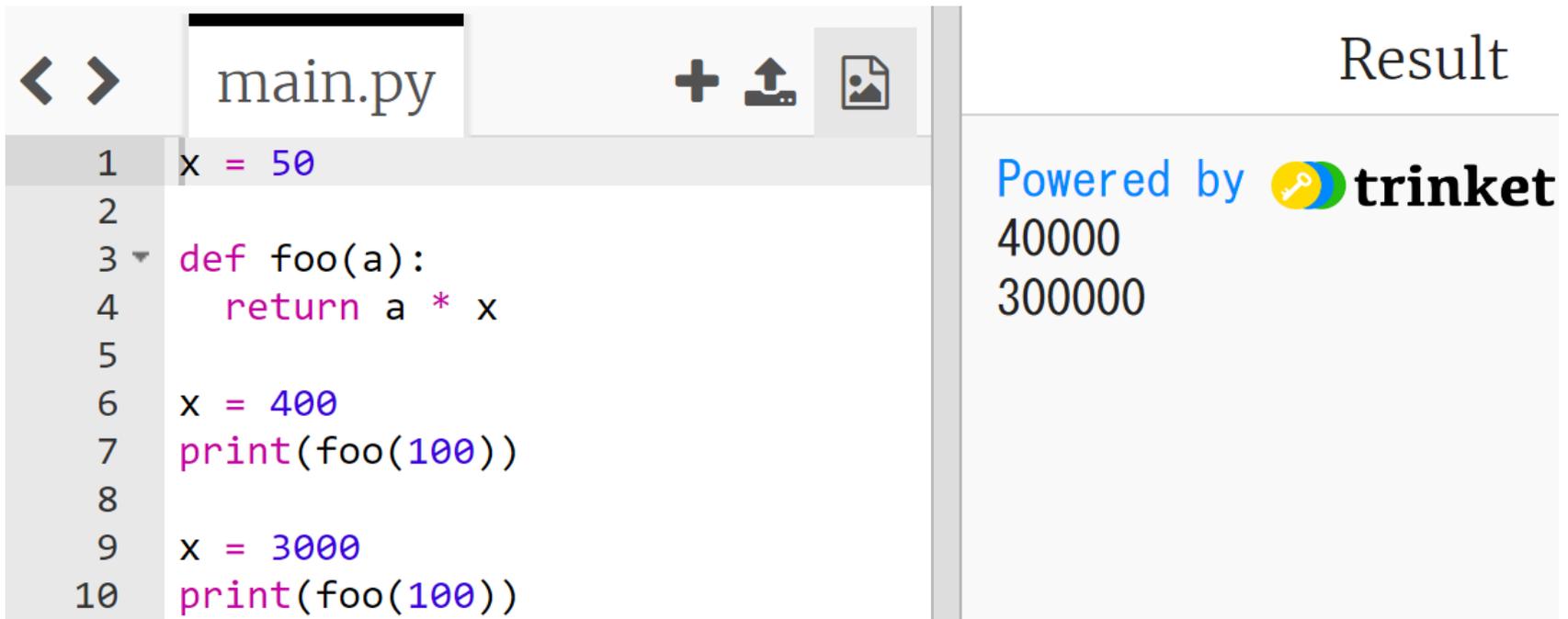
【トピックス】

- ・ 関数呼び出し

① trinket の次のページを開く

<https://trinket.io/python/d60de93fb8>

② 実行結果が、次のように表示されることを確認



The screenshot shows a Python code editor window titled 'main.py' with the following code:

```
1 x = 50
2
3 def foo(a):
4     return a * x
5
6 x = 400
7 print(foo(100))
8
9 x = 3000
10 print(foo(100))
```

To the right of the code editor is a 'Result' panel displaying the output of the code execution:

```
Powered by  trinket
40000
300000
```

## ステップ実行

ステップ実行により、プログラム  
実行の流れをビジュアルに観察

# Python Tutor

- Python Tutor というウェブサイトを利用しよう

<http://www.pythontutor.com/>

- Web ブラウザを使ってアクセスできる

- PythonTutor では, Pythonだけでなく, Java, C,, C++, JavaScript, Ruby など, 多くのプログラミング言語を学ぶことができる.



Python 3.6  
[known limitations](#)

```
1 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6 s = 0
7 for i in [1, 2, 3,
8     s = s + i
9 print(s)
```

[Edit this code](#)

just executed  
to execute

<< First < Prev Next >

Done running (16 s)

# Python Tutor の使用方法



- ① まず, **ウェブブラウザ**を開く
- ② **Python Tutor** を利用するために, 以下の URL にアクセス

**<http://www.pythontutor.com/>**

- ③ 「**Python**」 をクリック ⇒ **編集画面**が開く

---

## Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by [visualizing code execution](#). You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

**Over 15 million people in more than 180 countries** have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

# Python Tutor の編集画面



Python debugger - [pdb](#) interface to Python Tutor - Learn Python by visualizing code (also debug [JavaScript](#), [Java](#), [C](#), and [C++](#) code)

Write code in  「Python 3.6」になっている

1 |

**エディタ**  
(プログラムを書き換えることができる)

実行のためのボタン

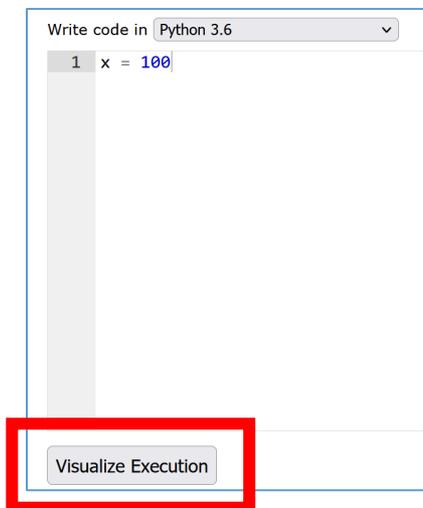
[Show code examples](#)



# Python Tutor でのプログラム実行

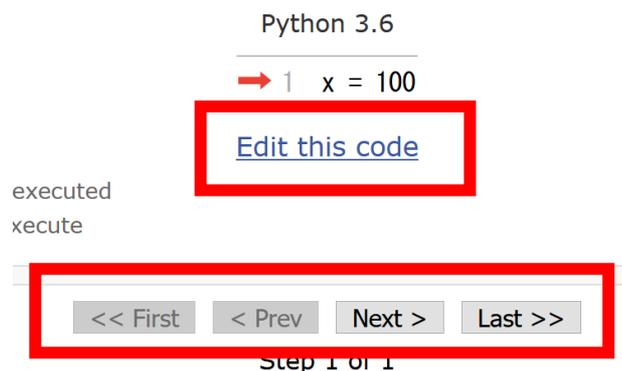
- Python Tutor は Python などのプログラムを書き実行できるサイト。ステップ実行、変数の値表示などの機能がある。
- Python Tutorのウェブサイトアクセス。「Python」を選択  
<https://www.pythontutor.com/>

メイン画面で、プログラムを書く



Visualize Execution ボタン

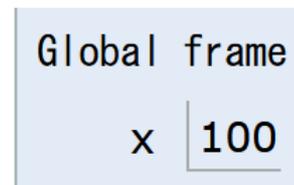
メイン画面に  
戻るには  
Edit this code



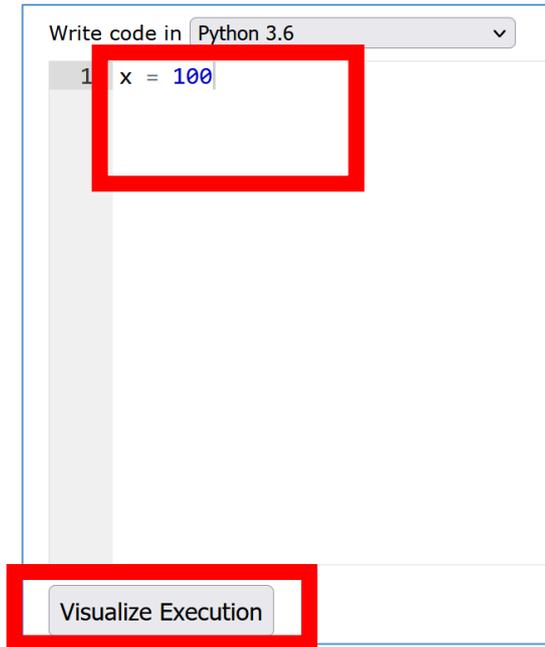
通常実行: Last  
ステップ実行: 他のボタン

変数の値を  
視覚的に  
確認できる

Frames



# Python Tutor でのプログラム実行手順



(1) 「**Visualize Execution**」をクリックして**実行画面**に切り替える

(2) 「**Last**」をクリック。



(3) 実行結果を確認する。

(4) 「**Edit this code**」をクリックして**編集画面**に戻る

# Python Tutor 使用上の注意点①



実行画面で、**赤いエラーメッセージ**が出ることがある

過去の文法ミスに関する確認表示。

基本的には、**無視して問題ない**

邪魔なときは「**Close**」

Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Python 3.6  
([known limitations](#))

→ 1 x = 100

[Edit this code](#)

→ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Step 1 of 1

[Customize visualization](#)

Frames Objects

You just fixed the following error:

```
x 1 x = 100!
```

**SyntaxError: invalid syntax (<string>, line 1)**

Please help us improve this tool with your feedback.  
What misunderstanding do you think caused this error?

Submit Close Hide all of these pop-ups

# Python Tutor 使用上の注意点②



「please wait ... executing」のとき，10秒ほど待つ。



- Python Tutor が混雑しているとき，「Server Busy . . .」 と表示される場合がある。
- このメッセージは，サーバが混雑していることを示す。
- 数秒から数十秒待つと自動で処理が始まるはずですが（しかし，表示が変わらないときは，操作をもう一度試してください）

## 演習

資料：16 ~ 24

### 【トピックス】

- Python Tutor
- ステップ実行
- 関数呼び出しにおけるジャンプ
- 関数内で使用される変数が消えるタイミング

# ステップ実行により確認できること

ステップ実行により，ジャンプの様子を観察

ジャンプの様子を示す矢印

```
Python 3.6
1 age = 30
2 if age <= 12:
3     print(500)
4 else:
5     print(1200)
```

[Edit this code](#)

Print output (drag lower right)

Frames

Global frame

age 30

変数の値の確認もできる

Step 3 of 3

<< First < Prev Next > Last >>

ステップ実行は、ボタンやスライダーでコントロール

# Python Tutor の起動



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  
**<https://www.pythontutor.com/>**

③ 「**Python**」 をクリック ⇒ **メイン画面**が開く

## Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by [visualizing code execution](#). You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

**Over 15 million people in more than 180 countries** have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

④ Python Tutor のエディタで次のプログラムを入れ、実行し、結果を確認する（あとで使うので消さないこと）

```
def foo(a):  
    return a * 1.1  
print(foo(100))  
print(foo(150))  
print(foo(400))
```



Print output (drag lower right)

```
110.00000000000001  
165.0  
440.00000000000006
```

結果を確認

（計算誤差がある。  
動作は正常）

「return a \* 1.1」の行は字下げが必要

「Visual Execution」をクリック。そして「Last」をクリック。結果を確認。  
「Edit this code」をクリックすると、エディタの画面に戻る

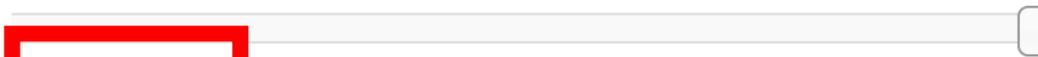
## ⑤ 「First」 をクリックして, 最初に戻る

Python 3.6  
([known limitations](#))

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

[Edit this code](#)

xecuted  
ecute



Navigation buttons: << First, < Prev, Next >, Last >>

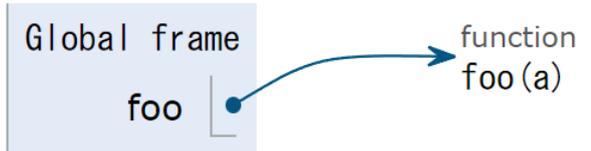
Done running (13 steps)

Print output (drag lower right corner to resize)

```
110.00000000000001  
165.0  
440.00000000000006
```

Frames

Objects



⑥ 「**Step 1 of 13**」と表示されているので、  
全部で、ステップ数は 13 あることが分かる  
(ステップ数と、プログラムの行数は違うもの)

Python 3.6  
([known limitations](#))

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

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First

< Prev

Next >

Last >>

Step 1 of 13

Print output (drag lower right corner)

Frames

Objects

## ⑦ 最初に戻したので

- すべての**オブジェクト**は消えている
- **赤い矢印**は**先頭**のところに戻っている

Python 3.6  
([known limitations](#))

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

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First

< Prev

Next >

Last >>

Step 1 of 13

Print output (drag lower right corner)

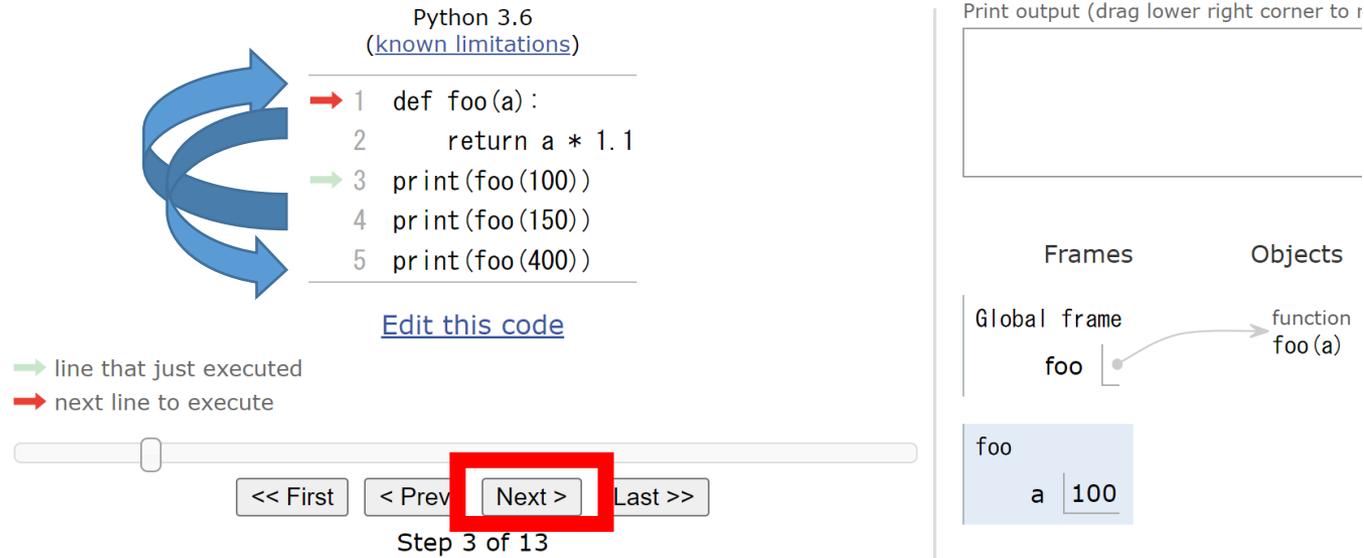
Frames

Objects

⑧ **ステップ**実行したいので、「Next」をクリックしながら、**矢印の動きを確認**.

※「Next」ボタンを何度か押し、それ以上進めなくなったら終了

見どころ  
foo との間で  
**ジャンプ**するところ



Python 3.6  
([known limitations](#))

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

[Edit this code](#)

→ line that just executed  
→ next line to execute

Print output (drag lower right corner to r

Frames      Objects

Global frame      function foo(a)

foo

a 100

<< First   < Prev   **Next >**   Last >>

Step 3 of 13

# ⑨ 終わったら、もう一度、「First」をクリックして、最初に戻る



Python 3.6  
([known limitations](#))

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

[Edit this code](#)

→ line that just executed  
→ next line to execute

Progress bar: [-----] [ ]

<< First | < Prev | Next > | Last >>

Done running (13 steps)

Print output (drag lower right corner to

```
110.00000000000001  
165.0  
440.00000000000006
```

Frames

Objects

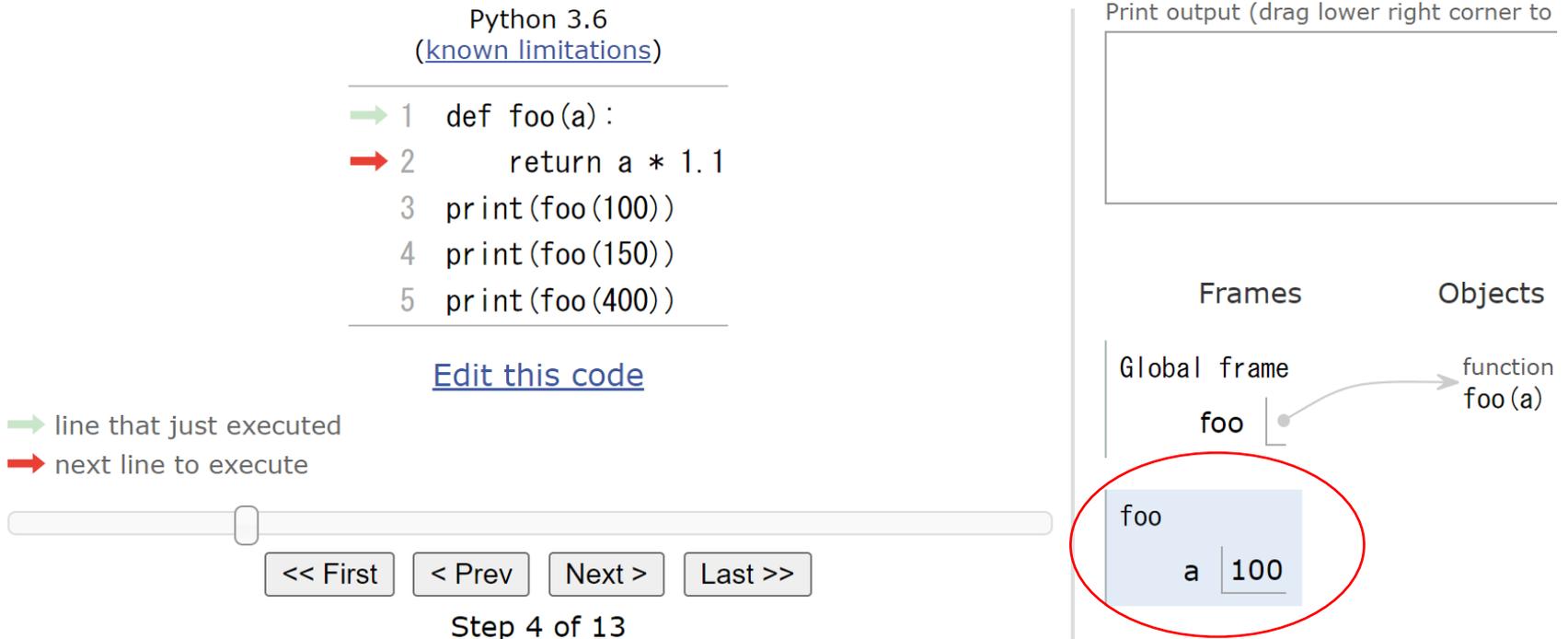
Global frame  
foo

function  
foo(a)

# ⑩ もう一度、ステップ実行.

今度は、**緑の矢印**を見ながら、**変数 a が生成、消去**されるタイミングを確認

**緑の矢印** : いま実行が終わった行



The screenshot shows a Python 3.6 IDE interface. On the left, a code editor displays the following code:

```
Python 3.6  
(known limitations)  
1 def foo(a):  
2     return a * 1.1  
3 print(foo(100))  
4 print(foo(150))  
5 print(foo(400))
```

A green arrow points to line 1, and a red arrow points to line 2. Below the code is a legend: a green arrow for "line that just executed" and a red arrow for "next line to execute". A progress bar and navigation buttons (<< First, < Prev, Next >, Last >>) are at the bottom, with "Step 4 of 13" displayed.

On the right, the "Print output" area is empty. Below it, the "Frames" and "Objects" panels are visible. The "Frames" panel shows a "Global frame" with a variable "foo" pointing to a "function foo(a)". The "Objects" panel shows the "function foo(a)". A red circle highlights a frame for "foo" containing the variable "a" with the value "100".

**関数 foo の実行中は、  
変数 a が現れる**