

# 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



The screenshot shows the trinket online Python editor interface. The file 'main.py' contains 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))
```

The right pane shows the output of the code execution:

Powered by  
40000  
300000

Two blue arrows point from the printed results to annotations on the right:

- An arrow points from the result '40000' to the annotation 'foo(100) の値は 40000 a 100 x 400'.
- An arrow points from the result '300000' to the annotation 'foo(100) の値は 300000 a 100 x 3000'.

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、書き替えて実行也可能）
- Python の標準機能を登載、その他、次の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

The screenshot shows the trinket.io web interface. At the top, there's a navigation bar with a home icon, 'My Trinkets', and a file name 's11-1'. Below the bar is the trinket logo and a 'Run' button, which is highlighted with a red box. To the left is a code editor window titled 'main.py' containing the following Python code:

```
1 age = 18
2 if age <= 11:
3     print(500)
4 else:
5     print(1800)
```

To the right of the code editor is a 'Result' panel where the output of the program will be displayed.

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

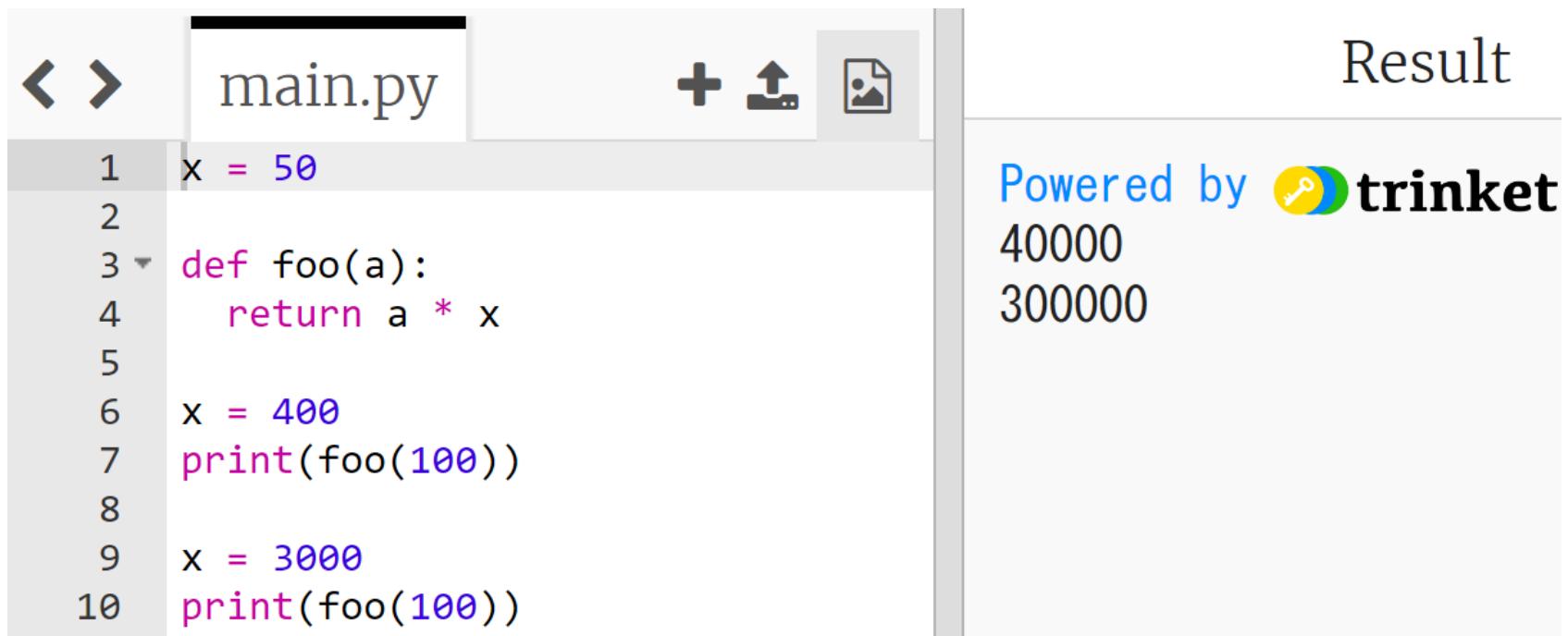
## 【トピックス】

- ・関数呼び出し

① trinket の次のページを開く

<https://trinket.io/python/d6ode93fb8>

② 実行結果が、次のように表示されることを確認



The screenshot shows the trinket.io interface. On the left, there is a code editor window titled "main.py" containing the following Python code:

```
x = 50
def foo(a):
    return a * x
x = 400
print(foo(100))
x = 3000
print(foo(100))
```

On the right, under the "Result" tab, the output is displayed as:

Powered by  trinket

Output
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](#)

<< 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 を利用するためには、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

「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)

Visualize Execution

実行のためのボタン

hide exited frames [default] ▾

inline primitives, don't nest objects [default] ▾

draw pointers as arrows [default] ▾

[Show code examples](#)

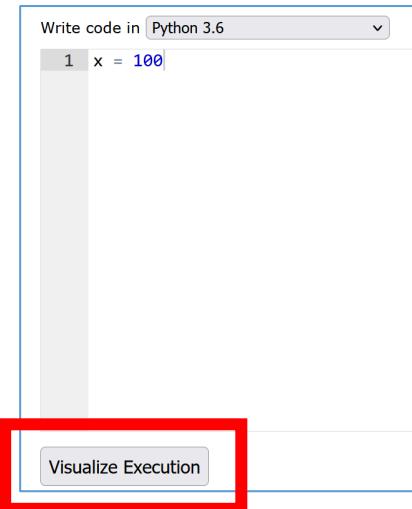
Generate permanent link



# 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 のウェブサイトにアクセス。「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

変数の値を  
視覚的に  
確認できる

Python 3.6

→ 1 x = 100

Edit this code

executed  
execute

<< First < Prev Next > Last >>

Step 1 of 1

Global frame

x 100

Frames

Global frame

x 100

通常実行: Last

ステップ実行: 他のボタン

# Python Tutor でのプログラム実行手順



Write code in Python 3.6

```
1 x = 100
```

Visualize Execution



Python 3.6

```
→ 1 x = 100
```

Edit this code

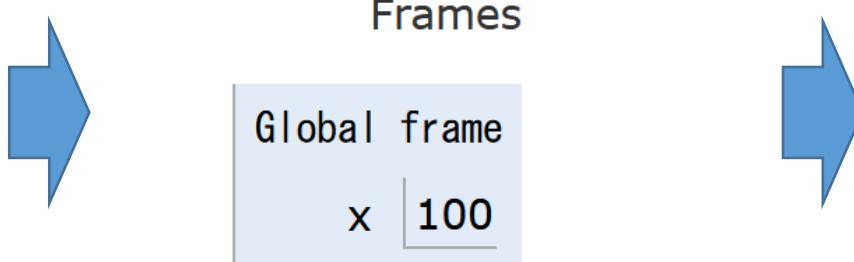
executed  
execute

<< First < Prev Next >> Last >>

Step 1 of 1



(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.

Python 3.6 (known limitations)

```
→ 1 x = 100
```

Edit this code

green line that just executed  
red next line to execute

(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: Visualize code in Python

Please wait ... your code is running (up to 10 seconds)

Write code in Python 3.6

```
1 k = 100
```

Please wait ... executing (takes up to 10 seconds)

A screenshot of the Python Tutor web application. At the top, it says "Python Tutor: Visualize code in Python". Below that is a message box with a red border containing the text "Please wait ... your code is running (up to 10 seconds)". Underneath is a text input field labeled "Write code in Python 3.6" with the code "1 k = 100" entered. At the bottom, there's another message box with a grey border containing the text "Please wait ... executing (takes up to 10 seconds)".

- 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（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



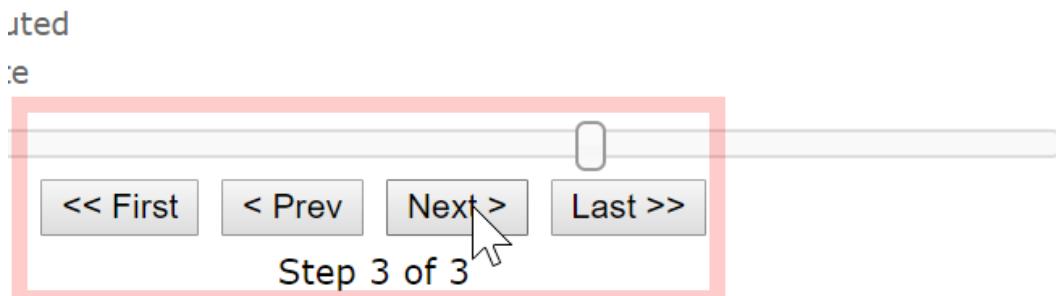
ステップ実行により、ジャンプの様子を観察

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](#)



ステップ実行は、  
ボタンやスライダーでコントロール

Print output (drag lower right)

Frames

Global frame

age 30

変数の値の  
確認もできる

# 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（あとで使うので消すこと）



```
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

<< First

< Prev

Next >

Last >>

Done running (13 steps)

Print output (drag lower right corner to resize)

```
110. 0000000000000001  
165. 0  
440. 0000000000000006
```

Frames

Objects

Global frame  
foo

function  
foo(a)



## 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 > >>

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

<< First < Prev Next > Last >>

Step 3 of 13

Print output (drag lower right corner to r)

Frames Objects

Global frame function foo(a)  
foo

foo a 100

# 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

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**

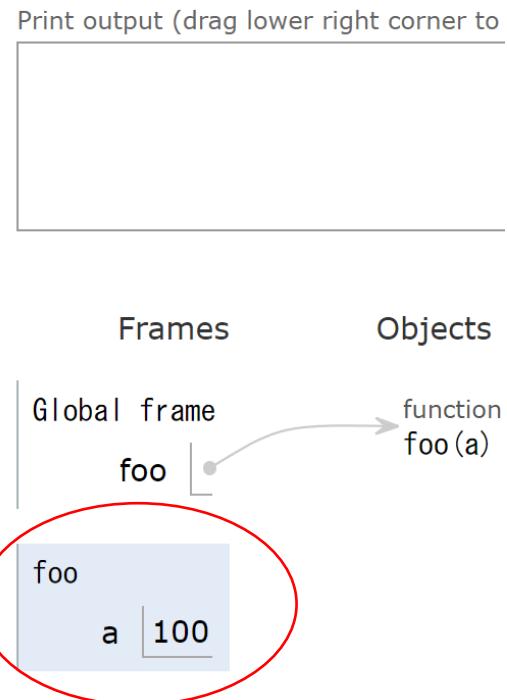
Python 3.6  
([known limitations](#))

```
→ 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 4 of 13

関数 foo の実行中は、  
変数 a が現れる