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

- **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

- 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

main.py

```
x = 50

def foo(a):
  return a * x

x = 400
print(foo(100))

x = 3000
print(foo(100))
```

Result

Powered by trinket

```
40000
300000
```

ステップ実行

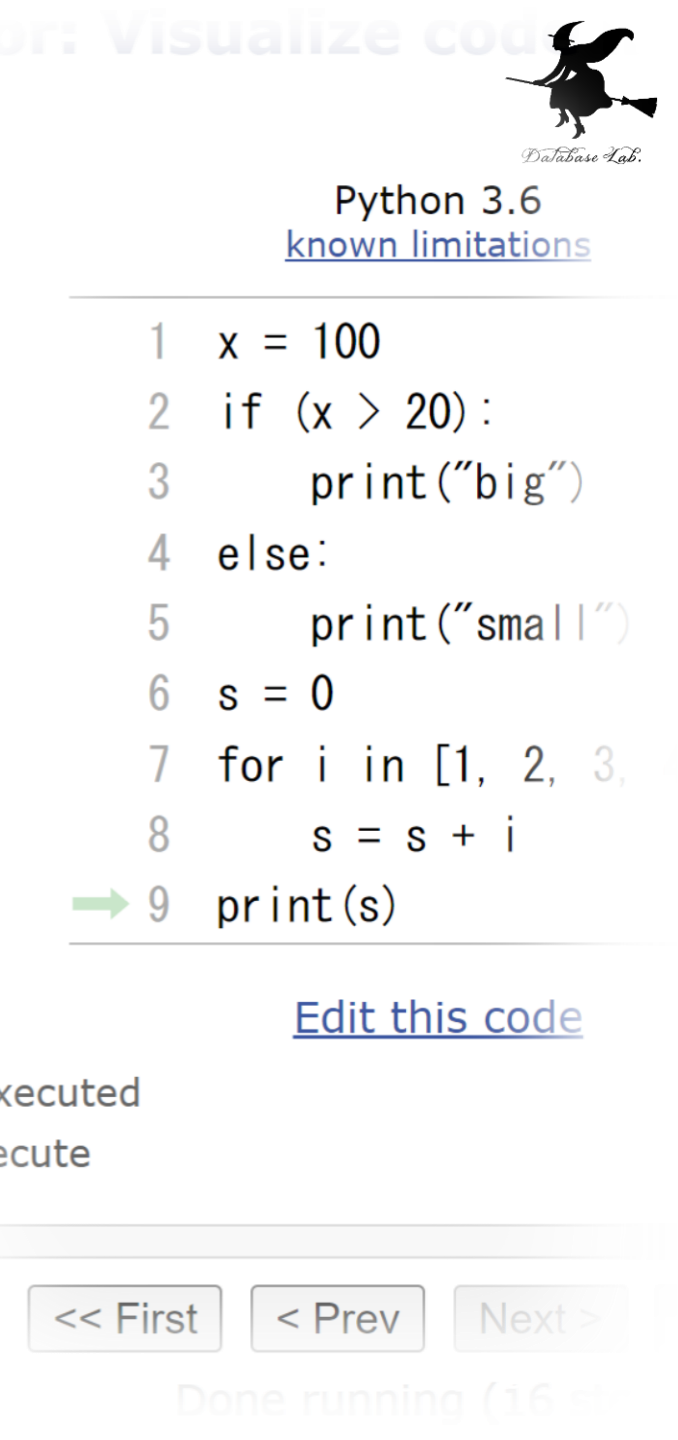
ステップ実行により、プログラム実行の流れをビジュアルに観察

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.



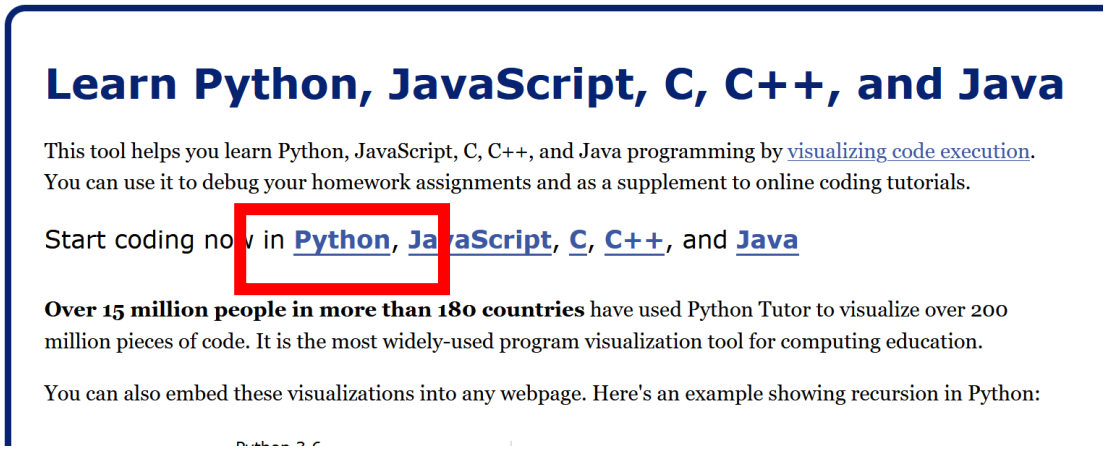
Python Tutor の使用方法

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



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]

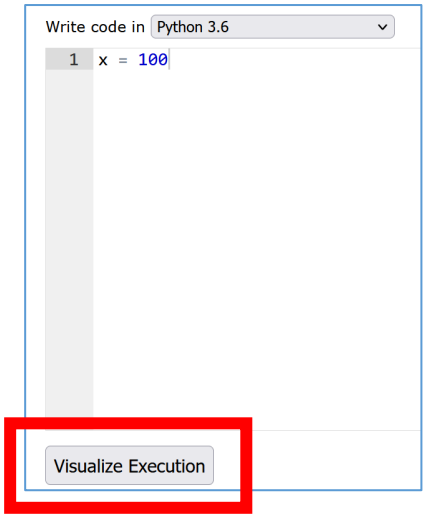
Show code examples

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/

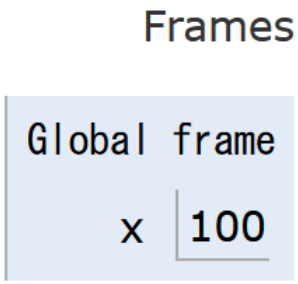
メイン画面で、プログラムを書く



Visualize Execution ボタン

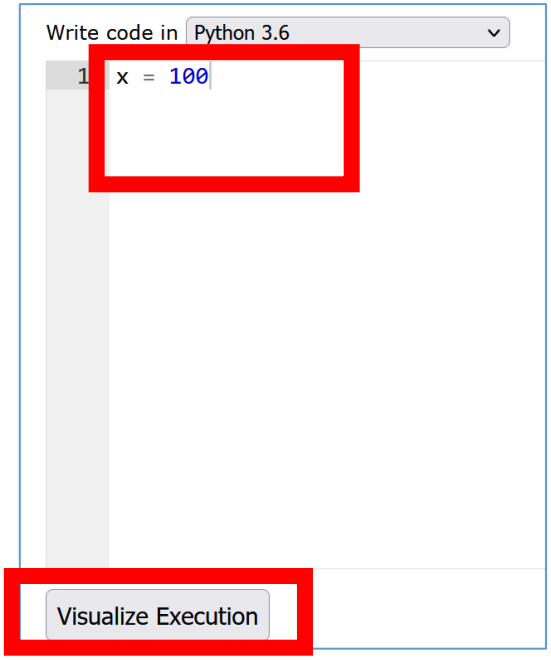
メイン画画に
戻るには
Edit this code

変数の値を
視覚的に
確認できる



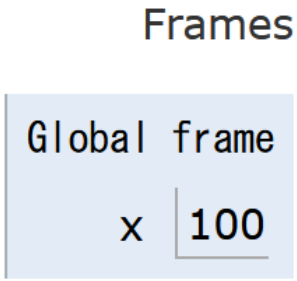
通常実行: Last
ステップ実行: 他のボタン

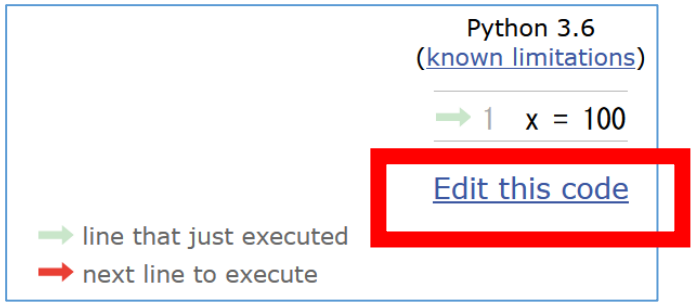
Python Tutor でのプログラム実行手順



(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) **<u>実行結果を確認</u>**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

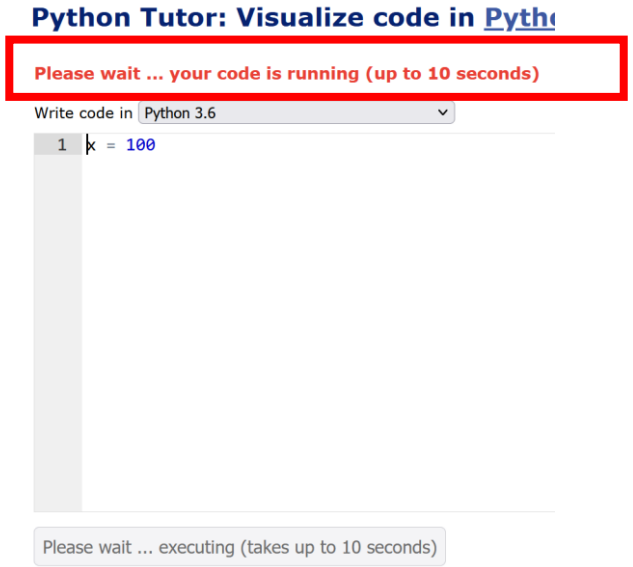
過去の文法ミスに関する確認表示.

基本的には，**無視**して問題ない

邪魔なときは「**Close**」

Python Tutor 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（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

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

Step 3 of 3

ステップ実行は、ボタンやスライダーでコントロール

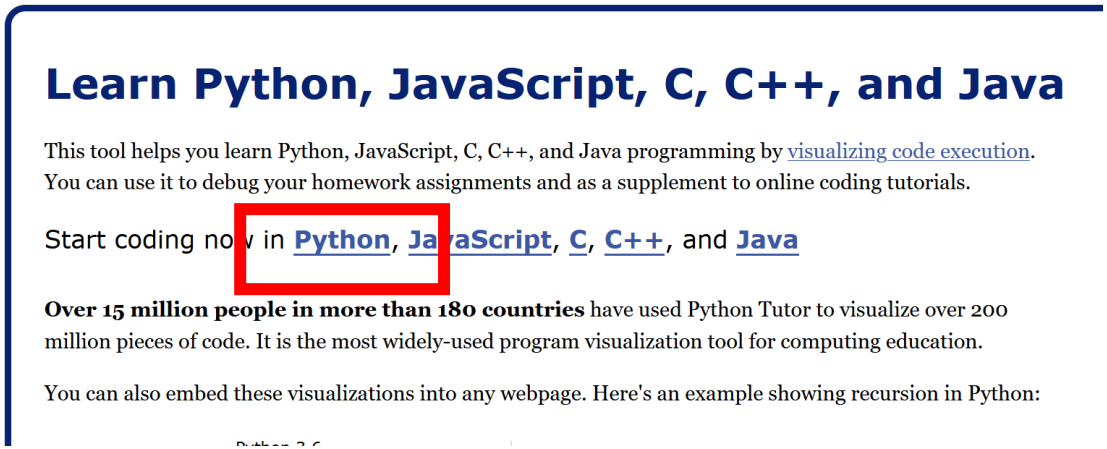
Python Tutor の起動

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

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

Done running (13 steps)

Print output (drag lower right corner to resize)

```
110.00000000000001
165.0
440.00000000000006
```

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 > | Last >>

Step 1 of 13

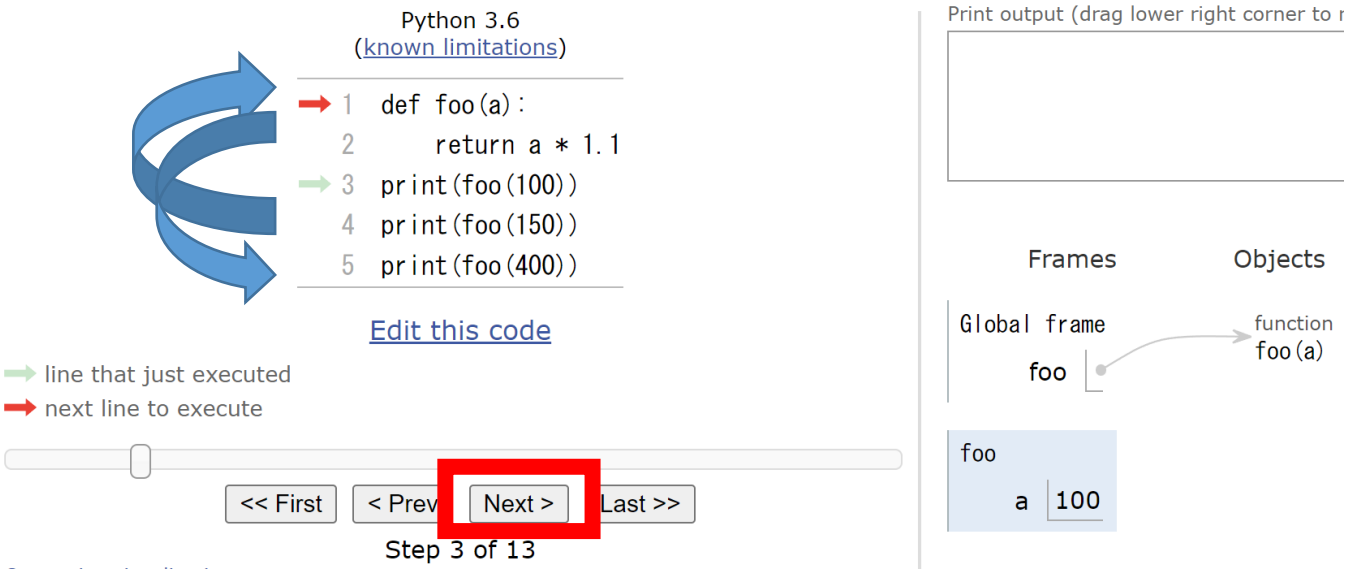
Print output (drag lower right corner

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

<u>見どころ</u>
foo との間で
ジャンプするところ



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

<< 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

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

Print output (drag lower right corner to

Frames | Objects

Global frame
foo → function foo(a)

foo
a 100

関数 foo の実行中は，
変数 a が現れる