



pf-2. Python 言語, 式, 変数

(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

2-1. Python 言語

2-2. 変数, データ型

2-3. 式と変数



2-1. Python 言語



問いかけ

- Python 言語とは何でしょうか？
- Python 言語のアプリケーションを動かすのに、どういう手順を踏むでしょうか？



Python 言語の 特徴

- 「入門者に学習しやすい」とされる
- 多数の拡張機能
- Python 言語システムのソースコードは公開されている

※ この資料では、**バージョン 3** を使用

Python の動作



```
x = 100
if x > 20:
    print("big")
else:
    print("small")

s = 0
for i in range(1, 6):
    s = s + i
print(s)
```

Python のソースコード

コンパイル (ビルド) と実行は
一度に行われる

→
コンパイル
(ビルド)

バイトコード

Python 仮想マシン

標準ライブラリ

全部がそろって,
1つのアプリケーション

これで, プラットフォーム
非依存を達成



2-2. 変数, データ型



- プログラミング学習を行えるオンラインサービス

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

- Web ブラウザを使う

- たくさんの言語を扱うことができる

Python, Java, C, C++, JavaScript, Ruby など



実習の指示

- 資料：10～14
- Python Tutor に関する次のことを理解しマスターする
 - Python Tutor の起動手順
 - Python Tutor の画面構成
 - Python Tutor は、オンラインのプログラム開発環境であること

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

← → ↻ ⓘ 保護されていない通信 | pythontutor.com ☆ APP 2

VISUALIZE CODE AND GET LIVE HELP

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

[Python Tutor](#) (created by [Philip Guo](#)) helps people overcome a fundamental barrier to learning programming: understanding what happens as the computer runs each line of code.

Write code in your web browser, see it visualized step by step, and get live help from volunteers.

Related services: [Java Tutor](#), [C Tutor](#), [C++ Tutor](#), [JavaScript Tutor](#), [Ruby Tutor](#)

Over five million people in more than 180 countries have used Python Tutor to visualize over 100 million pieces of code, often as a supplement to textbooks, lectures, and online tutorials.

[Visualize your code and get live help now](#)



Get live help!

Start private chat

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)
user_91c from Vélizy-Villacoublay, France needs help with Python3 - [click to help](#) minutes ago)
user_985 from Seattle, Washington, US needs help with Python3 - [click to help](#) (-

「Python 3.6」になっている

Write code in Python 3.6

1 |

エディタ
(プログラムを書き換えることができる)

Help improve this tool by completing a [short user survey](#)

Visualize Execution

Live Programming Mode

実行のためのボタン

hide exited frames [default] v

inline primitives but don't nest objects [default] v

draw pointers as arrows [default] v

パソコン演習. 計算



① Python Tutor のエディタで, 次のように入れる

```
print(100 * 200)
```

Write code in Python 3.6

```
1 print(100 * 200)
2
```

すべて **半角文字**

「*」は掛け算の記号

② 「Visual Execution」をクリック. そして「Last」をクリック. 結果を確認

The screenshot shows the Python Tutor interface. On the left, the code editor contains the code `print(100 * 200)`. Below the editor, there are buttons for 'Visualize Execution' and 'Live Programming Mode'. In the center, the 'Visual Execution' window shows the code being executed, with a red arrow pointing to the first line. Below this window, there are navigation buttons: '<< First', '< Prev', 'Next >', and 'Last >>'. The 'Last >>' button is highlighted with a red box. On the right, the 'Print output (drag)' window displays the result '20000', which is also highlighted with a red box.

パソコン演習. 計算



③ 「Edit this code」 をクリックすると, エディタの画面に戻る

The image shows a two-part interface. On the left, a code preview window displays the code `print(100 * 200)` with a red box around the `print(100 * 200)` line and a red arrow pointing to it. Below the code is a navigation bar with buttons for `<< First`, `< Prev`, `Next >`, and `Last >>`, and the text `Step 1 of 1`. A blue arrow points from this preview to the right. On the right, a larger code editor window is shown. At the top, it says 'Write code in Python 3.6'. The code area contains two lines: `1 print(100 * 200)` and `2`. Below the code area, there is a link to 'Help Improve this tool by completing a [short user survey](#)' and two buttons: 'Visualize Execution' and 'Live Programming Mode'.



データの種類

- データには種類がある
 - 整数
 - 浮動小数
 - 文字列
 - 辞書
 - 集合
 - bool (True/False)
 - バイト列 (バイナリともいう)

Python のデータの種類とデータ型名 (クラス名)



データの種類	データ型名 (クラス名)
整数	int
浮動小数	float
	complex
文字列	str
辞書	dict
集合	set
	frozenset
bool	bool
バイト列	bytes
	bytearray
	memoryview



Python の変数

- 変数には, データの値を代入できる.

```
a = 100
```

```
x = a * 20
```

- 「a = 100」のように書くと, x の値が 100 に変化する
- データの値は, いずれか1つのデータ型に分類される
- データ型名 (クラス名) を type 関数で取得できる

実習の指示

- 資料：19～22
- 次のことを理解しマスターする
 - 変数
 - データ型



変数 x の値を 100 に変化させる

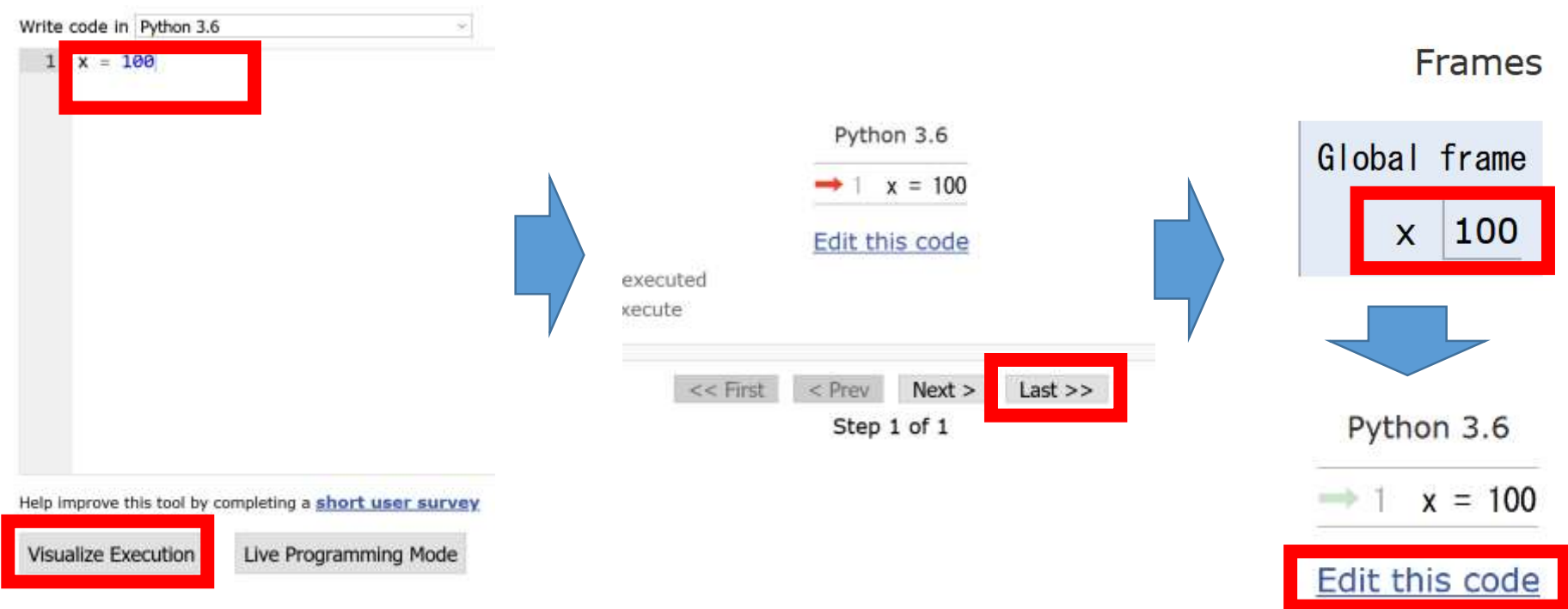
① エディタに, 次のように「x = 100」を入れる

Write code in Python 3.6

```
1 x = 100
```



- ② 「Visualize Execution」 をクリック.
- 「Last」 をクリック.
- 結果を確認する. 「x 100」となっている.
- 「Edit this code」 をクリックして戻る





③ 「a = 10」 に書き換えて、実行し、結果を確認しなさい

Write code in Python 3.6

```
1 a = 10
2 |
```

Frames

Global frame

a	10
---	----

④ 「b = 20」 に書き換えて、実行し、結果を確認しなさい

Write code in Python 3.6

```
1 b = 20
2 |
```

Frames

Global frame

b	20
---	----



変数のデータ型名を確認する

⑤ プログラムを次のように書き換えて、実行し、結果を確認しなさい

Write code in `Python 3.6`

```
1 b = 20
2 print(type(b))
```

Print output (drag to)

```
<class 'int'>
```

Frames

Global frame

b	20
---	----

⑥ プログラムを次のように書き換えて、実行し、結果を確認しなさい

Write code in `Python 3.6`

```
1 x = 1.2
2 print(type(x))
```

Print output (drag to)

```
<class 'float'>
```

Frames

Global frame

x	1.2
---	-----

まとめ



```
In [1]: x = 100  
  
In [2]: print(x)  
100  
  
In [3]: y = 200  
  
In [4]: print(y)  
200
```

Python コンソールの画面

- 変数は、データの値を覚えておくためのメモリ
- 変数には名前（変数名）がある
- 代入: 「x = 100」のように書くと、変数 x の値が 100 に変化する
- Python には、種々のデータ型がある
- データの値は、いずれか1つのデータ型に分類される



2-3. 式と変数

式と変数



- 式の実行結果として、値が得られる

```
1 print(100 * 1.1)
2 print(150 * 1.1)
3 print(200 * 1.1)
```



Print output (drag lower right corner to resize)

```
110.000000000000001
165.0
220.000000000000003
```

複数の**式**

Frames

Objects

実行結果

- 式の中に、変数名を書くことができる

```
r = 3
```

```
print( r * r * 3.14 )
```

式

実習の指示

- 資料：27～31
- 次のことを理解しマスターする
 - 式と変数

式と変数



① 次のように書き換えて、「Visualize Execution」をクリック。

「Last」ボタンをクリック。「300」と表示されるので確認する。「Edit this code」をクリックして戻る

```
1 x = 100
2 y = 200
3 print( x + y )
4
```



Use the Back and Forward buttons to jump there.

Step 1 of 3 Forward Last >>



Print output (drag lower right corner to resize)

300

Frames Objects

Global frame

x	100
y	200



Python 3.6

```
1 x = 100
2 y = 200
→ 3 print( x + y )
```

[Edit this code](#)



② 次のように書き換えて，実行し，結果を確認しなさい

Write code in Python 3.6

```
1 x = 100
2 y = 200
3 print( x * y )
4
```



Print output (drag lower r

20000

Frames

Global frame

x	100
y	200

結果の
「20000」が表示
されるので確認



③ 次のように書き換えて、実行し、結果を確認しなさい

/rite code in Python 3.6

```
1 x = 100
2 y = 200
3 print( ( x + 10 ) * y )
4
```



Print output (drag lo

22000

Frames

Global frame

x 100

y 200

結果の
「22000」が表示
されるので確認

三角形の面積



底辺が 2.5 で、高さが 5 のとき、
三角形の面積は、面積： 6.25

- ④ 次のように書き換えて、実行し、結果を確認しなさい

Write code in Python 3.6

```
1 teihen = 2.5
2 takasa = 5
3 print( teihen * takasa / 2 )
```

Print output (drag low)

6.25

Frames

Global frame

teihen	2.5
takasa	5

結果の
「6.25」が表示
されるので確認

円の面積



円周率を 3.14 とする. 半径が 3 の円
円の面積は : 28.26

⑤ 次のように書き換えて、実行し、結果を確認しなさい

Write code in Python 3.6

```
1 r = 3
2 print( r * r * 3.14 )
```



Print output (drag lo

28.26

Frames

Global frame

r 3

結果の
「28.26」が表示
されるので確認