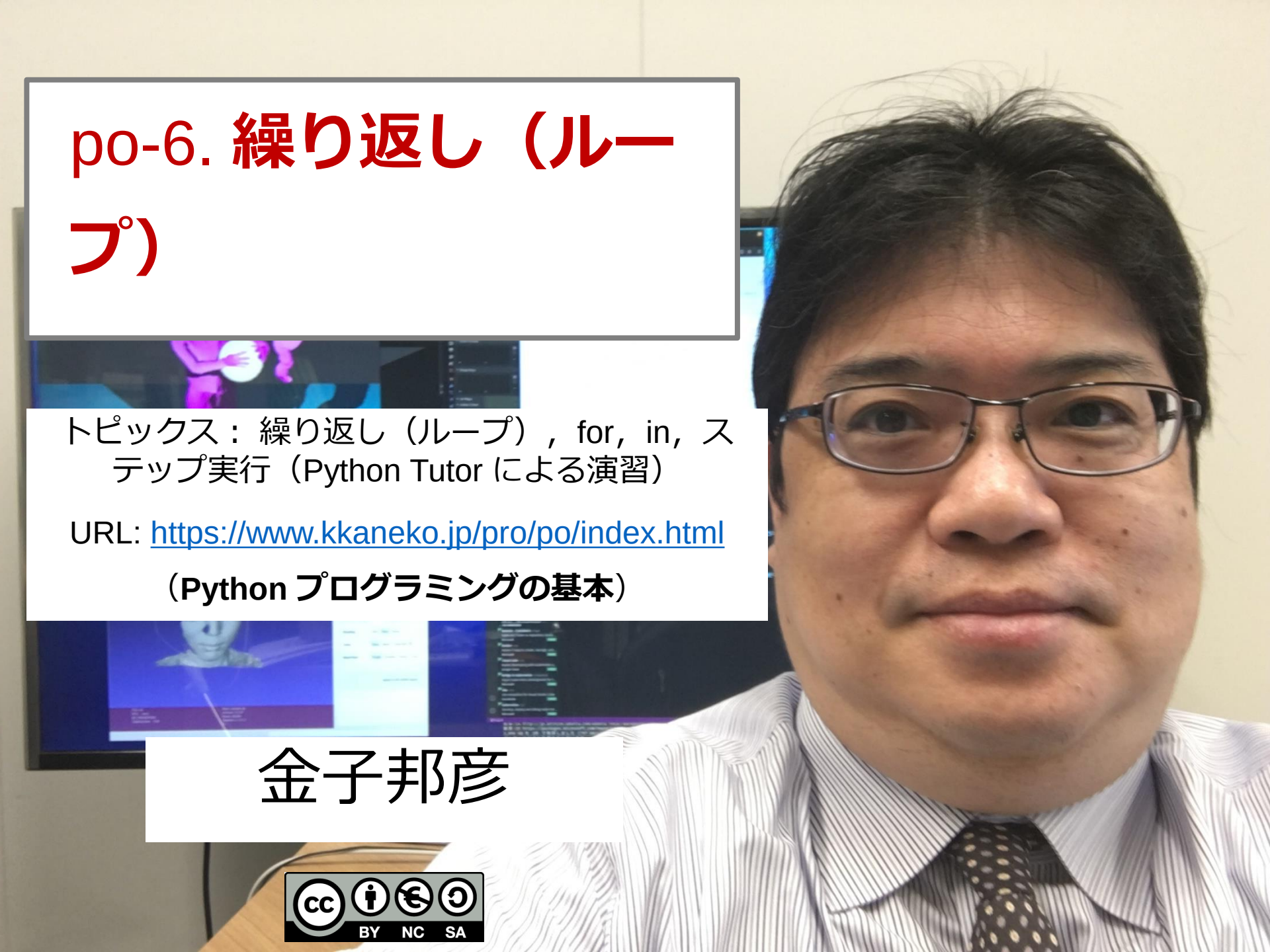
po-6. 繰り返し（ループ）



トピックス： 繰り返し（ループ），for，in，ステップ実行（Python Tutor による演習）

URL: https://www.kkaneko.jp/pro/po/index.html

（Python プログラミングの基本）

金子邦彦

物体の落下距離： **9.8 × (時間)² ÷ 2**

時間は 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

⇒ 同じ式の計算を 11 回繰り返し

```
1  x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2  for t in x:
3      print((9.8 / 2) * t * t)
```

Python プログラム

実行結果

Print output (drag lower right corner to resize)

```
0.0
4.9
19.6
44.1
78.4
122.5
176.4
240.1
313.6
396.9
490.0
```

全体まとめ

- **処理の繰り返し**により，**リスト**や**辞書**のすべての要素を処理できる
- **リスト**や**辞書**で**処理の繰り返し**を行うときは，**for**が便利である
- for による**繰り返し（ループ）**
  同じ処理や操作を繰り返す

```
1  x = [8, 6, 4, 2, 3]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
```

アウトライン

	項目
	復習
6-1	リストと繰り返し（ループ）
6-2	ステップ実行
6-3	辞書と繰り返し（ループ）

メソッドアクセス，代入

Python プログラムの例

```
x = 100
a = x + 200
enermy1 = hero.findNearestEnemy()
hero.attack(enemy1)
```

- **代入**：**オブジェクト名** ＋ 「**=**」
  ＋ 式または値またはメソッド呼び出し
- **メソッドアクセス**：**オブジェクト名** ＋ 「**.**」
  ＋ **メソッド名** ＋「**()**」 （引数を付けることも）

Python プログラムでは，その他にも，属性アクセス，関数呼び出し，制御，「*」，「+」などの演算子，コマンド，定義など

式の抽象化と関数，条件分岐，繰り返し（ループ）

- **式**の抽象化と関数

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

```
Print output (drag lower right
110.00000000000001
165.0
440.00000000000006
```

実行結果

- 条件分岐

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

x > 20 のとき***のみ***

print("big") が実行される

x ≦ 20 のとき***のみ***

print("small") が実行される

- 繰り返し（ループ）

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

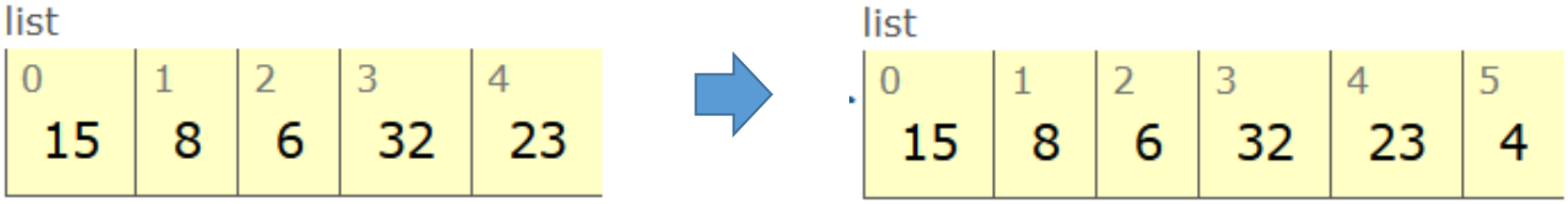
足し算の 5回繰り返し

0 + 1, 1 + 2, 3 + 3, 6 + 4, 10 + 5

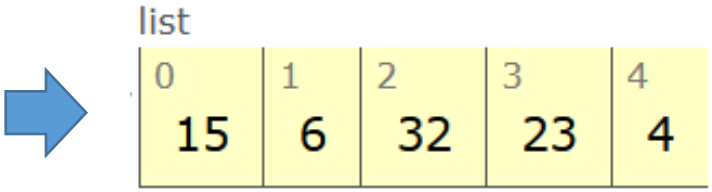
リスト

- **リスト**は，**同じ型の要素の並び**
- **リスト**の要素には**順序がある**．０から始まる番号（**添字**）が付いている
- リストは，要素の挿入，削除により，サイズが増減する

4 を末尾に挿入



8 の削除



辞書

- **辞書**は，**キー**と**値（バリュー）**の**ペア**の集まり
- **辞書**に，**同じ値**の**キー**は**2回以上登場しない**

4, "Orange" の削除　　**3 を "Black"** に置き換え

1	"Red"
2	"Yellow"
3	"Blue"

1	"Red"
2	"Yellow"
3	"Blue"
4	"Orange"

1	"Red"
2	"Yellow"
3	"Black"
4	"Orange"

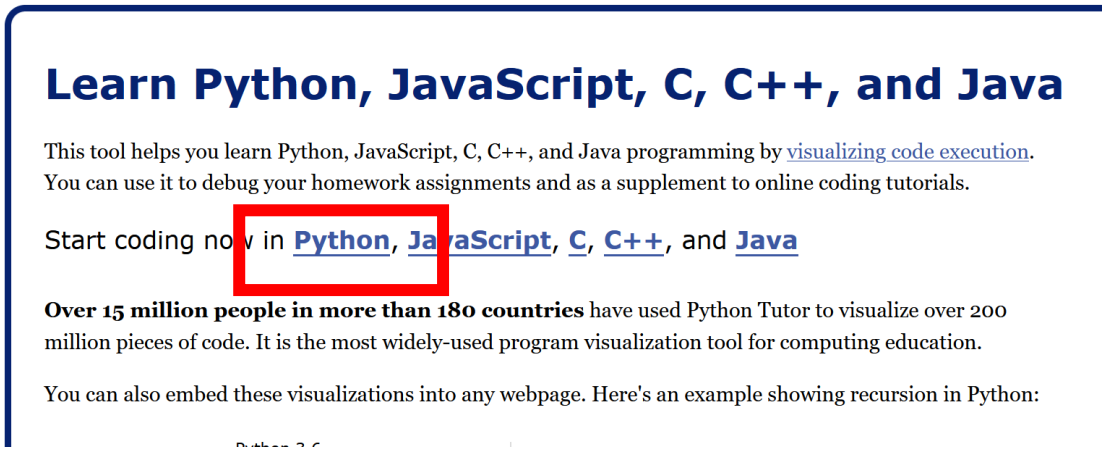
Python Tutor の起動

① **ウェブブラウザ**を起動する

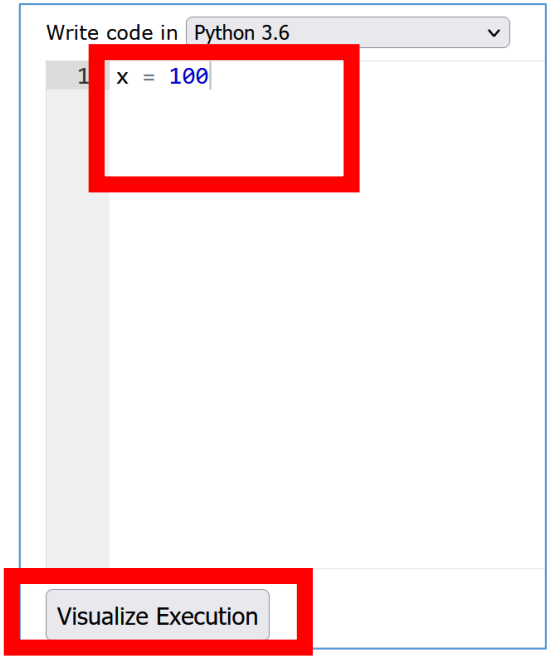
② **Python Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

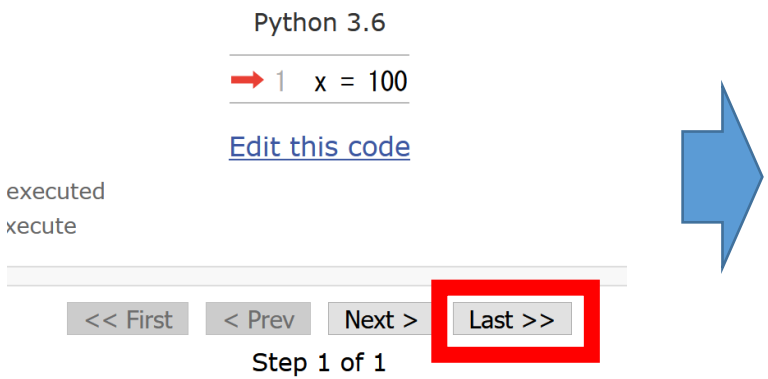
③ 「**Python**」をクリック　⇒ **編集画面**が開く



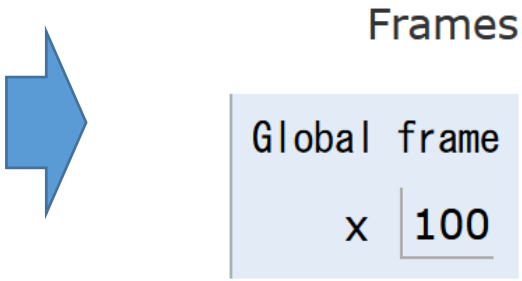
Python Tutor でのプログラム実行手順



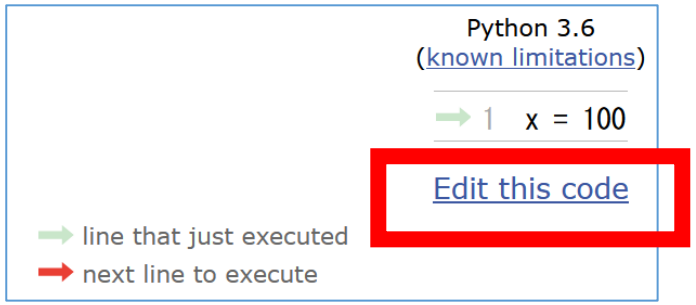
(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える



(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

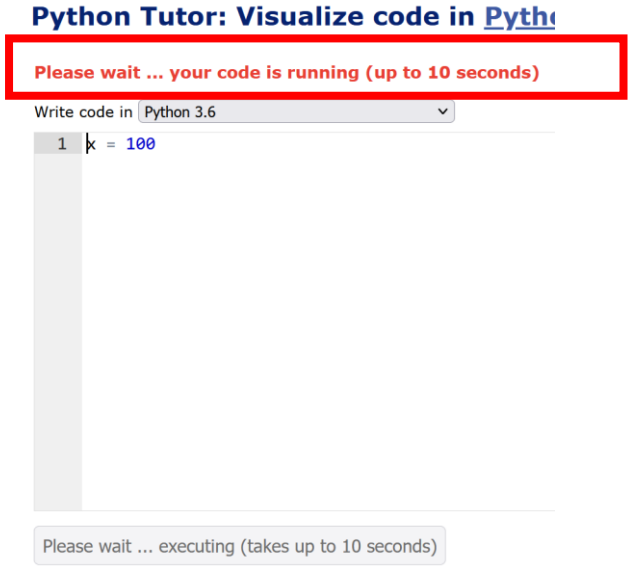
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  過去の文法ミスに関する確認表示

  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**.



→　**混雑しているとき**は，**「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）

6-1. リストと繰り返し（ループ）

繰り返し（ループ）

繰り返し（ループ）では，**同じ処理や操作を繰り返す**

物体の落下距離： **9.8 × (時間)² ÷ 2**

時間は 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

⇒ 同じ式の計算を 11 回繰り返し

```
1  x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2  for t in x:
3      print((9.8 / 2) * t * t)
```

Python プログラム

実行結果

```
Print output (drag lower right corner to resize)
0.0
4.9
19.6
44.1
78.4
122.5
176.4
240.1
313.6
396.9
490.0
```

繰り返しのプログラム例

```
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for t in x:
        print((9.8 / 2) * t * t)
```

リストの組み立て

繰り返し

for t in x の直後に「:」
（コロン）

字下げも正確に.
print((9.8 / 2) * t * t) の前に，「タブ」を 1つだけ

演習

資料：**17 ～ 20**

【トピックス】

- **for による繰り返し**

① Python Tutor のエディタで次のプログラムを入れる

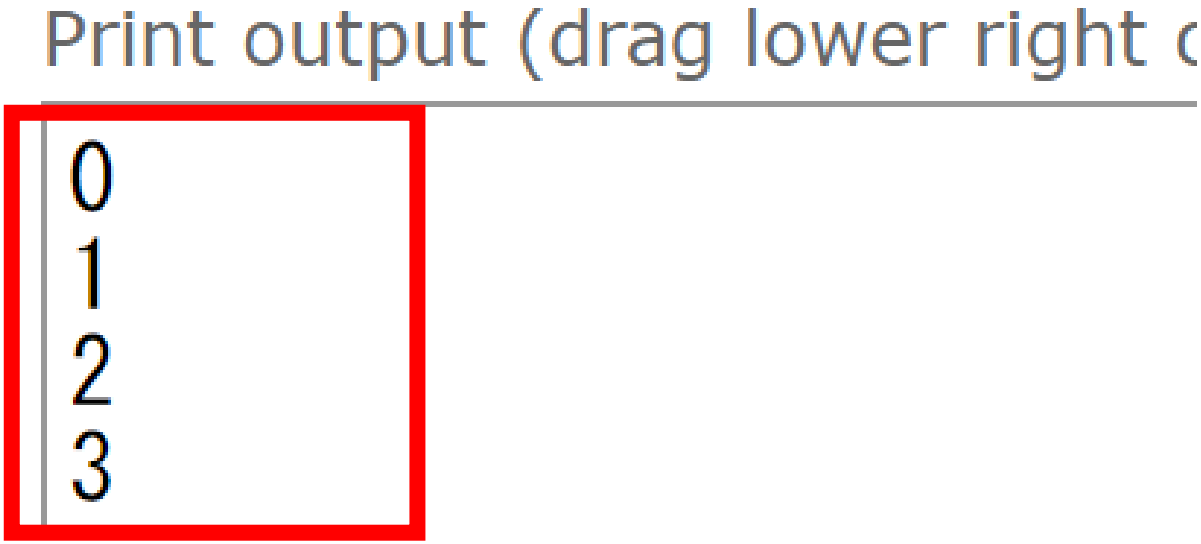
リストの要素を表示する

```
1  a = [0, 1, 2, 3]
2
3  for i in a:
4      print(i)
```

for i in a
の直後に「:」

字下げも正確に！
print(i) の前に，「タブ (Tab)」を 1つだけ

② 実行し，結果を確認する



結果を確認

「**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

③ Python Tutor のエディタで次のプログラムを入れる

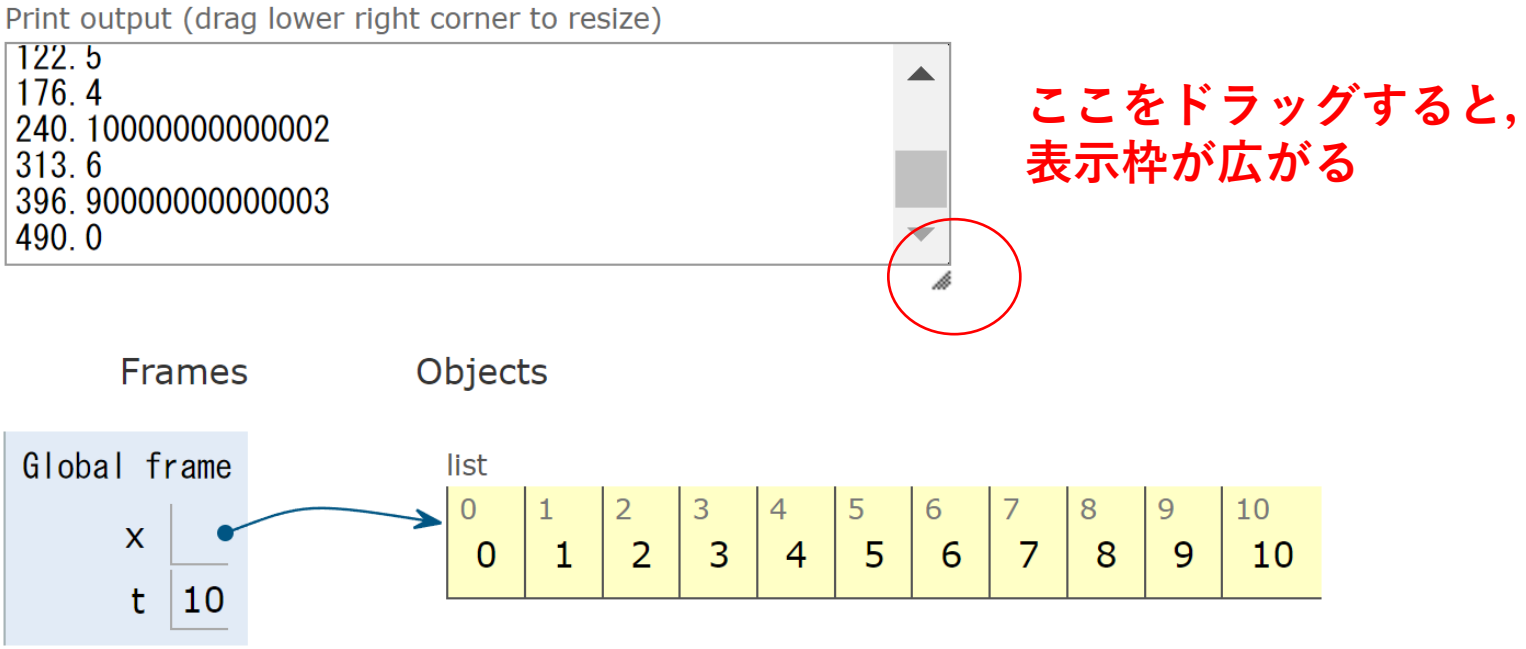
物体の落下．0〜10秒まで．1秒ごと

```
1  x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2  for t in x:
3      print((9.8 / 2) * t * t)
```

for t in x
の直後に「:」

字下げも正確に！
print((9.8 / 2) * t * t) の前に，「タブ (Tab)」を 1つだけ

④ 実行し，結果を確認する



「**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

6-2. ステップ実行

繰り返しのプログラム例

```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

実行結果

```
8.8
6.600000000000005
4.4
3.3000000000000003
2.2
```

繰り返しのプログラム例

リストの組み立て

```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

「y[i] = x[i] * 1.1」を i の値を変えながら 5回繰り返す

確認クイズ

次のプログラムで i の値はどのように変化するか？

i の値： □ → □ → □ → □ → □

```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

「y[i] = x[i] * 1.1」を
i の値を変えながら
5回繰り返す

答え合わせ

次のプログラムで **i の値はどのように変化する**か？

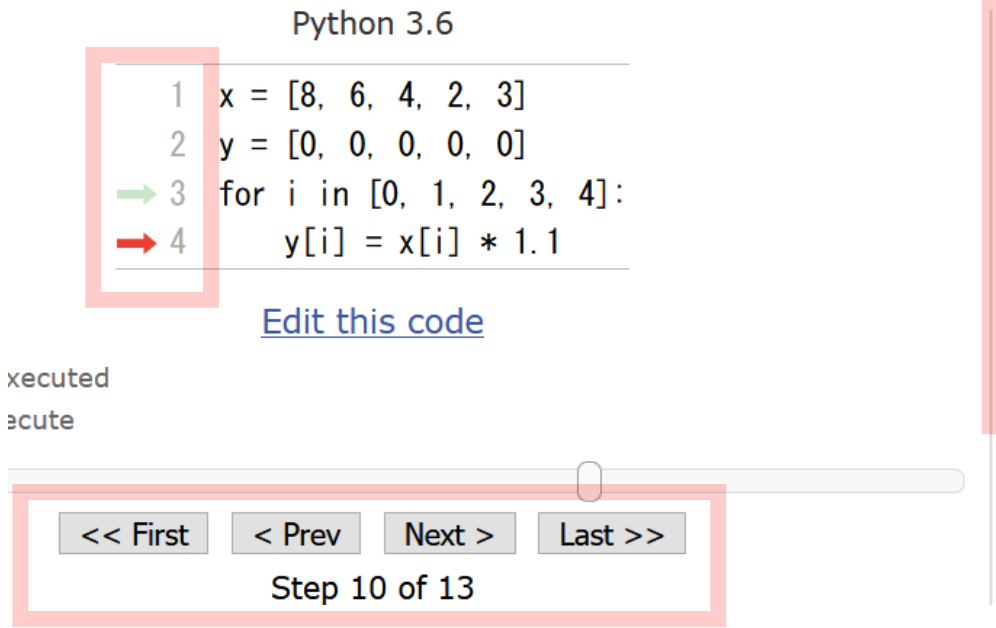
i の値：　0 → 1 → 2 → 3 → 4

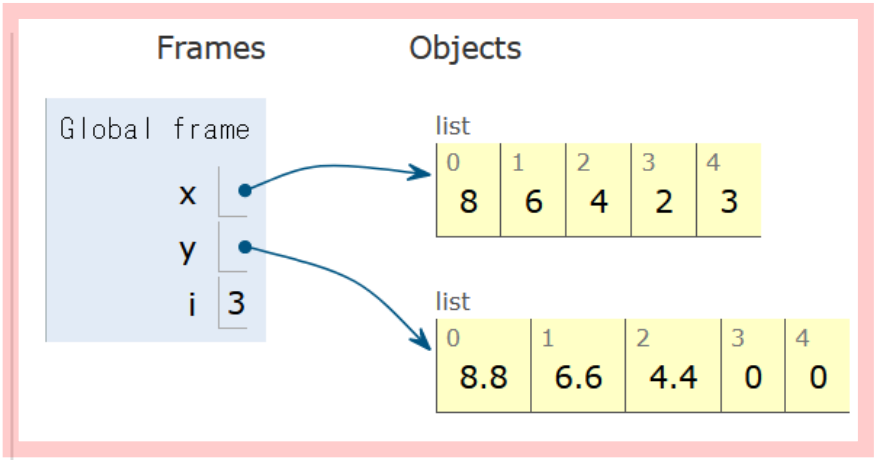
```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

「y[i] = x[i] * 1.1」を
i の値を変えながら
5回繰り返す

Python Tutor でのステップ実行

ステップ実行により，**プログラム実行の流れ**を確認できる





演習

資料：**28 ～ 33**

【トピックス】

- Python Tutor でのステップ実行の操作
- 変数の値の変化
- 実行の流れの変化（ジャンプ）

① Python Tutor のエディタで次のプログラムを入れる

8, 6, 4, 2, 3 の 1.1 倍を求め，結果を別のリストに保存

```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

for i in [0, 1, 2, 3, 4] の直後に「:」

字下げも正確に！ y[i] = x[i] * 1.1 と print(y[i]) の前に，「タブ (Tab)」を 1つだけ

②「**Visualize Execution**」をクリックして，実行開始

Write code in Python 3.6

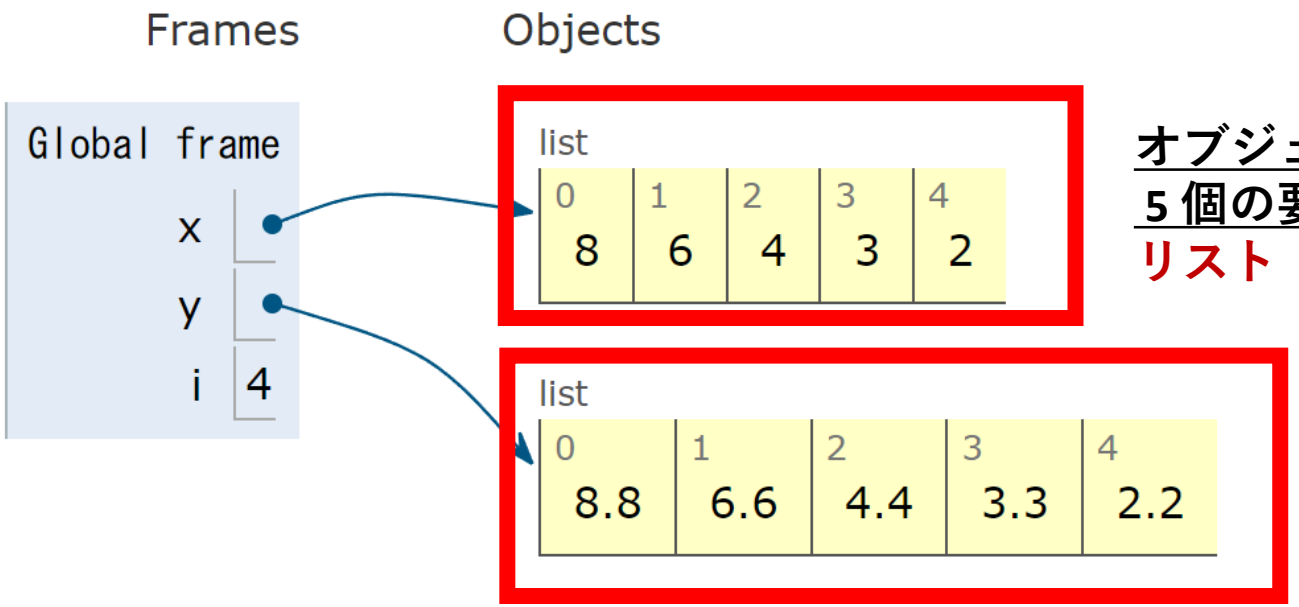
```
1  x = [8, 6, 4, 3, 2]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
5      print(y[i])
```

Visualize Execution

③ 実行し，結果を確認する

Print output (drag lower right corner to resize)

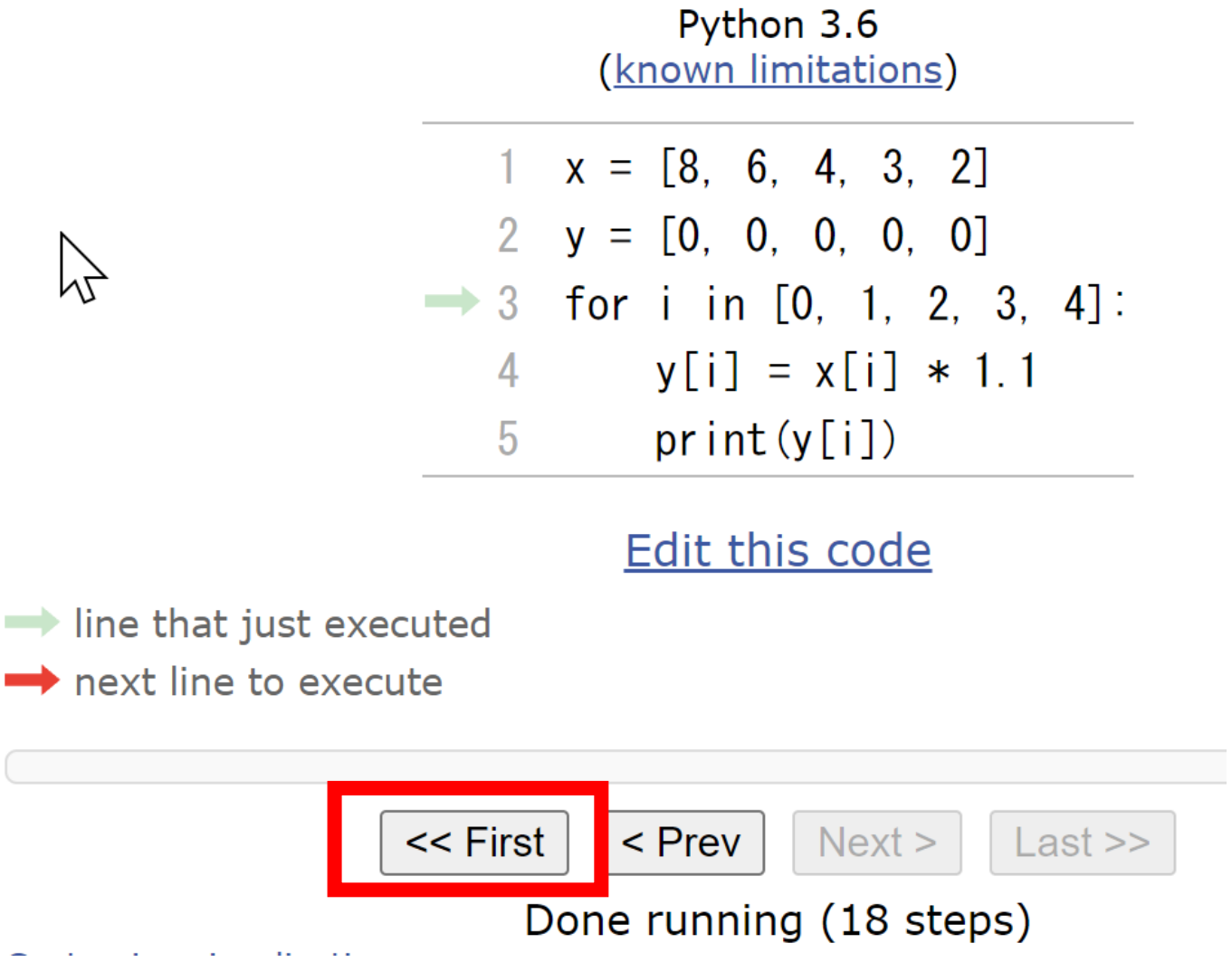
```
8.8
6.6000000000000005
4.4
3.3000000000000003
2.2
```



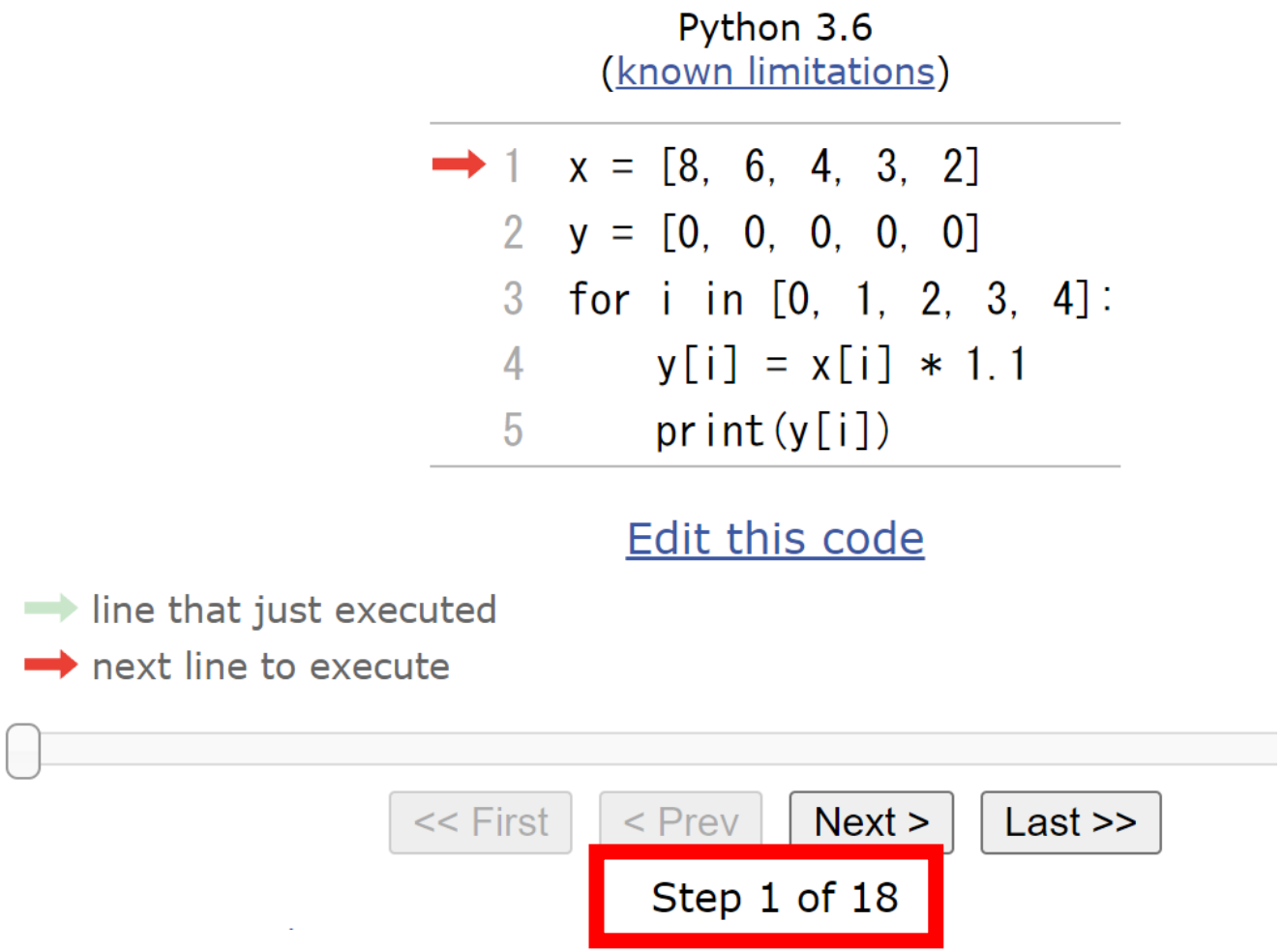
オブジェクト x は
5 個の要素が入った
リスト

オブジェクト y は
5 個の要素が入った
リスト

④「First」をクリックして，プログラム実行を先頭に戻す

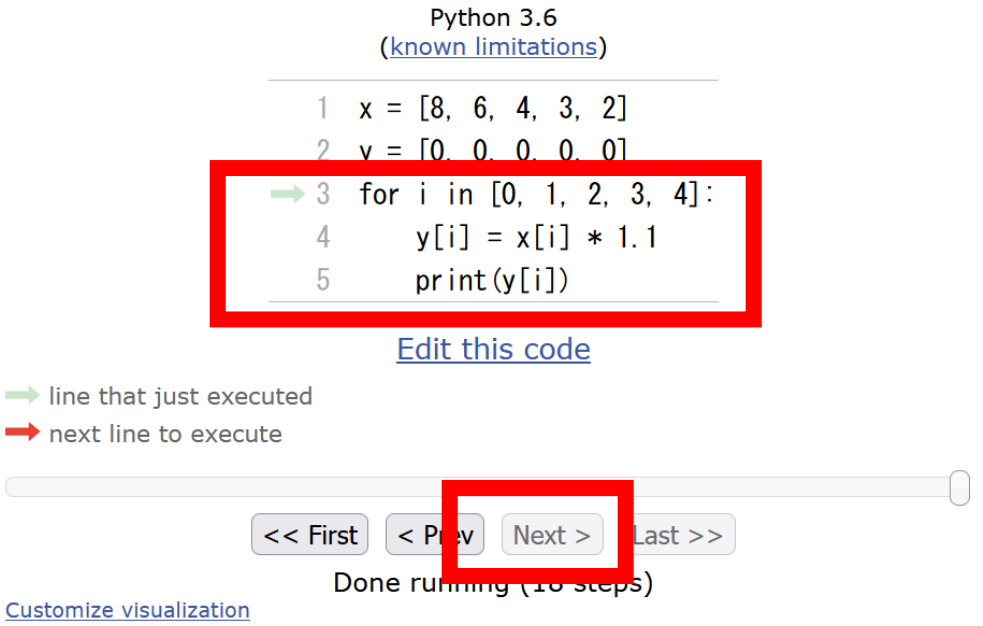


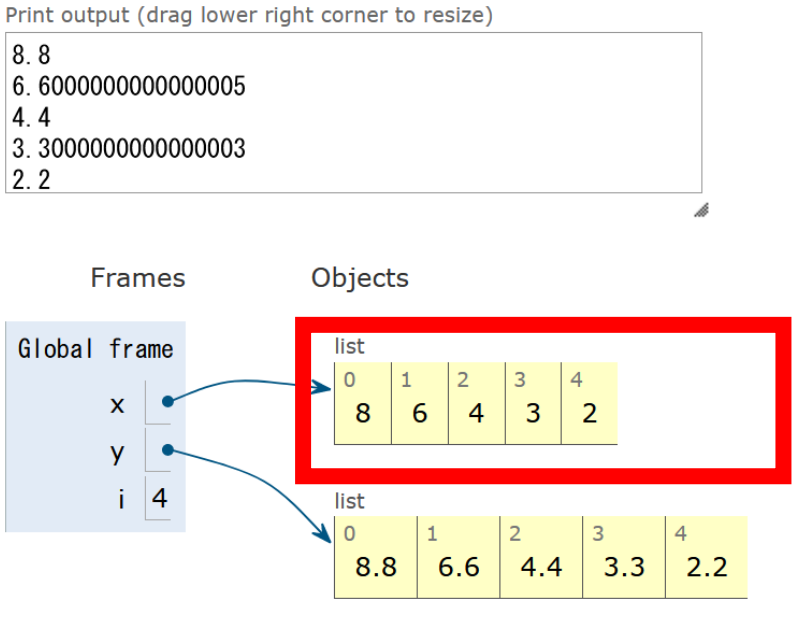
⑤「**Step 1 of 18**」と表示されているので，
全部で，ステップ数は **18** あることが分かる
（ステップ数と，プログラムの行数は**違うもの**）



⑥ **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





実行が進むと，
y の中身が更新される

見どころ
３行目，４行目、５行目が繰り返される

6-3. 辞書と繰り返し

辞書

辞書は，**キー**と**値（バリュー）**の**ペア**の集まり

キー	値（バリュー）
1	"Red"
2	"Yellow"
3	"Blue"

繰り返しのプログラム例

辞書の
組み立て

```
1  d = {1: "Red", 2: "Yellow", 3: "Blue"}
2
3  for i in d:
4      print(i)
```

辞書のキー 1, 2, 3 のそれぞれについて，値（バリュー）を表示

実行結果

Print output (drag lower right corner to resize)

```
1
2
3
```

演習

資料：**38 ～ 39**

【トピックス】

- **for による繰り返し（ループ）**

① Python Tutor のエディタで次のプログラムを入れる

```
1  d = {1: "Red", 2: "Yellow", 3: "Blue"}
2
3  for i in d:
4      print(i)
```

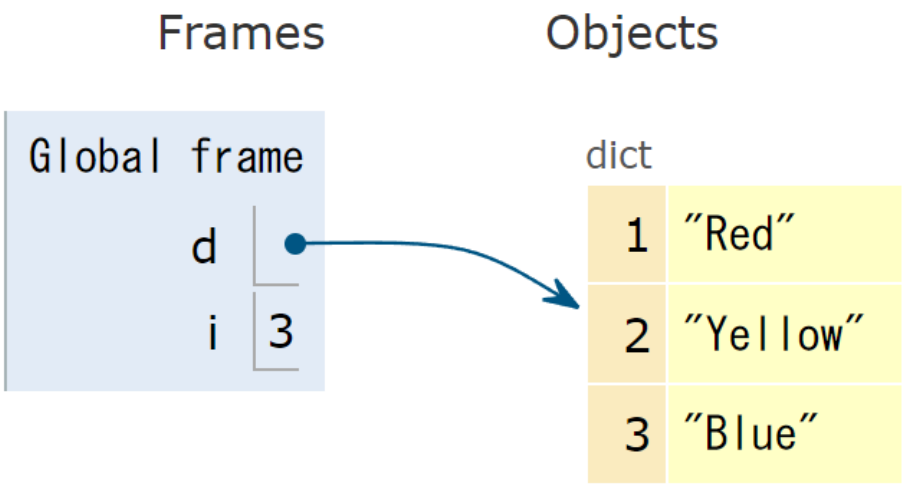
for i in d
の直後に「:」

字下げも正確に！
print(i) の前に，「タブ (Tab)」を 1つだけ

② 実行し，結果を確認する

Print output (drag lower right corner to resize)

```
1
2
3
```



「**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

全体まとめ

- **処理の繰り返し**により，**リスト**や**辞書**のすべての要素を処理できる
- **リスト**や**辞書**で**処理の繰り返し**を行うときは，**for**が便利である
- for による**繰り返し（ループ）**
  同じ処理や操作を繰り返す

```
1  x = [8, 6, 4, 2, 3]
2  y = [0, 0, 0, 0, 0]
3  for i in [0, 1, 2, 3, 4]:
4      y[i] = x[i] * 1.1
```

Python 関連ページ

- Python まとめページ

https://www.kkaneko.jp/tools/man/python.html

- Python 入門（スライド資料とプログラム例）

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

- Python プログラミングの基本（スライド資料とプログラム例）

https://www.kkaneko.jp/pro/po/index.html

- Python プログラム例

https://www.kkaneko.jp/pro/python/index.html

- 人工知能の実行（Google Colaboratory を使用）

https://www.kkaneko.jp/ai/ni/index.html

- 人工知能の実行（Python を使用）（Windows 上）

https://www.kkaneko.jp/ai/deepim/index.html