Lecture 6

プログラミング演習 I　その６

本日の演習の流れ

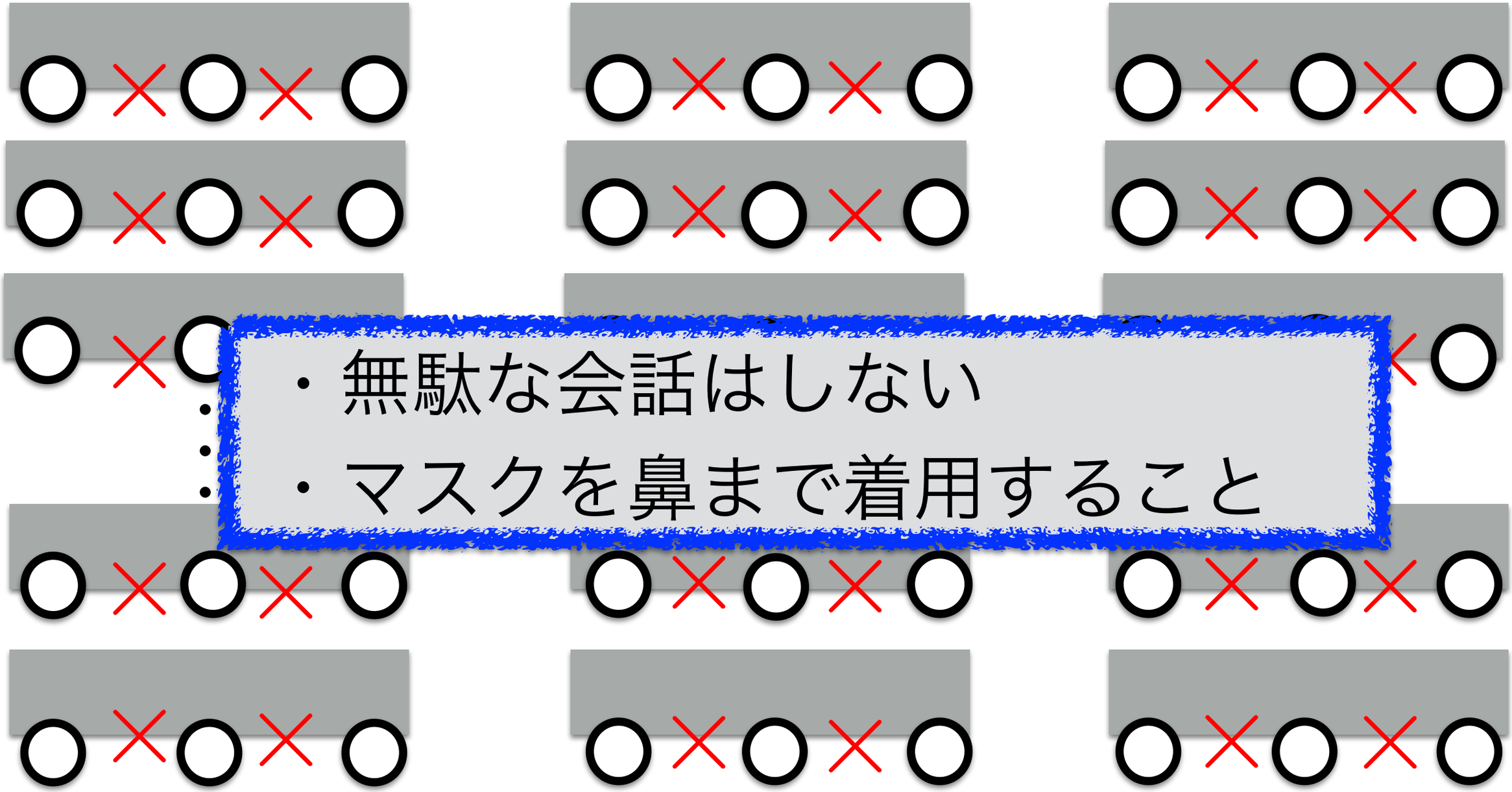
1. Pythonによる論理値の演算および比較演算子
2. Python のフロー制御
3. Pythonによる関数の利用
4. Pythonによる for文，while 文による繰り返し

授業のホームページ

http://amth.mind.meiji.ac.jp/courses/PE1/

教室での座席について

前



・無駄な会話はしない

・マスクを鼻まで着用すること

後

論理値の演算

演算	結果
x or y	xとyのどちかが真なら真
x and y	xとyの両方が真なら真
not x	xが偽なら真，xが真なら偽

真値：True

偽値：False

論理値の演算

1 Pythonの対話型シェルを利用し，論理値の演算を調べよ.

EX

```
[eginder@Elliotts-MacBook-Pro Pro5 % python
[>>> True or False
 True
```

比較演算子

演算	意味
<	より小さい
<=	以下
>	より大きい
>=	以上
!=	変数のデータが等しくない
==	変数のデータが等しい

論理値の演算

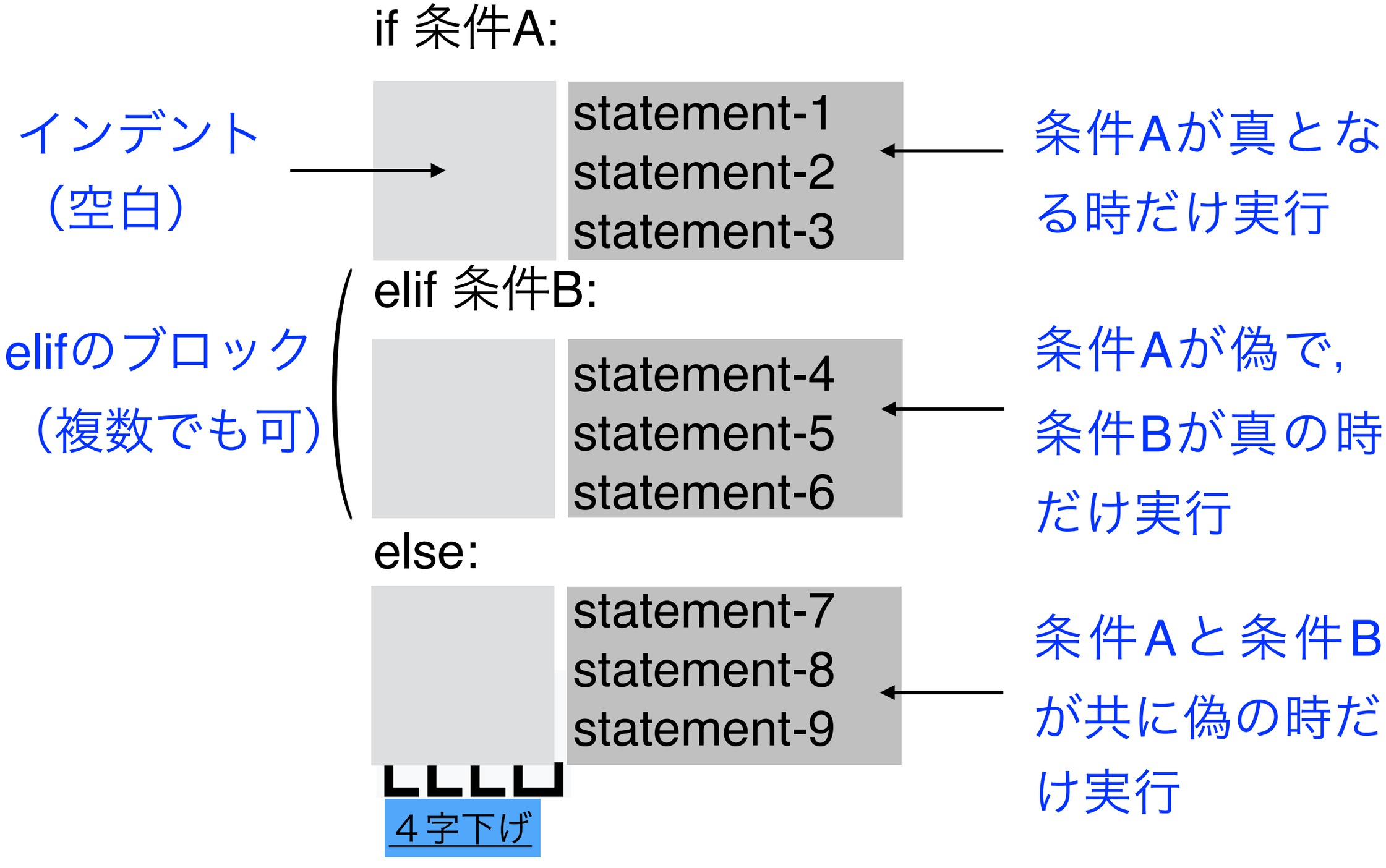
2 Pythonの対話型シェルを利用し，比較演算子を試してみよう．

EX

```
>>> 5 < 10
True
```

Python のフロー制御

if文の構成



Python のブロックについて

Pythonではインデント (文字下げ) が文法的に意味を持つ．特に，Python のブロックはインデントで決定されている．インデントには**４文字**のスペース（空白文字）を使うのが一般的である．以下スクリプトを実行して，比べてみよう．

スクリプト

```
x = 3
y = 5
if x == 3:
    if y == 5:
        print("congratulations!")
```

出力

```
congratulations!
```

```
x = 3
y = 5
if x == 3:
    if y == 10:
        print("Merry Xmas!")
    print("congratulations!")
```

```
congratulations!
```

```
x = 3
y = 5
if x == 7:
    if y == 7:
        print("no!")
print("congratulations!")
```

```
congratulations!
```

Python のブロックについて

スクリプト

```
1 x = 3
2 y = 5
3 if x == 3:
4     if y == 5:
5     print("congratulations!")
```

出力

```
  File "indent2.py", line 5
    print("congratulations!")
        ^
IndentationError: expected an indented block
```

Python のフロー制御

3 Pythonでは他のプログラミング言語と同様に if文, for文, while文などの構文が準備されている.

<u>if文</u>

Pythonで, 何らかの条件が成立するかどうかによって処理を振り分けたい場合に使う構文が if文である. 例を見てみよう.

```
1 x = 8
2 if x < 5:
3     print("x is smaller than 5")
4 elif x > 10:
5     print("x is bigger than 10")
6 elif x > 8:
7     print("x is bigger than 8")
8 else:
9     print("5 <= x <= 8")
```

← if.py

出力

```
Lec14_pro python if.py
5 <= x <= 8
```

Python のフロー制御

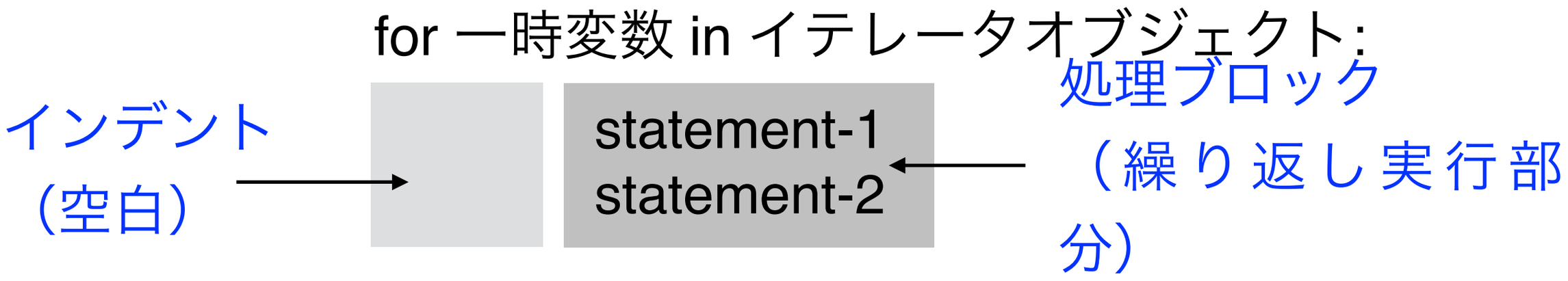
4 Cと同じく，if文はネストされることが可能である．次の例では，a == b が偽となった場合に実行されるelse文の直後のブロック中に，別のif文が入っている．

```
a = 3
b = 3
if a == b:
    print("a is equal to b")
else:
    if a < b:
        print('a is smaller than b')
    else:
        print('a is greater than or equal to b')
```

NOTE: if文に限らず，プログラムの様々な要素についてネスト構造を使うことが可能である．

Python のフロー制御

for文の構成



Python のフロー制御

for文

プログラム中で繰り返し処理を行いたい場合，すなわちループ処理を行いたい場合に使われるのが for文である．例を見てみよう．

5

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

出力

```
Lec14_pro python for.py
1
2
3
4
```

6

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

```
Lec14_pro python for2.py
3
2
1
```

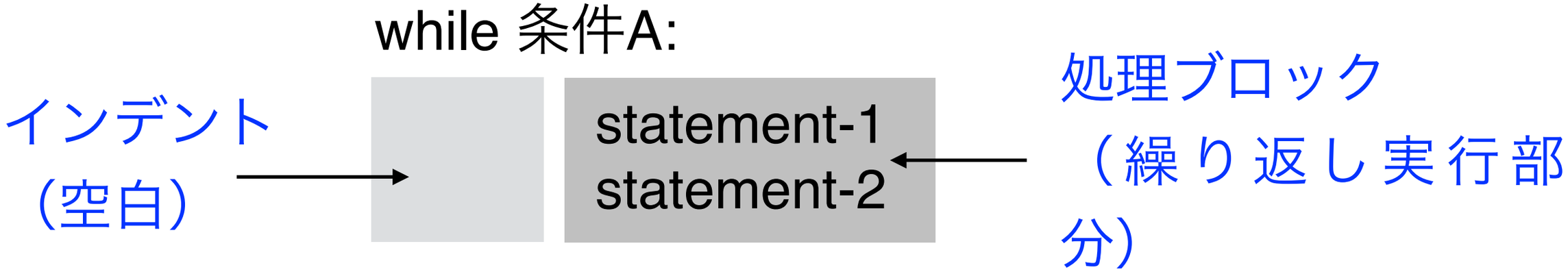
7

```
1 for i in range(1,5):
2     print(i)
```

```
Lec14_pro python for3.py
1
2
3
4
```

Python のフロー制御

while文の構成



Python のフロー制御

while文

特定の条件が成立している間は継続的に何かの処理を行いたい場合に使う．例を見てみよう．

8

```
1 i = 1
2 while i < 10:
3     print(i)
4     i = i + 1
```

出力

```
Lec14_pro python while.py
1
2
3
4
5
6
7
8
9
```

Python で関数定義の基本

関数の定義の構文

```
def 関数名 (引数1, 引数2, …, 引数n):
    statement-1
    …
    return something
```

インデント（空白）

関数の処理を書く

return 文はオプション

関数の定義の具体的な例を見てみよう.

9

```
1 def my_add(a, b):
2     print("The variables are %s and %s" %(a,b) )
3     return a + b
4
5 ans = my_add(3,4)
6 print(ans)
```

func.py

出力

```
 Lec14_pro python func.py
The variables are 3 and 4
7
```

Python で関数定義の基本

関数f(x)を以下で定義する．for文を用いて，x=0.0, 0.1, …, 1.0におけるf(x)の値を表示するプログラムを作成せよ．

10

```
1 def f(x):
2     return x*x*9*10 + 100
```

matplotlibを使ってPythonでグラフを作りましょう

matplotlibはPythonのパッケージ（関連する働きをする関数などのモジュールの集まり）です．すなわち，数をプロットしたり，グラフを作るのに役に立つモジュールです．以下を打ちましょう．

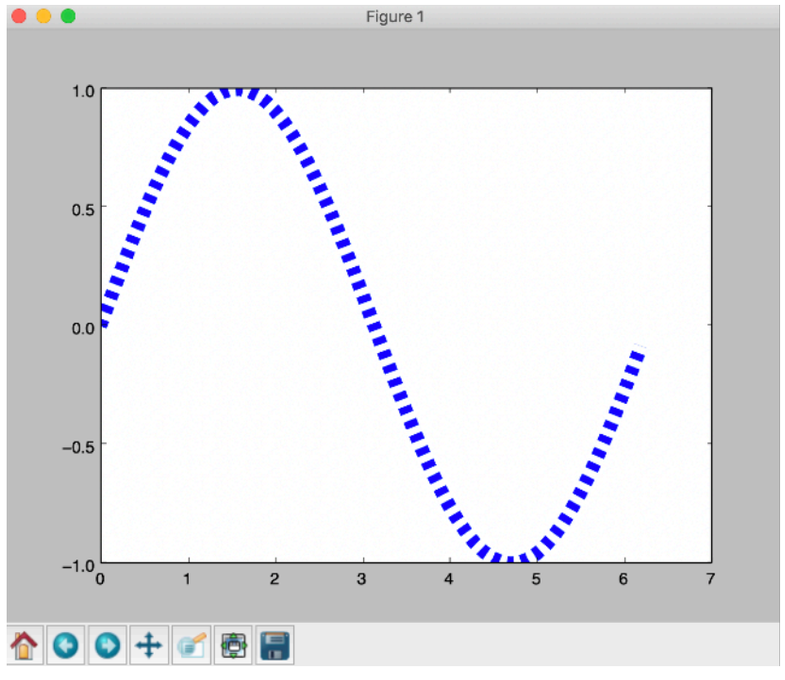
11

```
1  import numpy as np
2  import matplotlib as mpl
3  import matplotlib.pyplot as plt
4
5  mpl.rcParams['lines.linewidth']=10
6  mpl.rcParams['lines.linestyle']='--'
7
8  t = np.arange(0,2*np.pi,0.1)
9  plt.figure(1)
10 plt.plot(t,np.sin(t))
11 plt.show()
```

線幅を10 points 設定

線種を波線に設定

10で定義した関数もプロットしてみよう.



matplotlibを使ってPythonでグラフを作りましょう

アニメーションも簡単に作れます.

12

```
1  import numpy as np
2  import matplotlib as mpl
3  import matplotlib.pyplot as plt
4
5  mpl.rcParams['lines.linewidth']=10
6  mpl.rcParams['lines.linestyle']='--'
7
8  t = np.arange(0,2*np.pi,0.01)
9  plt.figure(1)
10 for i in range(1,30):
11     plt.plot(t,np.sin(t - 0.2*i))
12     plt.pause(0.1)
13     plt.clf()
```

← 一時停止

← 図をクリアする