

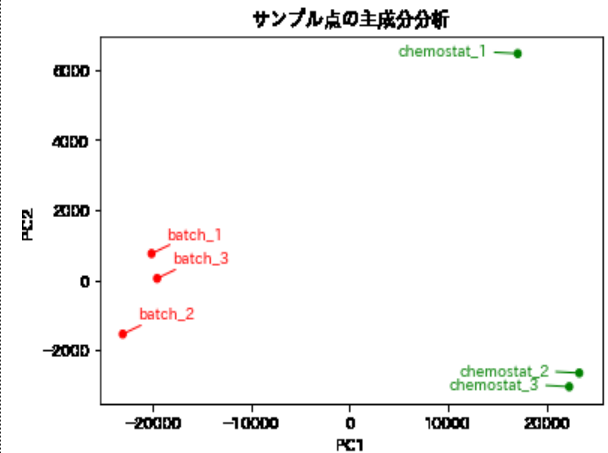
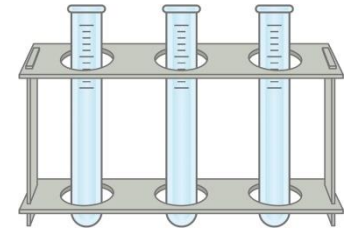
第3回

道具

VS

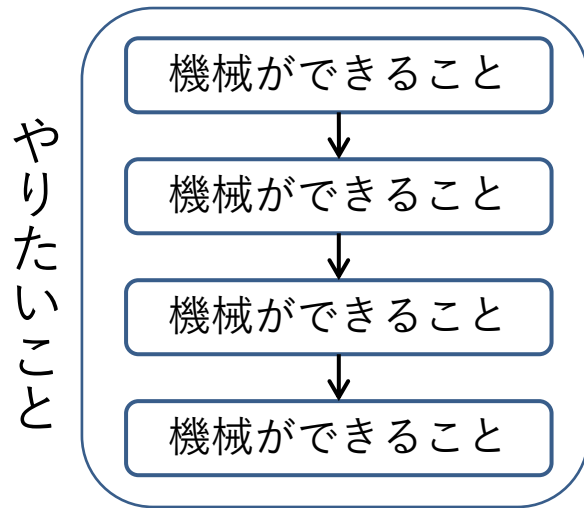
やりたいこと

- Pythonのデータ
 - 数値、文字列
 - 「リスト」と仲間
- 変数と代入
- Pythonの「実行制御」
 - 無指定 ~ 上から順に
 - 条件分岐 (if文)
 - ループ/繰り返し (for文)



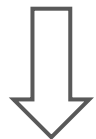
道具～プログラム

機械ができることに分割



たとえば

1から100までの和



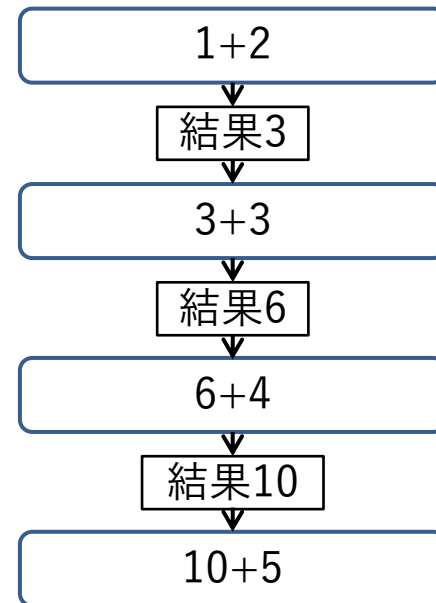
$((1+2)+3)+\dots+100$

似た遺伝子
配列を探す



Smith-Waterman
のやり方
(後の回で見る?)

順番にしかできない



- 順番の変更
 - 条件分岐
 - ループ } これから

機械にできること①

(プログラミング知識の即席ツアー)

変数

- 値を入れる箱
- 名前をつける

代入

- 右辺の値を計算して左辺に入れる

$x = 3 + 2$
 $x = x + 4$

向きがある
⇒ 等号と違う

$s = 'abc'$

中身は何でもいい

$t = s + 'de'$

文字列に対する「+」

おまけ `print()`

- `()`の中を印刷・表示する

`print(3+2)`

`print(x)`

機械にできること②

条件分岐

- 条件が成立すれば
<内側の処理>
成立しなければ
<内側の処理>

```
if (x<3):  
    print('3未満')  
else:  
    print('3以上')
```

<内側の処理>の始まりマーク

<内側の処理>が続いているマーク

ループ (繰返し)

- 指定した範囲で 繰り返す
<内側の処理>を

```
for u in [3, 4, 5, 6, 7]:  
    print(u)
```

uが順番に3, 4, 5, 6, 7の
値を取って繰り返す

1 周目はuの値は3

2 周目はuの値は4

5 周目はuの値は7 で終り

機械にできること③

- 基本的に

- 定数・変数
- 代入
- 条件分岐 (if)
- 繰返し (for)

に加えて

- (関数) 呼出し

- でほぼ使える。あと必要なのは

- 関数やクラスを自分で定義 ←あとで

もう少しだけ細かいことを追加 ⇨

追加① 算術演算子（+，-，...）

式の中に書ける+とか-とか

演算子	使い方	
+	$a + b$	足し算
-	$a - b$	引き算
*	$a * b$	掛け算
/	a / b	割り算
//	$a // b$	切捨ての割り算（切捨て除算の商）
%	$a \% b$	aをbで割った時の余り
**	$a ** n$	aのn乗（べき乗）

$$4 / 3 \Rightarrow 1.3333\dots$$

$$4 // 3 \Rightarrow 1$$

$$4 \% 3 \Rightarrow 0.3333\dots$$

追加② 条件の書き方① 比較演算

比較して、真Trueか、偽Falseかを返す

演算子	使い方	
==	a == b	aがbに等しい
!=	a != b	aがbに等しくない
>	a > b	aがbより大きい
>=	a >= b	aがbより大きいか等しい
<	a < b	aがbより小さい
<=	a <= b	aがbより小さいか等しい

3 == 3 ⇒ True
4 == 3 ⇒ False
4 != 3 ⇒ True
3 != 3 ⇒ False

4 > 3 ⇒ True
3 > 4 ⇒ False
3 > 3 ⇒ False
3 >= 3 ⇒ True

追加③ 論理演算子 (and, or, ...)

True・Falseの値を持つ式を組合せる

True and True	⇒ True	True or True	⇒ True
True and False	⇒ False	True or False	⇒ True
False and True	⇒ False	False or True	⇒ True
False and False	⇒ False	False or False	⇒ False

not True ⇒ False
not False ⇒ True

```
if ((x > y) or (x < y)):  
    print('等しくない')
```

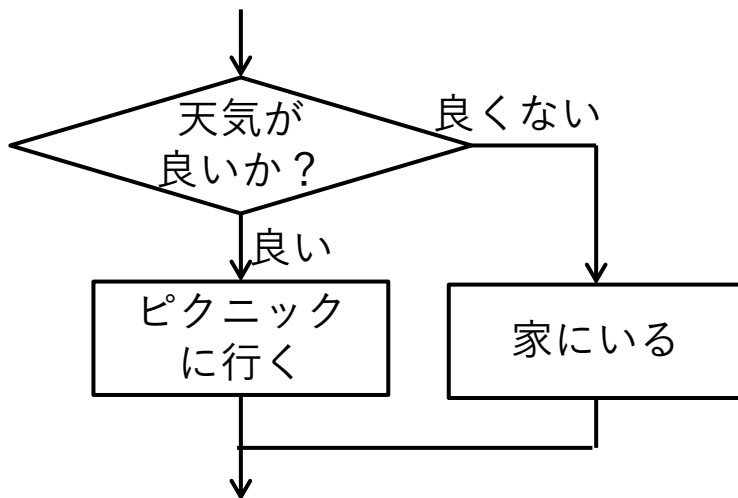
```
if ((x == y) and (y == z)):  
    print('xとzは等しい')
```


条件分岐 (if文)

条件の真偽によってやることを変える

天気が良いければ
ピクニックに行く
そうでなければ
家にいる

```
weather = 'fine'  
  
if weather=='fine':  
    print('picnic')  
else:  
    print('stay home')
```



```
age = 19  
  
if age>=19:  
    print('1900年代生まれ')  
else:  
    print('2000年代生まれ')
```

条件分岐とブロック

x = 3

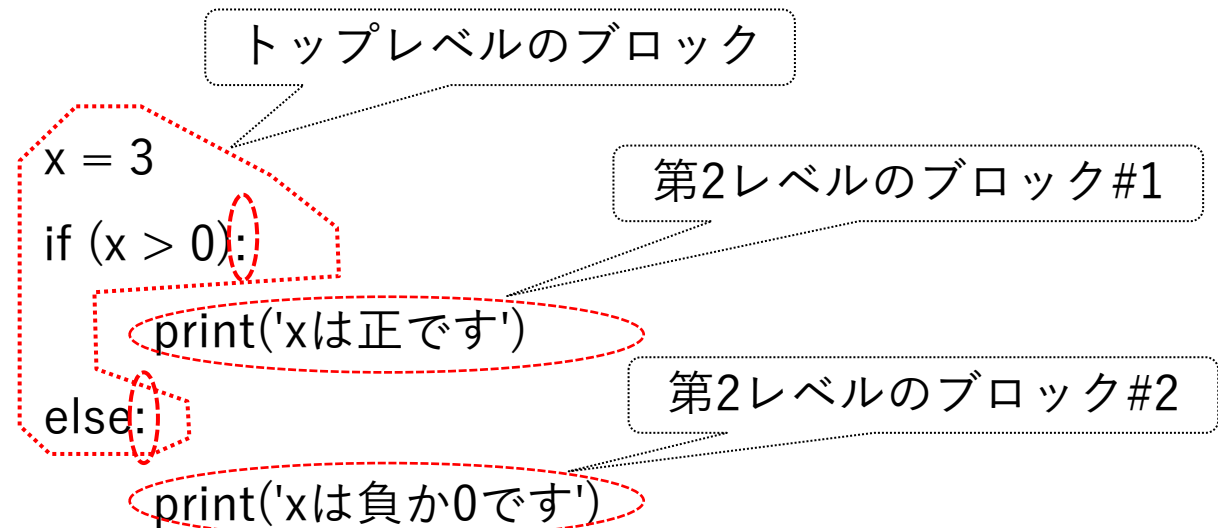
if (x > 0):

print('xは正です')

else:

print('xは負か0です')

段下げする (4文字)



条件分岐とブロック

```
x = 3
```

```
if (x > 0):
```

```
    if (y > 0):
```

```
        print('xは正です')
```

```
        print('yは正です')
```

```
    else:
```

```
        print('xは正、yは負か0')
```

```
else:
```

```
    print('xは負か0です')
```

- ブロック
- ブロックを入れ子にする
 - 段下げレベルでブロックを区別する

自分で考えてみよう

机上で考えてみてください

TAの人に、以下の問題をチェックしてもらおうこと

```
x = 3
y = 5
z = 0.5
if (((y-x)*0.6)+((z-x)/2))>=0):
    print('正か0です')
else:
    print('負です')
```

どう動くか？

- $(y-x)*0.6$
- $(z-x)/2$
- $((y-x)*0.6)+((z-x)/2)$
- $((y-x)*0.6)+((z-x)/2) \geq 0$
- 結果

```
x = 2
y = 1
z = 0.5
u = ((y-x)*0.6)-(z-x)
v = (((x*5)+y)//3)-(x+y)
if ((u>=0) and (v>=0)):
    print('正か0です')
else:
    print('負です')
```

どう動くか？

- u
- v
- 結果

繰返し (forループ)

```
t = [1, 5, 4, 9, 7, 2, 3, 6]
```

```
for u in t:  
    print(u)
```

出力結果は

1, 5, 4, 9, 7, 2, 3, 6

- リスト t がある
- t の要素を順番に (u という名前で) 取り上げて print(u) を実行する

```
t = [1, 5, 4, 9, 7, 2, 3, 6]
```

```
tmax = -10000
```

```
for u in t:
```

```
    if (u > tmax):  
        tmax = u
```

ブロック

```
print(tmax)
```

- for と if の入れ子
- for で回るたびに if を繰り返す
 - もし $u > tmax$ なら $tmax$ の値を u に置換える
- 全部回ったら $tmax$ を表示

TAチェック

机上で考えてみてください

TAの人に、以下の問題をチェックしてもらうこと

```
t = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum = 0
for u in t:
    sum = sum + u
print(sum)
```

どう動くか？

- 1周目が終わった時のsumの値
- 2周目が終わった時のsumの値
- 3周目が終わった時のsumの値
- (途中省略)
- 10周目が終わった時のsumの値
- print文の結果

```
t = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum = 0
for u in t:
    if (u%2 == 0):
        sum = sum + u
print(sum)
```

どう動くか？

- 1周目が終わった時のsumの値
- 2周目が終わった時のsumの値
- 3周目が終わった時のsumの値
- (途中省略)
- 10周目が終わった時のsumの値
- print文の結果

余裕のある人は実際にPythonを動かしてごらん下さい