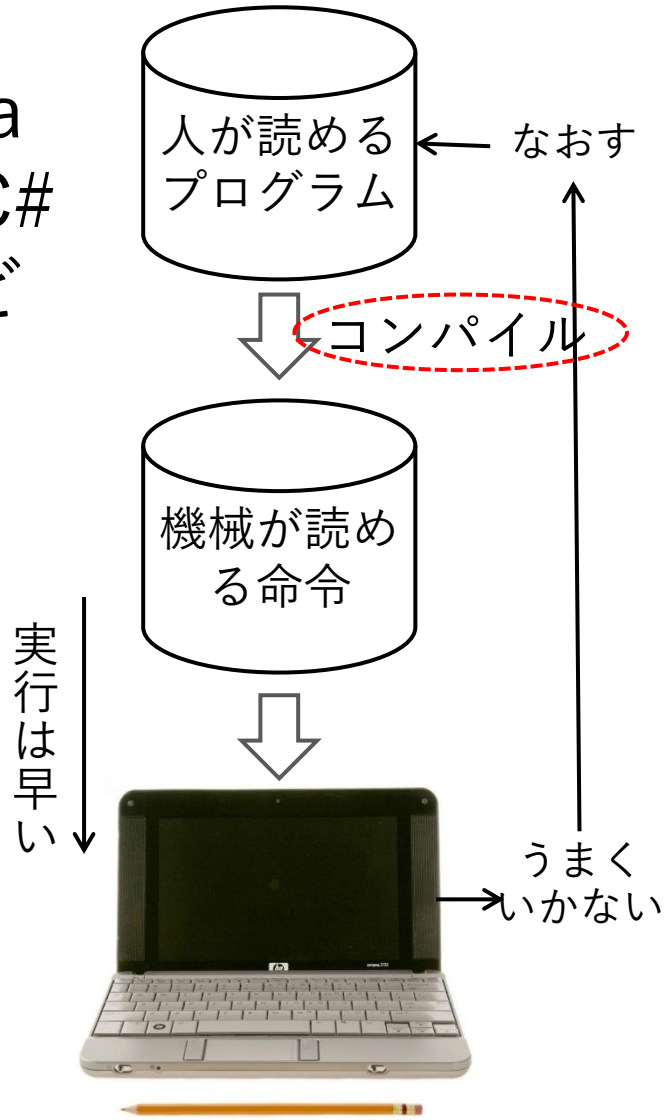
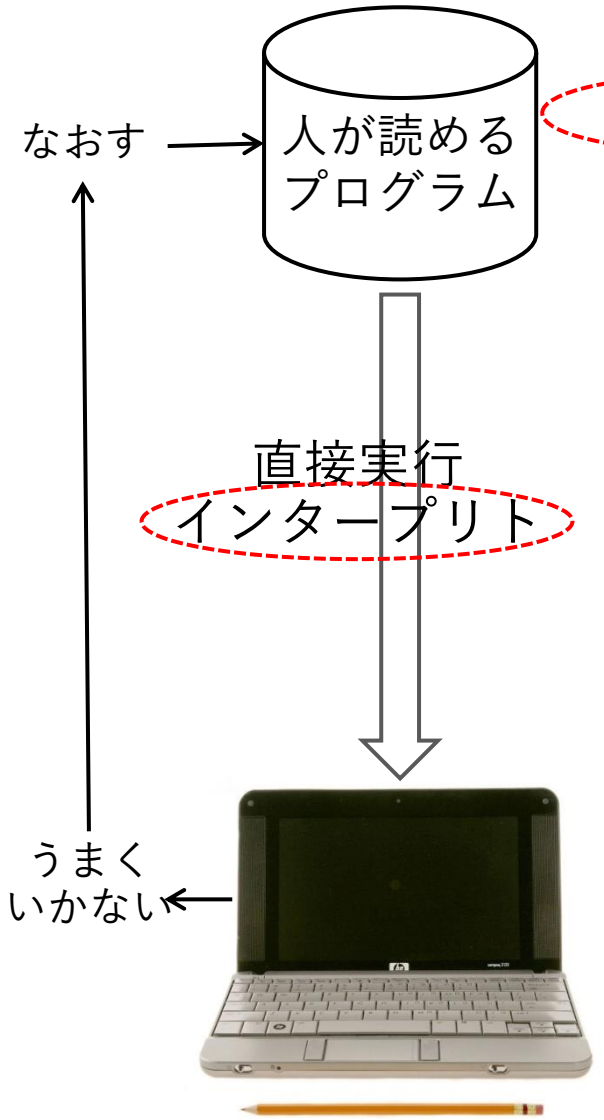
プログラム実行　２種類の環境

Java
C/C#
など

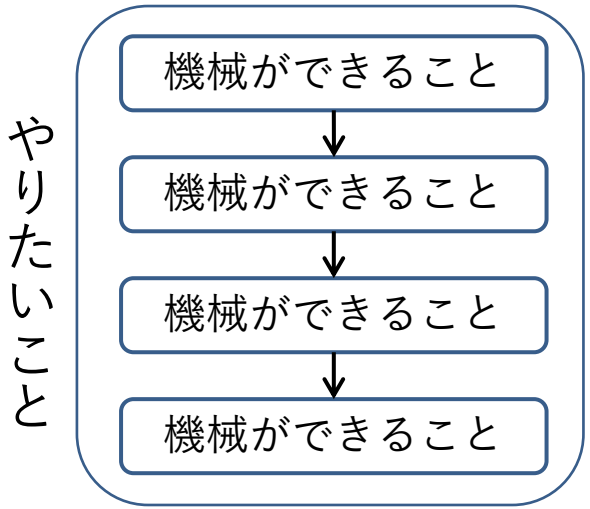


Python
Ruby
Basic
など

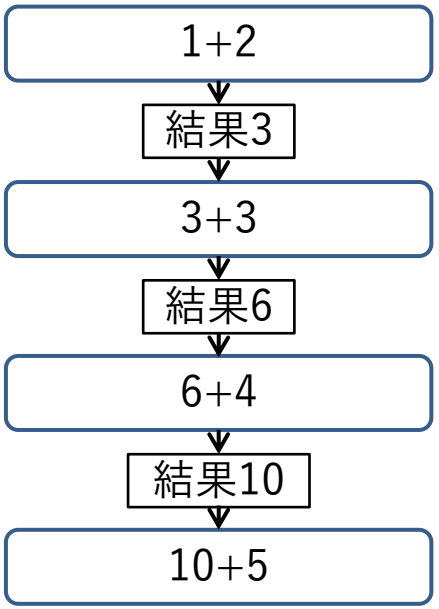


プログラムの構造

機械ができることに分割

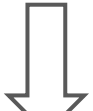


順番にしかできない



たとえば

1から100までの和



(((1+2)+3)+…)+100

似た遺伝子
配列を探す

Smith-Waterman
のやり方
(第14回目で見る?)

- 順番の変更
  - 条件分岐
  - ループ

} あとで

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
  - 条件分岐（if）
  - 代入
  - 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＋とかーとか

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　⇒ 1.3333...
4 // 3 ⇒ 1
4 % 3　⇒ 0.3333...

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
```

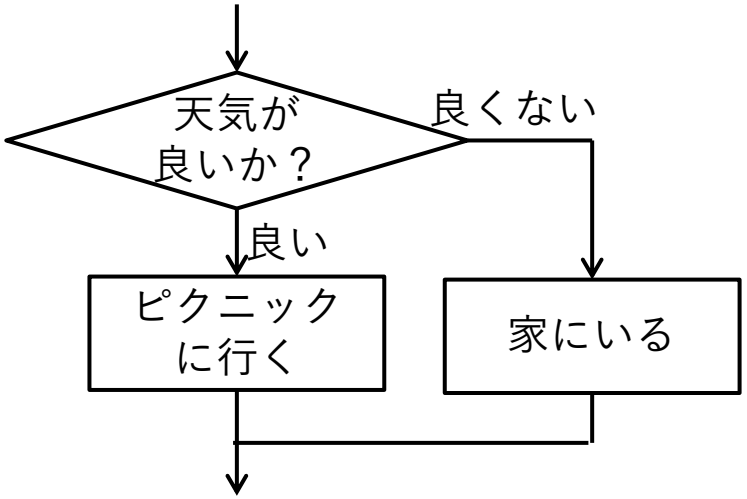
```
if ((x > y) or (x < y)):
    print('等しくない')

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')
```

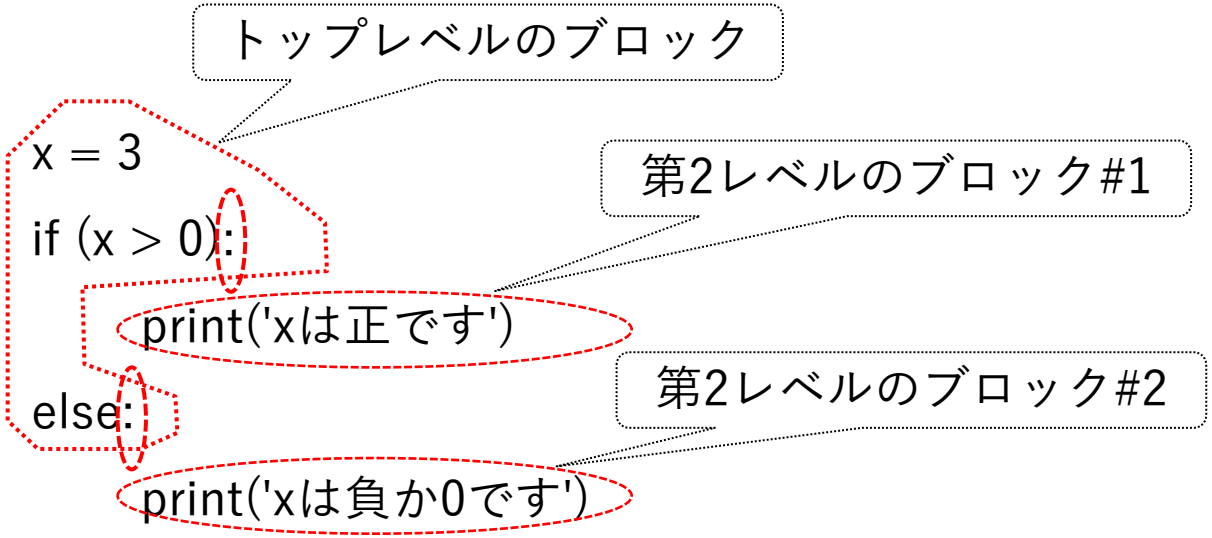
```
age = 19

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

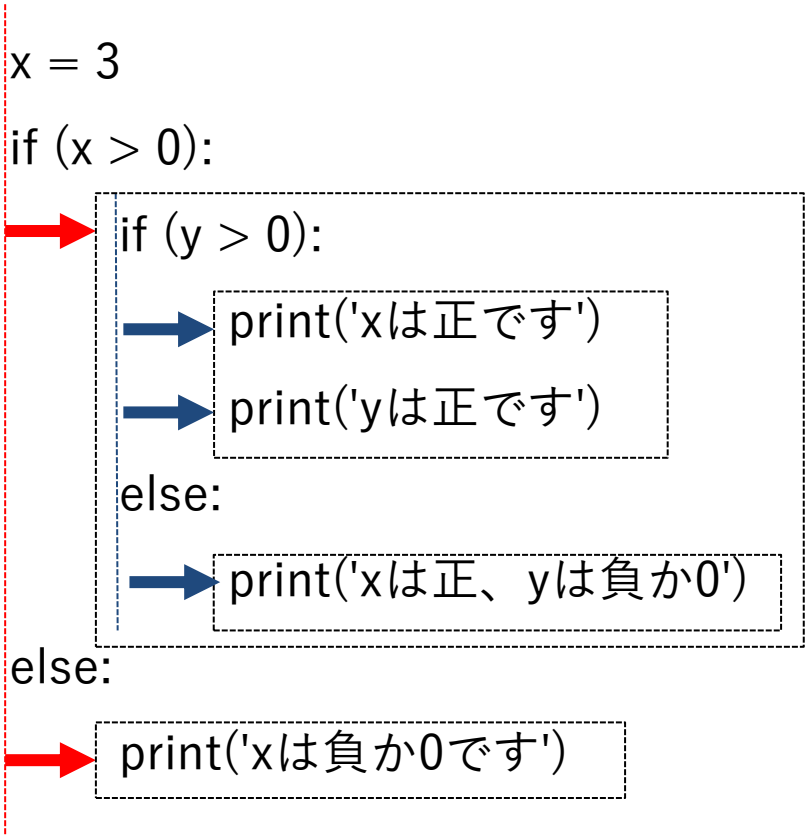
条件分岐とブロック

```
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



- 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

繰返し（forループ）

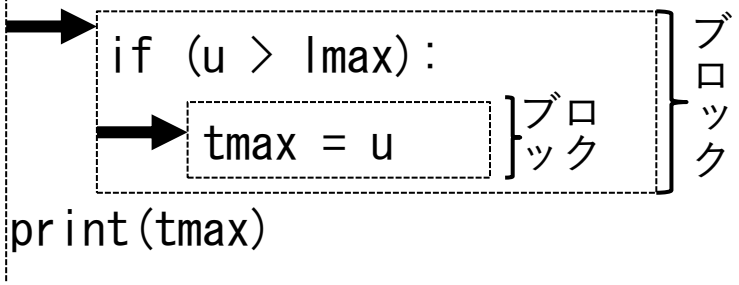
```
t = [1, 5, 4, 9, 7, 2, 3, 6]
for u in t:
    print(u)
```

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
  する

```
t = [1, 5, 4, 9, 7, 2, 3, 6]
tmax = -10000
for u in l:
    if (u > lmax):
        tmax = u
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

```
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')
```

出力はどうなるか

```
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')
```

出力はどうなるか