4-1-0　　オンデマンド授業　プログラミング・サマリ

プログラミング（システム設計やアプリ設計など）の流れ

①問題解決の考え方（算法）　　：**アルゴリズム**
②アルゴリズムをモデル化（図式化）：**フローチャート**（流れ図）
③アルゴリズムをコンピュータに実装：**コーディング**
④動作確認を行い、誤り（バグ）を修正：**デバッグ**

プログラミング

大学共通テスト出題予測
・DNCL（日本語記述）
・Python記述に近似

- 基本的な処理➡**順次処理・分岐処理・反復処理**
- 重要なポイント➡**変数、関数、乱数、配列、（プロンプト入力受入れ,無限ループ等）**
- Pythonコードに **#** を付けると以降は注釈となりコードから除外（➡コメントアウト）
- 基本構文からスタート、英語は This is a pen.　I have a book.からでした

(1) python基本作法

① 表示➡print()･･･(　)内が数値の場合はそのまま、文字の場合は’　’または”　”で囲う

② 変数➡a、b、c ～y 、z～age、fortune、price･･･自分で分かれば何でもよい

③ 演算子

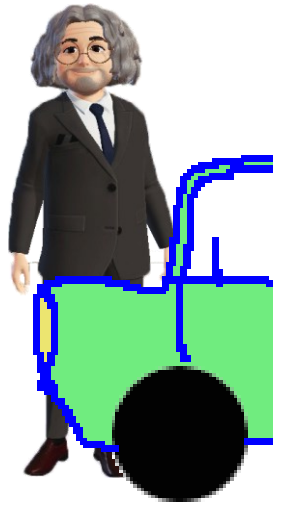
プログラミング（コーディング）		数学
n+2	# 加算	n + 2
n-2	# 減算	n － 2
n*2	# 乗法	n × 2
n/2	# 除法	n ÷ 2
n//2	# 商	n を2で割った商
n%2	# 剰余	n を2で割った余り
n**2	# 累乗	n^2
n=n+2	# 代入	n に n+2 を代入（n=1➡1を3に変えて代入）
n+2==7	# 等号	n+2=7
n!=2	# 不等号	n≠2
n>=2	# 以上	n≧2
2<=n	# 以下	2≦n

以下の構文(#以降は抜いてよい)をショートカットCtrl+C,Ctrl+Vを上手く使って実際にコーディングしてみよう。

```
n=5
print(n+2)      # 加算     n+2                                                    7
print(n-2)      # 減算     n-2                                                    3
print(n*2)      # 乗法     n×2                                                    10
print(n/2)      # 除法     n÷2                                                    2.5
print(n//2)     # 商       nを2で割った商                                         2
print(n%2)      # 剰余     nを2で割った余り                                       1
print(n**2)     # 累乗     n^2                                                    25
n=n+2           # 代入     nにn+2を代入(n=5➡5+2=7に置換代入)
print(n)        # nを表示                                                        7
print(5+2==7)   # 等号     5+2=7 ➡正しければ'true'、誤りならば'false'             True
print(5!=2)     # 不等号   5≠2  ➡正しければ'true'、誤りならば'false'             True
print(5+2==6)   # 等号     5+2=6 ➡正しければ'true'、誤りならば'false'             False
print(5!=5)     # 不等号   5≠5  ➡正しければ'true'、誤りならば'false'             False
print(5>=2)     # 以上     5≧2  ➡正しければ'true'、誤りならば'false'             True
print(2<=5)     # 以下     2≦5  ➡正しければ'true'、誤りならば'false'             True
print(10/3)     # 10÷3     3.333･･･ x 10^0,  33.33･･･x 10^-1,  0.3333･･･x 10^1,  3.3333333333333335
print(int(10/3))  # 10÷3 を整数integerで表示                                      3
print(float(10/3))# 10÷3 を小数floatで表示                                       3.3333333333333335
```

ブール型

(2) 道路を横断するアルゴリズム(問題解決の算法)・フローチャート



端子
処理
分岐
処理
分岐
処理
端子

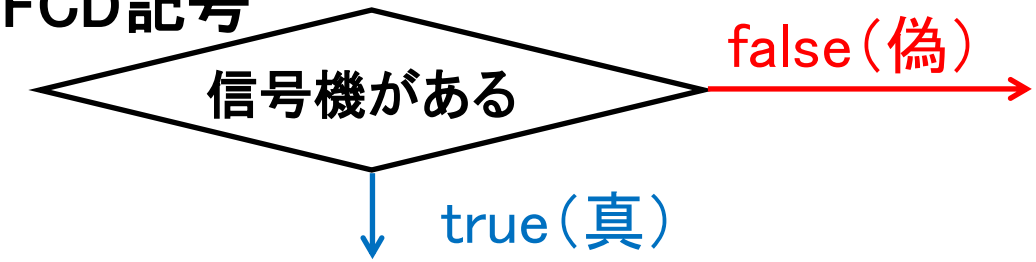
開始
信号機有無を確認
信号機がある
false(偽)
true(真)
信号を見る
待つ
反復
信号が青
true(真)
false(偽)
入れ子
右を見る
待つ
右方安全
true(真)
false(偽)
反復
左を見る
待つ
左方安全
false(偽)
true(真)
道を渡る
終了

入れ子(nest):条件の中に条件を入れる
右側が安全で、かつ左側も安全➡横断可

(3) 分岐処理

- 順次処理：上から下へ並ぶ処理
  - FCD順次処号

- 分岐処理：ある条件によって処理が別れる
  - FCD記号



- 端子：始、終　開始　終了
- 処理：道を渡る

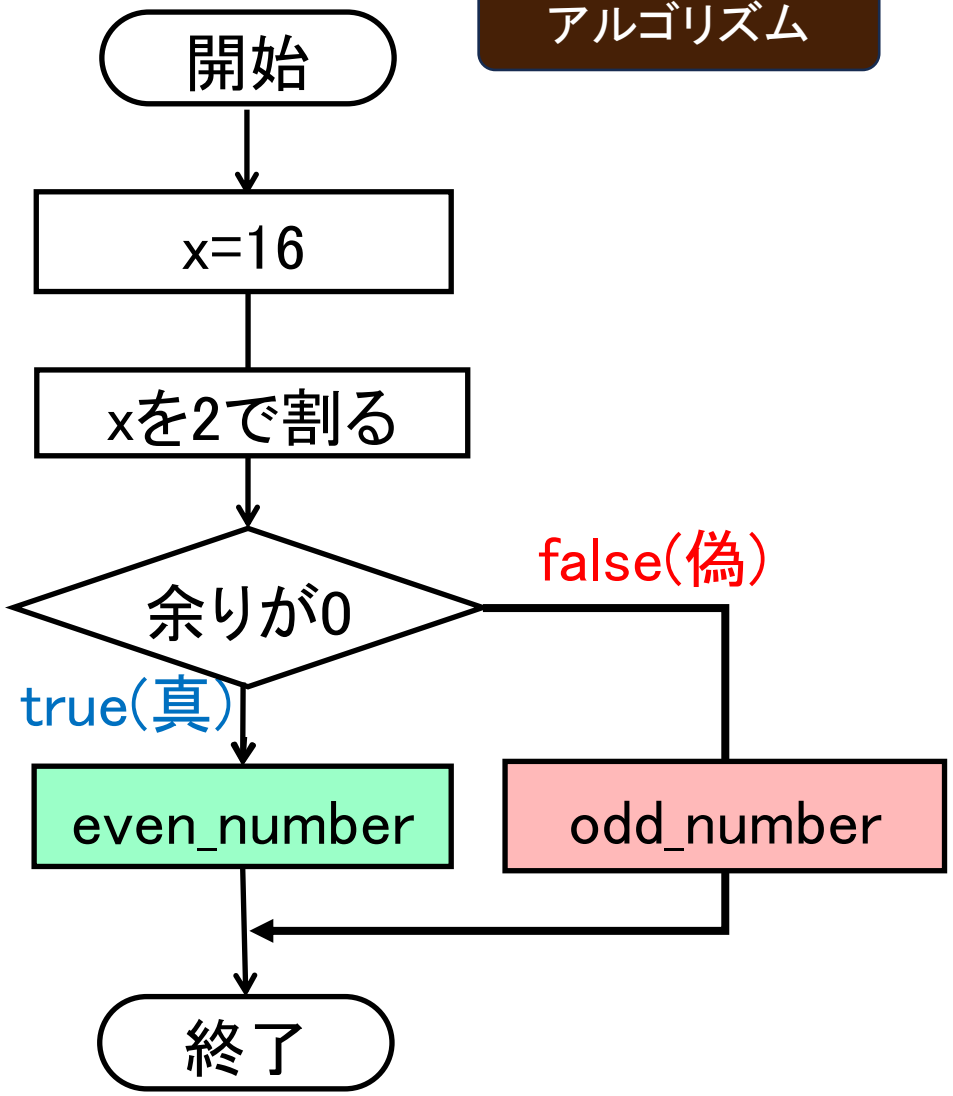
Python文法

```
if～条件1：
    処理1
elif～条件2 ：
    処理2
else：
    処理3
```

インデント(字下げ)
2～4字くらい

分岐処理の例題1　変数xが偶数(even_number)か奇数(odd_number)を判別せよ

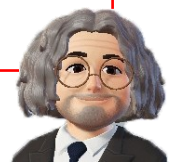
アルゴリズム



Pythonコード

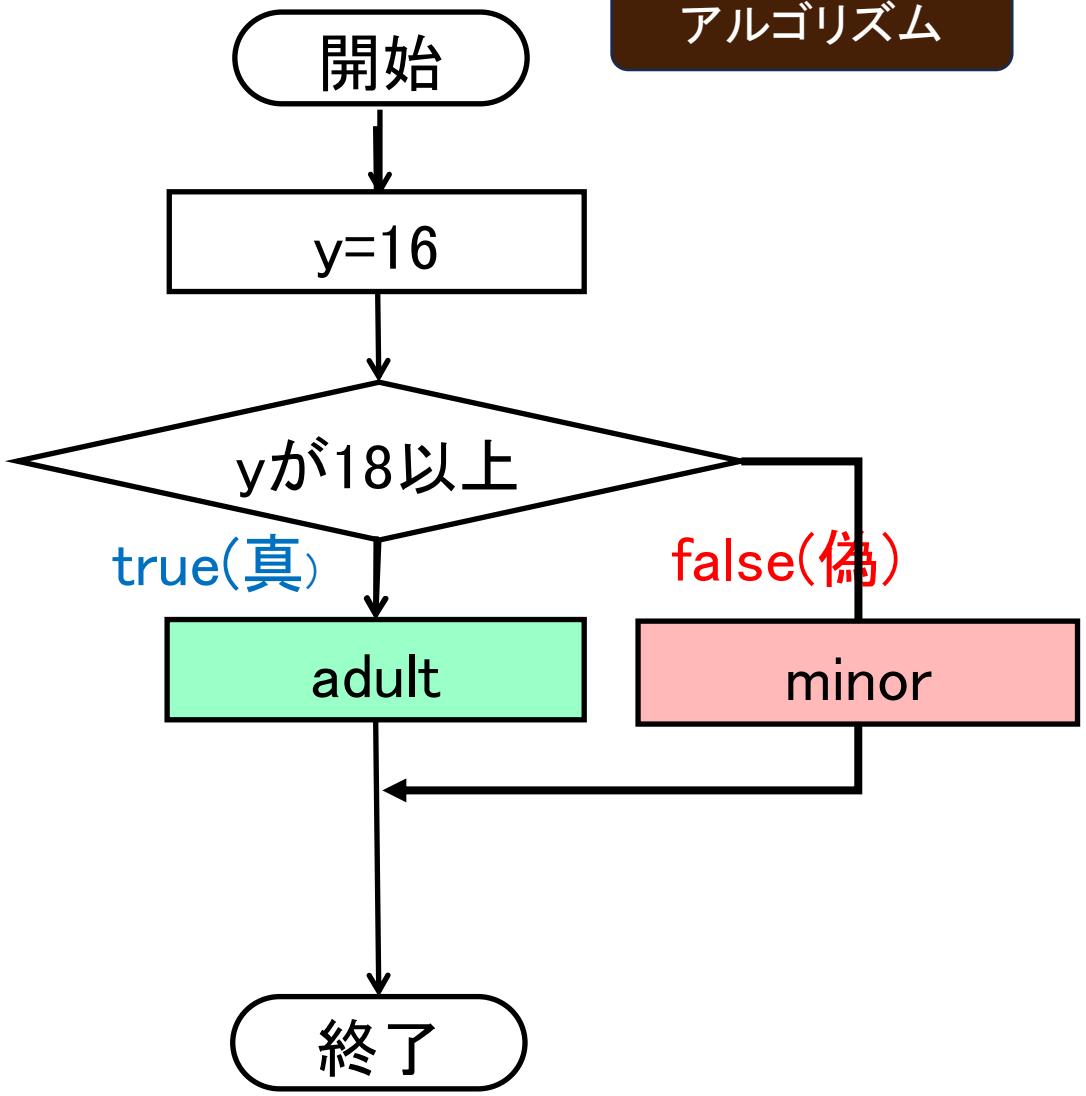
```
x=16
if x%2==0:
    print('even')
else:
    print('odd')
```

- ✓ if ,elif ,elseの先頭は揃える
- ✓ 行末に必ずコロン：を打つ
- ✓ 対応する処理は、2～4文字分の字下げ（インデント）を行う



分岐処理の例題2　年齢yが成人(adult)か未成年(minor)を判別せよ

アルゴリズム



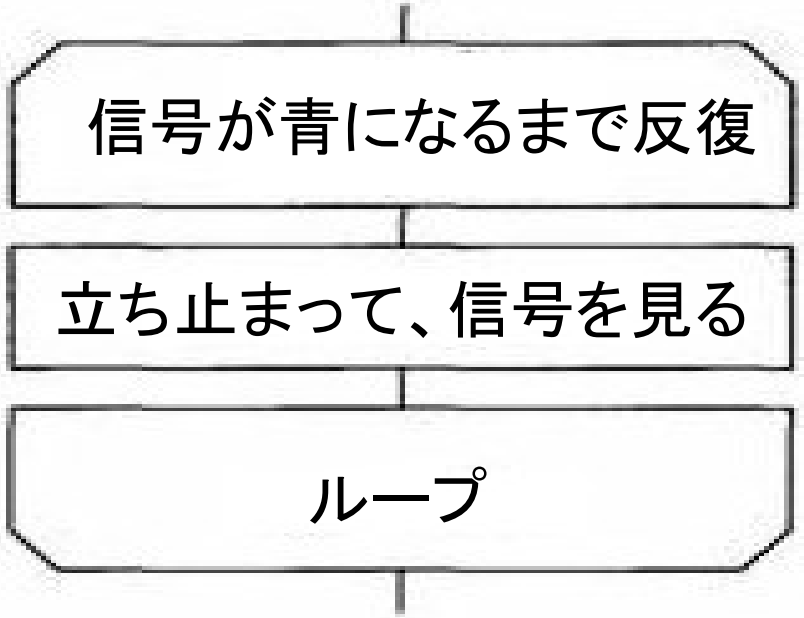
Pythonコード

```
y=16
if y>=18:
  print( 'adult')
else:
  print( 'minor')
```

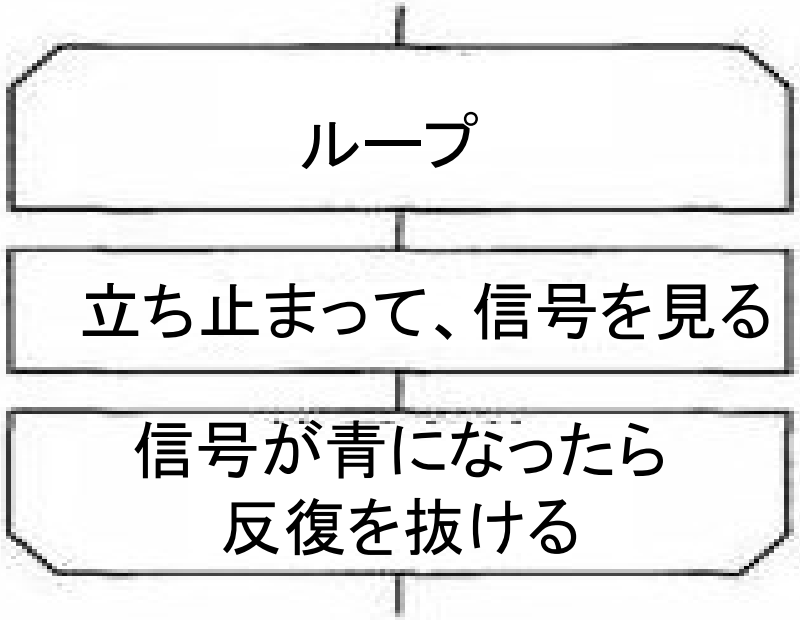
(4) 反復処理

- 反復処理：指定する条件内で処理を繰り返す･･･条件を設定しないと「無限ループ」となる

・前判定：先に反復条件を指定する



・後判定：処理の後で判定する



Python文法

① for文

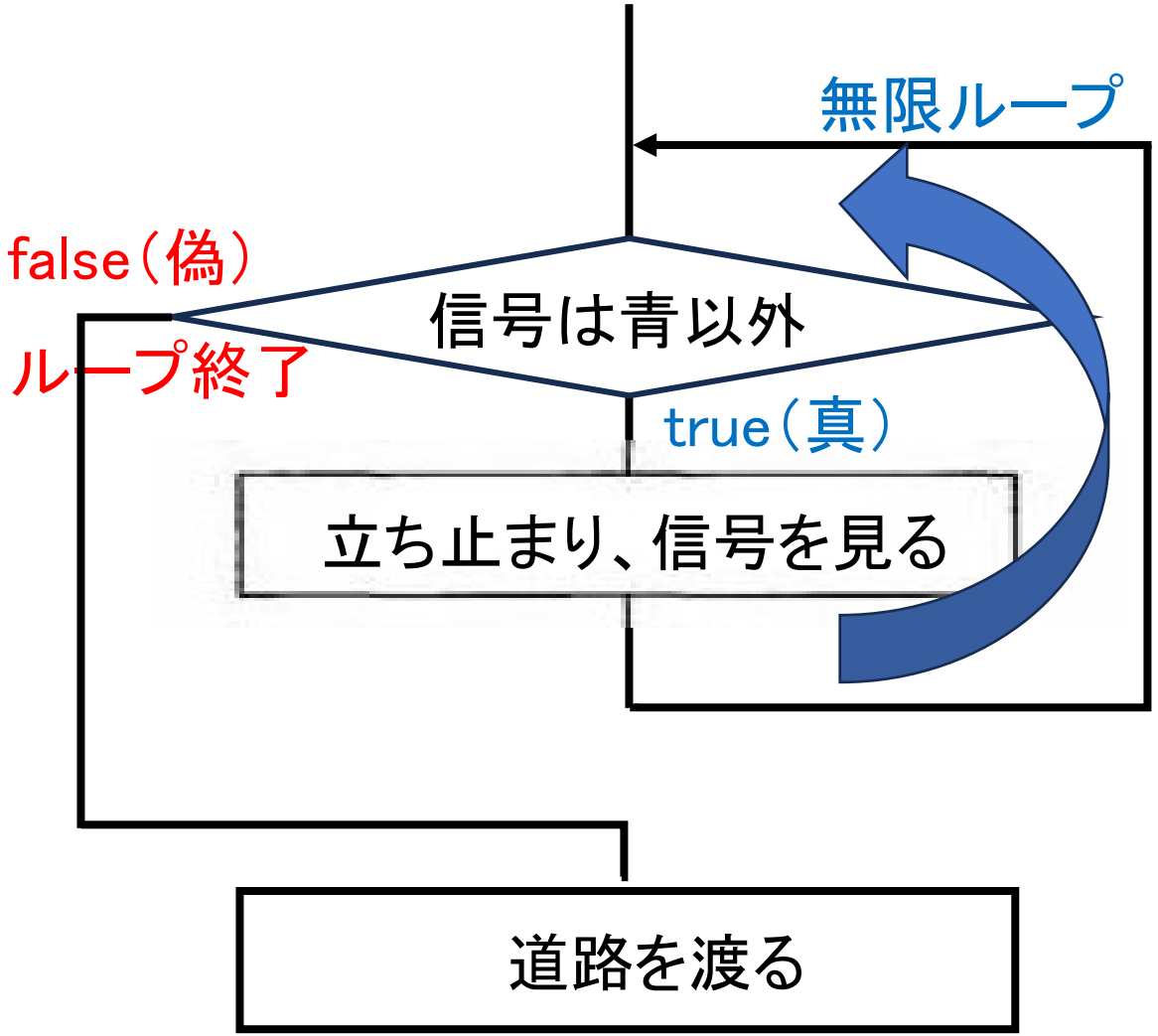
```
for n in range（条件）:
    処理
```

② while文

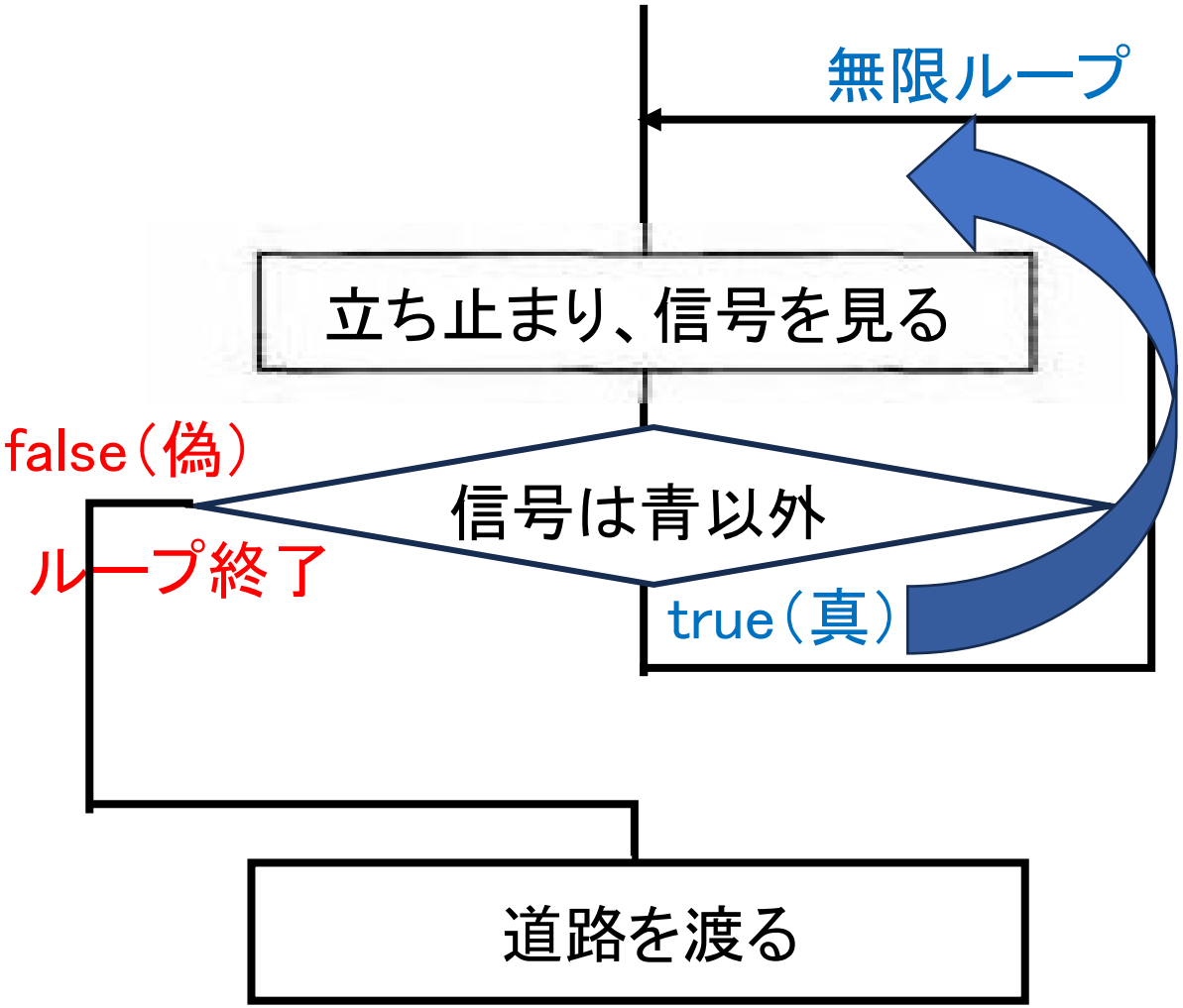
```
while 条件：
    処理
```

インデント(字下げ)
2～4字くらい

前判定：先に反復条件を指定する

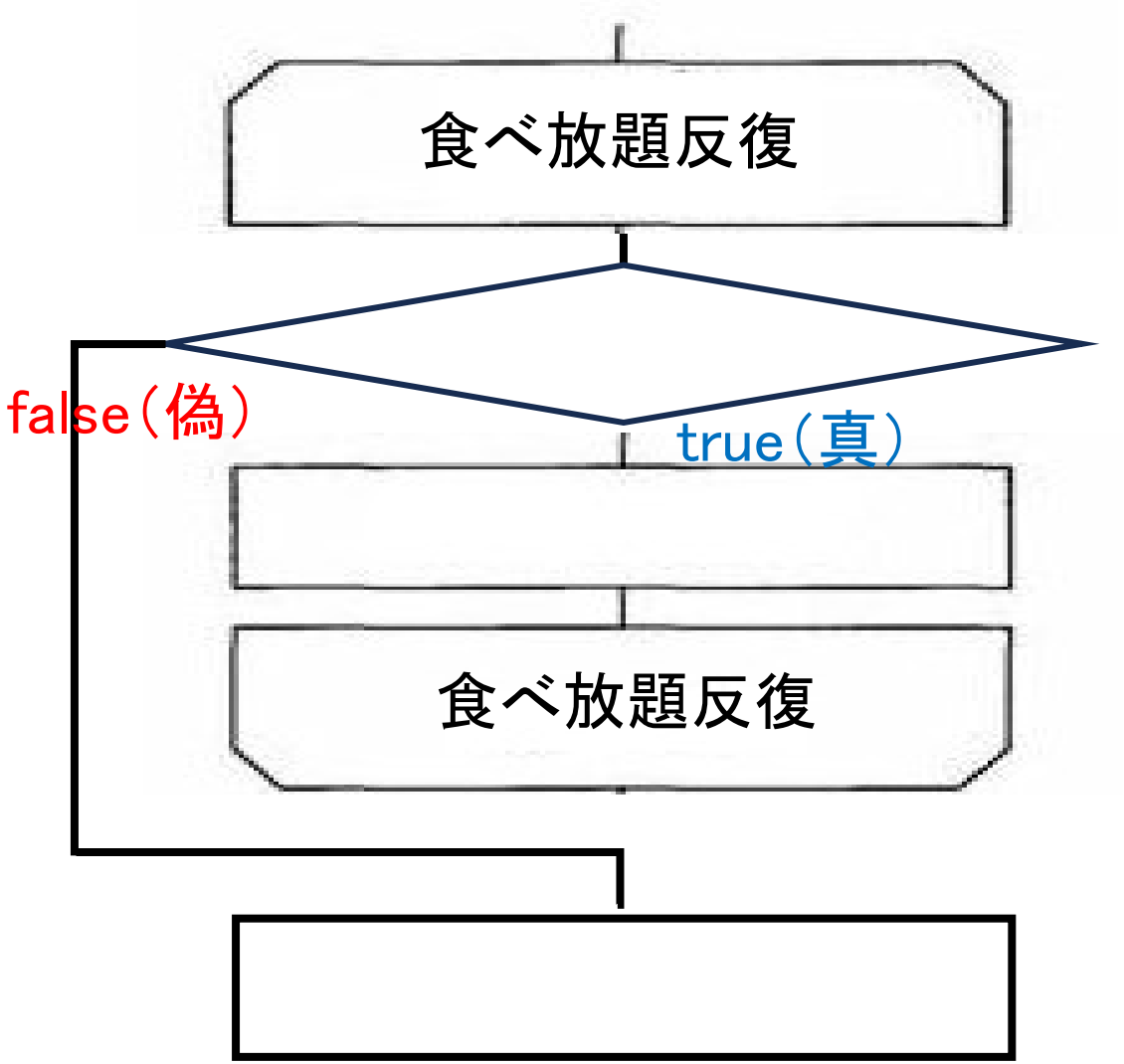


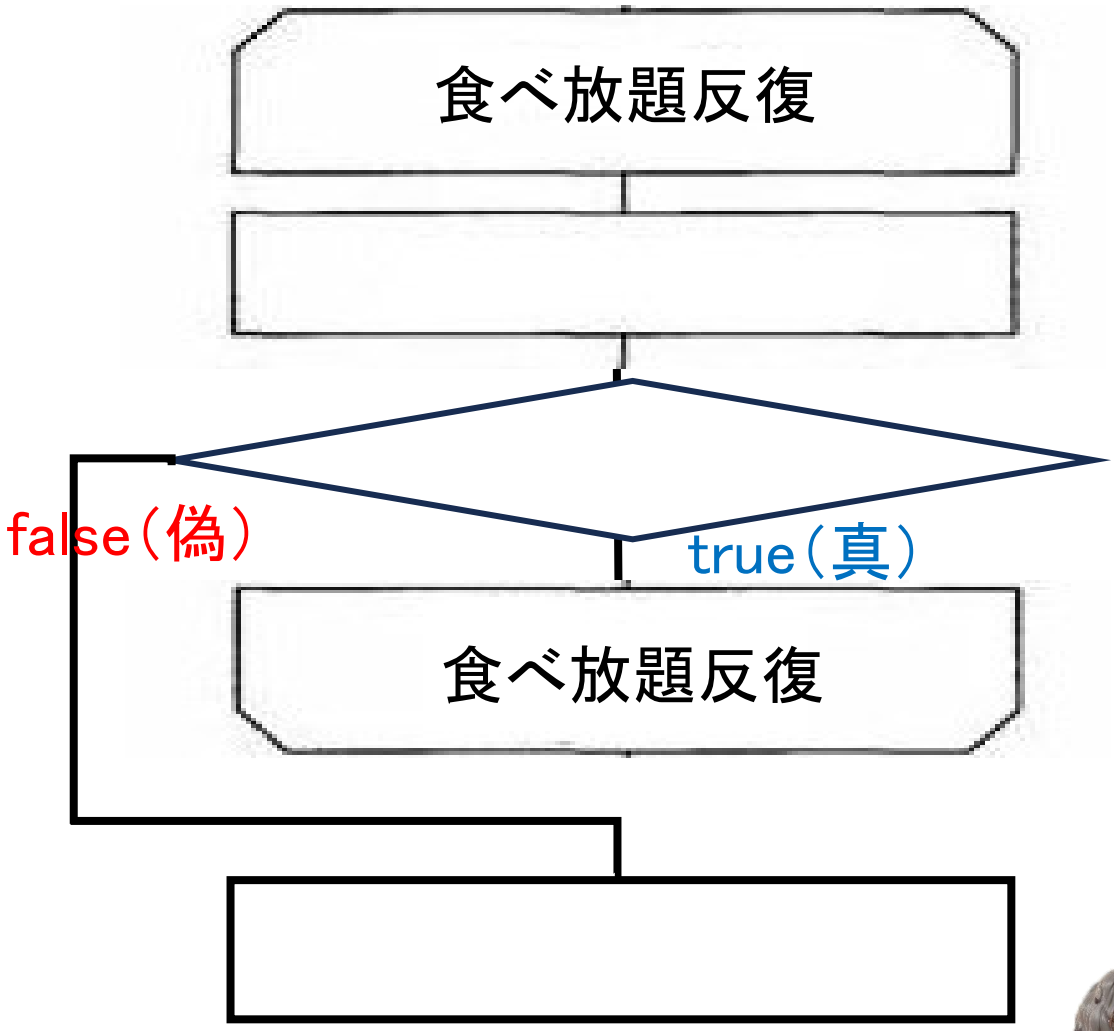
後判定：処理の後で判定する



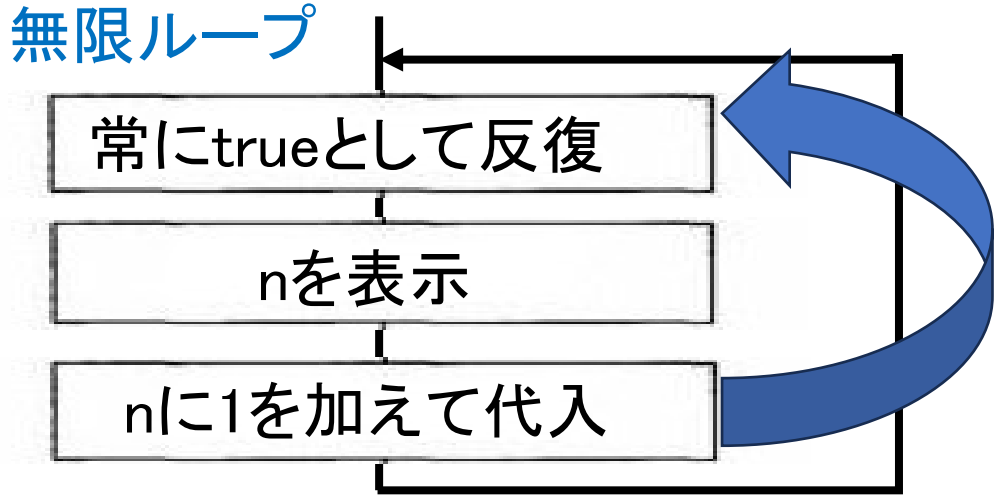
例題：満腹まで焼肉食べ放題の食事アルゴリズムをフローチャートで示せ。

満腹ではない　　お代わりする　　食事を終えて代金を払う



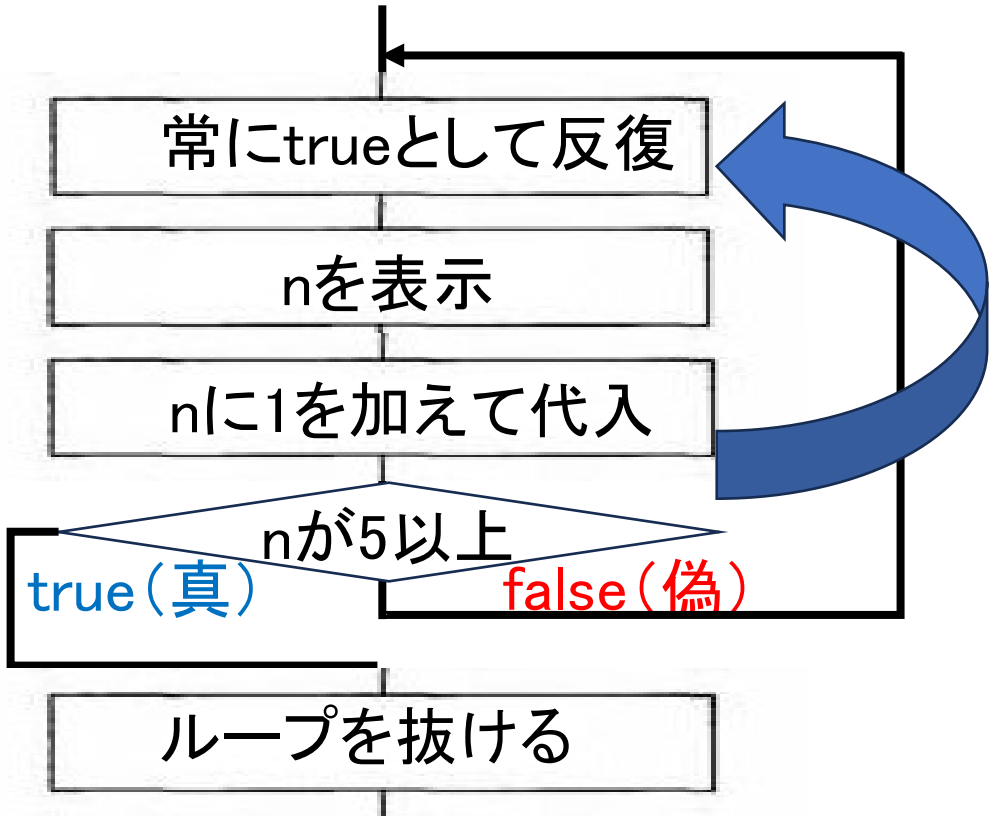


無限ループ



無限ループ例

```
n = 0                # nの初期値として0を代入
while True:          # 条件が常にTrue(=真)
    print(n)         # nを表示
    n += 1           # n = n+1 と同じ
```



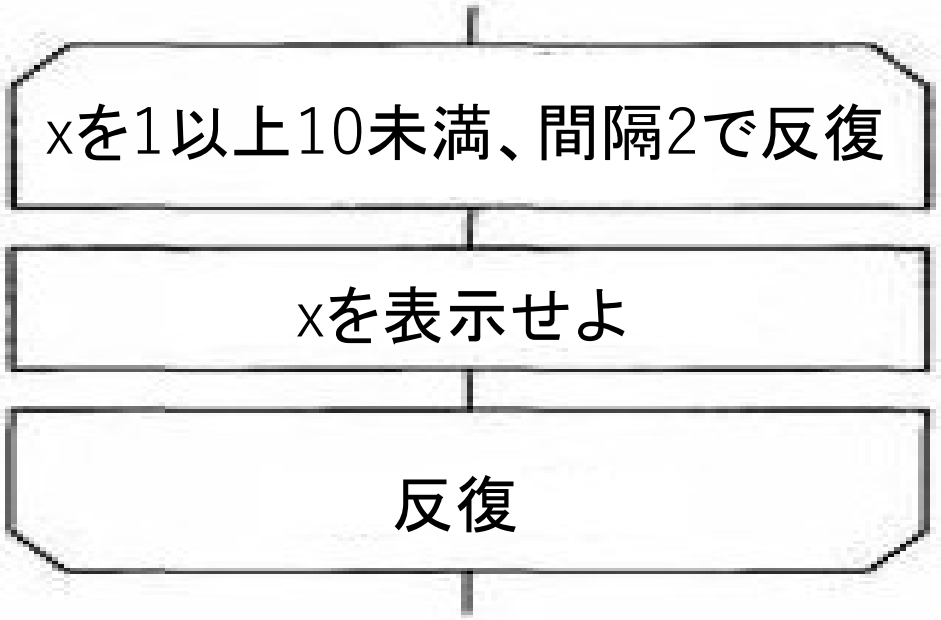
無限ループの抜け方例

```
n = 0
while True:
    print(n)
    n = n+1
    if n>=5:         # n >=5ならば
        break        # ループを抜ける
```

```
0
1
2
3
4
```

反復処理の例題1　変数xを1以上、10未満、間隔2で繰り返し表示せよ　for文

アルゴリズム



Pythonコード

スタート（以上）
ストップ（未満）
ステップ（間隔）

```
for x in range(1,10,2):
    print(x)
```

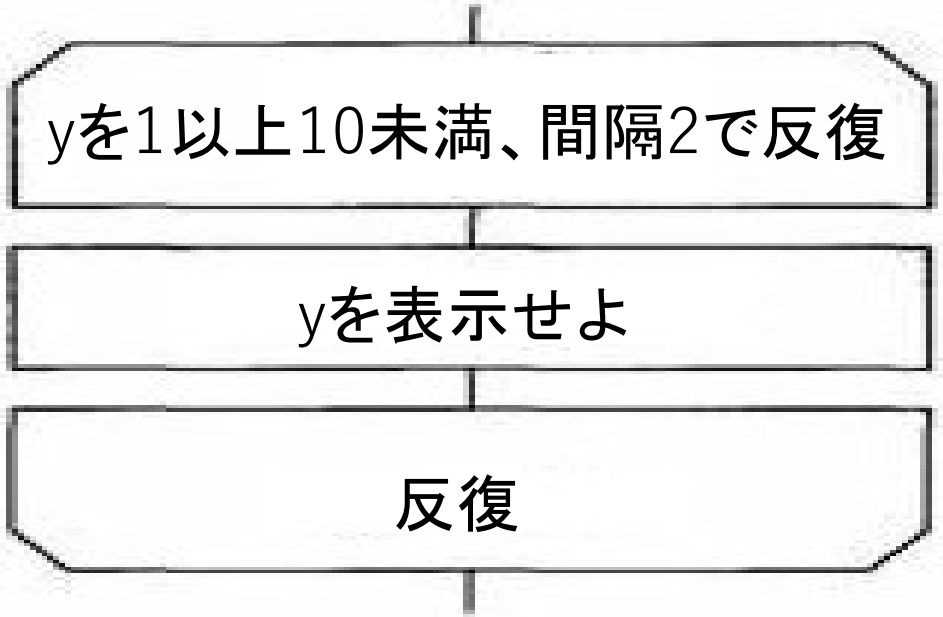
- ✓ for文行末に必ずコロン：を打つ
- ✓ 対応する処理は、2～4文字分の字下げ（インデント）を行う

rangeの扱い　range(start, stop, step)
start ～ stop未満でstepごとに連続した数値を返す。
start省略→0、step 省略→1
for x in range(5):　　0～5未満(0,1,2,3,4)の意味

```
1
3
5
7
9
```

反復処理の例題2　変数yを1以上、10未満、間隔2で繰り返し表示せよ while文

アルゴリズム　　Pythonコード



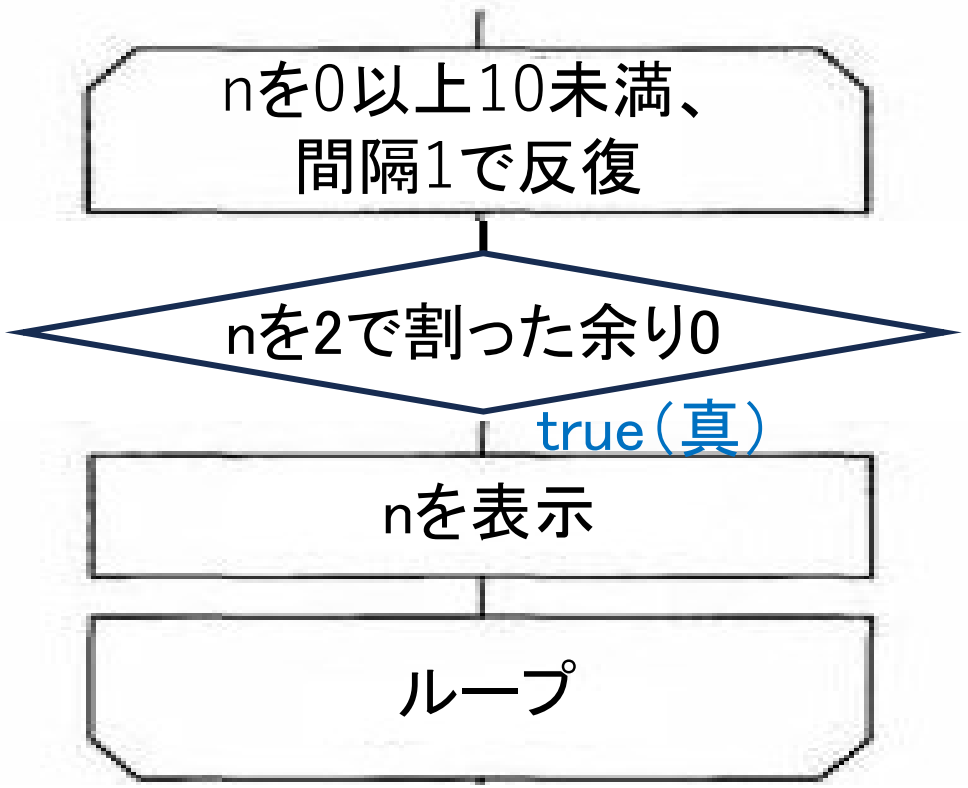

```
y=1                 # 初めはyに1を代入
while y<10:         # yは10未満とする
    print(y)        # yを表示
    y=y+2           # yにy+2を代入せよ
                    # 11は反復条件を抜ける
```

✓ while文行末に必ずコロン ： を打つ
✓ 対応する処理は、2～4文字分の字下げ（インデント）を行う

```
1
3
5
7
9
```

反復+分岐処理の例題　変数nを0以上、10未満の範囲で、偶数のみ表示せよ

アルゴリズム



Pythonコード

```
for n in range(0,10):
    if n%2==0:
        print(n)
```

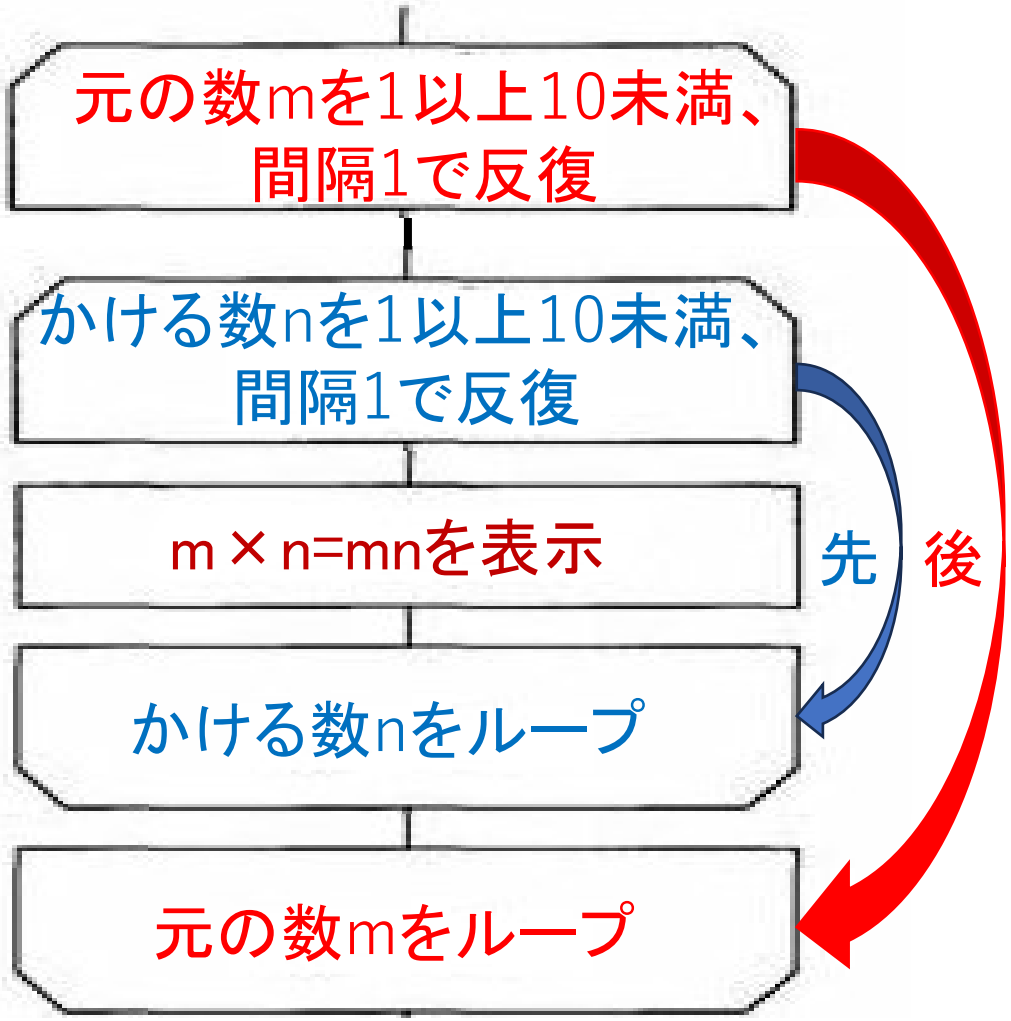
```
for n in range(0,10,1):
```

```
0
2
4
6
8
```

- ✓ for文行末に必ずコロン：を打つ
- ✓ if文行末に必ずコロン：を打つ
- ✓ それぞれ対応する処理は、2～4文字分の字下げ（インデント）を行う
- ✓ for文の中にif文が入れ子で入る

反復入れ子処理の例題　2つの変数m,nで掛け算九九を表示せよ

アルゴリズム



Pythonコード

```
for m in range(1,10):
    for n in range(1,10):
        print(m,'×',n,'=',m*n)
```

- ✓ 元の数mのfor文行末にコロン：を打つ
- ✓ かける数nのfor文行末にコロン：を打つ
- ✓ それぞれ対応する処理は、2～4文字分の字下げ（インデント）を行う
- ✓ for文の中にfor文が入れ子で入る

(5) input()関数：キーボードを使った文字列データ(プロンプト)の入力

```
x=input("what 's your name?")      # 入力値を変数xに代入する
print(x)                           # 表示
y=input("How old are you?")        # 入力値を変数yに代入する
print(y)                           # 表示➡inputで獲得した値は文字型
```

```
what's your name?john
john
How old are you?16
16
```

例題　年齢を回答させて、還暦までの年数を返すアプリ

```
y=input("How old are you?")                                  # 入力値を変数yに代入する
print(60-int(y),' years left until the 60th birthday')       # 還暦までの年数を表示
```

```
How old are you?16
44  years left until the 60th birthday
```

例題　名前n、身長h、体重wとする。bmi=w/(h*h)を算出し以下の判定を返すアプリ設計
参考：body mass index (BMI)
bmi25以上ならover weight、18.5<未満ならunder weight、18.5以上25未満ならhealthy weight
表示例： nさん、貴方のBMIは bmi、healthy weight

```
n = input('what is your name?')                          # nameを問う
h = input('How tall are you in meters?')                 # height（小数も有だね）を問う
w = input('How much is your weight in kilograms?')       # weight（小数も有だね）を問う
bmi=float(w)/float(h)**2                                 # bmi計算式（小数の計算）
if bmi>=25:                                              # 25以上ならば
    print(n, 'BMI= ' , bmi, 'over weight')                   # over weight
elif bmi <18.5:                                          # そうでなければ、18.5未満ならば
    print(n, 'BMI= ' , bmi, 'under weight')                  # under weight
else:                                                    # そうでなければ、
    print(n, 'BMI= ' , bmi , 'healthy weight')               # healthy weight
```

(6) 配列　複数の値を一括で扱うことができる番号付きの収納箱

Pythonコード　　注釈

index	0	1	2	3	4	5	6	7	8
value	3	8	6	1	9	5	4	2	7

```
list = [3,8,6,1,9,5,4,2,7]   # 変数listに配列を代入する･･･list, arrayなどがよく使われる
```

```
list.sort()                  # listを並べ替え（昇順）
print(list)                  # 表示する
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
list.sort(reverse=True)      # listを並べ替え（降順）
print(list)                  # 表示する
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
result = 0 in list           # 配列に0があるかの結果
print(result)                # 結果表示
```

```
False
```

```
result = 5 in list           # 配列に5があるかの結果
print(result)                # 結果表示
```

```
True
```

```
index = list.index(5)        # 配列にある5番目の値
print(index)                 # 結果表示
```

```
4
```

index	0	1	2	3	4	5	6	7	8
value	9	8	7	6	5	4	3	2	1

Pythonコード

注釈

index	0	1	2	3	4	5	6	7	8
value	4	5	2	1	3	5	3	5	1

```
list = [4,5,2,1,3,5,3,5,1]
```

```
c = list.count(5)        # 配列にある5の個数
print(c)
```

3

```
x=max(list)              # 配列の最大値
print(x)
```

5

```
y=min(list)              # 配列の最小値
print(y)
```

1

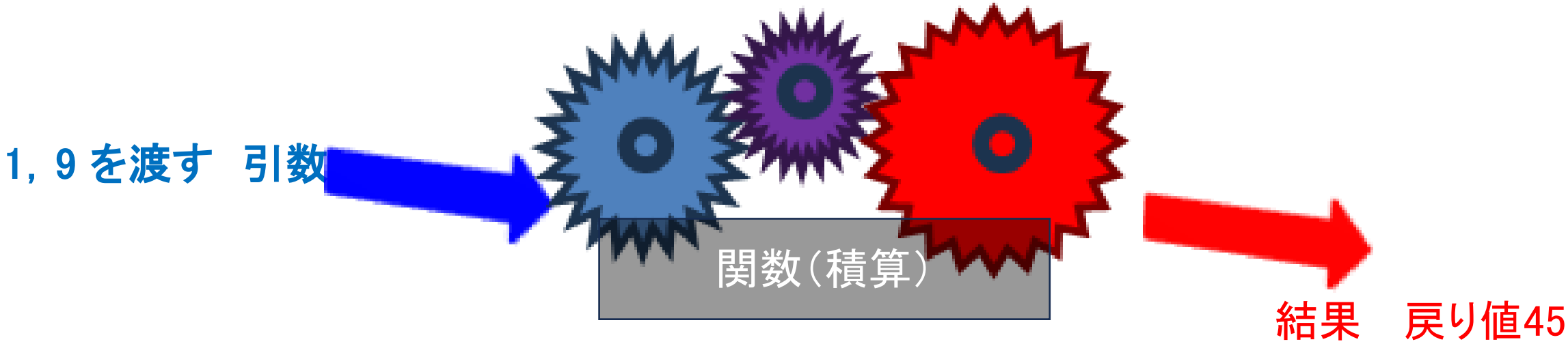
```
a=sum(list)              # 配列の合計値
print(a)
```

29

```
b=len(list)              # データの個数
print(b)
```

9

(7) 関数：複数処理をまとめてワンアクションで実行　自分で定義(define)できる。



例題　本体価格をxとして消費税10%を加えた税込み価格を求める関数を実装せよ

Pythonコード

注釈　# 以降はコメントアウト

```
def function(x):          # 関数function(x)を定義defineする
    price=x*1.1           # priceは本体価格xを1.1倍する税込価格
    return price          # 税込み価格（戻り値）を返せ
print(function(100))      # 本体価格を100（引数）として関数処理を実行し戻り値を表示
```

(8) モジュール、ライブラリ：目的に適する関数を組み合わせてツール化した仕組み

乱数モジュール
①賽の目

Pythonコード

注釈　# 以降はコメントアウト

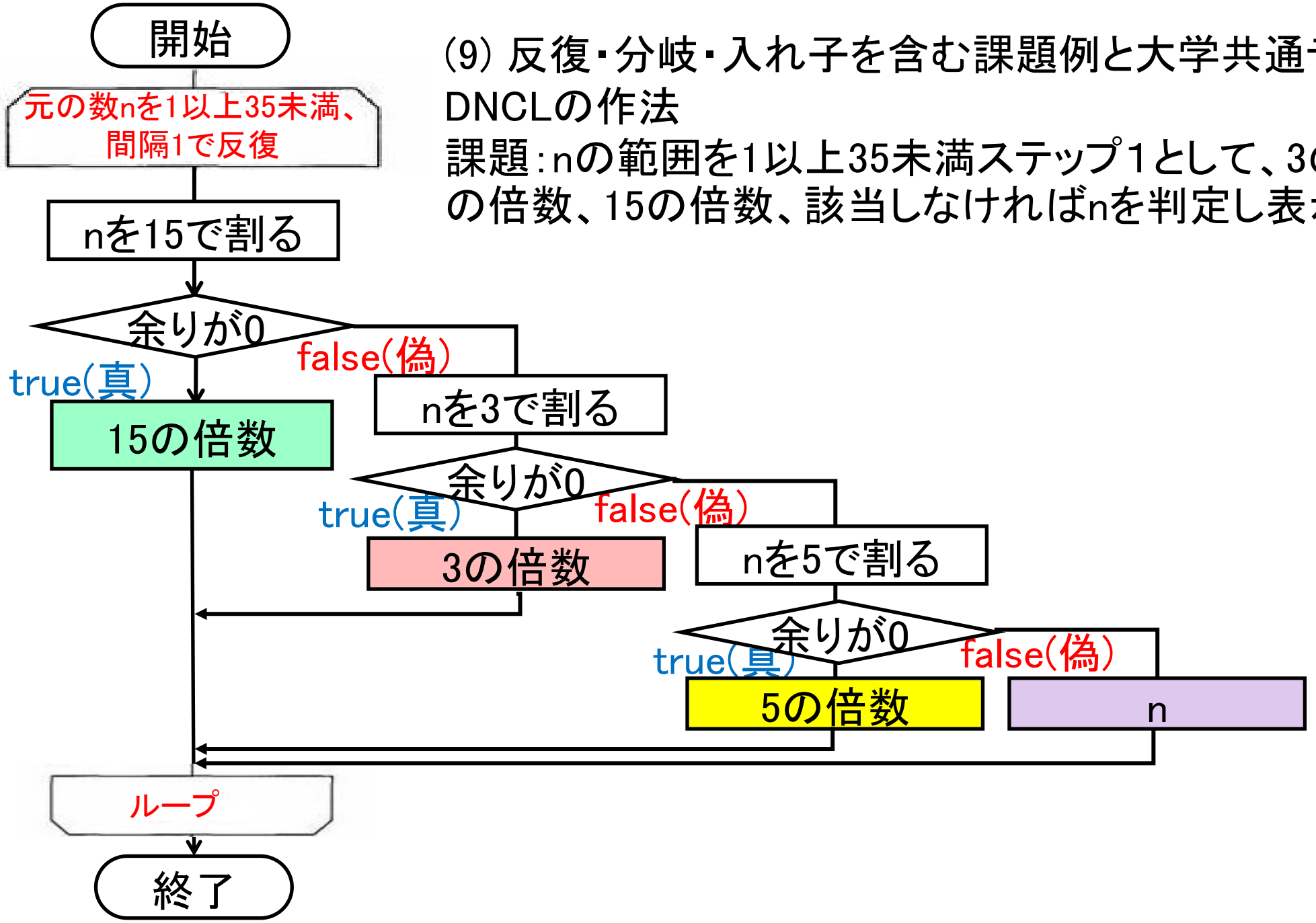
```
import random                  # 乱数を取り入れる
dice=[1,2,3,4,5,6]             # dice配列1～6
print(random.choice(dice))  # 乱数でdice配列から値を選択
```

②運勢　通称「おみくじアプリ」

Pythonコード

注釈

```
import random                                                                  # 乱数モジュール
fortune=[ ‘excellent’ , ‘luck’ , ‘good’ , ‘unknown’ , ‘bad luck’ , ‘terrible’ ]  # 運勢の配列
print(random.choice(fortune))                                                  # ランダム選択
```



(9) 反復・分岐・入れ子を含む課題例と大学共通テスト言語DNCLの作法

課題：nの範囲を1以上35未満ステップ1として、3の倍数、5の倍数、15の倍数、該当しなければnを判定し表示する

大学入試共通テストDNCL形式

```
(01)n を 1 から 35 まで 1 ずつ増やしながら:
(02) | もし n % 15 == 0 ならば:
(03) | | 表示する(“15の倍数”)
(04) | そうでなくもし n % 3 == 0 ならば:
(05) | | 表示する(”3の倍数”)
(06) | そうでなくもし n % 5 == 0 ならば:
(07) | | 表示する(”5の倍数”)
(08) | そうでなければ:
(09) |_ |_ 表示する(n)
```

Pythonコード

```
n=1
while n<35:
  if n%3==0 and n%5==0:
    print('Multiple of 15')
  elif n%3==0:
    print(”Multiple of 3”)
  elif n%5==0:
    print(' Multiple of 5')
  else:
    print(n)
  n=n+1
```

(10) 定番課題：素数Prime Number発見問題

- 素数Prime Number：「1」と「その数自身」でしか割りきれない数
- 試し割り法：整数 n を2以上の整数 i で順に割り、割り切れるか判定（素因数分解）
- 素因数が1つであれば素数と判断
- 判定が $\sqrt{n}$ までで十分な理由

① n が自然数 i で割り切れる場合、商を j とすると $n = ij$

② 小さい自然数から順に素因数か(割り切れるか)を確認していくため、j が i より小さい場合($i > j$)には既に素因数として検出済

具体例　$6 = 2 \times 3$ ➡ 6の素因数は「2と3」➡ $6 = 3 \times 2$ 、$6 = 2 \times 3$

故に　i と j が一致する数値である$\sqrt{n}$まで確認すればよい

60未満の素数を発見するアルゴリズム

```
for n in range(60):                          # nを 60未満で反復
  def sosu(n):                               # 素因数が無い数を見つける関数sosuを定義
    if n<2:                                  # nが 2未満の場合
      return False                           # 除外
    for i in range(2,int(n**0.5) + 1):       # iを2以上√60の整数値+1未満で反復
      if n % i == 0:                         # nをiで割って余りが0ならば素因数なので
        return False                         # 除外
    return True                              # 素因数を持たない n を戻り値として返す
  if sosu(n):                                # nが素数の場合
    print(n)                                 # 表示
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
```

大学入試共通テストDNCL形式

```
n を 0 から 60未満で 1 ずつ増やしながら:
 ｜ 関数sosu(n)を定義:
 ｜ ｜ nが2未満ならば:
 ｜ ｜ ｜ Falseを返す（除外）
 ｜ ｜ 2からnの平方根までの各整数iについて:
 ｜ ｜ ｜ nがiで割り切れるならば:
 ｜ ｜ ｜ ｜ Falseを返す（除外）
 ｜ ｜ True（素数）を返す
 ｜ もしsosu(n)ならば:
 ｜ ｜ nを表示する。
```

Pythonコード

```
for n in range(60):
  def sosu(n):
    if n<2:
      return False
    for i in range(2,int(n**0.5) + 1):
      if n % i == 0:
        return False
    return True
  if sosu(n):
    print(n)
```