# Pythonプログラミング～前半～

会津大学

復興知2022年度

# 講習会内容

- Pythonプログラミング
  - Pythonの紹介
  - 演算子，標準入出力，データ型，変数，プログラム実行
  - 条件分岐
  - 繰り返し処理
  - リスト
  - ライブラリの利用
    - 乱数の生成
    - グラフの作成
  - 関数

# プログラミング言語 Python

- プログラムがシンプルで，読みやすさ・書きやすさを重視した設計
- 様々なOSに対応 (Windows, macOS, Unix, Linuxなど)
- 人工知能(AI)，ロボット，データサイエンスなど，
  幅広い開発分野で利用


- **本講習が対象とするPythonのバージョンは3系（ver 3.10.x）**

# Pythonの起動・終了方法

❖ Pythonでプログラムを実行する方法は**2通り**ある

1. 対話モードで，直接コードを記述して実行する
2. ファイルにコードを記述し，読み込ませて実行する

❖ 対話モードとは，簡単にPythonプログラムを実行する環境

❖ windows環境における対話モードの起動は，
「スタートメニュー」→「Windowsシステムツール」→
「コマンドプロンプト」を起動後，コマンドプロンプトで

**python** と入力して，「Enter」キーを押す

❖ Mac, Linuxの場合、ターミナル上で **python3** と入力して，
「Enter」キーを押す

# 対話モードの起動・終了方法

- 対話モードの終了は，コマンドプロンプトで **quit()** と入力し，「Enter」キーを押す
- Pythonでは **#** 以降は説明コメントとなる
  - プログラムとして扱われない
- これ以降，↵ は「Enter」キーを押すことを意味する
- **>>>** はpythonのプログラムを待機している状態を表す

```
> python↵ # 対話モードを起動する
Python 3.10.4 (main, Jun 14 2022, 21:01:25)
[Clang 13.1.6 (clang-1316.0.21.2.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>> quit()↵ # 対話モードを終了する
```

# 対話モードでの数式実行

❖ 対話モードで簡単な数式を入力し実行することで,
数式の計算結果が表示される

```
>>> 4 + 3↵
7
>>> 5 - 3↵
2
>>> 7 * 2↵
14
>>> 8 / 2↵    # 整数同士の除算結果が小数となることに注意
4.0
>>> 3 + 4.0↵ # 整数と小数の計算結果は小数となることに注意
7.0
```

# 対話モードでの数式実行

- 計算の順番は，数学の数式と同様
  - 丸括弧( ) → 乗除計算 → 加減計算の順序で行う

```
>>> (3 + 4) / (2 + 3)↵ # 7 / 5 = 1.4
1.4

>>> 3 + 4 / 2 + 3↵      # 3 + 2.0 + 3 = 8.0
8.0
```

- 計算順番さえ分かれば，対話モードを簡易電卓として使用可能
  - 自由に試してみてください

# 算術演算子

- 計算（加算，減算，乗算，除算など）に関する演算子
  - 演算子: 各種の演算（計算，比較，代入など）を表す記号

| 演算子 | 意味 | 使用例 | 結果例 |
|---|---|---|---|
| + | 加算 | 5 + 3 | 8 |
| - | 減算 | 5 - 3 | 2 |
| * | 乗算 | 5 * 3 | 15 |
| / | 除算 | 5 / 3 | 1.6666666666666667 |
| % | 剰余（余り） | 5 % 3 | 2 |
| ** | 累乗 | 5 ** 3 | 125 |
| // | 切り捨て除算 | 5 // 3 | 1 |

# Pythonで文字列を扱う

- シングルクォーテーション ' か，ダブルクォーテーション " を使用して，扱う文字列を囲む
  - どちらの記号を使っても問題ないが，前後の囲む記号を統一すること

```
>>> 'abc123'↵
'abc123' # 文字列の場合，シングルクォーテーションが両端に表示される
```

- 文字列同士は，+演算子で繋げることができる

```
>>> "abc" + "xyz"↵
'abcxyz'
```

# Pythonで文字列を扱う | 応用事例

❖ **文字列 * 自然数**にした場合，文字列を繰り返して連結した文字列となる

```
>>> 'a' * 4↵ # 文字列aを4回繰り返した文字列
'aaaa'

>>> 'abc' * 3↵ # 文字列abcを3回繰り返した文字列
'abcabcabc'

>>> '123' * 2↵ # 文字列123を2回繰り返した文字列
'123123'

>>> 'a' * 8 + 'b' * 4↵
# 文字列aを8回繰り返した文字列と文字列bを4回繰り返した文字列との連結
# 文字列'aaaaaaaa'と文字列'bbbb'を連結
'aaaaaaaabbbb'
```

# 標準出力

- コマンドプロンプトに数値，文字列，演算結果を
  標準出力するには，**print()**を使用
  - **print(文字列)**とした場合，
    シングルクォーテーションが付かないことに注意

```
>>> print(987)↵        # 数字の出力
987

>>> print('qwerty')↵ # 文字列の出力
qwerty

>>> print(2 ** 6)↵    # 計算結果の出力
64
```

# Pythonにおける基本的データ型

| データ型 | 例 |
| --- | --- |
| 整数 (int) | 0, 12, -345, ··· |
| 小数 (float) | 0.12, -2.34, -45.67, ··· |
| 文字列 (string) | 'xyz', "0123", 'i4j5k6', ··· |
| 真偽値 (boolean) | True, False |

# データ型とキャスト

- キャスト: データ型を変換する
  - **小数から整数に変換した場合，小数点以下の情報が消える**

```
>>> float(10)↵    # 整数から小数にキャスト
10.0

>>> int(3.3)↵     # 小数から整数にキャスト
3

>>> str(666)↵     # 整数から文字列にキャスト
'666'

>>> int('1234')↵ # 文字列から整数にキャスト
1234
```

# 標準入力

- キーボードから標準入力するには，**input()**を使用する
  - 標準入力した値は文字列データとして扱われるため，数値として使用したい場合，適切にキャストを行う
  - 数値と文字列の足し算はエラーになるので注意

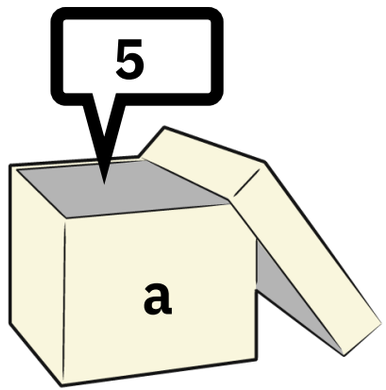
```
>>> input() + '456'↵      # キーボードで入力した値は文字列
123↵                      # キーボードで入力する
'123456'                  # 出力結果（文字列の足し算：123と456の連結）

>>> int(input()) + 456↵   # キーボードで入力した値を整数にキャスト
123↵                      # キーボードで入力する
579                       # 出力結果（整数の足し算：123+456の演算結果）
```

# 変数

- コンピュータのメモリ上の領域に名前を付けて値を格納する仕組み
  （**この仕組みを利用して，値を一時保存できる**）
  - 数値や文字列に付ける名札のようなもの
- 『**変数 = 初期値**』で変数の初期化を行う
  - **=** を代入演算子とよぶ（**= を等号の意味で使用しない**）
- 『**変数 = input()**』とすると，
  その変数に，キーボードから入力した値が格納される
- 『**print(変数)**』とすると，その変数に格納されている値を，
  表示することができる

**a = 5** とすると，
aという名前がついた箱の中に
整数の5というデータが入っているイメージ



# 変数 | 補足説明

- 『**print(変数1, 変数2, ･･･, 変数n)**』とすると，
  『**変数1 変数2 ･･･ 変数n**』と表示する
- 変数は自由に付けられるが，以下のルールがある
  - 英数字，アンダースコア _ のみ使用可能
  - 数字から始まらないこと（英字，アンダースコアはOK）
- 英字の大文字と小文字は区別される
  - **ABCとabcは別の変数**
- 予約語（Pythonの文法で定義されている名前）は使用不可
  - if，for，andなど

# 変数 | 応用事例

❖ 『**x, y = y, x**』とした場合，**変数xと変数yの値を交換する**

```
>>> x = 4↵
>>> y = 3↵
>>> x, y = y, x↵ # 変数xとyの値を交換する
>>> x↵
3
>>> y↵
4
```

❖ 『**x, y = y, x**』の部分は，以下のプログラムと同じ

```
>>> temp = x↵ # 変数を1つ用意して，値を保存しておく
>>> x = y↵
>>> y = temp↵ # 保存した値を代入する
```

# 変数の使用例

```
>>> a = 10↵                    # 変数aに10を代入
>>> print(a)↵                  # 変数aの値を表示
10

>>> b = int(input())↵          # キーボードから入力した値を変数bに代入
2↵                             # キーボード入力
>>> print(b)↵                  # 変数bの値を表示
2

>>> c = a + b↵                 # 変数cに変数aの値と変数bの値の和を代入
>>> print(c)↵                  # 変数cの値を表示
12

>>> print(a, b)↵               # 変数aとbの値を表示
10 2
```

# 変数の使用例

- 『変数 = 同じ変数 演算子 値』で変数の値を更新できる
  - プログラミング特有の書き方

例1: x = x + 4 (『変数xに4を加えた値』を『変数xに代入する』)

例2: y = y - 5 (『変数yに5を引いた値』を『変数yに代入する』)

```
>>> a = 11↵          # 変数aに11を代入
>>> print(a)↵        # 変数aの値を表示
11

>>> a = a * 11↵      # 変数aの値を更新する（11倍した値にする）
>>> print(a)↵        # 変数aの値を表示
121
```

# 複合代入演算子

- 変数に演算した結果を同じ変数に代入する演算子
  - 変数の値を更新する場合に使われることが多い

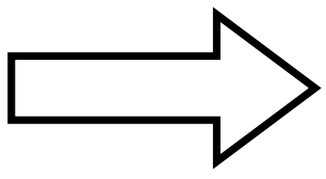
| 演算子 | 使用例 | 意味 |
|---|---|---|
| += | i += 1 | i = i + 1 |
| -= | i -= 2 | i = i - 2 |
| *= | i *= 3 | i = i * 3 |
| /= | i /= 4 | i = i / 4 |
| %= | i %= 5 | i = i % 5 |
| **= | i **= 6 | i = i ** 6 |
| //= | i //= 7 | i = i // 7 |

# Pythonプログラムの実行

- 対話モードではなく，ファイルにソースコードを作成し，作成したファイルを実行する
  - テキストエディタを使用して，「**.py**」ファイルを作成
    - **py** は拡張子
  - 上から書かれているソースコードを1行ずつ順番に実行する
- コマンドプロンプト上で『**python ファイル名.py**』とすることで実行できる（**dir**コマンドで実行したいファイルがあることを確認）
  - Mac，Linuxの場合『**python3 ファイル名.py**』で実行できる

```
print('Hello World')
```

**hello.pyを作成**

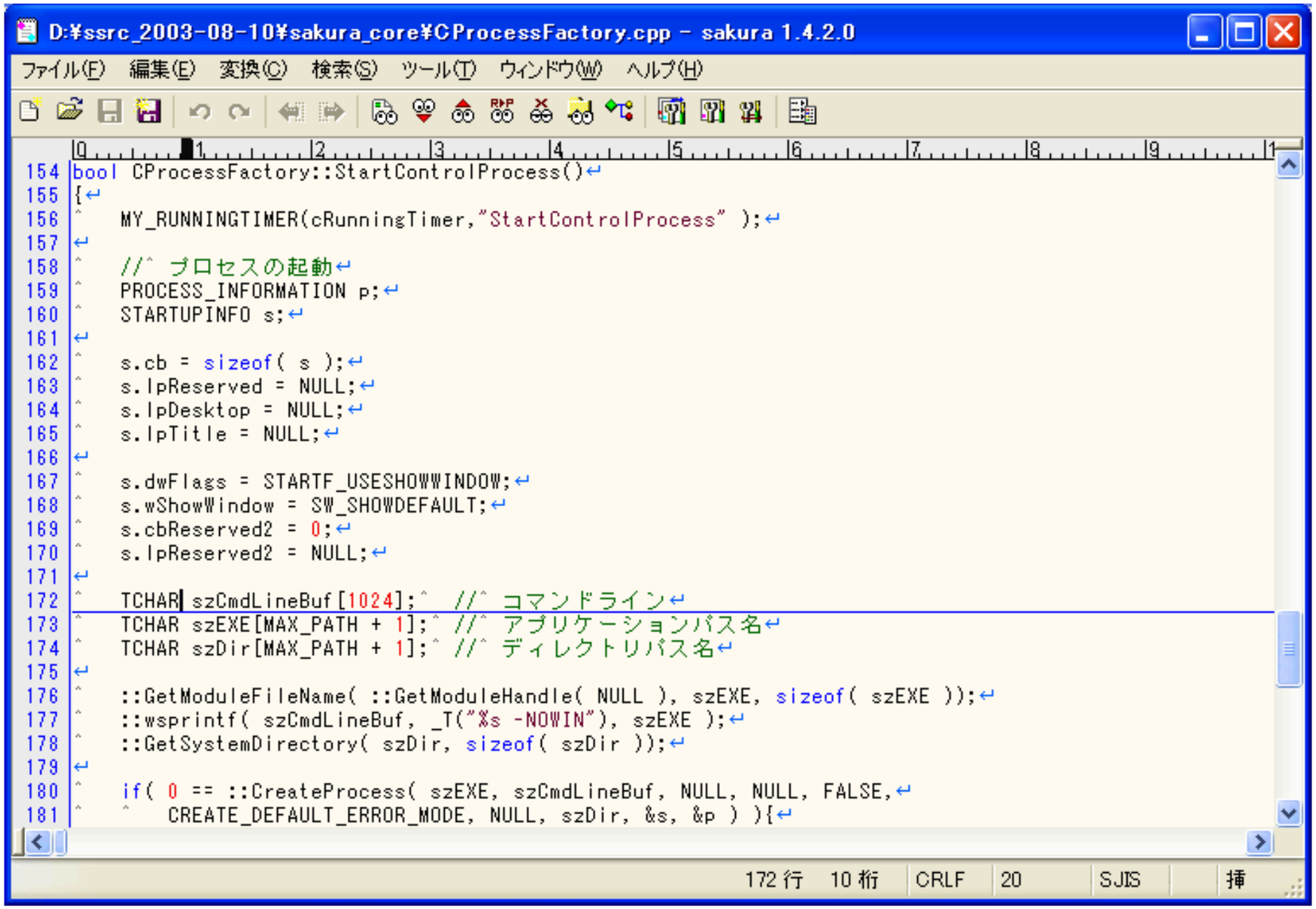


```
python hello.py↵
Hello World # 実行結果
```

**コマンドプロンプト上で実行**

# テキストエディタの導入

❖ 例: サクラエディタ（https://sakura-editor.github.io/）



# ここまでのまとめ演習1-A

1. 自分の名前をprint()を使って表示させてみよう
2. 2の10乗の値をprint()を使って表示させてみよう
3. 200円の商品を3つ，30円の商品を6つ購入しました，

   このとき消費税を10%としたときの値段をprint()を使って
   表示させてみよう
4. キーボードから文字列を入力および変数に代入し，
   以下のように出力するプログラム作成せよ

   （xxxはキーボードから入力した文字を表す）

```
Hello, xxx !
```

# ここまでのまとめ演習1-B

❖ Aizu Online Judgeからの出題

1. `Hello World` と一行に出力せよ
2. 1つの整数 x を読み込んで,
   その整数 x の３乗の値を出力するプログラムを作成せよ
3. 長方形の縦の長さ a と 横の長さ b を読み込んで,
   その長方形の面積と周の長さを求めるプログラムを作成せよ
4. 秒単位の時間 S を読み込んで, 「h:m:s」 の形式に変換して
   出力してください
   （hは時間，mは60未満の分，sは 60未満の秒とすること）

# 条件分岐

# 条件分岐

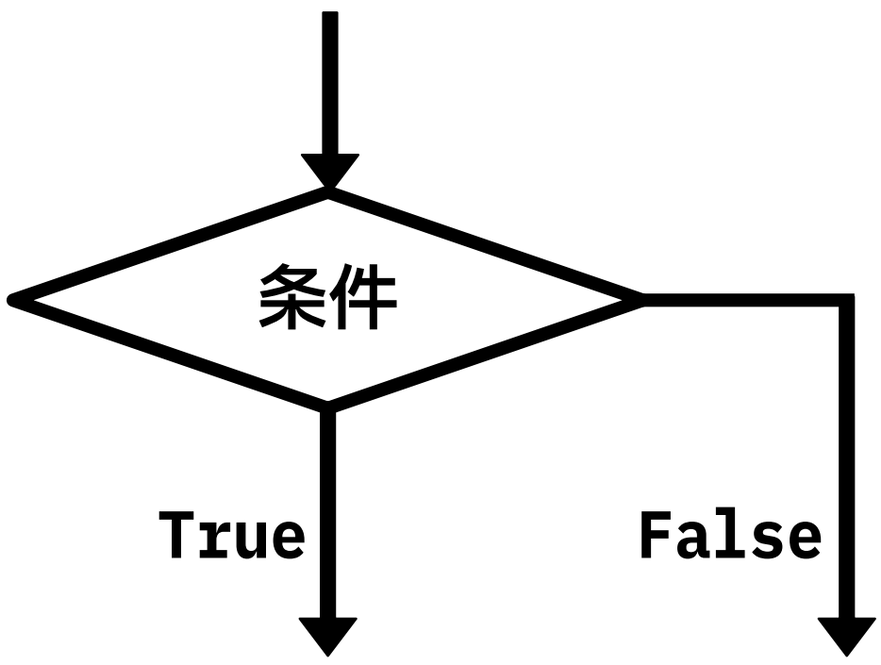
❖ 条件によって処理が異なるプログラムを作成する

例1:「**700円以上なら，**くじを引く」

例2:「**80点以上なら，**スコアAと表示する」

例3:「**20km以上40km未満なら，**中距離である」

❖ 条件を「満たしている場合」，「満たしていない場合」を，
**真偽値**(**True**/**False**)を使用して，プログラムを作成する



# 比較演算子

- 2つの数字や文字列の比較を行うときに使用する演算子
  - 加算や減算などの代数演算子よりも後に計算する
- この比較演算子を使用して条件分岐を行う

| 演算子 | 数学記号 | 意味 |
|---|---|---|
| == | = | 等しい |
| != | ≠ | 等しくない |
| < | < | より小さい |
| > | > | より大きい |
| <= | ≤ | 以下 |
| >= | ≥ | 以上 |

# 条件分岐

- 「**700円以上なら，**くじを引く」の条件をプログラムで表現
- **以下の例のように変数と比較演算子を使用して条件を記述する**
  - **条件を満たしている場合: True**

```
>>> cost = 800↵
>>> cost >= 700↵ # 条件
True # 条件を満たすため
```

  - **条件を満たしていない場合: False**

```
>>> cost = 100↵
>>> cost >= 700↵ # 条件
False # 条件を満たさないため
```

- 「**80点以上なら，**スコアAと表示する」の条件をプログラムで表現してみてください

# if文

- 条件分岐を行うためには**if文**を記述する
  - if文で記述されている条件を満たしている(True)の場合，if文の直後のコードブロックの処理が実行される
    - コードブロック: 字下げされている部分
  - 条件を満たしていない(False)の場合，コードブロックの処理が実行されない

```
if 条件:
    # Trueの場合の処理を
    # コードブロック内に書く
```

**【if文の記述方法】**

**コロン( : )を忘れずに，記述する**

```
x = int(input())
if x % 2 == 0:
    print('偶数です')
```

偶数判定プログラム例

**print箇所がコードブロック**

# コードブロック

- 条件分岐は1行以上のひとまとまりのコードブロックで表現
- Pythonでは，コードを字下げ（インデント）することで，
  コードブロックを作成可能
  - 字下げ: 行の先頭に決まった数の半角スペースを入力する
  - **Pythonは字下げを判断して，プログラムを実行するため，
    ブロック化がきちんとできていない場合，プログラムが
    意図した動作をしない（エラーとなることもある）**
  - 「Tab」キーを押すことで，設定したスペース数が入力される
  - **日本語の全角スペースを入力しないように注意**
    - **全角スペースは見つかりづらいエラーの原因の1つ**

# コードブロックの違いによる変化

❖ 以下のプログラムをテキストエディタに書いて,
実際に実行してみてください

```
x = 0
if x == 1:
    print('A')
print('AA')
```

```
# コードブロック1行と判断
AA # 1行出力する実行結果
```

```
x = 0
if x == 1:
    print('A')
    print('AA')
```

```
# コードブロック2行と判断
# 出力しない実行結果
```

# if-else文

- **if-else文**は，ifの条件に応じて実行される処理が異なる
  - 条件を満たす(**True**)場合
    - if文に続くコードブロックの処理が実行
  - 条件を満たさない(**False**)の場合
    - else文に続くコードブロックの処理が実行

```
if 条件式:
    # Trueのときの処理
else:
    # Falseのときの処理
```

**【if-else文の記述方法】**

**コロン( : )を忘れずに，記述する**

```
x = int(input())
if x % 2 == 0:
    print('偶数です')
else:
    print('奇数です')
```

偶数奇数判定プログラムの例

# if-elif-else文

- elif文は，複数の条件分岐（if, else以外の処理）で使用する
  - if文や他のelif文に続けて記述する
  - 3つ以上の条件分岐（多分岐）に使用する

```
if 条件1:
    # 条件1がTrueのときの処理
elif 条件2:
    # 条件2がTrueのときの処理
elif 条件3:
    # 条件3がTrueのときの処理
else:
    # 条件式1～3が全てFalseのときの処理
```

**【if-elif-else文の記述方法】**

**コロン( : )を忘れずに記述する**

# 論理演算子

- 2つ以上の条件を組み合わせ，真偽を判定する演算子
  - **and**: 論理積（A **かつ** B）
    - 2つの条件を満たしている場合: **`True`**
    - どちらか一方の条件が満たしていない場合: **`False`**
  - **`or`**: 論理和（A **または** B）
    - どちらか一方の条件を満たしている場合: **`True`**
    - 2つの条件を満たしていない場合: **`False`**

例: 「**20km以上40km未満なら，**中距離である」の条件を表現

- **`d >= 20 and d < 40`** # 変数dに距離の値を代入した場合

# ここまでのまとめ演習2-A

1. 整数を入力して，その値が「正の値」，「0」，
   「負の値」かどうかを判定するプログラムを作成せよ

   例1: 3を入力すると，「正の値」と出力

   例2: 0を入力すると，「0」と出力

   例3: -3を入力すると，「負の値」と出力

2. 西暦（整数）を入力して，その西暦がうるう年かどうかを
   判定するプログラムを作成せよ

   ❖ うるう年の条件は以下の通りである
   「400で割り切れる年」または，
   「4で割り切れて，かつ100で割り切れない年」

# ここまでのまとめ演習2-A

2. 西暦（整数）を入力して，その西暦がうるう年かどうかを判定するプログラムを作成せよ

   例1: 2000を入力すると「うるう年です」と出力

   例2: 2100を入力すると「うるう年ではない」と出力

3. 点数（整数）を入力して，以下のように出力するプログラムを作成せよ

   - 60～100の場合: Pass!
   - 0～59の場合: Do not give up!
   - それ以外の場合: Error

# ここまでのまとめ演習2-B

❖ Aizu Online Judgeからの出題

1. 2つの整数 a, b を読み込んで，a と b の大小関係を出力するプログラムを作成せよ
2. 3つの整数a, b, cを読み込み，それらが「a < b < c」の条件を満たす場合には”Yes”を，満たさない場合には”No”を出力するプログラムを作成せよ
3. 3つの整数を読み込み，それらを値が小さい順に並べて出力するプログラムを作成せよ
4. 3つの正の整数を入力し，それぞれの長さを3辺とする三角形が直角三角形である場合には”Yes”を，違う場合には”No”と出力するプログラムを作成せよ

# 繰り返し処理

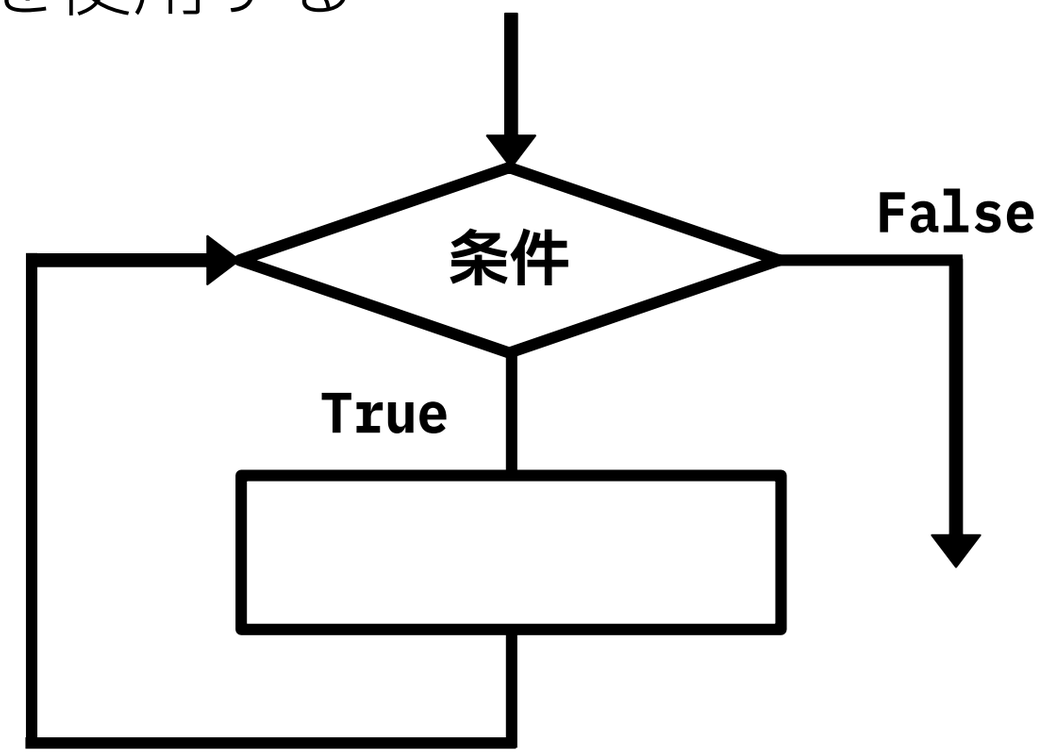
# 繰り返し処理

❖ 条件を満たすまで，コードブロックにあるプログラムを
繰り返し行うことができる

例1: サイコロの目で1が出るまで，サイコロを振り続ける

例2: 合計金額が2000円を超えるまで，お寿司を食べ続ける

❖ **while文**，**for文**を使用する



# while文

- while文を使用することで，コードブロックにあるプログラムを繰り返すことができる
  - while文の条件を満たしている間，プログラムが実行される

```
while 条件:
  # 条件を満たしている間,
  # コードブロックを
  # 繰り返して実行する
```

【while文の記述方法】

コロン( : )を忘れずに，記述する

```
i = 0
while i < 11:
  print(i)
  i += 1
```

0から10を1つずつ出力する
プログラム例

# while文による無限ループ

- while文の条件にTrueを与えると無限ループになる
- 無限ループがあるプログラムを実行すると，

  プログラムが終わらないため，**break文**を使用することで，
  無限ループを抜けることができる

- **while文を使用する場合，条件をきちんと定義する**
- 意図をせず無限ループに陥るプログラムを実行した場合，

  「ctrlキー」を押しながら，「cキー」を押すことで，プログラムを
  強制終了できる

```
while True:
    break # ここで強制的にループ抜ける
# breakが無いと無限ループになり，プログラムが終わらない
```

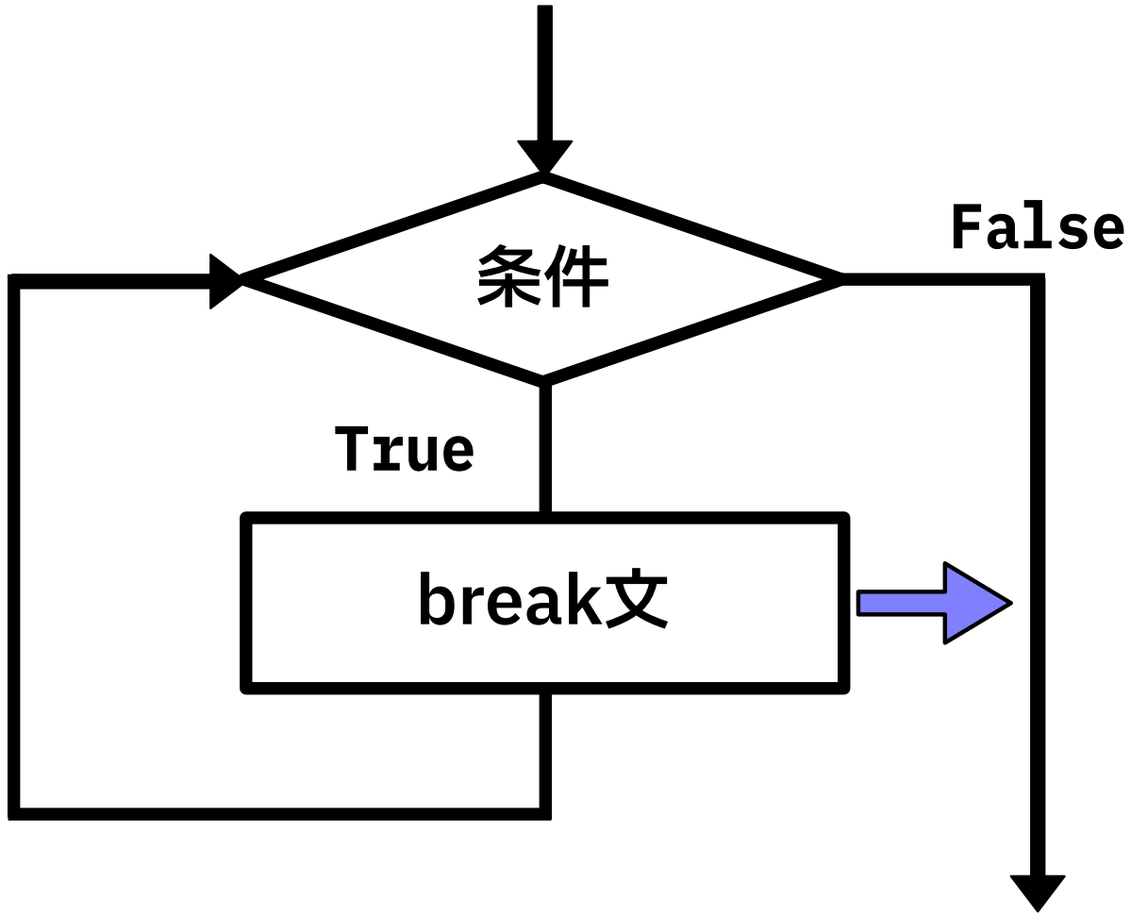
# break文

- 繰り返し処理の中で，break文を使用できる
  - break文が実行されると，即座にその繰り返し処理のコードブロックから抜ける

```
i = 0
while True:
   if i > 2:
      break
   print(i)
   i += 1
```

変数iが2を越えたら

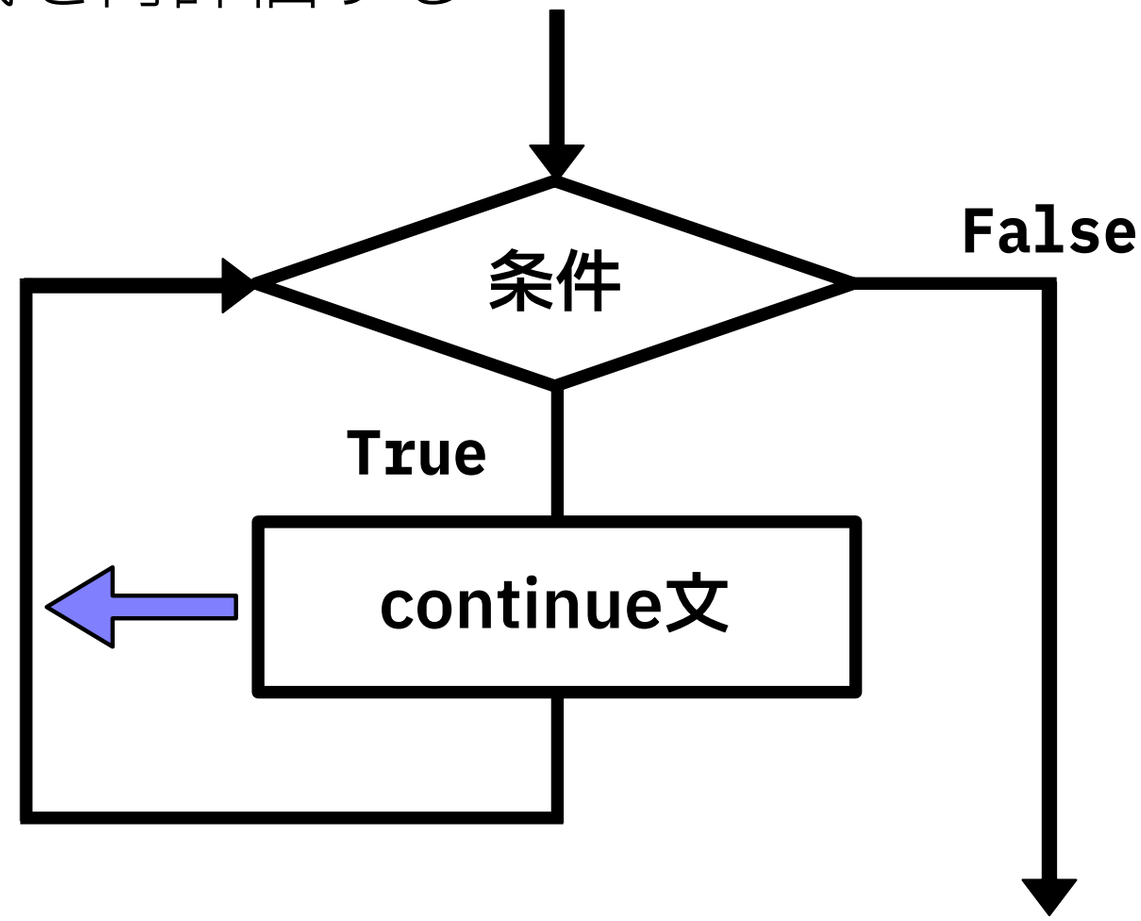
break文でループを抜ける



# continue文

- 繰り返し処理の中で，continue文を使用できる
  - continue文が実行されると，即座に繰り返し処理の先頭に戻り，繰り返しの条件式を再評価する

```
i = 0
while True:
    if i > 10:
        break
    if i % 2 == 0:
        i += 1
        continue
    print(i)
    i += 1
```




変数iが偶数時にループの先頭に戻る

continue文使用例

# for文

❖ 一定の回数だけコードのブロックを実行したいときは,

**for文**と**range()**を使用する

❖ 繰り返し回数に応じて,

**range(n)**は変数を 0, 1, 2, 3, …, n-1 と変化させる

```
for 変数 in range(繰り返し数):
    # 繰り返し処理
```

```
for i in range(3):
    print(i)
```

**【for文の記述方法】**

**コロン( : )を忘れずに，記述する**

繰り返し回数に応じて,

変数iが0, 1, 2と変化するプログラム例

# range()

- range()のカッコの中に，3つまでの値を書くことができる
  - 複数の値を書く場合，コンマで区切る
- range()のカッコの中に，2つの値を書いた場合，
  1番目の値から2番目の値を越えないように繰り返す
- range(m, n)は m, m+1, m+2, …, n-2, n-1 を表す

```
for i in range(3, 7):
    print(i)
```

繰り返し回数に応じて，

変数iが3, 4, 5, 6と変化するプログラム例

# range()

- range()のカッコの中に，3つの値を書いた場合,
  - 1番目の値から2番目の値を越えないように繰り返す
  - 3番目の値は繰り返しを行う際の変数の更新量を意味する
- 3番目の値が正の値:
  - 2番目の値以上にならないように繰り返す

  例: range(2，11，2)は，2, 4, 6, 8, 10 を表す
- 3番目の値が負の値：
  - 2番目の値以下にならないように繰り返す

  例: range(5，0，-1)は，5, 4, 3, 2, 1 を表す

# 繰り返し処理の例

❖ 繰り返し条件，繰り返ししたときに変化する変数を意識する

例: 1から100までの自然数の和を計算するプログラム

- ❖ 「自然数の和の結果」を保存する変数を作成する
- ❖ 0で初期化を行い，繰り返し処理の中で足していく

```
# while文
i = 0
sum = 0
while i <= 100:
    sum += i
    i += 1

print(sum)
```

```
# for文
sum = 0

for i in range(0, 101):
    sum += i

print(sum)
```

# ここまでのまとめ演習3-A

1. 『$1^3 + 2^3 + 3^3 + 4^3 + \cdots + 48^3 + 49^3 + 50^3$ 』を，
   繰り返し処理を利用したプログラムで求めよ

2. 「0を入力するまでおうむ返しする」プログラムを作成せよ
   - ❖ この問題は整数が入力されるものとする

3. 整数を入力して，その値を累積する（足し算する）ことを行い，
   累積値が30を超えたら，累積値と入力した回数を出力して終了する
   プログラムを作成せよ

# ここまでのまとめ演習3-B

❖ Aizu Online Judgeからの出題

1. 15個の "Hello World" を出力するプログラムを作成せよ
2. 3つの自然数a, b, cを読み込み，a から b までの自然数の中に，cの約数がいくつあるかを求めるプログラムを作成せよ
3. 5個の整数を入力し，それらの最小値，最大値，合計値を求めるプログラムを作成せよ
4. 入力した自然数が，「121」のように逆から数字を並べても同じ数になる回文数である場合には”Yes”を，
   違う場合には”No”と出力するプログラムを作成せよ

# 構造化プログラミング

❖ 1960年代後半にエドガー・ダイクストラによって提唱

**『1つの入口と1つの出口を持つプログラムは，**
**順次・選択・反復の3つの論理構造によって記述できる』**

1. 順次: 上から下へ順番にプログラムが実行される
2. 選択: 条件Aなら，処理aを行う（条件分岐）
   - ❖ `if`文，`elif`文，`else`文
3. 反復: 条件Bを満たす間，処理bを繰り返す（繰り返し処理）
   - ❖ `while`文，`for`文