

# Python講座

2016/5/12(木)

# 今回の内容

- ▶ 比較演算子(引用)
- ▶ 条件式の詳細①
- ▶ if文
- ▶ while文
- ▶ 条件式の詳細②
- ▶ 関数・組み込み関数①
- ▶ 組み込み関数②
- ▶ ユーザー定義(自作)関数
- ▶ 自作関数(例)
- ▶ 自作関数(注意点)
- ▶ turtle
- ▶ turtleに関する文
- ▶ 演習問題

# 比較演算子

▶ 以降のページに出てくる条件式に対して使用する

| Pythonでの書き方            | 数学での書き方    | 意味               |
|------------------------|------------|------------------|
| <code>a == b</code>    | $a = b$    | aはbと等しい          |
| <code>a != b</code>    | $a \neq b$ | aはbと等しくない        |
| <code>a &lt; b</code>  | $a < b$    | aはbより小さい         |
| <code>a &gt; b</code>  | $a > b$    | aはbより大きい         |
| <code>a &lt;= b</code> | $a \leq b$ | aはbより小さいか等しい(以下) |
| <code>a &gt;= b</code> | $a \geq b$ | aはbより大きい(以上)     |

# 条件式の詳細①

- ▶ 条件式から得られるものは、TrueかFalse
- ▶ 条件式が満たされている場合はTrue(真)
- ▶ 条件式が満たされていない場合はFalse(偽)
- ▶ 条件式から得られたものによって、if文やwhile文の処理の有無が変わる

# if文

- ▶ 条件式が満たされる場合、内容を実行する
- ▶ 条件式が満たされない場合、内容を実行しない



- ▶ 条件式がTrueである場合、内容を実行する
- ▶ 条件式がFalseである場合、内容を実行しない

# while文

- ▶ 条件式が満たされる間、内容を実行する
- ▶ 条件式が満たされない間、内容を実行しない



- ▶ 条件式がTrueである間、内容を実行する
- ▶ 条件式がFalseである間、内容を実行しない

# 条件式の詳細②

▶ 条件式的位置にTrue、Falseを直接書くことも可能

▶ if True:

```
    print("A")
```

```
>>> A
```

▶ while True:

```
    print("A")
```

```
>>> A A A A ...
```

▶ Trueのため、必ず実行される

※while Trueを使う場合は、breakを書かなければ無限ループするので注意

# 関数・組み込み関数①

- ▶ プログラムを再使用するためのしくみ
- ▶ print()、input()、sum()なども関数
- ▶ 正確には、組み込み関数というもの
  - ※Python自体に元々組み込まれている関数のこと

<http://docs.python.jp/3/library/functions.html>(3.5.1)

# 組み込み関数②

- ▶ 組み込み関数ではない関数の使用例
- ▶ 例：乱数
- ▶ `import random`
  - `random`の関数を組み込む
- ▶ `random.random()`、`random.randint(値a, 値b)`
  - `random`の関数を使う

# ユーザー定義(自作)関数

- ▶ 以下の作業を「関数を定義する」という
- ▶ `def` 関数名(`変数1`):      ▶ `def` 関数名(`変数1`):  
    内容                                      内容  
                                                `return 変数2`
- ▶ 関数名はわかりやすいものが好ましい
- ▶ 変数1は引数(`ひきすう`)と言い、内容で使用できる
- ▶ `return`でこの関数で返却するもの(`変数2`)を決める

# 自作関数(例)

```
def plus(x, y):  
    x+=y  
    return x
```

```
a=10  
b=5  
print(plus(a,b))
```

```
>>>  
15  
>>>
```

▶ 関数名 : plus

▶ 引数 : x、 y

▶ 内容 : xを+yする

▶ 返却 : x

↓

▶ a、 bを引数として  
a + bの値を返却する関数

# 自作関数(注意点)

- ▶ 関数内にreturnは複数書けるが、返却できるものは**1つのみ**
  - if文などで指定する場合が多い
- ▶ **関数外で使用している変数を関数内**では使用不可
  - 使う場合は**引数**で関数内に引っ張ってくる
- ▶ **関数内で使用している変数を関数外**では使用不可
  - 使う場合は**return**で関数外に返却
  - 逆に内外で変数名が同じでも問題なし

# turtle

- ▶ グラフィックを書くことができる
- ▶ `import turtle`  
→turtleを組み込む
- ▶ `t=turtle.Pen()`  
→タートルオブジェクト作成  
(初めて作るときはウィンドウも作成される)

# turtleに関する文

▶ <http://docs.python.jp/3.3/library/turtle.html>

|                                 |                              |
|---------------------------------|------------------------------|
| t.forward( <b>x</b> )           | まっすぐ <b>x</b> 進む             |
| t.left( <b>x</b> )              | 反時計回りに <b>x</b> 度回転          |
| t.right( <b>x</b> )             | 時計回りに <b>x</b> 度回転           |
| t.reset                         | 白紙にする                        |
| t.circle( <b>r</b> , <b>x</b> ) | 半径 <b>r</b> を反時計に <b>x</b> 度 |
| t.goto( <b>x</b> , <b>y</b> )   | 座標( <b>x</b> , <b>y</b> )に進む |
| t.up                            | 描画をしない                       |
| t.down                          | 描画をする                        |

# 演習問題①

## ▶ テーマ：

第一回の演習問題①「ノット30(仮)」の改良

## ▶ 改良点

① Player vs CPU(1~8人) 対戦人数を選択

② Playerの順番を選択可能(またはランダム)

※Playerの勝敗のみ、CPUの誰が勝ったかは不問

# 演習問題① 補足

- ▶ 作る関数は4つ
  1. 以下の3種類の関数をまとめた関数Game
  2. CPUの人数、Playerの順番を決める関数Select
  3. CPUのゲーム処理を行う関数CPU
  4. Playerのゲーム処理を行う関数Player
- ▶ Gameを実行するだけでノット30が遊べる形

## 演習問題②

- ▶ テーマ：六芒星の描画
- ▶ やること：turtleで図を描く
- ▶ 描画する際に使うもの  
`import math`  
`math.sqrt(値)`
- ▶ 値の平方根を返す

