第4回　関数
第5回　内包表記

目次

関数

- ここまでの内容で、いくつかの関数を使ってきました。
  - print関数、range関数、str関数、input関数など

- 関数は、プログラム内でよく利用する処理や作業を、手軽に呼び出せるように用意されているもので、Pythonで最初から用意されている関数を組み込み関数と呼びます。

- 関数の呼び出し（復習）
  - 構文 `戻り値を受け取る変数 = 関数名（引数1, 引数2, ・・・）`

関数を定義する

- Pythonでは新しい関数を自分で作る(定義する)こともできます。
  - 繰り返し利用するような処理や、他のプログラムでも再利用できそうな処理は、関数にまとめておくとプログラムを効率的に作ることができます。
  - 関数定義の構文
    - 関数はdef文で定義します。
    - defの後に関数名を置き、引数のリストを丸括弧で囲みます。引数がない場合は丸括弧の中は空にします。
    - 関数内の処理はブロックに書きます。
    - 呼び出し元に結果を返すにはreturn文を使います。

BMIを計算するcalBMI関数の定義例です。
- 引数として体重、身長(m)を取ります
- 最後にreturn 文で変数bmiの値を返します。

```
def calBMI(taiju, shincho):
    bmi = taiju / shincho / shincho
    if bmi > 25.0:
        print("太っている")
    else:
        print("太っていない")
    return bmi
```

関数を呼び出す

- いったん定義した関数は、呼び出せば何度でも実行できるので便利です。
- 実引数と仮引数
  - 関数呼び出しで指定する引数のことを実引数と呼びます。
  - 関数定義の引数のリストに記述した変数を仮引数と呼びます。「仮」とは、実際に関数が呼び出されるまでどのような値が入るかわからないからです。
- ローカル変数
  - 関数の中で使われている変数は関数の中でだけ有効です。
  - 関数の中と外で同じ名前の変数があったとしても、別のものとして扱われます。

仮引数

```
def calBMI(taiju, shincho):
    bmi = taiju / shincho / shincho
    if bmi > 25.0:
        print("太っている")
    else:
        print("太っていない")
    return bmi

bmi = 0
a = calBMI(60, 1.6)
print(a)
a = calBMI(80, 1.6)
print(a)
print(bmi)
```

2つのbmiは別扱い

実引数

実引数

```
太っていない
23.4375
太っている
31.25
0
```

引数のキーワード指定とデフォルト値

- 引数のキーワード指定
  - 複数の引数をとる関数を呼び出す場合、引数の順序に注意する必要があります。
  - 実引数に対して仮引数名をキーワード指定することで、順番にかかわらず目的の仮引数に値を渡すことができます。
  - 方法：引数指定の際に、「仮引数名=実引数」の形式で渡す

```
a = calBMI(shincho=1.6, taiju=90)
print(a)
a = calBMI(taiju=40, shincho=1.5)
print(a)
```

- 引数のデフォルト値
  - 関数定義の仮引数の後ろに「=値」として、デフォルト値を設定しておくことができます。デフォルト値が設定された引数は、呼び出す際に省略することができます。

```
def calBMI(taiju, shincho=1.5):
    bmi = taiju / shincho / shincho
     ・・・以下省略・・・
```

複数の戻り値

- 戻り値はreturn文で呼び出し元に返すことができますが、複数の戻り値を戻すこともできます。
  - 構文 `return 戻り値1, 戻り値2, ・・・`

- 呼び出し側での受け取り
  - 複数の戻り値を返す関数を呼び出した場合、結果はタプル型で受け取れます。
  - タプルでなく単独の変数として受け取るには、=の前に戻り値の数だけ変数をカンマで区切って並べます。

タプルで受け取る場合

```
def calBMI(taiju, shincho=1.5):
    bmi = taiju / shincho / shincho
    if bmi > 25.0:
        txt = "太っている"
    else:
        txt = "太っていない"
    return bmi, txt
a = calBMI(60, 1.6)
print(a)
```

```
(23.4375, '太っていない')
```

単独の変数で受け取る場合

```
a, b = calBMI(60, 1.6)
print(a)
print(b)
```

```
23.4375
太っていない
```

課題4

- p7のcalBMI関数では、身長はm（メートル）で指定する仕様になっていました。
- 私たちの普段の生活では、身長はcmで扱ったほうがわかりやすいので、shinchoをcmで指定できるようにcalBMI関数を修正してみてください。

```
a, b = calBMI(60, 1.6)
```

身長をmでなくcm で指定できるように！

```
a, b = calBMI(60, 160)
```

内包表記(コンプリヘンション)

- Pythonでは、リスト等に含まれる複数の要素に対して何か処理を行いたい場合はfor文を使うことができます。
- しかし、簡単な処理でもfor文の場合には複数行に渡るブロックの記述が必要です。

リストaの各要素を2乗したリストbを作成する例

```
a = [2, 4, 6, 8, 10]
b = []
for i in a:
    b.append(i*i)
print(b)
```

- 内包表記とは？
  - Pythonでは、内包表記という構文を使うと、リストやディクショナリを加工するような処理をブロックを使わずに1行で簡潔に書けるようになります。
  - しかも、実行速度も速くなります。

リスト内包表記(1)

- リスト内包表記は、あるシーケンスをもとにして、新しいリストを返す式として記述します。
  - 構文

リスト変数 = [一時変数による式 for 一時変数 in シーケンス]

  - 例:

```
a = [2, 4, 6, 8, 10]
b = []
for i in a:
    b.append(i*i)
print(b)
```

これまでのfor文の場合

＝

```
a = [2, 4, 6, 8, 10]
b = [i * i for i in a]
print(b)
```

内包表記の場合

リストa内の要素をiに取り出してきて、iを基にリストbを作る

リスト内包表記(2)

- 要素を追加する条件の指定
  - 内包表記にはifを追加することができ、条件がTrueの場合のみ要素が追加されます。
  - 構文

リスト変数 = [一時変数による式 for 一時変数 in シーケンス if 条件式]

  - 例:

```
a = [2, 4, 6, 8, 10]
b = []
for i in a:
    if i != 6:
        b.append(i*i)
print(b)
```

これまでのfor文の場合

＝

```
a = [2, 4, 6, 8, 10]
b = [i * i for i in a if i != 6]
print(b)
```

iが6以外なら追加

内包表記の場合

リスト内包表記(3)

- 元となる要素の値によって追加する値の式を切り替えたい場合
  - 先ほどのifは要素を追加するかしないかを記述するための表記でしたが、追加する要素の式を切り替えたい場合には、forの前にifを書きます。
  - 構文

リスト変数 = [条件式が成り立つ場合の式 if 条件式 else 条件式が成り立たない場合の式 for 一時変数 in シーケンス]

  - 例：bmiが25.0以上の場合「太っている」それ以外「太っていない」という文字列に変換しなさい。

```
bmi = [25.4, 23.2, 26.7]
b = ["太っている" if i >= 25.0 else "太っていない" for i in bmi]
print(b)
```

```
['太っている', '太っていない', '太っている']
```

リスト内包表記(4)

- 元となる要素のインデックスも追加する要素の値や条件に加味したい場合
  - enumerate関数を使うことで、元となるリストのインデックス値を取り出すことができ、追加する要素の値や条件の材料にすることができます。
  - 構文

リスト変数 = [一時変数による式 for インデックス, 一時変数 in enumerate(シーケンス)]

  - 例:インデックス値をもとに、番号名簿を作る

```
a = ["鈴木", "田中", "加藤"]
b = [str(i+1) + "番:" + j for i, j in enumerate(a)]
print(b)
```

```
['1番:鈴木', '2番:田中', '3番:加藤']
```

リスト内包表記(5)

- 複数リストの要素全ての組合せから新たなリストを作る
  - 内包表記では、forを複数記述することで、複数のリストのすべての要素の組合せから新たなリストを作成することができます。
  - 構文

リスト変数 = [一時変数iとjによる式 for 一時変数i in シーケンスA for 一時変数j in シーケンスB]

    - この構文では、シーケンスAとシーケンスBのすべての要素の組合せが評価されます。
  - 例:ある大学にはAからCまでの棟があり、それぞれ3階建である。A棟1階からC棟3階までの要素を持つリストを作りなさい。

```
a = ["A棟", "B棟", "C棟"]
b = ["1階", "2階", "3階"]
c = [i + j for i in a for j in b]
print(c)
```

```
['A棟1階', 'A棟2階', 'A棟3階', 'B棟1階', 'B棟2階', 'B棟3階', 'C棟1階', 'C棟2階', 'C棟3階']
```

リスト内包表記(6)

- 複数リストの同じ順番の要素から新たなリストを作る
  - zip関数を使うことで、複数のリストから同じ順番に要素を取り出して、新たなリストを作成することができます。
  - 構文

リスト変数 = [一時変数による式 for 一時変数i, 一時変数j in zip(シーケンスA, シーケンスB)]

  - 例:3名の生徒の英語と数学の合計点を求める。

```
eigo = [80, 35, 65]
sugaku = [64, 76, 35]
goukei = [i + j for i, j in zip(eigo, sugaku)]
print(goukei)
```

```
[144, 111, 100]
```

ディクショナリ内包表記

- リストと同様にディクショナリにおいても内包表記が可能です。ディクショナリの場合には波括弧{}を使います。
  - 構文

ディクショナリ変数 = [新たなキー:新たな値 for 一時キー変数,一時値変数 in 既存ディクショナリ.items()]

  - 例:市名をキー、市外局番を値とするディクショナリから、キーと値が逆のディクショナリを作る

```
a = {"静岡市":"054", "浜松市":"053", "沼津市":"055"}
b = {v:k for k, v in a.items()}
print(b)
```

```
{'054': '静岡市', '053': '浜松市', '055': '沼津市'}
```

課題5

- 3名の生徒の名前、英語の成績、数学の成績が、name、eigo、sugakuというリストで与えられたとします。
- 英語と数学の合計が120点以上の場合「名前:合格」、それ以外の場合「名前:不合格」というリストを内包表記で作ってください。
- whileを使った例を下記に示します。

```
name = ["鈴木", "田中", "加藤"]
eigo = [80, 35, 65]
sugaku = [64, 76, 35]
kekka = []
i = 0
while i < len(name):
    goukei = eigo[i] + sugaku[i]
    seiseki = ""
    if goukei >= 120:
        seiseki = "合格"
    else:
        seiseki = "不合格"
    kekka.append(name[i] + ":" + seiseki)
    i += 1
print(kekka)
```

```
['鈴木:合格', '田中:不合格', '加藤:不合格']
```