繰り返しとは？

レッスン4「プログラムの基本構造」で説明しましたが、プログラムの基本的な動きは「順次」「分岐」「繰り返し」の3つです。

繰り返しとは、決まった回数や条件を満たしてれば、同じ処理を実行するプログラム構造です。

for文

繰り返しの代表例がfor文です。

for文は、条件を満たしていれば、同じ処理をぐるぐる繰り返します。

そして、条件を満たさなくなったタイミングで、繰り返しがおわります。

例えば、for文で同じ処理を5回繰り返したい場合で考えてみます。

どうやったら5回をカウントできるでしょうか。

例えば、「1」からスタートして、1ずつ増やしていき「5」で終われば、5回です。

例えば、「0」からスタートして、1ずつ増やしていき「4」で終われば、これも5回です。

そういった処理をPythonのfor文で書いていきます。

Pythonでのfor文のきまりをみていきましょう。

```
for 変数 in range(繰り返し回数):
    繰り返し中に実行する処理
```

まずforを書きます。次に繰り返し回数を格納する変数を記述します。

ここで定義した変数のことをカウンタ変数といいます。

for文では、繰り返し処理をカウンタ変数によって制御します。

5回繰り返したい場合は、カウンタ変数に数字が順次代入され、5回目がきたら繰り返し処理が終わります。

カウンタ変数は、英語の「index」「Iterator」の頭文字「 i 」が使われることが多いです。

次にinと書いて、range。丸括弧を書きます。

丸括弧の中に繰り返したい回数を、数値で記述します。

最後にコロンを記述します。

コロンの次の行はインデントを下げて、繰り返したい処理を記述します。

繰り返したい処理が終わったら、一番上に戻り、繰り返し処理が終わりになるか判定します。

繰り返し回数分、繰り返していないなら、次のループをします。繰り返し回数分繰り返していたら、ループが終了します。

コードを書いてみましょう。

```
for i in range(5):
    print(i)
```

実行結果：

```
0
1
2
3
4
```

break

次に、break文についてです。

break文は、ある条件にあてはまったとき、繰り返し処理を終了させることができます。

例えば、「0」からスタートさせて、1ずつ増やしていき、「3」になったら繰り返しを終了するといったときです。

コードを書いてみます。

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

ここで、iが3にになったときに、breakする記述をします。

「3」でループを抜けるので、「0」から「2」まで表示されるはずです。

実行します。

実行結果：

```
0
1
2
```

「0」から「2」までが表示されました。

continue

次に、continue文についてです。

continue文は、繰り返し処理で、ある条件にあてはまったときにその処理をスキップしたい場合に使います。

例えば、「0」からスタートさせて、1ずつ増やしていったとき、「3」になったら「3」をスキップさせるという場合です。

コードを書いてみます。

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

if文でiが3にになったときに、処理をスキップさせる

continueを記述をします。

「3」をスキップするので、「0」「1」「2」「4」が表示されるはずです。

実行してみます。

実行結果：

```
0
1
2
4
```

「0」「1」「2」「4」が表示されました。

for文のネスト

for文の中にfor文を入れることもできます。

あるものの中に、それと同じ種類のものが入っている構造のことをネストといいます。

for文の中にfor文が入っている構造のことをfor文のネストといいます。

外側の繰り返しのカウンタ変数は「i」で、0から2まで回し、内側の繰り返しのカウンタ変数は「j」で、0から2まで回すという例で考えてみます。

外側のループの1周目の時に、内側のループが0から2までまわります。内側のループがまわりきったら、外側のループが2周目に入ります。

コードを書いてみましょう。

```
for i in range(3):
    for j in range(3):
        print(i, j, sep="-")
```

iとjの変化がわかるように、「i、ハイフン、j」を表示させてみましょう。

printの第一引数、第二引数をそれぞれi、jにして、sep引数をハイフンにすることで、i、ハイフン、jと表示できます。

iが0周目のときに、jが0から2までまわり、次にiが1周目の動きになる予想通りの結果となりました。

実行結果：

```
0-0
0-1
0-2
1-0
1-1
1-2
2-0
2-1
2-2
```

for文でリスト内を参照

```
arr = [2, 4, 6, 8, 10]
for i in arr:
    print(i)
```

最後に変数を使ってリストの中身を表示させてみましょう。

arrというリスト変数に「2」から「10」までの偶数「2,4,6,8,10」を代入。

for文のinの後にリストを書くことで、リストの中身が変数に一つずつ格納されます。

2から10までの偶数が表示されるはずです。

実行してみましょう。

実行結果：

```
2
4
6
8
10
```

2,4,6,8,10が表示されました。

変数を使って、たし上げていくこともできます。

```
arr = [2, 4, 6, 8, 10]
sum = 0

for i in arr:
    sum += i
print(sum)
```

まず、sumという変数を定義します。

そして、演算子のレッスンで説明した、リストの値を複合代入演算子を使って足し上げていきます。

表示させてみましょう。

実行結果：

```
30
```

足し算ができています。