第 2 回 Python 基礎文法ゼミ

ゼミ担当者　：　米田 有佑，戸松 祐太
開催日　：　2009 年 5 月 7 日

ゼミ内容: 本ゼミでは，Python の基本構文と例外処理の書き方について学習する．

1　条件分岐とループ

Java や C 言語と同じように，Python でも条件分岐やループがある．以下，Python での条件分岐やループ処理の文法について解説する．

1.1　if 分

決められた条件によって処理を振り分けたい場合に if 文を使う．if 文は，Fig. 1 に示すように if に続けて判断条件を記述する．条件の直後にはコロン（:）を記入する．条件によって実行したい部分はブロックとしてまとめて記述する．Python では，インデントを使ってブロックを表現する．

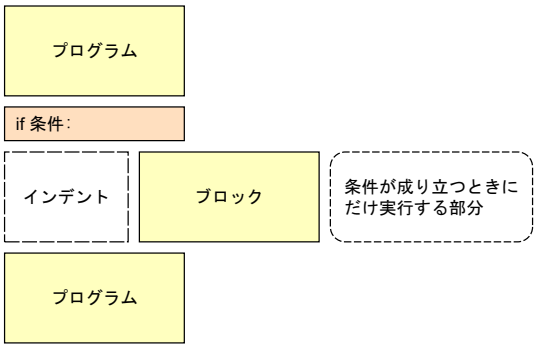

Fig. 1　if 文の記述方法

Python には，switch のような多分岐を行う専用の機能がないため，Fig. 2 のように else や elif 文を使って複数の条件を記述するなどする．最初の条件が成り立たなかった時，さらに別の条件を調べたい場合に elif 文を用いる．これは，Java や C 言語の else if 文に相当する．また，条件が成り立たなかったときに実行したい命令には else 文を用いる．

1.2　for 文

同じ処理を，決められた回数実行したいときに for 文を使う．for 文は，Fig. 3 に示すように for に繰り返し変数を添えて記述する．繰り返し変数に続けて in というキーワードを置き，さらにシーケンスを添える．if 文と同様に，右側にコロンを置くことで for 文が終わったことを示す．

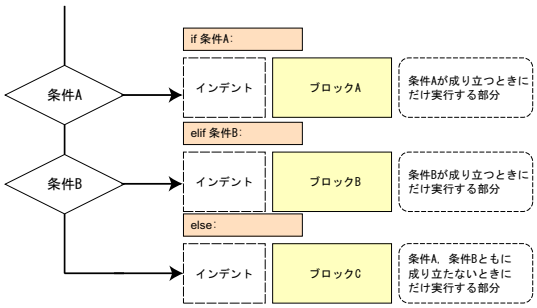

Fig. 2　else，elif 文の記述方法

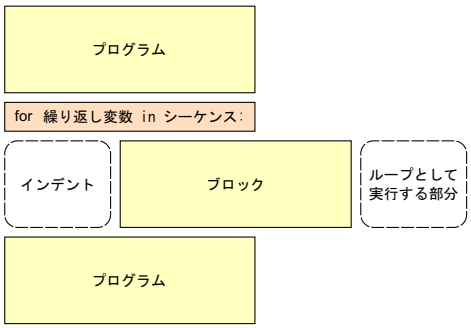

Fig. 3　for 文の記述方法

Fig. 4 に示すように，python の for 文は，シーケンスの要素を一つずつ取り出し，順番に処理する．ループの処理では，シーケンスの要素を順番に，繰り返し変数に代入していく．ループブロックは，シーケンスの要素の数だけ繰り返し実行される．Java や C 言語の様に，繰り返し数を代入するループカウンタは設けないので注意が必要である．

プログラムでは，回数を指定してループを実行するという処理をよく行う．繰り返しを行う回数を指定してループを実行する場合には，組み込み関数 range() を使うと便利である．range() を使うと，ループに利用する数値のリストを簡単に作成することができる．

1.3　break 文と continue 文

ループを実行する際，ある条件を満たしたらループを終了したい場合がある．そのような場合には break 文を利用する．break を実行すると，ループのブロックにあ

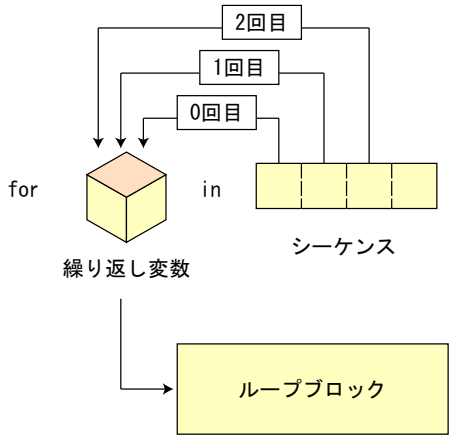

Fig. 4　for 文の処理の流れ

るその後のコードを実行せず，ループのブロックの直後からコードを実行する．
continue 文は，ループブロックの先頭まで処理を戻し，ループ処理を続ける．break と同様，continue 以降のブロックにあるコードは実行されないが，ループの処理を先頭から続けて行う点が異なる．通常，break 文も continue 文も if 文で条件分岐を行ったときに利用する．

1.4　while 文

プログラムでは，状態監視しながら，一定の条件を満たしている間繰り返しを行うという処理を実行する場合，for 文は適していない．そのような処理を実行したい場合には，while 文を使うと便利である．while の後には条件を記述する．条件が True（真）と判断される間，繰り返しを行う．条件の右には，if 文や for 文と同様にコロンを置く．

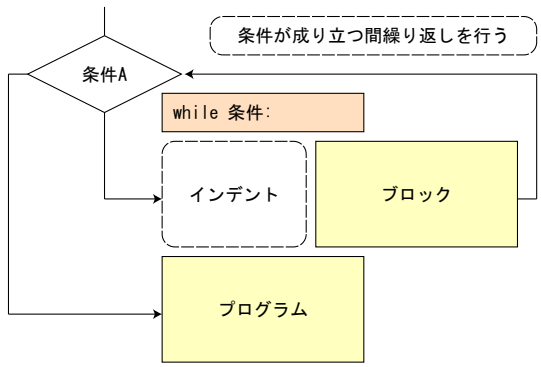

Fig. 5　while 文の記述方法

1.5　比較演算子と論理演算

比較を行うために記述する==の用な記号は比較演算子と呼ばれている．比較演算子を使って比較を行うと「True」または「False」という文字列のような値が返ってくる．比較を行っても，変数は変化しないことに注意が必要である．以下に，比較演算子の一覧を示す．

- $x == y$

  x と y が等しい場合に True となる．代入（=）とは異なるので注意が必要となる．

- $x! = y$，$x <> y$

  x と y が等しくない場合に True となる．感嘆符（!）の位置はイコールの左側となる．「><」という演算子はないので注意が必要である．

- $x > y$，$x < y$

  x と y の大きさを比較する．「>」は左側が大きい場合に True となる．「<」は右側が大きい場合に True となる．x と y が等しい場合は False になる．「$x < y < z$」のように，複数の演算子を組み合わせて書くこともできる．

- $x >= y$，$x <= y$

  x と y が等しい場合を含み，大きさを比較する．不等号の位置はイコールの左になる．「$x <= y <= z$」のように，複数の演算子を組み合わせて書くこともできる．

- $x \quad in \quad y$

  シーケンスに含まれる要素を比較する．x という要素が y というシーケンスに含まれる場合に True となる．

- *not*

  True と False を逆にする．「not x == y」のように比較式の前に記述する．in の場合は「x not in y」と記述する．

「18 歳以上 35 歳未満」というように，複数の比較結果を合わせた条件を使いプログラムを制御したい場合，and や or で複数の条件をつなぐことが可能である．「A が成り立ち，かつ B が成り立つ」という条件は「A and B」というように記述します．「A が成り立つか，または B が成り立つ」という「A or B」というように記述する．

2　関数

プログラムでは，よく使う処理を実行するために関数を利用する．関数には，処理を実行するための命令が書き込まれていて，呼び出すことで処理を実行する．また，自分で作ったコードのうちよく使う処理をまとめて関数を定義して利用可能である．

2.1 関数の利用

- 関数呼び出しは，関数名 (引数) の形で呼び出す．
- 関数の引数において，与える引数の数，順番，型は関数ごとに決まっている．そのため，組み込み関数を使用する場合は，関数ごとのルールをよく把握しておく必要がある．
- 関数を呼び出して，処理した結果が返ってくることがある．この結果の値を戻り値と呼ぶ．戻り値は，数値や文字列，リストといった Python のオブジェクトである．そのため，関数の戻り値は，文字列，変数などと同じように扱える．

2.2 関数の定義

Python では，新しい関数を作り，定義することが可能である．Fig. 6 に関数の定義を示す．

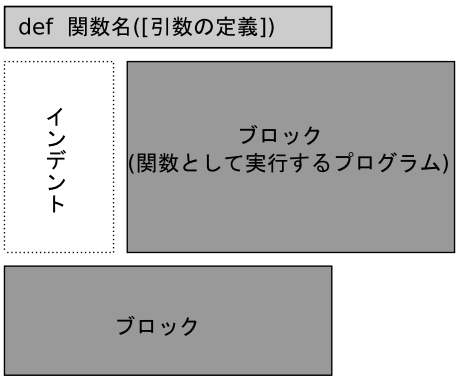

Fig. 6　関数の定義

2.2.1 関数の書き方

Python では，インデントによって一つのブロックを表す．そのため，インデントが非常に重要である．以下に簡単な関数を示す．

test.py

```
def test(h, w):
    print "Hello! " * h
    print "World! " * w

test(3, 5)
```

実行結果

```
Hello! Hello! Hello!
World! World! World! World! World!
```

また，return 文を用いて，関数で処理した結果を戻り値として関数の外に返すことも可能である．

test2.py

```
def test2(times, sportname="tennis "):
    return sportname * times

sport = test2(4)
print sport
```

実行結果

```
tennis tennis tennis tennis
```

2.2.2 関数とローカル変数

関数内で定義された変数は，ローカル変数となる．ローカル変数とは「決まった場所でのみ利用可能な変数」である．そのため，関数内で定義した変数は関数内でしか使用不可能である．

3 例外処理

例外処理は，プログラムの処理中に起こったエラーや状態の変化を伝えるために利用する仕組みである．プログラムで起こるエラーには以下の 2 つがある．

- プログラムを実行する前にわかるエラー (SyntaxError(構文エラー))

  例：IndentatuinErrior
- プログラムの実行中にわかるエラー (StandardError)

  例：NameError，TypeError，IndexError

3.1 例外処理の定義

例外処理では，try〜except ブロックを用いることで例外を捕まえることが可能である．Fig. 7 に例外処理の定義を示す．

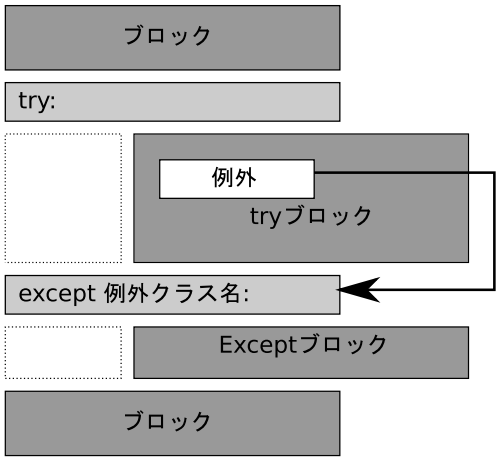

Fig. 7　例外処理

Fig. 7 に示したように，例外が発生すると，try ブロック以降のプログラム実行がスキップされる．

3.1.1 例外処理の書き方

以下に例外処理の例を示す.

test3.py

```
test = range(10)

try:
    print test[11]
except IndexError:
    print "要素数を超えています"
```

実行結果

```
要素数を超えています
```

test3.py では，シーケンスの要素数を超えて出力しようとするために，エラーを受け取って Except ブロックを実行する.

3.1.2 例外処理の書式

try 文に続く except には，特定の例外オブジェクトを捕まえるために例外のクラスを記述する．次に例外を処理するための書式を示す.

- except:

  全ての例外を受け取り，例外発生時の処理を行う

- except 例外クラス名:

  クラスを指定して，特定の例外だけを受け取る

- except 例外クラス名, 変数名:

  例外クラスと，例外オブジェクトを受け取る変数名を指定する

- else:

  例外が発生しなかった場合の処理を行いたいときに利用する

- finally:

  例外の発生に関わらず，必ず実行する場合に利用する

これらの書式を用いて様々なプログラムに対応させる.