スクリプト言語 Python の初心者向け
教材の作成

目　次

1　画面への出力

(1) 画面に出力する

画面に文字などを表示することを、プログラミングの世界では画面に出力すると呼んでいます。画面に文字などを出力したいときはどうするのかを学びましょう。まず下のように入力します。

sample2-1

```
print "ようこそ Python へ!"
print "Python をはじめましょう!"
```

sample2-1.py 実行画面

```
ようこそ Python へ!
Python をはじめましょう!
```

ここでは、画面出力する方法を学びます。画面出力する方法は print 文を使います。表示したい文章を print 文の後に記述します。その際表示したい文字の両側は” ”で囲むことが必要です。

構文

```
print " "(" "の中身は出力したい文字や値を記述します。)
```

(2) コメントを記述する

Python ではコメントを記述する場合、#コメント文というようにする。

まず、一番はじめに入力した、#記号でかこまれた行をみてください。

実は、Python のコンパイラは、#からの文字を無視して処理するというきまりになっています。

そのため、この部分にプログラムの処理とは直接関係のない自分の言

葉を「メモ」として入力しておくことができます。これを「コメント」といいます。通常は、そのコードの内容をメモしておくと便利でしょう。

Python だけでなく、多くのプログラム言語は、人間にとっては決して読みやすい言語ではありません。このようなコメントを書いておくことによって、読みやすいコードを作成していことができるのです。

(3) 改行のしくみを知る

, という記号は「改行」を行わないということです。, をつけると、その部分で改行を行わないということができるのです。ためしに、次のコードを入力してから適当な名前で保存してください。それから実行してみてください。

sample2-2.py

```
print "ようこそ Python へ!",
print "Python をはじめましょう!"
```

sample2-2.py 実行画面

```
ようこそ Python へ!Python をはじめましょう!
```

このように, を入力するとそこで改行されません。Python では、普通 print 文を入力すると自動的に改行されます。Python で改行したくない場合はカンマを入力しなければいけません。

重要

普通 Python **では、**print **文を使うと改行される。**
そこで、改行したくない場合は print **文の後にカンマを付ける。**

```
print "",
```

2　文字と数値

(1) 変換仕様のしくみを知る

Python では、文字列のほかにも、さまざまな文字や数値を使います。そこでこの節では、これまでのコードを応用して、Python の基本となる文字・数値・文字列の書き方を学ぶことにしましょう。

まず最初に、次のようなコードを入力して下さい。

sample2-3.py

```
print "%s は文字です。"  % ("A")
print "%d は整数です。"  % (123)
print "%f は小数です。"  % (10.5)
```

sample2-3.py 実行画面

```
A は文字です。
123 は整数です。
10.500000 は小数です。
```

%s は文字 (文字列) を、%d は整数を、%f は小数を表示することができます。変換仕様を使うと、文字列中に文字や数値を埋め込んで出力することができます。ちなみに変換仕様とは文字や数値を文字列中に埋め込んで表示するものです。

(2) 文字

Python では文字列、文字はどちらも””でくくります。

3　変数

変数は、ローマ字のアルファベットか下線で始まり、あとはアルファベット・数字・下線のみで名前を作り、それに値を代入することで使用を宣言する。上の例では、関数 raw_input() で入力された値を answer や number という変数に代入することで、新たな変数を宣言している。また in_tax には、計算の結果を代入して新たな変数を宣言している。

変数は、後で値の代わりに使う事ができるだけではなく、新たな値を代入する事もできる。値を変える事ができるので、「変数」と言う。

(1) 変数の使い方

これまで、print 文で表示していたのは、文字列や数値そのものであった。このように値そのものを「リテラル」と言う。しかし、これではプログラムの再利用は難しい。そこで、「変数」を用意し、そこに値を入れ、実際の print 文には変数を使うようにすると応用の範囲が広がる。変数はプログラムの最初の方で「宣言」しておく。

(2) 変数の例

```
name = "山中麻衣子"
university = "大谷大学"
old = 22
print "私は", name, "です。"
print "大学は", university, "です。"
print "年齢は", old, "歳です。"
```

4　識別式

(1) 代入と変数

(i) 代入の基本形式

変数	=	右辺

＝の左辺には必ず変数。＝の右辺には、文字列・数値・リストなどの直接の値か、変数か、計算式。変数の場合には、その変数に入っている値、計算式の場合にはその計算の結果の値が代入される。

変数は、名前と値からなっている。頭の中に「引き出し」のようなものを想像しよう。個々の「引き出し」にはラベルが貼ってある。その「引き出し」の中に物を入れる。ラベルが「変数の名前」であり、中に入れる物が「変数の値」である。つまり、

変数の名前	=	値

は、ラベルの貼った引き出しに、物を入れる。

という操作をしていることになる。また、「引き出し」に物をしまう、というのは、カードに値を書いて、そのカードの引き出しに入れることである。

変数に代入される値、つまり頭の中の引き出しにしまうカードに書かれる値には、これまで学習したものでは、

1. 数値。形式:1, 3, 5, 8.3, 100, 32.0 などの数。
2. 文字列。形式:`"Otani University"`, `"`**数を入力して下さい。**`"`など、一重または二重の引用符で囲まれたテキスト。
3. リスト。形式：`[2, 4, 6, 8]`, `["abc", "def", "123", "456"]`

など、項目をカンマで区切って、大括弧で囲ったもの。これについては、後で述べる。

がある。また、

1. 変数。代入の左辺以外では、変数の値が使われる。変数の値は数値、文字列リストのいずれかなので代入の右辺に使える。
2. 計算式。計算した結果が、数値、文字列、リストになるので、これも代入の右辺に使える。

「引き出し」は最初に使うときに、新しい引き出しにラベルを貼って中に物をしまうことで、その引き出しを使うことを「宣言」する必要がある。

変数が代入の左辺以外で使われるときには、その値が使われる。これは、そのラベルの付いた引き出しからカードを取り出して、そこに書かれた値を計算などに使うということである。

5　型

Python に型はありますが、宣言する必要はありません。最初に代入したときに、それが整数なのか、実数なのか、文字列なのか、あるいはリストなのか、などの型が決まります。型によってできることとできないことの違いがあります。

6　変数の宣言

(1) 変数を宣言する

Python の場合は事前に変数の宣言をする必要がありません。唯単に値を入れればその時点で変数領域が確保されます。

7　変数の利用

(1) 変数に値を代入する

変数に特定の値を記憶させる事ができます。このことを、「値を代入する」と言います。代入をするには、次のように=という記号を使って記述します。

例

```
a = 3
```

構文　変数への代入

変数名 = 式

(2) 変数の値を出力する

sample3-1.py

```
a = 3
print "変数 a の値は", a, "です。"
```

sample3-1.py **実行画面**

```
変数 a の値は 3 です。
```

重要

変数には、=を使って値を代入する。

(3) 変数を初期化する

構文

識別式 = 式

(4) 変数の値を変更する

sample3-2.py

```
a = 3
print "変数 a の値は",a ,"です。"


a = 5
print "変数 a の値を変更しました。"
print "変数 a の新しい値は",a ,"です。"
```

sample3-2.py 実行画面

```
変数 a の値は 3 です。
変数 a の値を変更しました。
変数 a の新しい値は 5 です。
```

(5) ほかの変数の値を代入する

sample3-3.py

```
num1 = 3
print "変数 num1 の値は", num1, "です。"
num2 = num1
print "変数 num1 の値を変数 num2 に代入しました。"
print "変数 num2 の値は", num2, "です。"
```

sample3-3.py 実行画面

```
変数 num1 の値は 3 です。
変数 num1 の値を変数 num2 に代入しました。
変数 num2 の値は 3 です。
```

3 行目を見て下さい。

ここでは、=記号の右側に数値ではなく、変数の num1 を記述しています。すると、変数 num2 には、「変数 num1」の値が代入されることになります。画面をみると、確かに、変数 num2 に変数 num1 の値である 3 が格納されていることがわかりますね。このように、変数の値をほかの変数に代入するということもできるのです。

(6) 値の代入についての注意

sample3-4.py

```
num = 3.14
print "変数 num の値は %d です。" % (num)
```

sample3-4.py 実行画面

```
変数 num の値は 3 です。
```

変数 num に小数を代入すると、実行画面では小数点以下が切り捨てられてしまいます。これは、変数である num には、整数値しか格納できないことになっているからです。

(7) 変数の宣言位置についての注意

Python では、変数の宣言をする必要がないので、特に注意はありません。

8　キーボードからの入力

(1) キーボードから入力する

この章の応用として、ユーザーにキーボードからいろいろな文字や数値を入力させて、その値を出力するコードを記述してみましょう。キーボードから入力を受け付けるコードを学ぶと、さらに柔軟なプログラムが作成できるようになります。キーボードから入力するコードは、次のようなスタイルで記述します。

◆構文◆ sample3-5.py **「キーボードから整数を入力する」**

```
answer = raw_input("整数を入力してください。")
print answer, "が入力されました。"
```

sample3-5.py **実行画面**

```
整数を入力してください。
10（キーボードから入力すると…）
10 が入力されました。（入力した数値が出力されます）
```

このプログラムを実行すると、「整数を入力してください。」というメッセージが画面に出力されます。そして、コンピュータはユーザーからの入力を待つ状態になります。

ここで「10」をキーボードから入力し、Enter キーを押してみましょう。すると、画面にも「10 が入力されました。」と出力されるはずです。

何度もプログラムを実行し、いろいろな数値を入力してみてください。このコードを使うと、さまざまな数値を出力することができるはずです。

◇重要◇

質問を出して、それに対する答をユーザーから受け取るには、

raw_input() 関数を使う。raw_input() 関数の返す値を変数に代入することで、ユーザーの返答を保存できる。入力された値を数値として計算に使いたいときは、整数に変換するには int() 関数を、実数に変換するには float() 関数を使う。

(2) 小数を入力する

次に整数値以外の入力をためしてみることにしましょう。まず小数値を入力してみます。次のコードを入力してみてください。

sample3-6.py「小数を入力する」

```
number = raw_input("円の半径を入力してください。")
in_menseki = float(number) * float(number) * 3.14
print "円の面積は", in_menseki, "です。"
```

sample3-6.py 実行画面

```
円の半径を入力してください。
2
円の面積は 12.56 です。
```

このプログラムは、小数値を読み込んで画面に出力しています。

9　レッスンのまとめ

練習

1．次のように画面に出力するコードを記述してください。

```
あなたは何歳ですか？
23
```

あなたは 23 **歳です。**

2. 次のように画面に出力するコードを記述してください。

円周率の値はいくつですか？
3.14
円周率の値は 3.140000 **です。**

3. 次のように画面に出力するコードを記述してください。

アルファベットの最初の文字はなんですか？
a
アルファベットの最初の文字は a **です。**

4. 次のように画面に出力するコードを記述してください。

身長を入力してください。
165.2
体重を入力してください。
52.5
身長は 165.200000 **センチです。**
体重は 52.500000 **キロです。**

5. 次のように画面に出力するコードを記述してください。

身長と体重を入力してください。
165.2
52.5
身長は 165.200000 **センチ**:**体重は** 52.500000 **キロです。**

10　式と演算子

(1) 式のしくみを知る

コンピュータは、いろいろな処理を、「計算」することによって行います。そこで、この章では最初に、式というものについて学ぶことにしましょう。

(2) 式の値を出力する

sample4-1.py「式の値を出力する」

```
sum = 1 + 2
print "1+2 は",sum,"です。"
sum1 = 3 * 4
print "3 × 4 は",sum1,"です。"
```

sample4-1.py 実行画面

```
1+2 は 3 です。
3 × 4 は 12 です。
```

(3) いろいろな演算をする

sample4-2.py「変数の値を使う」

```
num1 = 2
num2 = 3
sum = num1 + num2
print "変数 num1 の値は %d です。" % (num1)
print "変数 num2 の値は %d です。" % (num2)
print "num1+num2 の値は %d です。" % (sum)
```

```
num1 = num1 + 1
print "変数 num1 の値に 1 をたすと %d です。" % (num1)
```

sample4-2.py 実行画面

```
変数 num1 の値は 2 です。
変数 num2 の値は 3 です。
num1+num2 の値は 5 です。
変数 num1 の値に 1 をたすと 3 です。
```

(4) キーボードから入力した値をたし算する

sample4-3.py「たし算プログラム」

```
answer = int(raw_input("整数を入力して下さい。"))
answer2 = int(raw_input("整数を入力して下さい。"))
sum = answer + answer2
print "足し算の結果は", sum, "です。"
```

sample4-3.py 実行画面

```
整数 1 を入力してください。
5
整数 2 を入力してください。
10
たし算の結果は 15 です。
```

11　演算子の種類

(1) いろいろな演算子

Python には、＋ 演算子以外にもたくさんの演算子があります。演算子の種類を次の表に示しておきましょう。

```
+   加算
-   減算
*   乗算
/   除算
%   剰余
() タプル
[] リスト
< <= > >= == <> 比較演算子
```

sample4-4.py「いろいろな演算子を利用する」

```
num1 = 10
num2 = 5

print "num1 と num2 にいろいろな演算子を行います。"
print "num1+num2 は %d です。" % (num1+num2)
print "num1-num2 は %d です。" % (num1-num2)
print "num1*num2 は %d です。" % (num1*num2)
print "num1/num2 は %d です。" % (num1/num2)
```

sample4-4.py 実行画面

```
num1 と num2 にいろいろな演算を行います。
num1+num2 は 15 です。
```

```
num1-num2 は 5 です。
num1*num2 は 50 です。
num1/num2 は 2 です。
```

(2) 代入演算子

次に、代入演算子について学びましょう。代入演算子は、これまで、変数に値を代入する際に使ってきた「=」という記号のことです。この演算子は、通常の=の意味である「等しい」(イコール) という意味ではないことは、すでに説明しました。代入演算子は、左辺の変数に右辺の値を代入するという機能をもつ演算子なのです。代入演算子は=だけでなく、=とほかの演算を組み合わせたバリエーションもあります。次の表をみてください。これらの代入演算子は、ほかの演算と代入を同時に行う複合的な演算子となっています。たとえば、+=演算子をみてみましょう。

```
a += b
```

+=演算子は、変数 a の値に変数 b の値を足し算し、その値を変数を a に代入する。という演算を行います。+ 演算子と=演算子の機能をあわせたような機能をもっているのです。

このように、四則演算などの演算子 (●としておきます) と組み合わせた複合的な代入演算子を使った文である

```
a ●= b
```

は、通常の代入演算子である=を使って、

```
a = a ● b
```

と書き表すことができます。

つまり、次の 2 つの文はどちらも、変数 a の値と b の値をたして変数 a に代入するという処理をあらわしています。

a += b

a = a + b

なお、複合的な演算子では、

a + = b

などと、+ と=の間にスペースをあけて記述してはいけません。ためしに、+=演算子を使ってコードを記述してみましょう。

sample4-5.py**「複合的な代入演算子を使う」**

```
sum = 0;
num = 0;

sum = raw_input("1 番目の整数を入力してください")
sum1 = raw_input("2 番目の整数を入力してください")
sum2 = raw_input("3 番目の整数を入力してください")

in_tax = int(sum) + int(sum1) + int(sum2)

print ("3 つの数の合計は", in_tax,"です。")
```

sample4-5.py **実行画面**

```
1 番目の整数を入力してください。
1(任意の数字)
2 番目の整数を入力してください。
3(任意の数字)
3 番目の整数を入力してください。
```

```
4(任意の数字)
3 つの数の合計は 8 です。
```

このサンプルでは、raw_input というものを使っています。
質問を出して、それに対する答をユーザーから受け取るには、raw_input() 関数
を使います。

raw_input() 関数の返す値を変数に代入することで、ユーザーの返答を保存する
事ができます。

入力された値を数値として計算に使いたいときは、整数に変換するのには int()
関数を、実数に変換するには float() 関数を使います。

12　演算子の優先順位

演算子には優先順位がある。優先順位を変更するには () を使う。

(1) 同じ優先順位の演算子を使う

左から優先的に評価される場合を左結合という。右から優先的に評価される場合を右結合という。

13　型変換

(1) 大きなサイズの型に代入する

sample4-6.py「大きなサイズの型に代入する」

```
inum = 160;
print "身長は %d センチです。" % (inum)
print "double 型の変数に代入します。"
dnum = inum;
print "身長は %f センチです。" % (dnum)
```

sample4-6.py 実行画面

```
身長は 160 センチです。
double 型の変数に代入します。
身長は 160.000000 センチです。
```

このコードでは、int 型の変数の値を double 型の変数に代入しています。すると、int 型の値は double 型に変換されて代入されることになります。第 3 章で、型のサイズについて学んだことを思い出してください。一般に Python では、大きなサイズの型の変数に小さなサイズの型の値を代入することができます。代入するときに型が変換されるのです。型が変換されることを、型変換といいます。

(2) 小さなサイズの型に代入する

sample4-7.py「小さなサイズの型に代入する」

```
dnum = 160.5;


print "身長は %f センチです。" % (dnum)
print "int 型の変数に代入します。"
```

```
inum = dnum
print "身長は %d センチです。" % (inum)
```

sample4-7.py **実行画面**

```
身長は 160.500000 センチです。
int 型の変数に代入します。
身長は 160 センチです。
```

今度はさきほどとは逆に、double 型の変数の値を int 型の変数に代入しています。double 型の値が int 型に変換されて代入されるわけです。ただし、小さなサイズの型に変換した場合には、その型であらわせない部分は、上のように切り捨てられることになります。たとえば 160.5 という値は、int 型の変数にそのまま格納することはできません。小数点以下が切り捨てられて 160 という整数が格納されることになります。小さなサイズの型の変数に代入する場合には、このように値の一部が失われてしまう場合があるので注意してください。

(3) 型変換の関数

なお、型が変換されるときには、型を変換することをコードの中に明示的に書いておくということができるようになっています。次のキャスト演算子という演算子をみてください。sample4-7.py を書き換えてみると、

sample4-7.py**「小さなサイズの型に代入する」**

```
dnum = 160.5;


print "身長は %f センチです。" % (dnum)
```

```
print "int 型の変数に代入します。"
inum = int(dnum)
print "身長は %d センチです。" % (inum)
```

sample4-7.py 実行画面

```
身長は 160.500000 センチです。
int 型の変数に代入します。
身長は 160 センチです。
```

実行結果はさきほどと同じようになります。ただし、今度は小さなサイズの型に変換することをはっきりとコード中に書きあらわすことができました。キャスト演算子を使えば、型を変換することを明示的に記述することができるのです。

(4) 異なる型どうしで演算する

型変換は足し算、引き算、掛け算、割り算といった四則演算などを行う場合にも行われる場合があります。次の例をみてください。

sample4-8.py「異なる型の演算をする」

```
d = 2;
pi = 3.14;
print "直径が %d センチの円の" % (d)
print "円周は %f センチです。" % (d*pi)
```

sample4-8.py 実行画面

```
直径が 2 センチの円の
円周は 6.280000 センチです。
```

ここでは、int 型の d の値と double 型の pi の値の掛け算をしています。Python では一般的に、演算子に異なる型のオペランドを記述した場合、

一方のオペランドを大きなサイズのほうに型変更してから演算を行うといきまりになっています。つまり、ここでは int 型の d の値の「2」が double 型の数値 (2.0) に変換されてから、掛け算が行われるのです。得られる式の値も double 型の値となります。

(5) 同じ型どうしで演算する

それでは、同じ型どうしで演算を行う場合はどうなるのでしょうか？この場合は同じ型どうしで演算が行われ、結果もその型の値となります。しかし、この演算が予想外の結果となってしまうコードもあります。

次の例をみてください。

sample4-9.py「同じ型の演算をする」

```
num1 = 5
num2 = 4
div = num1 / num2
print "5/4 は %f です。" % (div)
```

sample4-9.py 実行画面

```
5/4 は 1.000000 です。
```

このままだと 5/4 をしたつもりなのに、思った通りの答になっていません。なので、下記のように書き換えると、

sample4-9.py

```
num1 = 5
num2 = 4
div = float(num1) / float(num2)
print "5/4 は %f です。" % (div)
```

sample4-9.py **実行画面**

```
5/4 は、1.250000 です。
```

今度は思ったとおりに出力されました。このようにコードを書き換えることによって、float 型の演算が行われるようになるのです。結果も float 型となり、「1.25…」という答えを出力することができます。演算をする場合は、オペランドの型に十分注意しておく必要があります。

14　レッスンのまとめ

練習

1. 次の計算結果を出力するコードを記述してください。

0-4

3.14 × 2

5 ÷ 3

30 ÷ 7 あまりの数

(7+32) ÷ 5

2. 次のように整数を入力して、正方形の面積を出力するコードを記述してください。

```
正方形の辺の長さを入力してください。
3
正方形の面積は 9 です。
```

3. 次のように高さと底辺を入力して、三角形の面積を出力するコードを記述して　ください。

（ヒント:三角形の面積=（高さ×底辺) ÷ 2)

```
三角形の高さを入力してください。
3
三角形の底辺を入力してください。
5
三角形の面積は 7.500000 です。
```

4. 次のように正負を反転して出力するコードを記述してください。

```
整数を入力してください。
3
正負を反転すると-3 です。
```

5. キーボードから 5 科目のテストの点数を入力して、合計値と平均値を次のように出力するコードを記述してください。

```
科目 1 の点数を入力してください。
52
科目 2 の点数を入力してください。
68
科目 3 の点数を入力してください。
75
科目 4 の点数を入力してください。
83
科目 5 の点数を入力してください。
36
5 科目の合計点は 314 点です。
```

```
5 科目の平均点は 62.800000 点です。
```

15　関係演算子と条件

(1) 条件のしくみを知る

条件とは普段私たちが日常で使うことがあります。

例えば、もし明日晴れだったら遠足へ出かける…もし明日雨だったら中止にする。というようなものが条件です。もし～ならば～ということが条件というものです。Python ではこういった条件を使用してプログラムを書くことができます。その方法を次の節で学びましょう。

(2) 条件を記述する

それでは、実際に条件を Python の式であらわしてみましょう。私たちは、1 より 3 が大きいことを、

$3 > 1$

という不等式であらわすことがあります。たしかに、3 は 1 より大きい数値なので、この不等式は「正しい」といえます。一方、この不等式はどうでしょうか。

$3 < 1$

この式は「正しくない」ということができます。Python でも、＞のような記号を使うことができ、上の式は「真」、下の式は「偽」であると評価されます。つまり、3 ＞ 1 や 3 ¡ 1 という式は Python の条件ということができるのです。条件をつくるために使う ＞ 記号などは、関係演算子と呼ばれています。

(3) 関係演算子を使う

それでは、関係演算子を使って、いくつか条件を記述してみましょう。

```
5 > 3    （この条件の評価は真です。）

5 < 3    （この条件の評価は偽です。）

a == 6    （この条件の評価は変数 a の値によって異なります。）

a != 6    （この条件の評価は変数 a の値によって異なります。）
```

「5 ＞ 3」という条件は、3 よりも 5 が大きいので、式の値は真になることがわかりますね。また、「5 ＜ 3」という条件は、3 よりも 5 が大きいので、式の値は偽になります。条件の中には変数を使うこともできます。たとえば、「a == 6」という条件は、変数 a の値が 6 であれば真になります。一方、変数 a の内容が 3 や 10 であった場合は偽になります。このように、そのときの変数の値によって条件があらわす値が異なるわけです。同様に、「a!=6」は、a が 6 でない値のときに真となる条件となっています。なお、!=や==は 2 文字で 1 つの演算子ですから、!と=の間に空白を入力したりしてはいけません。ところで、=演算子が代入演算子と呼ばれていたことを思い出してください。(第 4 章)。かたちは似ていますが、==は異なる種類の演算子（関係演算子) です。この 2 つの演算子は、実際にコードを書く際にたいへん間違いやすい演算子となっています。しかも、a=0 を a==0 と間違って書いても、コンパイラにはエラーを発見することができません。よく注意して入力する

ようにしてください。

重要

=（代入演算子）と==（関係演算子）を間違えないこと。

16　if 文

(1) if 文のしくみを知る

条件のしくみを実際に Python で処理するには、if 文を使います。if 文は条件が真の場合に指定した文を処理するという構文です。では、実際にコードを入力してみましょう。

sample5-1.py「if 文を使う」

```
res = int(raw_input("整数を入力してください。"))


if res == 1:
    print "1 が入力されました。"
print "処理を終了します。"
```

sample5-1.py 実行画面

```
整数を入力してください。
1
1 が入力されました。
処理を終了します。
```

sample5-1 では、条件 res == 1 が真であればのそのまま処理されます。したがって、「1 が入力されました。」「処理を終了します。」と出力されます。偽の場合は処理されません。したがって、「処理を終了します。」

とだけ出力されます。実際にどうなるのか、次の実行画面を見てください。

sample5-1.py **実行画面その** 2

```
整数を入力してください。
10
処理を終了します。
```

今度は、res == 1 という条件が偽になるため、そのまま処理されません。したがって、実行したときの画面は上のようになります。このように、if 文を使うと、条件が真のときにだけ、処理を行うことができます。

(2) if 文で複数の文を処理する

if 文では複数の文を処理することができます。やり方は次の構文の通りである。

構文

```
if（条件）
    文 1
    文 2
```

これで、文は一行ずつ処理されます。では、実際に次のコードを入力してください。

sample5-2.py「複数の文を処理する if 文を使う」

```
res = int(raw_input("整数を入力してください。"))


if res == 1:
    print "1 が入力されました。"
    print "1 を選択しました。"
```

```
print "処理を終了します。"
```

sample5-2.py **実行画面**

```
整数を入力してください。
1
1 が入力されました。
1 を選択しました。
処理を終了します。
```

ユーザーが 1 を入力した場合は条件が真となるので、ブロック内の処理が順に行われ、2 行分の文字列が出力されます。もし 1 以外の数値を入力した場合は、ブロック内の処理は行われず、次のようになります。

sample5-2.py **実行画面その** 2

```
整数を入力してください。
10
処理を終了します。
```

さきほどの結果と比べると、行われていない処理があるのがわかりますね。

17　if〜else 文

(1) if〜else 文のしくみを知る

今までは条件を使用して、ユーザーが入力したものが真であれば、処理が行われていました。

ここでは、ユーザーが入力したものが偽の場合でも、処理できる方法を学びましょう。構文は次の通りです。

構文

```
if（条件）
    文 1
else
    文 2
```

この構文では、条件が真の場合に文 1 を処理し、偽の場合に文 2 を処理します。それでは、if～else 文を試してみるために、次のコードを入力してみましょう。

sample5-3.py「if～else 文を使う」

```
res = int(raw_input("整数を入力してください。"))


if res == 1:
    print "1 が入力されました。"
else:
    print "1 以外が入力されました。"
```

sample5-3.py 実行画面その 1

```
整数を入力してください。
1
1 が入力されました。
```

sample5-3.py 実行画面その 2

```
整数を入力してください。
10
1 以外が入力されました。
```

この通り条件が真の場合は if の中が処理されます。また、偽の場合は else の中が処理されます。

18　if〜elif〜else

(1) if〜elif〜else のしくみを知る

if 文では、2 つ以上の条件を判断させて処理するバリエーションをつくることもできます。これが if〜elif〜else です。この構文を使えば、2 つ以上の条件に応じた処理ができるようになります。

構文

```
if（条件）
    文 1
    文 2
elif if（条件 2）
    文 3
    文 4
elif if（条件 3）
    文 5
else
    文 6
```

では、実際にコードを入力してみましょう。次のコードを入力してください。

sample5-4.py「if〜elif〜else を使う」

```
\begin{Verbatim}
res = int(raw_input("整数を入力してください。"))


if res == 1:
    print "1 が入力されました。"…
elif res == 2:
```

```
    print "2 が入力されました。"…
else:
    print "1 か 2 を入力してください。"…
```

sample5-4.py 実行画面その 1

```
整数を入力してください。
1
1 が入力されました。
```

sample5-4.py 実行画面その 2

```
整数を入力してください。
2
2 が入力されました。
```

sample5-4.py 実行画面その 3

```
整数を入力してください。
3
1 か 2 を入力してください。
```

1 を入力した場合、最初の条件は真になるので、　が処理され、ほかの部分は処理されません。

2 を入力した場合、最初の条件が偽となるため、次の条件を判断します。2 番目の条件は真となるので、今度は　が処理されます。

1 や 2 以外を入力した場合（12 番目の条件とも偽の場合)、必ず　が処理されます。

このように、if elif では、いくつもの条件を判断していって、複雑な処理を行うことができるのです。

19　switch 文

(1) switch 文のしくみを知る

Python には switch 文はありません。単純に if…elif…elif…というように elif を重ねていきます。switch 文がなくても、問題は if…elif…で解いて下さい。

20　論理演算子

(1) 論理演算子のしくみを知る

これまで、いろいろな条件を使った条件判断文を記述してきました。

このような文の中で、もっと複雑な条件を書ければ便利な場合があります。

普段私たちが使ってる言葉で例をあげてみます。

成績が「5」であり、かつ、お金があったら→旅行へ行く。

こういうことは Python でも使うことができます。

その場合論理演算子という演算子を使います。

論理演算子とは条件をさらに評価して、真または偽の値を得るという機能をもっているものです。

それでは、論理演算子を使ったコードを、具体的にみてみましょう。

```
5 > 3 && 3==4…
a==6 or a>y=12…
!(a==6)…
```

この条件は偽です。　この条件は変数 a の値が 6 または 12 以上のとき真になります。　この条件は変数 a の値が 6 でないときに真とな

ります。

(2) 複雑な条件判断処理をする

さて、いままで学んだ if 文などの中で論理演算子を使えば、より複雑な条件を判断する処理ができるようになります。

さっそく、論理演算子を使ってみましょう。次のコードを入力してください。

sample5-5.py「論理演算子を使って条件を記述する」

```
res = raw_input("あなたは男性ですか？ Y または N を入力してください。")


if res == "Y" or "y":
    print "あなたは男性ですね。"
elif res == "N" or "n":
    print "あなたは女性ですね。"
else:
    print "Y か N を入力してください。"
```

sample5-5.py 実行画面その 1

```
あなたは男性ですか？ Y または N を入力してください。
Y
あなたは男性ですね。
```

sample5-5.py 実行画面その 2

```
あなたは男性ですか？ Y または N を入力してください。
n
あなたは女性ですね。
```

sample5-5.py **実行画面その** 3

```
あなたは男性ですか？ Y または N を入力してください。
m
Y か N を入力してください。
```

この例題では、キーボードから入力した文字にしたがって処理を行っています。文字には、Y と y のように大文字と小文字がありますが、ここでは、大文字・小文字の区別なく処理したいと考えました。そこで、この例題では、if 文の条件に、論理演算子 or を使ってみたのです。

or を使って条件を記述すれば、Y または y を入力したときに、同じ処理を行うことができます。

練習

1. キーボードから整数値を入力させ、場合に応じて次のようなメッセージを出力するコードを記述してください。

値が偶数だった場合―「○は偶数です。」

値が奇数だった場合―「○は奇数です。」

(ただし○は入力した整数)

```
整数を入力してください。
1
1 は奇数です。
```

2. キーボードから 2 つの整数値を入力させ、場合に応じて次のようなメッセージを出力するコードを記述してください。

値が同じ場合―「2 つの数は同じ値です。」

それ以外の場合―「○より×のほうが大きい値です。」(ただし○、×は入力した整数。○ < ×)

```
2 つの整数を入力してください。
1
3
1 より 3 のほうが大きい値です。
```

3. キーボードから整数値を入力させ、場合に応じて次のようなメッセージを出力するコードを記述してください。

値が 0～10 の場合―「正解です。」それ以外の場合―「間違いです。」

```
0 から 10 までの整数を入力してください。
1
正解です。
```

4. キーボードから文字を入力させ、場合に応じて次のようなメッセージを出力するコードを記述してください。

A、B、C の場合―「正解です。」それ以外の場合―「間違いです。」

```
A～C までの文字を入力してください。
C
正解です。
```

5. キーボードから 1 から 5 までの 5 段階の成績を入力させ、成績に応じて次のようなメッセージを出力するコードを記述してください。

```
成績          メッセージ
1       成績は 1 です。もっとがんばりましょう。
2       成績は 2 です。もう少しがんばりましょう。
```

```
3       成績は 3 です。さらに上をめざしましょう。
4       成績は 4 です。たいへんよくできました。
5       成績は 5 です。たいへん優秀です。
```

```
成績を入力してください。
3
成績は 3 です。
さらに上をめざしましょう。
```