

## 変数

### 1.1 変数とは?

変数とは数値や文字などを入れることができる入れ物。

### 1.2 変数の名前

変数名 (変数の名前) にはルールがある。

1. 変数に使える文字は次の文字だけ

- アルファベット (a ~ z, A ~ Z)
- 数字 (0 ~ 9)
- アンダースコア (\_)

変数名の先頭 (1 文字目) にアンダーバー (\_) を使用できるが <sup>き</sup> 避けた方が良い

2. 数字は変数名の先頭には使えない

3. 予約語 (Python ですでに使われている名前) は変数名にはできない

予約語の例

if, elif, else, False, True, try, for, continue, break, and, or, not など

4. アルファベットの大文字小文字は別の変数として区別される

変数名として使える名前の例

a, abc, ABC, a1, xxx, abc\_xyz, \_a

変数名として使えない名前の例

1a, 2b, \$a, 1\_, if, else, and, not

### 1.3 代入

コード 1.1 代入

```
1 a = 5
2 b = "ABC"
3 c
```

変数には数値や文字を入れることができる。このことを代入という。

= があるが数学とは<sup>ちが</sup>い、左の例では右辺の値 (5) を左辺の変数 (a) に代入することとなる。また、代入をしていない変数は中身がなく、この状態のことを未定義という。(左の例では、3 行目の変数 c は未定義)

## 数式

### 2.1 使える記号

記号	意味	例	結果
+	加算 (足し算)	$5 + 8$	13
-	減算 (引き算)	$90 - 10$	80
*	乗算 (掛け算)	$4 * 7$	28
/	除算 (割り算)	$7 / 2$	3.5
//	切り捨て除算	$7 // 2$	3
%	剰余 (あまり)	$7 \% 3$	1
**	累乗	$3 ** 4$	81
()	括弧 (カッコ)	$(2 + 4) * 4$	24

### 2.2 優先順位

優先順位	記号	意味
1	()	括弧 (カッコ)
2	**	累乗
3	*	乗算 (掛け算)
	/	除算 (割り算)
	//	切り捨て除算
	%	剰余 (あまり)
4	+	加算 (足し算)
	-	減算 (引き算)

優先順位が同じ場合は左から順に計算される。また、演算に使用できる記号はほかにもたくさんある。

## 型 (type)

### 3.1 データの型の種類

型名	意味	例
int	整数	1, -12, 2022, 10, など
float	小数	3.14..., 0.5, 12.53, など
str	文字列	'hello', "こんにちは", など

数学では整数は小数 (実数) の中に含まれるが、コンピューターの世界では小数と整数の扱いが異なるので明確に別物。文字列は'(シングルクォーテーション)', "(ダブルクォーテーション) 記号で囲んだ部分のこと。

### 3.2 型と四則演算

- int 型と float 型の四則演算

数学の四則演算と同じ。

int 型と float 型で四則演算を行うときは、int 型を float 型として型の変換へんかんがされ計算が行われる。

- str 型と四則演算

- str 型では数学のような四則演算はできず、引き算・割り算はできない
- str 型同士の足し算は文字列の結合をする (文字列をくっつける)  
(掛け算と違い str 型と int 型の足し算はできない)
- str 型と int 型の掛け算は文字列をかけた分だけ繰り返す  
(足し算と違い str 型同士で掛け算はできない)

コード 3.2 str 型と四則演算

```
1 # str型の四則演算（足し算と掛け算）
2 a = "1"
3 b = "2"
4 print(a + b) # str型の足し算
5 print(a * 3) # str型の掛け算
```

## 出力

```
12
111
```

## 説明

str 型の変数 a と b が結合されたため出力が 12 となった。  
a \* 3 で 1 を 3 回繰り返すため、出力が 111 となった。

## 3.3 str 型の数字

コード 3.3 int 型と str 型

```
1 # int型とstr型の出力
2 int_val = -30 # int_valはint型(整数)
3 str_val = '-30' # str_valはstr型(文字列)
4 print(int_val)
5 print(str_val)
```

## 出力

```
-30
-30
```

## 説明

出力の見え方は全く同じでも、-30 が int 型か str 型かという違いがある。

if 文

もし (条件式) ならば (処理) を実行する, というように条件に合った時だけ (処理) を行うためのもの.

4.1 if 文の書き方

コード 4.4 if 文の基本構文-その 1

```
1 if 条件式 :
2     処理
```

コード 4.4  
条件が一つするとき

コード 4.5 if 文の基本構文-その 2

```
1 if 条件式 :
2     処理1
3 else :
4     処理2
```

コード 4.5  
条件が一つとその条件以外の  
ときに処理をしたいとき

コード 4.6 if 文の基本構文-その 3

```
1 if 条件式1 :
2     処理1
3 elif 条件式2 :
4     処理2
5 else :
6     処理3
```

コード 4.6  
条件が複数のときとそれらの条件  
以外のときに処理をしたいとき

注意

- 条件式の後ろには必ずコロン (:) をつける
- 条件式の後ろには必ずインデント (字下げ) をする
- else の後ろには条件式は書かない

4.2 条件式で使う記号

記号	意味	例	例の意味
==	等しい	$x == 5$	$x$ は 5 と等しい
!=	等しくない	$x != 3$	$x$ は 3 と等しくない
>	より大きい	$5 > 2$	5 は 2 より大きい
<	より小さい	$2 < 3$	2 は 3 より小さい
>=	以上	$a >= 0$	$a$ は 0 以上
<=	以下	$b <= 0$	$b$ は 0 以下

等しい事を示す記号は, 2 つの等号 (==) が使われている.  
1 つの等号 (=) では代入になってしまうので注意  
また, 数学では以上・以下は  $\geq \cdot \leq$  と書くが,  
プログラミングではそのように書くことが  
できないため代わりに  $>= \cdot <=$  と書く.

## 4.3 if文の例

コード 4.7 if文の例-その1

```
1 # 変数numの値が2の倍数かどうかの判定
2 if num % 2 == 0 :
3     print("num は2の倍数です")
4 else :
5     print("num は2の倍数ではありません")
```

コード 4.8 if文の例-その2

```
1 # 変数numが正の数か負の数か0かどうかの判定
2 if num > 0 :
3     print("num は正の数")
4 elif num < 0 :
5     print("num は負の数")
6 else :
7     print("num は0")
```

コード 4.7

num の値が 2 の倍数ならば,  
”num は 2 の倍数です” と出力.  
num の値が 2 の倍数でないならば,  
”num は 2 の倍数ではありません” と出力.  
2 の倍数の判定は `num % 2 == 0` で行うことができる.

コード 4.8

num が 0 より大きいとき (`num > 0`)  
”num は正の数” と出力.  
num が 0 より小さいとき (`num < 0`)  
”num は負の数” と出力.  
それ以外のとき (つまり num が 0 のとき)  
”num は 0” と出力.

## 複雑な if 文

### 4.4 if 文の仕組み

if 文の条件式からは True/False という値が返ってくる. これらは bool 型という「分類の値を持つ型」の値である.

条件式があっているときは True 条件式が間違っているときは False

if 文が実行される時は条件式が True のときである.

### 4.5 論理演算

複数の条件があるときに使う.

#### 4.5.1 and(論理積)

複数の条件がすべて合っていてほしいときに使う.

すべての条件が合っているときその条件式全体は True となる.

コード 4.9 and の例

```
1 if num % 3 == 0 and num % 5 == 0 :
2     print("3 と 5 の倍数です")
3 else :
4     print("3 と 5 の倍数でもありません")
```

#### 説明

num が 3 の倍数かつ 5 の倍数ならば,  
”3 と 5 の倍数です” と出力.  
num が 3 の倍数かつ 5 の倍数でないならば,  
”3 と 5 の倍数でもありません” と出力.

4.5.2 or(論理和)

複数の条件のどれか一つでも合っていてほしいときに使う。  
どれか一つでも条件が合っているときその条件式全体は True となる。

コード 4.10 or の例

```
1 if str == "中学生" or str == "小学生" :
2     print("中学生か小学生です")
```

説明

str が "中学生" か "小学生" ならば、  
"中学生か小学生です" と出力。

4.5.3 not(否定)

True ならば False, False ならば True となる。

コード 4.11 not の例

```
1 if not (num % 2 == 0) :
2     print("奇数です")
3 else :
4     print("偶数です")
```

説明

num が偶数でないならば, "奇数です" と出力。  
num が偶数ならば, "偶数です" と出力。

4.6 計算の優先順位 (再び)

優先順位が高いほど先に計算が行われる。同じ優先順位の場合は左から右へ順に計算される。(数学の計算と同じ)

優先順位	同じ演算のときの優先順位	
高	括弧 (かっこ)	
	算術演算	累乗 (**)
		乗算 (*), 除算 (/), 切り捨て除算 (/), 剰余 (%)
		加算 (+), 減算 (-)
	比較演算	>, >=, <, <=
低	論理演算	not(否定)
		and(論理積)
		or(論理和)

この表の同じ枠内での優先順位は同じ。

## for 文

決められた回数繰り返したいときに使う。繰り返し文ともいう。

## 5.1 for 文の書き方