プログラミング初級（Python）

値の演算

早稲田大学グローバルエデュケーションセンター

ブール値と数値と文字列を操る

新しいプログラミング言語を学ぶ唯一の方法は、
その言語でプログラムを書くことだ。

— デニス・リッチー

値の演算

- **ブール値**（TrueまたはFalseの値を持つ）
- **整数**（12、100000000などの小数点以下がない数値）
- **浮動小数点数**（3.14159のように小数点以下の部分がある数値、または、1.0e8のような指数表現。100000000.0<10の8乗>の意味。）
- **文字列**（"Hello"や"Python"などのクォーテーションで囲まれた文字または文字の並び。）
- これらの値は例えみると**原子**のようなプリミティブなもの。
- これらの値を組み合わせて**リスト**や**タプル**、**辞書**や**集合**などの**分子**を作る方法はそれらの演算方法も含めて後の講義で解説。
- このスライドでは、True、12、3.14159、"Hello"などの**リテラル**やそれらの代入値である**変数**を用いた**演算**を説明する。
- 演算とは数式の示す通りに所要の数値を計算すること。**四則演算**、**論理演算**、**比較演算**、**代入演算**などを意味する語。

算術演算子

演算子	意味	例
+	加算	2 + 3
-	減算	2 – 3
*	乗算	2 * 3
**	べき乗	2 ** 3
/	除算	6 / 3
//	除算 (切り捨て)	7 // 3
%	剰余演算 (余りを計算)	8 % 3

論理演算子

演算子	意味	例
`and`	論理積	True and False
`or`	論理和	True or False
`not`	否定	$not\ True$

比較演算子

演算子	意味	例
==	等しい	a == True
!=	等しくない	a! = True
>	大なり	a > 1
<	小なり	a < 1
>=	大なりイコール	a >= 1
<=	小なりイコール	a <= 1

代入演算子

演算子	意味	例	x の変化
=	変数の値を代入	a = 5	
+=	左辺 + 右辺で加算後，左辺に代入	a += 3	a が 5 → 8 へ変化
-=	左辺 - 右辺で減算後，左辺に代入	a -= 2	a が 5 → 3 へ変化
*=	左辺 * 右辺で乗算後，左辺に代入	a *= 2	a が 5 → 10 へ変化
**=	左辺 ** 右辺でべき乗演算後，左辺に代入	a **= 2	a が 5 → 25 へ変化
/=	左辺 / 右辺で除算後，左辺に代入	a /= 2	a が 5 → 2.5 へ変化
//=	左辺 // 右辺で除算後，左辺に代入 (切り捨て)	a //= 2	a が 5 → 2 へ変化
%=	左辺 % 右辺で除算後，左辺に代入 (剰余演算)	a %= 2	a が 5 → 1 へ変化

- Cや Javaには**インクリメント演算子 ++** があるが Python にはない。
- **a++** とする場合 **a += 1** もしくは **a = a + 1** とする必要がある。
- **デクリメント演算子 --** も同様。**a -= 1**もしくは**a = a - 1**とする。

ブール値の演算（データ型の表記：bool）

```
>>> True
True
>>> False
False
>>> True * False
0
>>> True + False
1
>>> True and False
False
>>> True or False
True
>>> not True
False
>>> print(True)
True
>>> x = True
>>> print(x)
True
>>> bool(True)
True
```

```
>> bool(1)
True
>>> bool(314)
True
>>> bool(-314)
True
>>> bool(3.14)
True
>>> bool('waseda')
True
>>> bool('university')
True
>>> bool(False)
False
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool('')
False
>>> bool("")
False
```

空文字と呼ぶ。

空文字と呼ぶ。

整数の演算（データ型の表記：int）

```
1  >>> 3
   3
2  >>> 0
   0
3  >>> 03
     File "<stdin>", line 1
       03
       ^
   SyntaxError:(略)
4  >>> +3
   3
5  >>> -3
   -3
6  >>> 1,000,000
   (1, 0, 0)
7  >>> 1_000_000
   1000000
8  >>> 3_1_4
   314
9  >>> 3 + 1
   4
```

```
10 >>> 3 - 1
   2
11 >>> 1 - 4
   -3
12 >>> 3 + 1 + 4
   8
13 >>> 3 + 1 - 4 + 1 - 5 - 9
   -13
14 >>> 1+5       +               9
   15
15 >>> 3 * 1
   3
16 >>> 1 * 3
   3
17 >>> 3 * 1 * 4 * 1 * 5 * 9
   540
18 >>> 4 / 3
   1.3333333333333333
19 >>> 4 // 3
   1
20 >>> 4 % 3
   1
```

整数の演算（データ型の表記：int）

```
>>> 3 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZezoDivisionError: division by zero
>>> 3 // 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZezoDivisionError: integer division or
modulo by zero
>>> a = 31
>>> a
31
>>> a - 4
27
>>> a
31
>>> a = a -4
>>> a
27
```

```
>>> a = 31
>>> temp = a - 4
>>> a = temp
>>> a
27
>>> a = 31
>>> a -= 4
>>> a
27
>>> a = 31
>>> a += 4
>>> a
35
>>> a = 31
>>> a *= 4
>>> a
124
>>> a = 31
>>> a /= 4
>>> a
7.75
```

整数の演算（データ型の表記：int）

```
>>> a = 31
>>> a //= 4
>>> a
7
>>> 31 % 4
3
>>> divmod(31, 4)
(7, 3)
>>> 31 // 4
7
>>> 31 % 4
3
>>> 3 ** 4
81
>>> 3.0 ** 4
81.0
>>> 4 ** 3.0
81.0
>>> 0 ** 3
0
```

```
>>> print(0 ** 3)
0
>>> print(0**3)
0
>>> print(0**      3)
0
>>> 0*      *3
  File "<stdin>", line 1
    0*      *3
            ^
SyntaxError:invalid syntax
>>> 3 + 1 * 4
7
>>> 3 + (1 * 4)
7
>>> -3 ** 4
-81
>>> -(3 ** 4)
-81
>>> (-3) ** 4
81
```

複数を演算子を使う場合、数学と同様に優先順位がある。

括弧をつけても良い。

これは期待していない？

優先順位の低い演算を先に実行する場合、括弧をつけると分かりやすくなる。

10進数以外の数値リテラル（整数）

- 2進数の整数リテラル: 0と1で表現。
  - リテラルの例：**0b**01111, **0B**101など。
  - **0b**はプレフィックス。
  - プリフィックスは接頭辞とも呼び2進数であることを示すのに使う。
  - `b`はbinary（バイナリ）の略。ラテン語のbinarius（2つのもの）から。
- 8進数の整数リテラル: 0～7で表現。
  - リテラルの例： **0o**314, **0O**413など。
  - **0o**または**0O**（ゼロと英字のオー）は8進数であることを示すプレフィックス。
  - `o`はoctal（オクタル）の略。ラテン語のocto（8）に由来。
- 16進数の整数リテラル: 0～9, a～fで表現。アルファベットは大文字表記可。
  - リテラルの例： **0x**ff, **0X**aa2b, 0x314A, 0X314a
  - **0x**は16進数であることを示すプレフィックス。
  - `x`はhexadecimal（ヘクサデシマル）の略。ラテン語のhexa（6）とdecimus（10）に由来。

整数の演算（データ型の表記：int）

```
>>> 10
10
>>> 0b10
2
>>> 0B10
2
>>> 0o10
8
>>> 0O10
8
>>> 0o19
  File "<stdin>", line 1
    0o19
       ^
SyntaxError:invalid digit '9' in octal
literal
>>> 0x10
16
>>> 0X1f
16
```

プレフィックスで基数を指定しないと Pythonは数値を10進数とみなす。

プレフィックスで基数２を指定。１０進数の数値リテラルを算出してくれる。

プレフィックスで基数８を指定。１０進数の数値リテラルを算出してくれる。

プレフィックスで基数16を指定。１０進数の数値リテラルを算出してくれる。

```
>>> 0x1g
  File "<stdin>", line 1
    0x1g
      ^
SyntaxError: invalid hexadecimal
literal
>>> value = 65
>>> bin(value)
'0b1000001'
>>> oct(value)
'0o101'
>>> hex(value)
'0x41'
>>> chr(65)
'A'
>>> ord('A')
65
>>> int(True)
1
>>> int(3.14)
3
```

浮動小数点数の演算（データ型の表記：float）

```
>>> 3.
3.0
>>> 3.0
3.0
>>> 03.0
3.0
>>> 3e0
3.0
>>> 3e1
30.0
>>> 3.0e1
30.0
>>> 3.0 * (10 ** 1)
30.0
>>> million = 1_000_000.0
>>> million
1000000.0
>>> 1.0_0_1
1.001
>>> float(True)
1.0
```

```
>>> float(False)
0.0
>>> float(31)
31.0
>>> float('31')
31.0
>>> float('31.4')
31.4
>>> float('-3.1')
-3.1
>>> float('3.1e4')
31000.0
>>> 31 + 4.
35.0
>>> False + 0
0
>>> False + 0.
0.0
>>> True + 0
1
>>> True + 0.
1.0
```

後半に続く