1 はじめに

このファイルでは「演算子の説明」と,「各型における演算子の挙動」を記しています.

2 目次

3 演算子とは

wikiによると

> プログラミング言語などで, 各種の演算を表す記号のことを指す

とのこと.

4 代数演算子

数値型の代数計算を行う際に用いられる演算子たちです.

演算例	意味	備考
`+` a	正数	
`-` a	負数	
a `+` b	加算	
a `-` b	減算	

演算例	意味	備考
a `*` b	乗算	
a `/` b	除算	C言語などとは違い, int/intでもfloatで返す
a `%` b	a/bの余り	
a `**` b	aのb乗	
a `//` b	割り算, 小数点以下切り捨て	C言語における `a/b` と同じ感じ

```
a = 5; b = 2


print("a + b ->", a + b)
print("a - b ->", a - b)
print("a * b ->", a * b)
print("a / b ->", a / b) #C言語との違いに注意
print("a % b ->", a % b)
print("a ** b ->", a ** b)
print("a // b ->", a // b)
```

少し注意が必要なのは, `//` と `/` についてです.
`//` が商, `%` は余り, `/` では完全に割り切るといった感じです.

これらの代数演算子を複数利用した時, 実行の優先順位というものが存在し, 必ずしも先頭から順に演算が行われるとは限りません.
(実行の優先順位は意外と細かく別れており, 複雑なので, 今回は説明を省略します.)

```
print( 3 + 5 + 7 / 5)    # => 3 + 5 + (7/5) になってしまう
```

そのため, 計算順序を明確にするために `()` を用いることができます.

```
print( (3 + 5 + 7) / 5 )
```

5 ビット演算子

数値を2進数と見て, 論理演算を行うことができます.
matplotlibではビット演算を行うようなことはないと思うので, ここの欄は飛ばしても大丈夫です.

演算例	内容
`~` a	ビット反転
a `&` b	AND, 論理積
a `¦` b	OR, 論理和
a `^` b	XOR, 排他的論理和
a `<<` b	aをbビット分左シフト
a `>>` b	aをbビット分右シフト

6 代入演算子

変数へ値を入れる際に利用される演算子たちです.

`=` が主ですが, 代数演算子やビット演算子と `=` を合体させたような記法もあります.

```
a = 10; print("= 10 ->", a) # 通常の代入
a += 5; print("+= 5 ->", a) # a = a + 5 と同じ
a -= 8; print("-= 8 ->", a) # a = a - 8 と同じ
a *= 2; print("*= 2 ->", a) # a = a * 2 と同じ
a //= 7; print("//= 7 ->", a) # a = a // 7 と同じ

print()

a = 11; print("= 11 ->", a)
a %= 7; print("%= 7 ->", a)   # a = a % 7 と同じ
a **= 2; print("**= 2 ->", a) # a = a ** 2 と同じ
a /= 3; print("/= 3 ->", a)   # a = a / 3 と同じ
```

また, 他の言語でよく見られる `インクリメント` (`a++` , `++a`)や `ディクリメント` (`a--` , `--a`)はないため, Pythonでは `a += 1` , `a -= 1` という風に書いていくことになります.

ビットの代入演算子も同様にありますが, ここでは紹介を省かせていただきます.

7 比較演算子

比較演算子を用いると二つの値の関係についてのブール値(真偽)が得られます.
`=` と何かがくっつく演算子は, `=` が右側に必ず来ると覚えておきましょう.

演算例	`True` なら
a `==` b	a が b と等しい
a `!= b`	a が b と等しくない
a `<` b	a が b より小さい
a `>` b	a が b より大きい
a `<=` b	a が b 以下
a `>=` b	a が b 以上
a `<>` b	a が b と等しいくない
a `is` b	a が b と等しい
a `is not` b	a が b と等しくない
a `in` b	a が b の中に含まれる
a `not in` b	a が b の中に含まれない

```
print( "3 == 5 ->",  3 == 5 )
print( "5 == 5 ->",  5 == 5 )

print( "10 is 20 ->", 10 is 20 )
print( "3 in [1, 2, 3, 4] ->", 3 in [1, 2, 3, 4] )
print( "3 in [1, 2] ->", 3 in [1, 2] )
```

8 ブール演算子

ブール値に対する演算子がいくつかあります.

演算例	内容
a `and` b	a も b も `True` => `True`
a `or` b	a か b が `True` => `True`
`not` a	`True/False` の内容を反転

```
print( "True and True ->", True and True )
print( "True and False ->", True and False )

print()

a = 10; b = 10; c = -99
print( "a == b and a == c ->", a == b and a == c )
print( "a == b or a == c ->", a == b or a == c )

print()

print( "a == b ->", a == b )
print( "not (a == b) ->", not (a == b) )
print( "a == c ->", a == c )
print( "not (a == c) ->", not (a == c) )
```

9 文字列と演算子

文字列に対しても一部の演算子が利用できます.

```
# 文字列同士の結合 +
print( '"ho" + "ge" ->', "ho" + "ge")

# 文字列をn回繰り返す *
print( '" ¦" * 5 ->', " ¦" * 5)
```

```
# 文字列の比較
print( '"AAA" == "AAA" ->', "AAA" == "AAA" )

# 文字列の順序比較
print( '"AAA" > "AAB" ->',  "AAA" > "AAB" )
print( '"あああ" < "あいう" ->',  "あああ" < "あいう" )

# 文字列内に特定の文字が含まれるか
print( '"a" in "abc" ->', "a" in "abc" )
print( '"x" in "abc" ->', "x" in "abc" )
```

10 リストと演算子

リストでも演算子を用いていくつか特定の動作を行えます.

```
# リストの比較
print( "[1, 2, 3] == [1, 2] ->", [1, 2, 3] == [1, 2] )
print( "[1, 2, 3] == [1, 2, 3] ->", [1, 2, 3] == [1, 2, 3] )

# リストの結合
print( "[1, 2, 3] + [4, 5] ->", [1, 2, 3] + [4, 5] )

# リストの内容の反復
print( "[1, 2] * 3 ->", [1, 2] * 3 )
```