位置引数のタプル化による可変個引数の受渡し

本章では、2値の最大値を求める関数 *max2* と、3値の最大値を求める関数 *max3* を作りました。**List 9-17** の *max2more* は、**2個以上の任意の個数**の値の最大値を求める関数です。

List 9-17 chap09/list0917.py

```
# ２個以上の任意の個数の値の最大値を求める
def max2more(a, b, *num):
    """２個以上の任意の個数の値の最大値を求める"""
    max = a if a > b else b
    for n in num:
        if n > max:
            max = n
    return max

print('max2more(1, 2)           = ', max2more(1, 2))
print('max2more(1, 2, 3)        = ', max2more(1, 2, 3))
print('max2more(1, 2, 3, 4, 5) = ', max2more(1, 2, 3, 4, 5))
print('max2more(1)              = ', max2more(1))
```

0個以上の実引数をタプルとして受け取る

実行結果

```
max2more(1, 2)           =  2
max2more(1, 2, 3)        =  3
max2more(1, 2, 3, 4, 5) =  5
```

b に対する実引数が欠如

```
Traceback (most recent call last):
  File "MeikaiPython\chap09\lit0917.py", line 14, in <module>
    print('max2more(1)              = ', max2more(1))
TypeError: max2more() missing 1 required positional argument: 'b'
```

プログラムと実行結果を対比しましょう。

アステリスク * の付いた仮引数 *num* は、0個以上の任意の個数の値を受け取ります（**Fig.9-10**）。

3番目を含め、それ以降のすべての実引数が、タプルにパックされます。そのタプルが渡されて、仮引数 *num* に代入される、という仕組みです。

▶ 図aの場合は、“空タプル”が作られて関数 *max2more* の *num* に渡されます。

呼出し側の実引数の個数が任意、すなわち可変個となることから、**可変個実引数**（*arbitrary argument*）と呼ばれます。

なお、関数本体では、“a”と“b”と“タプル *num* の全要素”の、最大値を求めます。

▶ 最後の呼出し *max2more(1)* は、仮引数 b に対して実引数を与えていないため、エラーとなります。

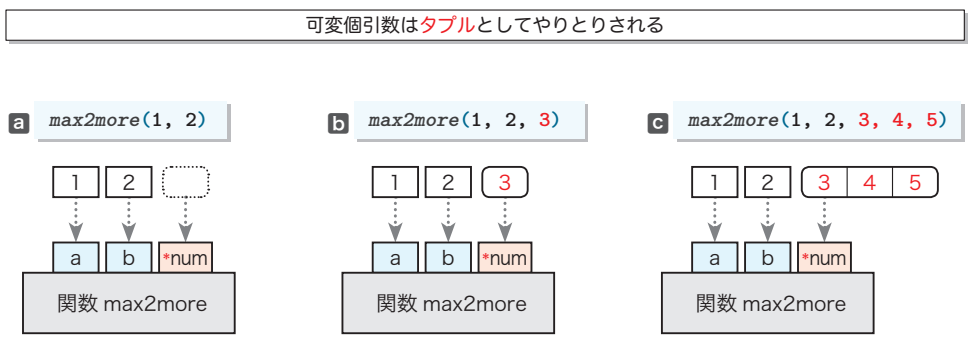

Fig.9-10 可変個引数の受渡し

アステリスク＊付きで宣言された仮引数がタプルであることは、**List 9-18** のプログラムで確認できます。

List 9-18 chap09/list0918.py

```
# 可変個引数の情報を表示する

def print_args(*args):
    """可変個の引数を受け取るargsの情報を表示"""
    print('type(args) = ', type(args))
    print('len(args)  = ', len(args))
    print('args       = ', args)

print_args()
print()
print_args(1, 2, 3)
```

実行結果

```
type(args) =  <class 'tuple'>
len(args)  =  0
args       =  ()

type(args) =  <class 'tuple'>
len(args)  =  3
args       =  (1, 2, 3)
```

関数 *print_args* では、唯一の仮引数 *args* が、アステリスク＊付きで宣言されています。そのため、この関数は、**0個以上の任意の個数**の引数を受け取ることになります。

重要 アステリスク＊付きで宣言された仮引数は、0個以上の任意の個数の値をタプルとして受け取る。呼出し側の可変個の実引数は、暗黙裏にパックされる。

次は、**1個以上の任意の個数**の引数を受け取る関数を作ります。それが、**List 9-19** のプログラムです。関数 *print_sum* は、引数の和を、式を表示しながら求めます。

List 9-19 chap09/list0919.py

```
# 引数の和を表示しながら求める

def print_sum(a, *no):
    """引数の和を返却（式も表示）"""
    sum = a
    print(a, end='')
    n = len(no)
    if n > 0:
        print(' + ', end='')
        for i in range(n - 1):
            sum += no[i]
            print(no[i], '+ ', end='')
        sum += no[n - 1]
        print(no[n - 1], end='')
    print(' =', sum)
    return sum

print_sum(5)
print_sum(9, 3)
print_sum(3, 6, 8, 2, 7)
```

実行結果

```
5 = 5
9 + 3 = 12
3 + 6 + 8 + 2 + 7 = 26
```

▶ プログラムの詳細な解説は省略します。

可変個引数は、その性格上、仮引数の末尾に置くのが基本です。可変個引数よりも後ろに置く仮引数は、位置引数ではなく、キーワード引数に限定されます。