str.format 関数

v2.6 以降に追加された。
モジュロ演算子は将来削除予定、とのこと。

```
print('key={0}, value={1}'.format(key, value))
```

{n} という書式で、C# と同じ。
元はどっかの言語から取ってきたんだろうけど。

名前付き引数のバリエーションの例。

```
print('sent={sent_bytes} recv={recv_bytes}'.format(sent_bytes=s, recv_bytes=r))
```

ディクショナリーを使うときは、{} の中に、format 関数の引数位置と [] でディクショナリーのキーを指定する。キーは文字列として指定する必要はない。

```
print('OS={0[OS]} WINDIR={0[WINDIR]} USERNAME={0[USERNAME]}'.format(os.environ))
```

format 関数の引数に指定したオブジェクトのプロパティも使用できる。

```
print('Python v.{0.version} on {0.plathome}.'.format(sys))
```

{n} には、コロンで区切ってフォーマット指定子による整形が可能。桁数指定の例。

```
print('Today is {0}-{1:02}-{3:02}.'.format(year, month, date))
```

小数点 2 ケタの例。

```
print('Average: {0:.2f}'.format(calculated_avg))
```

モジュロ演算子（旧式）

モジュロ演算子 % による文字列フォーマット。
基本形：

```
print("%s is a %s" % ("This", "pen"))
print("Today is %04d-%02d-%02d." % (2010, 1, 1))
```

文字列の中に、printf 風のフォーマット指定文字を含める。
% 演算子でタプルを第 2 オペランドとして置く。
そうすると、第 1 オペランドとなった文字列中のフォーマット指定文字の部分が、第 2 オペランドの要素で置き換えられる。

第 1 オペランドの文字列中に含まれるフォーマット指定文字の個数と、第 2 オペランドのタプル

の要素数は、一致させる必要がある。
また、データ型も合わせる必要がある（タプルの要素のデータ型に合わせたフォーマット指定文字を指定する必要がある）。

```
print("Today is %s-%s-%s." % (2010, 1, 1))
TypeError: ...
```

モジュロ演算子による文字列フォーマットは、単に評価されたフォーマット後の文字列を返すだけなので、変数の右辺や関数の引数にも使用できる。

また、第 2 オペランドとしてタプルではなくディクショナリーを指定可能。

```
print("Today is %(year)04d-%(mon)02d-%(date)02d." % {"date":1, "mon":1, "year":2010}
```

丸括弧でディクショナリーのキーを指定しているだけ。これで順不同のデータに対してもフォーマットできる。