

Python String Formatting


by Dr. Sethavidh Gertphol

Outline

- String format method
- Format specifier
- f-string

String format method

- สตริงใน Python มี method ชื่อ `format()` ที่ใช้แสดงค่าและกำหนดรูปแบบการแสดงผลตรงตำแหน่งที่ระบุไว้ในสตริงได้
- ภายในสตริงจะต้องมี **replacement field** ที่ระบุด้วย `{}` ซึ่งเป็นตำแหน่งที่จะนำค่ามาแสดงผล
- ค่าที่จะนำมาแสดงผลจะส่งเป็น **argument** ใน `format()`



The diagram illustrates the components of a Python string format call. An orange box labeled "replacement field" has an arrow pointing to the curly braces `{}` in the string `'Python {} is the newest version.'`. Another orange box labeled "argument" has an arrow pointing to the value `3.11` inside the `format()` method call.

```
>>> 'Python {} is the newest version.'.format(3.11)
'Python 3.11 is the newest version.'
```

การแสดงค่าใน Replacement field

- **automatic**: replacement field ไม่มีการระบุลำดับหรือชื่อตัวแปร การแสดงผลจะทำตามลำดับของ argument ของ format()

```
>>> "{}'s midterm score is {}".format('Sethavidh', 32)
'Sethavidh's midterm score is 32'
```

- **positional**: replacement field มีตัวเลขกำกับ จะนำ argument ที่ลำดับตรงกับตัวเลขนั้นมาแสดงผล

```
>>> name='Sethavidh'
>>> "{1}'s midterm score is {0}".format(8*4, name)
'Sethavidh's midterm score is 32'
```

argument เป็น expression ได้

- **keyword**: replacement field มี keyword กำกับ จะนำค่าของ keyword นั้นมาแสดง

```
>>> "{fname}'s midterm score is {}".format(32, fname='Sethavidh')
'Sethavidh's midterm score is 32'
```

- ใช้ automatic กับ positional ปนกันไม่ได้

Value formatting

- สามารถกำหนดรูปแบบการแสดงค่าได้ผ่าน **format specifier** หลังเครื่องหมาย **:** ใน replacement field
- **Format specifier** กำหนดได้ตามลำดับดังนี้ (ไม่จำเป็นต้องใช้ทุกรูปแบบ)
- **[[fill]align][sign][#][0][width][grouping_option][.precision][type]**
- **width:**
 - เป็นเลขจำนวนเต็ม
 - กำหนดขนาดพื้นที่ที่จะแสดงค่า ถ้าค่ามีขนาดใหญ่กว่าพื้นที่ที่กำหนดจะใช้ขนาดของค่า

{:10} หมายถึงให้เตรียมพื้นที่แสดงค่าขนาด **10** ตัวอักษร

```
>>> '{:10}'.format('hello')
'hello '
>>> '{:10}'.format(2.345)
' 2.345'
```

ค่าที่เป็นสตริงจะชิดซ้าย

ค่าที่เป็นตัวเลขจะชิดขวา

Fill and Align

- **Align:** กำหนดให้แสดงค่าชิดซ้าย ขวา หรือตรงกลาง
 - < ชิดซ้าย, > ชิดขวา, ^ ตรงกลาง, = พิเศษสำหรับค่าที่เป็นตัวเลข (อธิบายทีหลัง)

```
>>> '{:<10}'.format(2.345)
'2.345      '
>>> '{:^10}'.format('hello')
'   hello   '
```

- **Fill:** ใส่อักขระใด ๆ ก่อนหน้าสัญลักษณ์ของ **align** จะเติมช่องว่างด้วยสัญลักษณ์นี้

```
>>> '{:_<10}'.format(2.345)
'2.345_____ '
>>> '{:*^10}'.format('hello')
'**hello***'
```

Sign and 0

- **sign**: กำหนดสัญลักษณ์สำหรับค่าบวก (ค่าลบใช้ - อยู่แล้ว)
 - +, - (ไม่มีสัญลักษณ์สำหรับค่าบวก), ' ' (ใช้ช่องว่างเป็นสัญลักษณ์)
- **0**: เติมค่า 0 ระหว่างสัญลักษณ์ +/- และเลขตัวแรก

```
>>> '{:+10}'.format(2.345)
'+2.345'
```

```
>>> '{:+010}'.format(2.345)
'+00002.345'
```

เตรียมพื้นที่ว่าง 10 ช่อง

- **= (align)** เมื่อใช้กับตัวเลข จะพิมพ์ตัวเลขชิดขวาแต่พิมพ์สัญลักษณ์ +/- ชิดซ้าย

```
>>> '{:=+10}'.format(2.345)
'+      2.345'
>>> '{:_+=10}'.format(2.345)
'+____2.345'
```

ใส่ 0 ระหว่างสัญลักษณ์
บวกลบและเลขตัวแรก

Format type

- ระบุวิธีการรับค่าเพื่อแสดงผล
- integer format
 - d: decimal, b: binary, o: octal, x/X: hexadecimal
 - c: unicode symbol ของค่าจำนวนเต็ม
 - n: แสดงเลขฐานสิบเหมือน d แต่ใส่สัญลักษณ์คั่นหลักด้วย (แต่ต้อง set locale ก่อน)

```
>>> '{:b}'.format(15)
'1111'
>>> '{:o}'.format(15)
'17'
>>> '{:x}'.format(15)
'f'
>>> '{:X}'.format(15)
'F'
```

```
>>> '{:c}'.format(3585)
'ก'
```

```
>>> import locale
>>> locale.setlocale(locale.LC_NUMERIC, 'en_US')
>>> '{:n}'.format(3585)
'3,585'
```


Format type

- #: สำหรับเลขฐาน ให้ใส่ 0b, 0o, 0x/0X กำกับด้วย
- Floating point
 - f/F: fixed point
 - e/E: exponential (scientific)
 - g/G: automatic
 - n: ใส่สัญลักษณ์แบ่งหลักด้วย
 - %: ทำเป็นเปอร์เซ็นต์
- s: string format

```
>>> '{:#b}'.format(15)
'0b1111'
>>> '{:#o}'.format(15)
'0o17'
>>> '{:#x}'.format(15)
'0xf'
```

```
>>> '{:f}'.format(1234.56789)
'1234.567890'
>>> '{:e}'.format(1234.56789)
'1.234568e+03'
>>> '{:n}'.format(1234.56789)
'1,234.57'
>>> '{:%}'.format(1234.56789)
'123456.789000%'
```

ทศนิยม 6 หลักเป็น
default

Precision and grouping

- ใช้ . ตามด้วยเลขจำนวนเต็ม ก่อนหน้า **type** เช่น .2f
- ถ้า **type** เป็น f/F หรือ e/E หรือ g/G จะเป็นการกำหนดจำนวนเลขหลังจุดทศนิยม

```
>>> '{:.2f}'.format(1234.56789)
'1234.57'
>>> '{:.2e}'.format(1234.56789)
'1.23e+03'
```

- ถ้า **type** เป็น s จะเป็นการระบุจำนวนอักขระที่จะแสดง

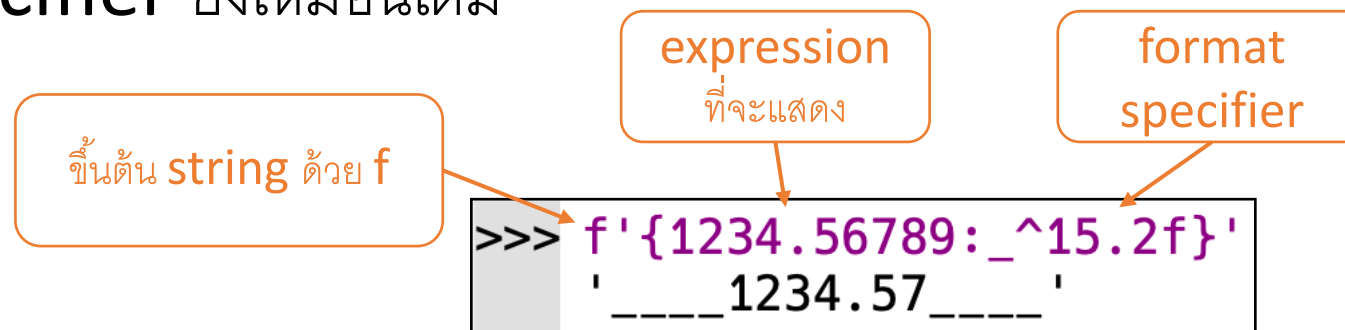
```
>>> '{:.2s}'.format('hello world')
'he'
```

- grouping: ใช้ , หรือ _ คั่นหลัก ไม่ใช้ร่วมกับ **type** n

```
>>> '{:_}'.format(1234.56789)
'1_234.56789'
>>> '{:,}'.format(1234.56789)
'1,234.56789'
```

f-string

- เป็นเทคนิคการ **format string** แบบใหม่ที่เริ่มใช้ได้จาก **Python 3.6** เป็นต้นไป
- สามารถนำค่าหรือ **expression** ที่ต้องการแสดงมาใส่ใน **{}** ได้เลย แทนที่จะต้องนำไปเป็น **argument** ของ **.format()**
- ต้องใส่ **f** หรือ **F**string ด้วย
- รูปแบบ **specifier** ยังเหมือนเดิม



Expression inside {}

- สามารถใช้ตัวแปรและ **expression** ใน {} ได้ไม่จำเป็นต้องเป็นแค่ค่า

```
>>> balance=16458.3452
>>> f'Your balance is {balance:,.2f} Baht'
'Your balance is 16,458.35 Baht'
```

```
>>> difficulty = 2
>>> f'Your score is {(15+12)*difficulty:**6}'
'Your score is **54**'
```

- เรียกฟังก์ชันใน **f-string** ได้เช่นกัน

```
>>> s = 'hello world'
>>> f"The number of characters in '{s}' is {len(s)-s.count(' ')}."
"The number of characters in 'hello world' is 10."
```

References

- Alex Martelli, et. al., "Python in a Nutshell, 4th Edition", O'Reilly Media, Inc., January 2023.