1.基本使用

1.1 简单使用

f-string用大括号 {} 表示被替换字段，其中直接填入替换内容：

```
>>> name = 'Andy'
>>> f'Hello, my name is {name}'
'Hello, my name is Andy'

>>> number = 7
>>> f'My lucky number is {number}'
'My lucky number is 7'

>>> price = 19.99
>>> f'The price of this book is {price}'
'The price of this book is 19.99'
```

1.2 表达式求值与函数调用

f-string的大括号 {} 可以填入表达式或调用函数，Python会求出其结果并填入返回的字符串内：

```
>>> f'A total number of {24 * 8 + 4}'
'A total number of 196'

>>> f'Complex number {(2 + 2j) / (2 - 3j)}'
'Complex number (-0.15384615384615388+0.7692307692307692j)'

>>> name = 'Andy'
>>> f'My name is {name.lower()}'
'My name is andy'

>>> import math
>>> f'The answer is {math.log(math.pi)}'
'The answer is 1.1447298858494002'
```

1.3 引号、大括号与反斜杠

f-string大括号内所用的引号不能和大括号外的引号定界符冲突，可根据情况灵活切换 ' 和 "：

```
>>> f'I am {"Andy"}'
'I am Andy'
>>> f'I am {'Andy'}'
  File "<stdin>", line 1
    f'I am {'Andy'}'
                ^
SyntaxError: invalid syntax
```


若 ' 和 " 不足以满足要求，还可以使用 ''' 和 """：

```
>>> f"He said {"I'm Andy"}"
  File "<stdin>", line 1
    f"He said {"I'm Andy"}"
                ^
SyntaxError: invalid syntax

>>> f'He said {"I'm Andy"}'
  File "<stdin>", line 1
    f'He said {"I'm Andy"}'
                  ^
SyntaxError: invalid syntax

>>> f"""He said {"I'm Andy"}"""
"He said I'm Andy"
>>> f'''He said {"I'm Andy"}'''
"He said I'm Andy"
```

大括号外的引号还可以使用 \ 转义，但**大括号内不能使用 \ 转义**：

```
>>> f'''He\'ll say {"I'm Andy"}'''
"He'll say I'm Andy"
>>> f'''He'll say {"I\'m Andy"}'''
  File "<stdin>", line 1
SyntaxError: f-string expression part cannot include a backslash
```

f-string大括号外如果需要显示大括号，则应输入连续两个大括号 {{ 和 }}：

```
>>> f'5 {"{stars}"}'
'5 {stars}'
>>> f'{{5}} {"stars"}'
'{5} stars'
```

上面提到，f-string大括号内不能使用 \ 转义，事实上不仅如此，f-string大括号内根本就不允许出现 \。如果确实需要 \，则应首先将包含 \ 的内容用一个变量表示，再在f-string大括号内填入变量名：

```
>>> f"newline: {ord('\n')}"
  File "<stdin>", line 1
SyntaxError: f-string expression part cannot include a backslash

>>> newline = ord('\n')
>>> f'newline: {newline}'
'newline: 10'
```


1.4 多行f-string

f-string还可用于多行字符串：

```
>>> name = 'Andy'
>>> age = 27
>>> f"Hello!" \
... f"I'm {name}." \
... f"I'm {age}."
"Hello!I'm Andy.I'm 27."
>>> f"""Hello!
...     I'm {name}.
...     I'm {age}."""
"Hello!\n    I'm Andy.\n    I'm 27."
```

2.自定义格式：对齐、宽度、符号、补零、精度、进制等

f-string采用**{content:format}** 设置字符串格式，其中 content 是替换并填入字符串的内容，可以是变量、表达式或函数等，format 是格式描述符。采用默认格式时不必指定 {:form at}，如上面例子所示只写 {content} 即可。

关于格式描述符的详细语法及含义可查阅Python官方文档，这里按使用时的先后顺序简要介绍常用格式描述符的含义与作用：

2.1 对齐相关格式描述符

格式描述符	含义与作用
`<`	左对齐（字符串默认对齐方式）
`>`	右对齐（数值默认对齐方式）
`^`	居中

示例：

```
>>> s = 3.1415926
>>> f'{s:<13}'
'3.1415926    '
>>> f'{s:>13}'
'    3.1415926'
>>> f'{s:^13}'
'  3.1415926  '
```


2.2 数字符号相关格式描述符

格式描述符	含义与作用
`+`	负数前加负号（`-`），正数前加正号（`+`）
`-`	负数前加负号（`-`），正数前不加任何符号（默认）
` `（空格）	负数前加负号（`-`），正数前加一个空格

注：仅适用于数值类型。

示例：

```
>>> f'{123:+}'
'+123'
>>> f'{-123:+}'
'-123'
>>> f'{123: }'
' 123'
>>> f'{-123: }'
'-123'
```

2.3 数字显示方式相关格式描述符

格式描述符	含义与作用
`#`	切换数字显示方式

注1：仅适用于数值类型。
注2：`#` 对不同数值类型的作用效果不同，详见下表：

数值类型	不加 `#`（默认）	加 `#`	区别
二进制整数	`'1111011'`	`'0b1111011'`	开头是否显示 `0b`
八进制整数	`'173'`	`'0o173'`	开头是否显示 `0o`
十进制整数	`'123'`	`'123'`	无区别
十六进制整数（小写字母）	`'7b'`	`'0x7b'`	开头是否显示 `0x`
十六进制整数（大写字母）	`'7B'`	`'0X7B'`	开头是否显示 `0X`

示例：

```
>>> f'{123:#0b}'
'0b1111011'
>>> f'{123:#0o}'
'0o173'
>>> f'{123:#0x}'
'0x7b'
>>> f'{123:#0X}'
'0X7B'
```


2.4宽度与精度**相关格式描述符

格式描述符	含义与作用
`width`	整数 `width` 指定宽度
`0width`	整数 `width` 指定宽度，开头的 `0` 指定高位用 `0` 补足宽度
`width.precision`	整数 `width` 指定宽度，整数 `precision` 指定显示精度

- 注1：0width 不可用于复数类型和非数值类型，width.precision 不可用于整数类型。
- 注2：width.precision 用于不同格式类型的浮点数、复数时的含义也不同：用于 f、F、e、E 和 % 时 precision 指定的是小数点后的位数，用于 g 和 G 时 precision 指定的是有效数字位数（小数点前位数+小数点后位数）。
- 注3：`width.precision` 除浮点数、复数外还可用于字符串，此时 `precision` 含义是只使用字符串中前 `precision` 位字符。

示例：

```
>>> a = 123.456
>>> f'a is {a:8.2f}'
'a is   123.46'
>>> f'a is {a:08.2f}'
'a is 00123.46'
>>> f'a is {a:8.2e}'
'a is 1.23e+02'
>>> f'a is {a:8.2%}'
'a is 12345.60%'
>>> f'a is {a:8.2g}'
'a is  1.2e+02'

>>> s = 'hello'
>>> f's is {s:8s}'
's is hello   '
>>> f's is {s:8.3s}'
's is hel     '
```

2.5 千位分隔符相关格式描述符

格式描述符	含义与作用
,	使用 , 作为千位分隔符
_	使用 _ 作为千位分隔符

- 注1：若不指定 , 或 _，则f-string不使用任何千位分隔符，此为默认设置。
- 注2：, 仅适用于浮点数、复数与十进制整数：对于浮点数和复数，, 只分隔小数点前的数位。
- 注3：_ 适用于浮点数、复数与二、八、十、十六进制整数：对于浮点数和复数，_ 只分隔小数点前的数位；对于二、八、十六进制整数，固定从低位到高位每隔四位插入一个 _（十进制整数是每隔三位插入一个 _）。

示例：

```
>>> a = 1234567890.098765
>>> f'a is {a:f}'
'a is 1234567890.098765'
>>> f'a is {a:,f}'
'a is 1,234,567,890.098765'
>>> f'a is {a:_f}'
'a is 1_234_567_890.098765'

>>> b = 1234567890
>>> f'b is {b:_b}'
'b is 100_1001_1001_0110_0000_0010_1101_0010'
>>> f'b is {b:_o}'
'b is 111_4540_1322'
>>> f'b is {b:_d}'
'b is 1_234_567_890'
>>> f'b is {b:_x}'
'b is 4996_02d2'
```

2.6 格式类型相关格式描述符

基本格式类型

格式描述符	含义与作用	适用变量类型
`s`	普通字符串格式	字符串
`b`	二进制整数格式	整数
`c`	字符格式，按unicode编码将整数转换为对应字符	整数
`d`	十进制整数格式	整数
`o`	八进制整数格式	整数
`x`	十六进制整数格式（小写字母）	整数
`X`	十六进制整数格式（大写字母）	整数
`e`	科学计数格式，以 `e` 表示 `×10^`	浮点数、复数、整数（自动转换为浮点数）
`E`	与 `e` 等价，但以 `E` 表示 `×10^`	浮点数、复数、整数（自动转换为浮点数）
`f`	定点数格式，默认精度（`precision`）是6	浮点数、复数、整数（自动转换为浮点数）
`F`	与 `f` 等价，但将 `nan` 和 `inf` 换成 `NAN` 和 `INF`	浮点数、复数、整数（自动转换为浮点数）
`g`	通用格式，小数用 `f`，大数用 `e`	浮点数、复数、整数（自动转换为浮点数）
`G`	与 `G` 等价，但小数用 `F`，大数用 `E`	浮点数、复数、整数（自动转换为浮点数）
`%`	百分比格式，数字自动乘上100后按 `f` 格式排版，并加 `%` 后缀	浮点数、整数（自动转换为浮点数）

常用的特殊格式类型：标准库 `datetime` 给定的用于排版时间信息的格式类型，适用于 `date`、`datetime` 和 `time` 对象

格式描述符	含义	显示样例
`%a`	星期几（缩写）	`'Sun'`
`%A`	星期几（全名）	`'Sunday'`
`%w`	星期几（数字，`0` 是周日，`6` 是周六）	`'0'`
`%u`	星期几（数字，`1` 是周一，`7` 是周日）	`'7'`
`%d`	日（数字，以 `0` 补足两位）	`'07'`
`%b`	月（缩写）	`'Aug'`
`%B`	月（全名）	`'August'`
`%m`	月（数字，以 `0` 补足两位）	`'08'`
`%y`	年（后两位数字，以 `0` 补足两位）	`'14'`
`%Y`	年（完整数字，不补零）	`'2014'`
`%H`	小时（24小时制，以 `0` 补足两位）	`'23'`
`%I`	小时（12小时制，以 `0` 补足两位）	`'11'`
`%p`	上午/下午	`'PM'`
`%M`	分钟（以 `0` 补足两位）	`'23'`
`%S`	秒钟（以 `0` 补足两位）	`'56'`

%f	微秒（以 0 补足六位）	'553777'
%z	UTC偏移量（格式是 ±HHMM[SS]，未指定时区则返回空字符串）	'+1030'
%Z	时区名（未指定时区则返回空字符串）	'EST'
%j	一年中的第几天（以 0 补足三位）	'195'
%U	一年中的第几周（以全年首个周日后的星期为第0周，以 0 补足两位）	'27'
%w	一年中的第几周（以全年首个周一后的星期为第0周，以 0 补足两位）	'28'
%V	一年中的第几周（以全年首个包含1月4日的星期为第1周，以 0 补足两位）	'28'

3.综合示例

```
>>> a = 1234
>>> f'a is {a:^#10X}'       # 居中，宽度10位，十六进制整数（大写字母），显示0X前缀
'a is   0X4D2    '

>>> b = 1234.5678
>>> f'b is {b:<+10.2f}'     # 左对齐，宽度10位，显示正号（+），定点数格式，2位小数
'b is +1234.57  '

>>> c = 12345678
>>> f'c is {c:015,d}'       # 高位补零，宽度15位，十进制整数，使用,作为千分分割位
'c is 000,012,345,678'

>>> d = 0.5 + 2.5j
>>> f'd is {d:30.3e}'       # 宽度30位，科学计数法，3位小数
'd is           5.000e-01+2.500e+00j'

>>> import datetime
>>> e = datetime.datetime.today()
>>> f'the time is {e:%Y-%m-%d (%a) %H:%M:%S}'    # datetime时间格式
'the time is 2018-07-14 (Sat) 20:46:02'
```