ماژول string

آخرین به‌روزرسانی: ۰۱ خرداد ۱۴۰۵

ماژول `string` یکی از ماژول‌های **کتابخانه استاندارد پایتون** است که مجموعه‌ای از ثابت‌های رشته‌ای، کلاس‌های قالب‌بندی پیشرفته و توابع کمکی را در اختیار برنامه‌نویس می‌گذارد. این ماژول را نباید با **نوع داده استرینگ** اشتباه گرفت؛ `str` همیشه در دسترس است و نیازی به ایمپورت ندارد، اما `string` یک ماژول جداگانه است که باید صریحاً ایمپورت شود.

ماژول `string` سه بخش اصلی دارد:

- ثابت‌های رشته‌ای آماده مثل `ascii_letters` و `digits`
- کلاس `Formatter` برای سفارشی‌سازی قالب‌بندی رشته
- کلاس `Template` برای جایگزینی ساده و ایمن متغیرها در رشته‌ها

فهرست مطالب:

برای استفاده از این ماژول، ابتدا باید آن را ایمپورت کنید:

```
import string
```

ثابت‌های رشته‌ای

ماژول string مجموعه‌ای از ثابت‌های آماده تعریف کرده که در اعتبارسنجی داده‌های ورودی، تولید رمز تصادفی، پردازش متن و ... بسیار پرکاربرد هستند.

```
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'

>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'

>>> string.digits
'0123456789'

>>> string.hexdigits
'0123456789abcdefABCDEF'

>>> string.octdigits
'01234567'

>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

>>> string.whitespace
' \t\n\r\x0b\x0c'

>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;
<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'
```

یکی از کاربردهای واقعی ثابت‌های رشته‌ای ماژول string، اعتبارسنجی ورودی کاربر است:

```
import string

def is_valid_username(username):
    allowed = string.ascii_letters + string.digits + "_-"
    return (
        3 <= len(username) <= 30
        and all(c in allowed for c in username)
        and username[0] in string.ascii_letters
    )

print(is_valid_username("ali_rezaei"))   # output: True
print(is_valid_username("123abc"))       # output: False – digits at start
print(is_valid_username("ab"))           # output: False – too short
print(is_valid_username("ali rezaei"))   # output: False – space not allowed
```

در این مثال، تابع ما بررسی می‌کند که نام کاربری انتخابی، بین ۳ تا ۳۰ کاراکتر باشد، تمام کاراکترهای آن مجاز باشند (اعضای متغیر `allowed`) و کاراکتر اول آن فقط حروف الفبا باشد.

کلاس Template

کلاس `Template` یک روش ساده و ایمن برای جایگزینی متغیرها در رشته‌ها فراهم می‌کند. این کلاس از نماد `$` برای مشخص کردن placeholderها استفاده می‌کند و برای موقعیت‌هایی مناسب است که قالب متن توسط کاربر یا فایل پیکربندی تعریف می‌شود.

```
from string import Template

t1 = Template("Hello, $name!")
t2 = Template("${name}_file.txt")
```

در صورتی که نوشتن نماد `$` به تنهایی منجر به ابهام در کد شود (مانند مثال t2 در بالا)، بهتر است از ساختار `${}` برای تعریف آن استفاده شود.

پس از این که قالب‌های متن را با آبجکت‌هایی از کلاس `Template` تعریف کردیم، می‌توان با متد `substitute` به آنها مقداردهی کرد:

```
>>> t1.substitute(name="Reza")
'Hello, Reza!'

>>> t2.substitute(name="report")
'report_file.txt'
```

همچنین می‌توان تمامی placeholderها را توسط یک دیکشنری مقداردهی کرد:

```
t3 = Template("Dear $name, Your order #$order_id will send to $city.")
data = {"name": "Ali", "order_id": "ORD-9810", "city": "Isfahan"}

>>> t3.substitute(data)
'Dear Ali, Your order #ORD-9810 will send to Isfahan.'
```

در این حالت، اگر یکی از کلیدهای دیکشنری مقادیر ناقص باشد، متد substitute خطای KeyError می‌دهد. جهت جلوگیری از این خطا، می‌توان از متد مشابه safe_substitute استفاده کرد. این متد در صورت نبود کلید، به جای نمایش خطا، placeholder را دست‌نخورده رها می‌کند و برای موقعیت‌هایی ایمن‌تر است که قالب ممکن است متغیرهای بیشتری داشته باشد:

```
>>> t4 = Template("Hello $name, your score is $score and rank is $rank.")
>>> t4.safe_substitute(name="Ali", score=95)
'Hello Ali, your score is 95 and rank is $rank.'
```

برتری امنیتی Template نسبت به f-string و format: اگرچه خروجی نهایی Template می‌تواند کاملاً مشابه با خروجی f-string یا متد format استرینگ‌ها باشد، اما Template در ماژول string فقط عمل جایگذاری ساده متن را انجام می‌دهد و هیچ عبارت پایتونی را اجرا یا ارزیابی (evaluate) نمی‌کند. به همین دلیل وقتی داده‌ها از کاربر، فرم، API یا فایل خارجی می‌آیند، استفاده از Template معمولاً امن‌تر از f-string یا format است.

در f-string اگر متن قالب از کاربر گرفته شود، ممکن است کد پایتون اجرا شود؛ چون f-string می‌تواند عبارت‌های پایتونی را تفسیر کند. متد format هم می‌تواند در بعضی شرایط مشکل‌ساز باشد؛ چون به ویژگی‌ها و attributeهای اشیاء دسترسی دارد.

کلاس Formatter

کلاس Formatter پیاده‌سازی داخلی متد format استرینگ‌ها را در اختیار برنامه‌نویس می‌گذارد تا بتواند رفتار قالب‌بندی را سفارشی نماید. در حالت معمول، متد format یا f-string کافی است، اما Formatter برای ساخت زیرکلاس‌های با رفتار دلخواه طراحی شده است.

```
from string import Formatter

class UpperFormatter(Formatter):
    def format_field(self, value, format_spec):
        return super().format_field(str(value).upper(), format_spec)

fmt = UpperFormatter()

print(fmt.format("Hello {name}", name="reza"))

# output: Hello REZA
```

در این مثال، کلاس UpperFormatter با ارث‌بری از کلاس Formatter ساخته شده و سپس متد format_field آن به نحوی بازنویسی شده که تمام حروف placeholder را به صورت بزرگ بنویسد.

مثال واقعی از کاربرد ماژول

در یک سناریوی فرضی، شما سیستمی برای تولید ایمیل‌های اطلاع‌رسانی می‌نویسید. قالب ایمیل‌ها از یک فایل پیکربندی می‌آید (و نباید از f-string استفاده شود)، رمزهای یکبار مصرف تولید می‌شود و خروجی باید به صورت جدول تراز شده باشد.

در این مثال از کلاس Template ماژول string پایتون برای قالب ایمیل ایمن، از ثابت string.digits برای تولید کد یک‌بار مصرف (OTP) و از ترکیب ثوابت ascii_uppercase و digits برای تولید کد مرجع استفاده شده و نمایش نهایی، فرمت‌بندی و تراز شده است.

کدهای این مثال به همراه خروجی نهایی را می‌توانید از اینجا دانلود کنید.

سوالات متداول

1. تفاوت ماژول `string` با نوع داده `str` چیست؟

`str` یک نوع داده پیش‌ساخته پایتون و همیشه در دسترس است. ماژول `string` یک ماژول از کتابخانه استاندارد پایتون است که باید با ایمپورت شود و شامل ثابت‌های آماده، کلاس `Template` و کلاس `Formatter` است. این دو مکمل یکدیگرند.

---

2. چه زمانی از `Template` به جای `f-string` استفاده کنیم؟

کلاس `Template` را انتخاب کنید وقتی: (۱) قالب متن از خارج برنامه می‌آید مثل فایل، پایگاه داده، ورودی کاربر، (۲) می‌خواهید از آسیب‌پذیری‌های امنیتی جلوگیری کنید، (۳) قالب‌های بین‌المللی‌سازی (`i18n`) می‌سازید. اما اگر قالب داخل کد خودتان است و به سادگی و سرعت نیاز دارید `f-string` را انتخاب کنید.

---

3. تفاوت `substitute` و `safe_substitute` چیست؟

متد `substitute` اگر کلیدی در قالب وجود داشته باشد ولی مقدارش تأمین نشده باشد، خطای `KeyError` می‌دهد اما `safe_substitute` در این حالت آن `placeholder` را دست‌نخورده رها می‌کند و خطایی نمی‌دهد. وقتی مطمئن نیستید همه‌ی کلیدها ارائه می‌شوند، `safe_substitute` ایمن‌تر است.

---

4. آیا ثابت‌های ماژول `string` وابسته به تنظیمات زبان سیستم هستند؟

خیر. تمام ثابت‌های ماژول `string` کاملاً مستقل از تنظیمات زبان سیستم (`locale`) هستند و همیشه همان مقادیر ثابت را دارند.

جهت کسب اطلاعات بیشتر می‌توانید به **مستندات رسمی پایتون برای ماژول `string`** مراجعه کنید.

دسته: کتابخانه استاندارد - پایتون - دانش‌نامه برنامه‌نویسی