تابع str

صفحه اصلی / دانش‌نامه / پایتون / توابع پیش‌ساخته

آخرین به‌روزرسانی: ۲۶ اردیبهشت ۱۴۰۵

تابع `str` در پایتون یک تابع پیش‌ساخته (`built-in`) است که یک آبجکت را به استرینگ تبدیل می‌کند. این تابع یکی از پایه‌ای‌ترین توابع پایتون است و در طیف گسترده‌ای از کاربردها، از نمایش داده‌ها گرفته تا رمزگشایی (`decoding`) داده‌های باینری، مورد استفاده قرار می‌گیرد.

فهرست مطالب:

سینتکس تابع str

سینتکس کلی تابع `str` به دو صورت زیر است:

```
str(object)
str(object, encoding, errors="strict")
```

پارامترها یا آرگومان‌های این سینتکس عبارتند از:

- آرگومان `object` آبجکتی است که می‌خواهیم به استرینگ تبدیل شود. می‌تواند هر نوع داده‌ای باشد: عدد صحیح، اعشاری، لیست، دیکشنری، کلاس دلخواه و غیره. اگر آرگومانی وارد نشود، استرینگ خالی "" برگردانده می‌شود.

- آرگومان `encoding` (فقط در سینتکس دوم) نام استاندارد کدگذاری کاراکتر را مشخص می‌کند؛ مانند `utf-8`، `ascii` یا `latin-1`. این آرگومان فقط زمانی معتبر است که `object` یک آبجکت از نوع `bytes` یا `bytearray` باشد.
- آرگومان اختیاری `errors` (فقط در سینتکس دوم) نحوه مدیریت خطاهای رمزگشایی را تعیین می‌کند. مقدار پیش‌فرض آن `strict` است که در صورت بروز خطا یک استثنا ایجاد می‌کند. مقادیر دیگر شامل `ignore` (نادیده گرفتن خطا) و `replace` (جایگزین کردن کاراکتر ناشناخته با ?) هستند.

مقدار بازگشتی تابع `str` همیشه یک رشته متنی (استرینگ) از نوع `str` است.

راهنمای جامع استرینگ (str)

در مورد نوع داده استرینگ یا همان `str` در پایتون بیشتر بدانید: **استرینگ (str)**

کاربردهای تابع str

ساده‌ترین مثال از کاربرد این تابع می‌تواند حالت زیر باشد:

```
>>> str(42)
'42'
>>> str(3.14)
'3.14'
>>> str(True)
'True'
>>> str()
''
```

یا می‌توان داده‌های باینری را با مشخص کردن `encoding` به رشته متنی تبدیل کرد:

```
>>> str(b"سلام", encoding="utf-8")
'سلام'
>>> str(b"Python", encoding="ascii")
'Python'
>>> str(b"Python")
"b'Python'"
```

توجه کنید که پاس دادن آبجکت bytes بدون آرگومان encoding، داده را رمزگشایی نمی‌کند و فقط نمایش متنی آبجکت bytes را برمی‌گرداند.

رایج‌ترین کاربردهای تابع str عبارتند از:

- تبدیل اعداد و سایر انواع داده به رشته متنی برای الحاق (concatenation) یا نمایش،
- رمزگشایی داده‌های باینری دریافتی از فایل یا شبکه،
- دریافت نمایش متنی آبجکت‌های دلخواه از طریق متد جادویی `__str__`،
- اعتبارسنجی و آماده‌سازی داده‌ها برای خروجی به کاربر.

رمزگشایی bytes با encoding: یکی از کاربردهای مهم تابع str در پردازش داده‌های باینری است. وقتی داده‌ای از یک فایل، سوکت شبکه یا API دریافت می‌کنید، اغلب به صورت bytes است و برای تبدیل آن به رشته قابل خواندن به encoding نیاز دارید:

```
# encoding with utf-8
>>> data = b"\xd8\xb3\xd9\x84\xd8\xa7\xd9\x85"
>>> str(data, encoding="utf-8")
'سلام'
```

در این مثال، خروجی تابع str مشابه وقتی است که متد deocode را روی data اثر بدهیم:

```
>>> data = b"\xd8\xb3\xd9\x84\xd8\xa7\xd9\x85"
>>> data.decode("utf-8")
'سلام'
```

مدیریت خطا با پارامتر errors انجام می‌شود:

```
>>> bad_data = b"\xff\xfeسلام"

# strict mode (default)
>>> str(bad_data, encoding="ascii", errors="strict")
UnicodeDecodeError: ...

# ignore mode
>>> str(bad_data, encoding="ascii", errors="ignore")
''

# replace mode
>>> str(bad_data, encoding="ascii", errors="replace")
'??سلام'
```

سفارشی‌سازی خروجی تابع str: وقتی تابع str روی یک آبجکت دلخواه فراخوانی می‌شود، پایتون متد `__str__` آن کلاس را صدا می‌زند. اگر کلاس این متد را تعریف نکرده باشد، تابع str به تابع repr بازگشت می‌کند. با تعریف `__str__` می‌توانید نمایش دلخواه آبجکت خود را کنترل کنید:

```
class Product:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def __str__(self):
        return f"{self.name} - {self.price:,} Rials"

laptop = Product("Laptop", 45000000)
print(str(laptop))

# output: Laptop - 45,000,000 Rials
```

در کد بالا، اگر متد جادویی `__str__` را تعریف نمی‌کردید، خروجی مشابه زیر می‌شد:

```
# output:
<__main__.Product object at 0x000001FEC5306300>
```

مثال واقعی از تابع str

در یک سناریوی فرضی، شما یک سیستم گزارش‌گیری دارید که اطلاعات کاربران را از یک پایگاه داده دریافت کرده و آن‌ها را به صورت یک فایل متنی ذخیره می‌کند. داده‌ها ممکن است شامل انواع مختلف باشند و باید همه آن‌ها به رشته متنی تبدیل شوند:

```
users = [
    {"name": "Ali Rezaei", "age": 28, "score": 94.5, "active": True},
    {"name": "Sara Mohamadi", "age": 35, "score": 87.0, "active": False},
    {"name": "Reza Karimi", "age": 22, "score": 76.3, "active": True},
]

lines = []
for user in users:
    line = (
        "name: " + user["name"] + " | " +
        "age: " + str(user["age"]) + " | " +
        "score: " + str(user["score"]) + " | " +
        "status: " + str(user["active"])
    )
    lines.append(line)

report = "\n".join(lines)
print(report)
```

نمونه خروجی این کد به صورت زیر خواهد بود:

```
name: Ali Rezaei | age: 28 | score: 94.5 | status: True
name: Sara Mohamadi | age: 35 | score: 87.0 | status: False
name: Reza Karimi | age: 22 | score: 76.3 | status: True
```

در این مثال، تابع پایتونی `str` مقادیر عددی و بولین را به رشته متنی تبدیل می‌کند تا بتوان آن‌ها را با عملگر + به هم الحاق کرد؛ بدون این تبدیل، پایتون خطای `TypeError` می‌داد.

تفاوت تابع `str` با `repr` و `f-string`

هر سه روش خروجی استرینگ تولید می‌کنند، اما هدف و رفتار آن‌ها متفاوت است:

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f"({self.x}, {self.y})"

    def __repr__(self):
        return f"Point(x={self.x}, y={self.y})"

p = Point(3, 7)

print(str(p))    # output: (3, 7)
print(repr(p))   # output: Point(x=3, y=7)
print(f"{p}")    # output: (3, 7)
print(f"{p!r}")  # output: Point(x=3, y=7)
```

به طور کلی:

- تابع `str` نمایش خوانا و دوستانه برای کاربر نهایی تولید می‌کند و از متد `__str__` استفاده می‌کند.
- تابع `repr` نمایش دقیق و اشکال‌زدایی‌پذیر تولید می‌کند که ایده‌آل برای توسعه‌دهنده است و از متد `__repr__` استفاده می‌کند.
- `f-string` به صورت پیش‌فرض از `__str__` استفاده می‌کند، اما با `!r` می‌توان به `__repr__` سوئیچ کرد.

سوالات متداول

1. تفاوت تابع `str` و تابع `repr` در پایتون چیست؟

هر دو تابع یک رشته متنی تولید می‌کنند، اما `str` نمایش خوانا و دوستانه برای کاربر نهایی ارائه می‌دهد، در حالی که `repr` نمایش دقیق و بازتولیدپذیر برای اشکال‌زدایی تولید می‌کند. اگر یک کلاس، متد `__str__` نداشته باشد، تابع `str` به طور خودکار به تابع `repr` بازگشت می‌کند.

---

2. آیا تابع `str` و `f-string` یکسان هستند؟

هر دو از متد `__str__` آبجکت استفاده می‌کنند؛ اما `f-string` علاوه بر این، امکان قالب‌بندی پیشرفته با `format_spec` را هم فراهم می‌کند. `f-string` در اکثر موارد خواناتر است، اما `str` زمانی مفیدتر است که بخواهید تبدیل را به صورت تابعی یا در داخل عبارات دیگر انجام دهید.

---

3. اگر `str` را بدون آرگومان فراخوانی کنم چه اتفاقی می‌افتد؟

تابع `str` بدون آرگومان یک استرینگ خالی "" برمی‌گرداند. این رفتار مشابه نوشتن "" در کد است.

---

4. آیا تابع `str` برای الحاق استرینگ‌ها بهتر است یا `f-string`؟

برای الحاق چند مقدار به هم، `f-string` خواناتر و معمولاً سریع‌تر است. اما تابع `str` زمانی کاربرد دارد که بخواهید یک مقدار منفرد را به صراحت به رشته متنی تبدیل کنید یا آن را به عنوان آرگومان به تابع دیگری بدهید.

جهت کسب اطلاعات بیشتر می‌توانید به **مستندات رسمی پایتون برای تابع `str`** مراجعه کنید.

دسته: توابع پیش‌ساخته - پایتون - دانش‌نامه برنامه‌نویسی