تابع bool

آخرین به‌روزرسانی: ۰۱ خرداد ۱۴۰۵

تابع bool در پایتون یک تابع پیش‌ساخته (built-in) است که مقدار درستی هر آبجکت پایتون را ارزیابی می‌کند و یکی از دو مقدار True یا False را برمی‌گرداند. این تابع از روال استاندارد آزمون درستی پایتون استفاده می‌کند تا تعیین کند آیا یک آبجکت در یک بافت بولین معادل درست است یا نادرست. از نظر فنی، bool زیرکلاسی از int است و نمی‌توان از آن زیرکلاس دیگری ساخت.

فهرست مطالب:

سینتکس تابع bool
قوانین ارزیابی درستی در پایتون
کاربردهای تابع bool
مثال واقعی از تابع bool
تفاوت تابع bool با استفاده ضمنی در شرط‌ها
سوالات متداول

سینتکس تابع bool

سینتکس کلی تابع bool به صورت زیر است:

```
bool(object=False)
```

پارامترها یا آرگومان‌های این سینتکس عبارتند از:

- آرگومان اختیاری object می‌تواند هر آبجکت پایتونی باشد که می‌خواهیم مقدار درستی آن را ارزیابی کنیم. این آرگومان فقط موقعیتی است و نمی‌توان آن را به صورت کلیدواژه‌ای پاس داد (از پایتون 3.7 به بعد). اگر آرگومانی وارد نشود، مقدار پیش‌فرض False برگردانده می‌شود.

مقدار بازگشتی تابع `bool` همیشه یکی از دو مقدار `True` یا `False` است.

راهنمای جامع بولین (`bool`)

در مورد نوع داده بولین یا همان `bool` در پایتون بیشتر بدانید: **بولین (`bool`)**

قوانین ارزیابی درستی در پایتون

تابع `bool` بر اساس روال استاندارد آزمون درستی (`truth testing procedure`) پایتون عمل می‌کند. پایتون هر آبجکتی را `True` می‌داند مگر آنکه یکی از مقادیر `falsy` باشد. این مقادیر عبارتند از:

- ثابت‌های `False` و `None`
- صفر در هر نوع عددی: `0`، `0.0`، `0j`
- دنباله‌ها و مجموعه‌های خالی: `""`، `[]`، `()`، `{}`، `range(0)`، `set()`
- آبجکت‌هایی که متد `__bool__` آن‌ها `False` یا متد `__len__` آن‌ها 0 برمی‌گرداند

تمام مقادیر دیگر، از نظر پایتون `True` هستند.

کاربردهای تابع `bool`

ساده‌ترین مثال از کاربرد این تابع می‌تواند حالت زیر باشد:

```
>>> bool()
False
>>> bool(1)
True
>>> bool("")
False
>>> bool(" ")
True
>>> bool("0")
True
>>> bool([0, False])
True
>>> bool(())
False
>>> bool({"key": 0})
True
>>> bool(None)
False
```

نکته مهم: استرینگ "0" با عدد 0 فرق دارد. `bool("0")` مقدار `True` برمی‌گرداند چون یک استرینگ غیرخالی است، در حالی که `bool(0)` مقدار `False` برمی‌گرداند.

رایج‌ترین کاربردهای تابع `bool` عبارتند از:

- تبدیل صریح یک مقدار به بولین برای استفاده در شرط‌ها و دستورات کنترلی،
- بررسی اینکه آیا یک متغیر مقدار معنادار دارد یا خیر،
- اعتبارسنجی ورودی‌ها و مقادیر بازگشتی توابع،
- سفارشی‌سازی رفتار بولین کلاس‌های دلخواه از طریق متد جادویی `__bool__`.

سفارشی‌سازی رفتار `bool` با متد `__bool__`: وقتی تابع `bool` روی یک آبجکت از کلاس دلخواه فراخوانی می‌شود، پایتون ابتدا متد `__bool__` و در صورت نبود آن متد `__len__` را صدا می‌زند. اگر هیچ‌کدام تعریف نشده باشند، آبجکت به صورت پیش‌فرض `True` ارزیابی می‌شود:

```python
class ShoppingCart:
    def __init__(self, items):
        self.items = items

    def __bool__(self):
        return len(self.items) > 0

    def __len__(self):
        return len(self.items)

empty_cart = ShoppingCart([])
full_cart = ShoppingCart(["Laptop", "Keyboard"])

print(bool(empty_cart))  # output: False
print(bool(full_cart))   # output: True

if full_cart:
    print("Your card is NOT empty.")
# output: Your card is NOT empty.
```

در شرط `if` پایتون به طور ضمنی تابع `bool` مربوط به آبجکت `full_cart` را فراخوانی می‌کند.

مثال واقعی از تابع `bool`

در یک سناریوی فرضی، شما یک سیستم ثبت‌نام دارید و می‌خواهید اعتبارسنجی کنید که فیلدهای ضروری فرم پر شده‌اند. تابع `bool` به شما اجازه می‌دهد این بررسی را به صورت مختصر و خوانا انجام دهید:

```python
def validate_registration(form_data):
    required_fields = ["username", "email", "password"]
    errors = []

    for field in required_fields:
        value = form_data.get(field, "")
        if not bool(value): # check for required field
            errors.append(f"Field '{field}' is required!")

    # check for password length
    password = form_data.get("password", "")
    if bool(password) and len(password) < 8:
        errors.append("Password must be at least 8 characters!")

    return {"is_valid": not bool(errors), "errors": errors}

form1 = {"username": "ali_r", "email": "ali@example.com", "password": ""}
form2 = {"username": "reza", "email": "reza@example.com", "password": "secure123"}

result1 = validate_registration(form1)
result2 = validate_registration(form2)

print(result1)
print(result2)
```

نمونه خروجی این کد به صورت زیر خواهد بود:

```
{'is_valid': False, 'errors': ["Field 'password' is required!"]}
{'is_valid': True, 'errors': []}
```

در این مثال، `bool(value)` بررسی می‌کند که آیا فیلد پر شده (استرینگ غیرخالی) است یا نه و `bool(errors)` تعیین می‌کند که آیا خطایی وجود دارد یا خیر.

تفاوت تابع bool با استفاده ضمنی در شرط‌ها

در اکثر موارد نیازی به فراخوانی صریح تابع bool نیست چون پایتون آن را به صورت ضمنی در دستورات if و while و عملگرهای منطقی اعمال می‌کند:

```
name = "Ali"

# روش ضمنی – پایتونیک‌تر و رایج‌تر
if name:
    print("Name has value.")

# روش صریح – نتیجه یکسان اما کمتر رایج
if bool(name):
    print("Name has value.")
```

فراخوانی صریح تابع bool زمانی ارزش دارد که:

- بخواهید مقدار بولین را در یک متغیر ذخیره کنید: is_valid = bool(value)
- بخواهید مقدار درستی را برای استفاده در محاسبات عددی به 0 یا 1 تبدیل کنید
- بخواهید در یک list comprehension یا پایپ‌لاین تابعی مقادیر truthy را استخراج کنید:

```
values = [0, 1, "", "Hi", None, True, [], [1, 2]]
truthy_flags = list(map(bool, values))
print(truthy_flags)
# output: [False, True, False, True, False, True, False, True]
```

سوالات متداول

1. تفاوت استفاده مستقیم و ضمنی از بولین در شرط `if` چیست؟

از نظر نتیجه یکسانند؛ `if x` معادل `if bool(x)` است. اما تابع `bool` زمانی مفید است که بخواهید مقدار بولین را صریحاً در یک متغیر ذخیره کنید یا در محاسبات عددی استفاده کنید یا به عنوان تابع به `map` بدهید.

---

2. آیا می‌توان رفتار تابع `bool` را برای کلاس‌های دلخواه تغییر داد؟

بله، با تعریف متد `__bool__` در کلاس می‌توانید مشخص کنید که آبجکت‌های آن کلاس در بافت بولین چه مقداری داشته باشند. اگر متد `__bool__` تعریف نشده باشد، پایتون از `__len__` استفاده می‌کند؛ آبجکت‌هایی با طول صفر `falsy` و بقیه `truthy` محسوب می‌شوند.

---

3. چرا `bool` زیرکلاس `int` است؟

این طراحی عمدی است و باعث می‌شود `True` و `False` در محاسبات عددی مستقیماً قابل استفاده باشند (`True == 1` و `False == 0`). این ویژگی سازگاری با کدهای قدیمی‌تر را نیز حفظ می‌کند و الگوهایی مثل شمارش مقادیر `truthy` را ممکن می‌سازد.

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

دسته: توابع پیش‌ساخته - پایتون - دانش‌نامه برنامه‌نویسی