معرفی `pytest` و کاربرد آن

`pytest` یک چارچوب تست قدرتمند و ساده برای پایتون است که به‌طور گسترده در صنعت استفاده می‌شود. این ابزار به شما امکان می‌دهد که تست‌های واحد خود را به راحتی بنویسید و اجرا کنید. برخلاف `unittest` که به صورت پیش‌فرض برای نوشتن تست‌ها به کلاس‌های خاص و متدهای از پیش تعریف‌شده نیاز دارد، `pytest` بسیار ساده‌تر است و به‌طور خودکار تست‌ها را شناسایی و اجرا می‌کند.

---

1. تفاوت‌های `pytest` با `unittest` و مزایای استفاده از آن

- **ساده‌تر بودن**: در `pytest` نیازی به نوشتن کلاس یا متدهای خاص برای تست‌ها نیست. فقط کافی است که نام تابع تست را با `_test` شروع کنید.
- **پشتیبانی از assert**: در `pytest` می‌توانید مستقیماً از دستور `assert` برای بررسی نتایج استفاده کنید، که خوانایی کد را افزایش می‌دهد.
- **گزارش‌گیری بهتر**: `pytest` به‌طور پیش‌فرض گزارش‌های دقیق و خوانا از تست‌ها ارائه می‌دهد.
- **پشتیبانی از Fixtureها**: در `pytest` می‌توانید از **fixtures** برای راه‌اندازی و پاک‌سازی داده‌های تست استفاده کنید.
- **پشتیبانی از تست‌های همزمان**: `pytest` از کتابخانه‌هایی مانند `pytest-asyncio` برای انجام تست‌های غیرهمزمان پشتیبانی می‌کند.
- **گسترش‌پذیری**: `pytest` از پلاگین‌ها برای گسترش قابلیت‌های خود پشتیبانی می‌کند.

---

2. نصب و راه‌اندازی `pytest` در پروژه

برای نصب `pytest` در پروژه، کافی است که از pip استفاده کنید:

```
pip install pytest
```

بعد از نصب، می‌توانید به راحتی تست‌ها را بنویسید و اجرا کنید.

---

3. نوشتن تست‌های ساده با `pytest` و اجرا کردن آنها

در `pytest` برای نوشتن تست‌ها تنها کافی است که نام تابع تست را با `_test` شروع کنید. سپس از دستور `assert` برای مقایسه نتایج استفاده کنید.

نمونه ساده از تست با `pytest`

```python
# math_operations.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
# test_math_operations.py
from math_operations import add, subtract

def test_add():
    assert add(2, 3) == 5
```

```
def test_subtract():
    assert subtract(5, 3) == 2
```

در اینجا:

- `test_add` و `test_subtract` تست‌های ساده‌ای هستند که از دستور `assert` برای بررسی نتایج استفاده می‌کنند.
- این تست‌ها با نام‌گذاری صحیح توابع (شروع با `_test`) توسط `pytest` شناسایی می‌شوند.

اجرای تست‌ها با `pytest`

برای اجرای تست‌ها، فقط کافی است که دستور زیر را در ترمینال وارد کنید:

```
pytest
```

این دستور به‌طور خودکار تمام فایل‌هایی که نام آنها با `_test` شروع می‌شود را شناسایی کرده و تست‌ها را اجرا می‌کند.

---

4. استفاده از `assert` در `pytest` برای بررسی نتایج

در `pytest`، برای بررسی نتایج، به‌جای استفاده از متدهای خاص مانند `()assertEqual` یا `()assertTrue`، به سادگی از دستور `assert` استفاده می‌کنیم. این دستور می‌تواند هر شرطی را بررسی کند و در صورت نادرست بودن، یک استثنا ایجاد خواهد شد.

مثال استفاده از `assert` در `pytest`

```
def test_addition():
    result = 2 + 3
    assert result == 5  # بررسی اینکه نتیجه برابر 5 است
```

در اینجا:

- اگر `result` برابر با 5 نباشد، `pytest` خطا خواهد داد و پیامی مبنی بر شکست تست نشان خواهد داد.

مزایای استفاده از `pytest`

- **سادگی و خوانایی**: به‌سادگی می‌توانید تست‌های خود را بنویسید و از `assert` برای بررسی نتایج استفاده کنید.
- **گزارش‌دهی و دیباگ بهتر**: `pytest` به شما گزارشی دقیق از وضعیت تست‌ها می‌دهد و به شما کمک می‌کند تا سریع‌تر مشکلات را شناسایی کنید.
- **پشتیبانی از ویژگی‌های پیشرفته**: از جمله پشتیبانی از fixtures، پلاگین‌ها، و تست‌های همزمان.

در مجموع، `pytest` ابزار بسیار قدرتمند و ساده‌ای است که برای تست‌نویسی در پایتون مورد استفاده قرار می‌گیرد و بسیاری از مشکلات و محدودیت‌های موجود در `unittest` را برطرف می‌کند.