

Atividade 1 – Testes Unitários e o Stack Overflow
Aluno: André Lucas Ávila Lima

Relatório sobre o problema escolhido no stackoverflow

Título:PATH issue with pytest 'ImportError: No module named ...'

Link:<https://stackoverflow.com/questions/10253826/path-issue-with-pytest-importerror-no-module-named>

Link do repositório Github:

https://github.com/fre200tey623/Teste_Software_2024_Lima_Andre

Configurando o ambiente

Abra o terminal e utilize o comando:

```
pip install pytest
```

Posteriormente verifique a instalação para garantir que o Pytest foi instalado corretamente

```
pytest --version
```

Problema apresentado



I used *easy_install* to install [pytest](#) on a Mac and started writing tests for a project with a file structure likes so:

428



```
repo/  
|--app.py  
|--settings.py  
|--models.py  
|--tests/  
    |--test_app.py
```

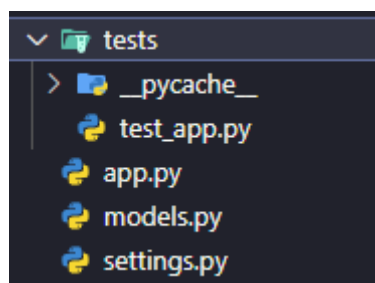
Run `py.test` while in the *repo* directory, and everything behaves as you would expect.

But when I try that same thing on either Linux or Windows (both have pytest 2.2.3 on them), it barks whenever it hits its first import of something from my application path. For instance, `from app import some_def_in_app`.

Do I need to be editing my [PATH](#) to run *py.test* on these systems?

`python` `unit-testing` `pytest`

Para ilustrar esse problema, criei um projeto seguindo a estrutura apresentada, com um código de exemplo:



Conteúdo de cada arquivo:

test_app.py:

A primeira função, `test_create_user`, é testada para garantir que ela cria um usuário com o nome e a idade corretos e o adiciona a uma lista que simula um banco de dados. A segunda função, `test_get_user`, é testada para verificar se ela pode encontrar e retornar um usuário pelo nome. Um teste adicional verifica se a função `test_get_user_not_found` retorna `None` quando tenta buscar um usuário que não existe.

```
tests > test_app.py > ...
1  import pytest
2  from app import create_user, get_user
3  from settings import DATABASE
4
5  def setup_function():
6      DATABASE.clear()
7
8  def test_create_user():
9      user = create_user("Alice", 30)
10     assert user.name == "Alice"
11     assert user.age == 30
12     assert len(DATABASE) == 1
13
14  def test_get_user():
15     create_user("Bob", 25)
16     user = get_user("Bob")
17     assert user is not None
18     assert user.name == "Bob"
19     assert user.age == 25
20
21  def test_get_user_not_found():
22     user = get_user("Charlie")
23     assert user is None
24
```

`app.py`:

A função `create_user` cria um novo usuário com um nome e idade fornecidos, e em seguida adiciona esse usuário a uma lista que simula um banco de dados. A função `get_user` busca nessa lista por um usuário com o nome especificado e o retorna, se encontrado. Se o usuário não for encontrado, a função retorna `None`.

```

app.py > get_user
1  from models import User
2  from settings import DATABASE
3
4  def create_user(name, age):
5      user = User(name=name, age=age)
6      DATABASE.append(user)
7      return user
8
9  def get_user(name):
10     for user in DATABASE:
11         if user.name == name:
12             return user
13     return None
14

```

models.py:

A classe possui um `__init__`, que é um construtor utilizado para inicializar os atributos de instância, `name` e `age`, sempre que um novo objeto `User` é criado. A classe também inclui o método `__repr__`, que retorna uma representação em string do objeto `User`.

```

models.py > ...
1  class User:
2
3      def __init__(self, name, age):
4          self.name = name
5          self.age = age
6
7      def __repr__(self):
8          return f"User(name={self.name}, age={self.age})"
9

```

settings.py:

Responsável por simular um banco de dados em memória para armazenar objetos de usuário no aplicativo. À medida que novos usuários são criados usando a função `create_user`, eles são adicionados a essa lista.

```

settings.py > DATABASE
1  DATABASE = []

```

Ao executar o código utilizando o comando pytest no terminal, obtive o seguinte erro, semente ao também encontrado pela pessoa que escreveu esse tópico no stackoverflow:






```
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.3.2, pluggy-1.5.0
rootdir: G:\Área de Trabalho\python teste\repo
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_app.py
ImportError while importing test module 'G:\Área de Trabalho\python teste\repo\tests\test_app.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\Andre Lima\AppData\Local\Programs\Python\Python311\Lib\importlib\__init__.py:126: in import_module
Traceback:
Traceback:
Traceback:
Traceback:
Traceback:
Traceback:
C:\Users\Andre Lima\AppData\Local\Programs\Python\Python311\Lib\importlib\__init__.py:126: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
tests\test_app.py:4: in <module>
    from app import create_user, get_user
E   ModuleNotFoundError: No module named 'app'

===== short test summary info =====
ERROR tests/test_app.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
===== 1 error in 0.14s =====
```

Até o momento de escrita deste relatório, no stackoverflow foram encontradas várias respostas sobre esse problema, as quais irei simular algumas e capturar os resultados obtidos:

1. Resposta com 483 votos e com verificado


483






I'm not sure why py.test does not add the current directory in the PYTHONPATH itself, but here's a workaround (to be executed from the root of your repository):

```
python -m pytest tests/
```

It works because Python adds the current directory in the PYTHONPATH for you.

Share Follow

answered Dec 7, 2015 at 18:21

 **Apteryx**
6,273 ● 3 ● 18 ● 19

Ao executar a solução no meu terminal obtive ótimos resultados com todos passando pelos testes e retornando um resultado positivo:


```
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.3.2, pluggy-1.5.0
rootdir: G:\Área de Trabalho\python teste\repo
collected 3 items

tests\test_app.py ... [100%]

===== 3 passed in 0.03s =====
```

Obs.: Ao pesquisar sobre obtive a seguinte justificativa: A execução de pytest como um módulo com o comando `python -m pytest tests/` funciona porque, ao usar a opção `-m`, o Python adiciona automaticamente o diretório atual ao `PYTHONPATH`. Isso permite que o Python trate as importações como absolutas, localizando corretamente módulos e pacotes a partir da raiz do projeto. Como resultado, evita problemas de importação que podem ocorrer quando pytest é executado diretamente, pois esse método garante que o caminho de importação inclua o diretório do projeto, facilitando o acesso a todos os módulos necessários para os testes.

2. Resposta com 487 votos


487

Recommended approach for `pytest>=7`: use the `pythonpath` setting

Recently, `pytest` has added a new core plugin that supports `sys.path` modifications via the [pythonpath configuration value](#). The solution is thus much simpler now and doesn't require any workarounds anymore:

`pyproject.toml` example:

```
[tool.pytest.ini_options]
pythonpath = [
    ".",
]
```

`pytest.ini` example:

```
[pytest]
pythonpath = .
```

Ao executar esse teste eu usei o comando `pytest` no meu terminal obtive ótimos resultados com todos passando pelos testes e retornando um resultado positivo:





```
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.3.2, pluggy-1.5.0
rootdir: G:\Área de Trabalho\python teste\repo
configfile: pytest.ini
collected 3 items

tests\test_app.py ... [100%]

===== 3 passed in 0.01s =====
```

Obs.: Ao pesquisar sobre obtive a seguinte justificativa: “Com a versão 7 ou superior do `pytest`, foi introduzido um novo plugin que permite modificar o `sys.path` por meio da configuração `pythonpath`, tornando a gestão de caminhos de importação mais simples e eliminando a necessidade de soluções alternativas. Agora, você pode especificar as entradas de caminho diretamente nos arquivos de configuração `pyproject.toml` ou `pytest.ini`. Além disso, múltiplas entradas de caminho são suportadas, permitindo configurações como `[tool.pytest.ini_options] pythonpath = [".", "src"]`, que incluem tanto o diretório do projeto quanto o diretório `src` no `sys.path`.

3. Resposta com 159 votos




I had the same problem. I fixed it by adding an empty `__init__.py` file to my `tests` directory.

159 Share Follow

edited Jul 10, 2014 at 1:38

answered Sep 24, 2013 at 1:20

**Aron Curzon**
2,634 ● 2 ● 21 ● 16

Ao executar a solução no meu terminal obtive ótimos resultados com todos passando pelos testes e retornando um resultado positivo:





```
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.3.2, pluggy-1.5.0
rootdir: G:\Área de Trabalho\python teste\repo
collected 3 items

tests\test_app.py ... [100%]

===== 3 passed in 0.01s =====
```

Porém depois dessa postagem um usuário comentou a seguinte afirmação afirmando que não seria uma boa prática, segue a mensagem traduzida para o português: “Observe que isso NÃO é recomendado pelo py.test: evite arquivos “`__init__.py`” em seus diretórios de teste. Desta forma, seus testes podem ser executados facilmente em uma versão instalada do mypkg, independentemente do pacote instalado, se ele contém os testes ou não.”

4. Resposta com 66 votos




Run `pytest` itself as a module with: `python -m pytest tests`


66 This happens when the project hierarchy is, for example, `package/src package/tests` and in tests you import from `src`. Executing as a module will consider imports as absolute rather than relative to the execution location.

Share Follow

edited Jan 21, 2023 at 16:31

answered Feb 2, 2018 at 9:37

**Peter Mortensen**
31.4k ● 22 ● 109 ● 132

**Stefano Messina**
2,024 ● 1 ● 19 ● 23

Para a execucao desse teste eu coloquei o app.py dentro da pasta src como descrito e escutei o comando “`python -m pytest tests`”, obtendo resultado positivos para os testes

```
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.3.2, pluggy-1.5.0
rootdir: G:\Área de Trabalho\python teste\repo
collected 3 items

tests\test_app.py ... [100%]

===== 3 passed in 0.03s =====
```

Obs.: Ao pesquisar sobre, encontrei a seguinte justificativa: Ao executar pytest como um módulo, utilizando o comando `python -m pytest`, o Python trata as importações como absolutas, utilizando a hierarquia de diretórios do projeto em vez da localização de execução do script. Isso possibilita ao Python encontrar módulos como `src.app` de maneira precisa, evitando equívocos na importação. Os módulos são acessados da raiz do projeto, garantindo que as dependências sejam resolvidas de forma correta, especialmente em projetos que usam uma estrutura de diretórios com subdiretórios, como `src` para o código-fonte.