



Framework de Pruebas para Python

¿Qué es pytest?

- Es un framework para Python que ofrece la recolección automática de los tests, aserciones simples, soporte para fixtures, debuggeo y mucho más.
- También está nose, doctest, testify.

Porque pytest

- Una prueba es código que ejecuta código. Cuando empiezas a desarrollar una nueva característica para tu proyecto Python, podrías formalizar sus requisitos como código. Cuando lo haces, no sólo documentas la forma en que se utilizará el código de tu implementación, sino que también puedes ejecutar todas las pruebas automáticamente para asegurarte siempre de que tu código se ajusta a tus requisitos.

Ejemplo 1

- Supongamos que ha escrito una función que valida una dirección de correo electrónico. Sin utilizar Expresiones Regulares o pruebas de DNS para validar las direcciones de correo electrónico. Sólo nos aseguramos de que haya exactamente un signo @ en la cadena a comprobar.

Ejemplo 1

Casos de Prueba

- Válidos:
 - test@example.org
 - user123@subdomain.example.org
 - john.doe@email.example.org
- No válidos
 - not valid@example.org contains space
 - john.doe missing @ sign
 - john,doe@example.org contains comma

Ejemplo 1

Creamos y activamos un entorno virtual

```
[kralos]--[!main ≡]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• ▶ python -m virtualenv venvtest
created virtual environment CPython3.12.1.final.0-64 in 1349ms
creator CPython3Windows(dest=D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0\venvtest, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\kralos\AppData\Local\pypa\virtualenv)
added seed packages: pip==23.3.1
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
[kralos]--[!main ≡]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• ▶ .\venvtest\Scripts\activate
o (venvtest) r[kralos]--[!main ≡]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
▶
```

Ejemplo 1

```
• .venvtest\Scripts\activate
(venvtest) r[kralos]--[p/main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• pip list
Package Version
-----
pip      23.3.1

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venvtest) r[kralos]--[p/main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• pip install pytest
Collecting pytest
  Downloading pytest-7.4.4-py3-none-any.whl.metadata (7.9 kB)
Collecting iniconfig (from pytest)
  Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)
Collecting packaging (from pytest)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pluggy<2.0,>=0.12 (from pytest)
  Downloading pluggy-1.3.0-py3-none-any.whl.metadata (4.3 kB)
Collecting colorama (from pytest)
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading pytest-7.4.4-py3-none-any.whl (325 kB)
  325.3/325.3 kB 2.3 MB/s eta 0:00:00
Downloading pluggy-1.3.0-py3-none-any.whl (18 kB)
Downloading packaging-23.2-py3-none-any.whl (53 kB)
  53.0/53.0 kB ? eta 0:00:00
Installing collected packages: pluggy, packaging, iniconfig, colorama, pytest
Successfully installed colorama-0.4.6 iniconfig-2.0.0 packaging-23.2 pluggy-1.3.0 pytest-7.4.4

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venvtest) r[kralos]--[p/main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
(venvtest) r[kralos]--[p/main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• pip list
Package Version
-----
colorama 0.4.6
iniconfig 2.0.0
packaging 23.2
pip      23.3.1
pluggy  1.3.0
pytest   7.4.4

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venvtest) r[kralos]--[p/main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
•
```

Ejemplo 1

```
code > pytest > example_0 > src > main.py > is_valid_email_address
1 import string
2
3 def is_valid_email_address(s):
4     s = s.lower()
5     parts = s.split('@')
6     if len(parts) != 2:
7         # Not exactly one at-sign
8         return False
9     allowed = set(string.ascii_lowercase + string.digits + '._-')
10    for part in parts:
11        if not set(part) <= allowed:
12            # Characters other than the allowed ones are found
13            return False
14    return True

code > pytest > example_0 > src > test > test_main.py > test_valid_email_has_only_allowed_chars
1 from src.main import is_valid_email_address
2
3 def test_regular_email_validates():
4     assert is_valid_email_address('test@example.org')
5     assert is_valid_email_address('user123@subdomain.example.org')
6     assert is_valid_email_address('john.doe@email.example.org')
7
8 def test_valid_email_has_one_at_sign():
9     assert not is_valid_email_address('john.doe')
10
11 def test_valid_email_has_only_allowed_chars():
12     assert not is_valid_email_address('john.doe@example.org')
13     assert not is_valid_email_address('not valid@example.org')
```

```
(venvtest) r[kralos]--[P]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
> pytest
===== test session starts =====
platform win32 -- Python 3.12.1, pytest-7.4.4, pluggy-1.3.0
rootdir: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0
collected 3 items

src\test\test_main.py ...

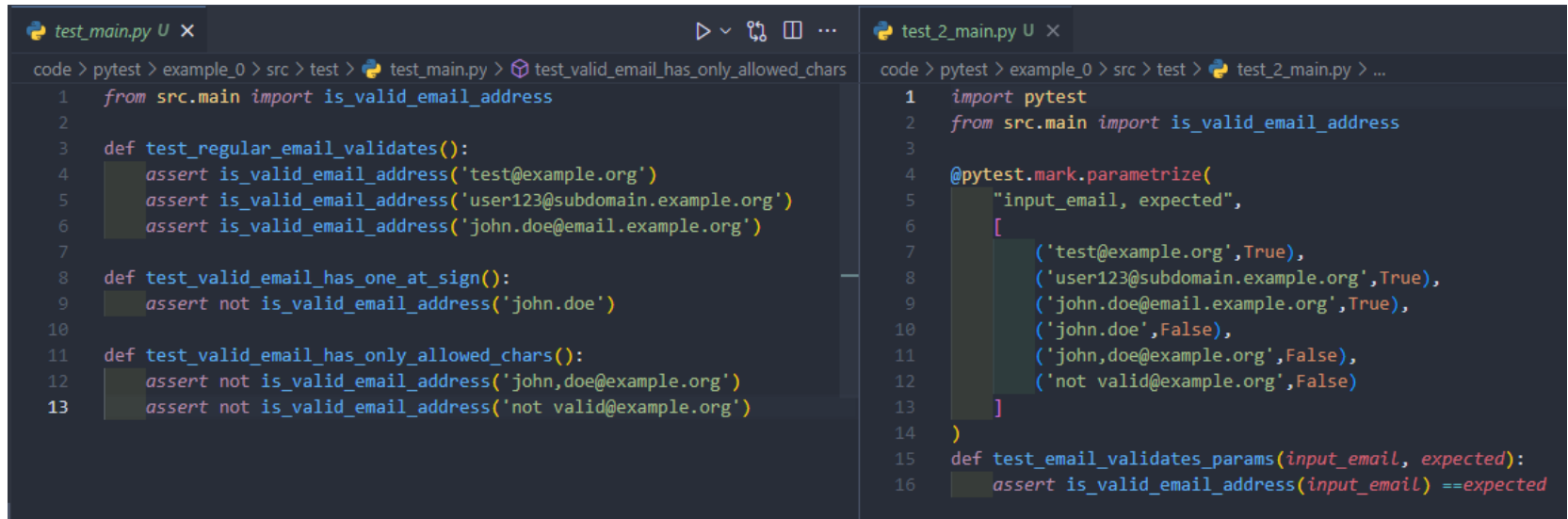
===== 3 passed in 0.02s =====
(venvtest) r[kralos]--[P]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
```

```
(venvtest) r[kralos]--[P]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
> pytest -v
===== test session starts =====
platform win32 -- Python 3.12.1, pytest-7.4.4, pluggy-1.3.0 -- D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0\venvtest\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0
collected 3 items

src\test\test_main.py::test_regular_email_validates PASSED [ 33%]
src\test\test_main.py::test_valid_email_has_one_at_sign PASSED [ 66%]
src\test\test_main.py::test_valid_email_has_only_allowed_chars PASSED [100%]

===== 3 passed in 0.03s =====
(venvtest) r[kralos]--[P]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
```

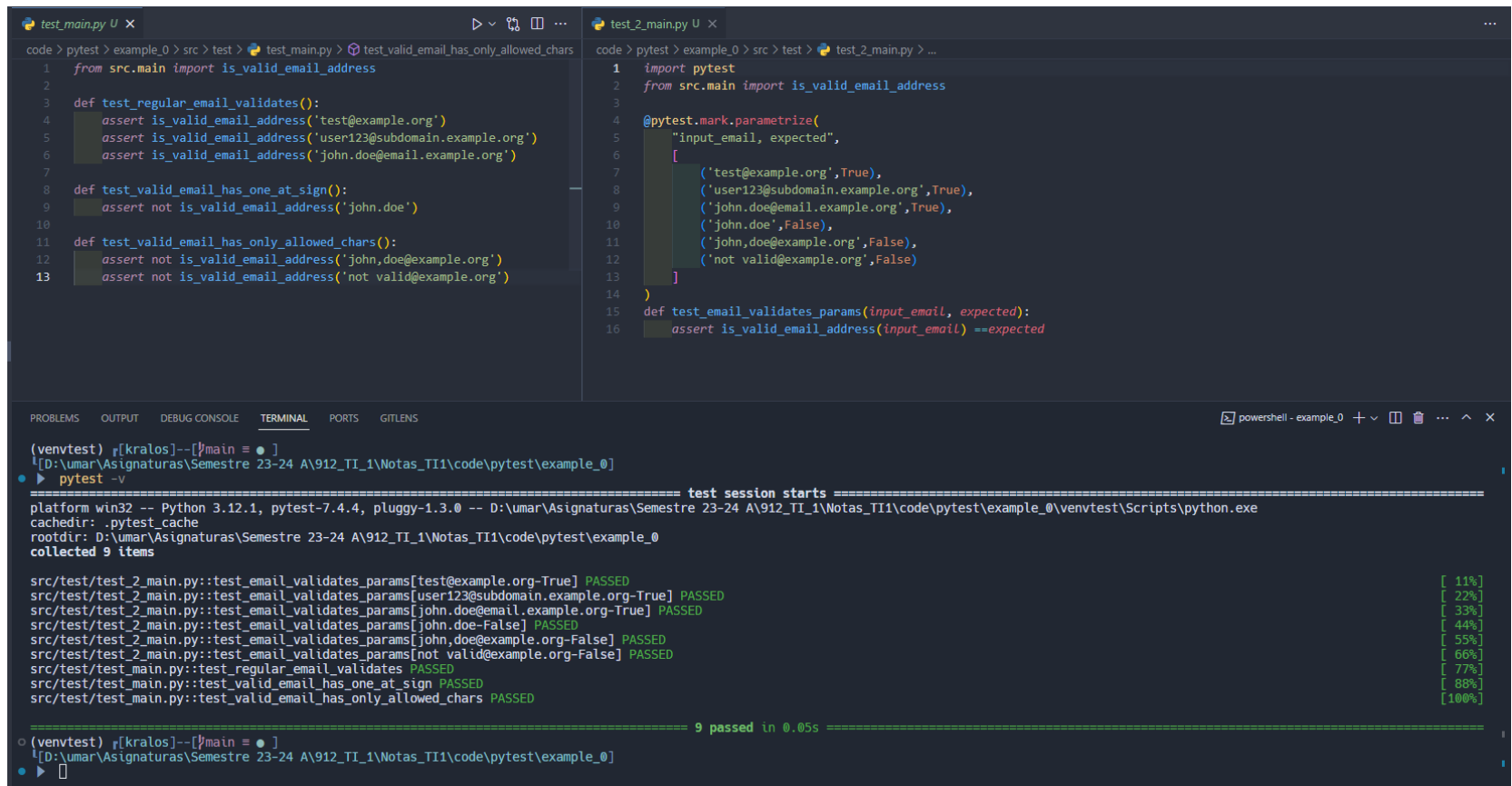

Ejemplo 1



```
test_main.py U X
code > pytest > example_0 > src > test > test_main.py > test_valid_email_has_only_allowed_chars
1  from src.main import is_valid_email_address
2
3  def test_regular_email_validates():
4      assert is_valid_email_address('test@example.org')
5      assert is_valid_email_address('user123@subdomain.example.org')
6      assert is_valid_email_address('john.doe@email.example.org')
7
8  def test_valid_email_has_one_at_sign():
9      assert not is_valid_email_address('john.doe')
10
11 def test_valid_email_has_only_allowed_chars():
12     assert not is_valid_email_address('john,doe@example.org')
13     assert not is_valid_email_address('not valid@example.org')
```

```
test_2_main.py U X
code > pytest > example_0 > src > test > test_2_main.py > ...
1  import pytest
2  from src.main import is_valid_email_address
3
4  @pytest.mark.parametrize(
5      "input_email, expected",
6      [
7          ('test@example.org', True),
8          ('user123@subdomain.example.org', True),
9          ('john.doe@email.example.org', True),
10         ('john.doe', False),
11         ('john,doe@example.org', False),
12         ('not valid@example.org', False)
13     ]
14 )
15 def test_email_validates_params(input_email, expected):
16     assert is_valid_email_address(input_email) == expected
```

Ejemplo 1



The image shows a code editor with two files open: `test_main.py` and `test_2_main.py`. The `test_main.py` file contains three test functions: `test_regular_email_validates`, `test_valid_email_has_one_at_sign`, and `test_valid_email_has_only_allowed_chars`. The `test_2_main.py` file contains a parametrized test `test_email_validates_params` and a simple test `test_email_validates_params`. The terminal window at the bottom shows the output of running `pytest` on the `example_0` directory. The output indicates that 9 tests passed in 0.05s.

```
code > pytest > example_0 > src > test > test_main.py > test_valid_email_has_only_allowed_chars
1 from src.main import is_valid_email_address
2
3 def test_regular_email_validates():
4     assert is_valid_email_address('test@example.org')
5     assert is_valid_email_address('user123@subdomain.example.org')
6     assert is_valid_email_address('john.doe@email.example.org')
7
8 def test_valid_email_has_one_at_sign():
9     assert not is_valid_email_address('john.doe')
10
11 def test_valid_email_has_only_allowed_chars():
12     assert not is_valid_email_address('john.doe@example.org')
13     assert not is_valid_email_address('not valid@example.org')
```

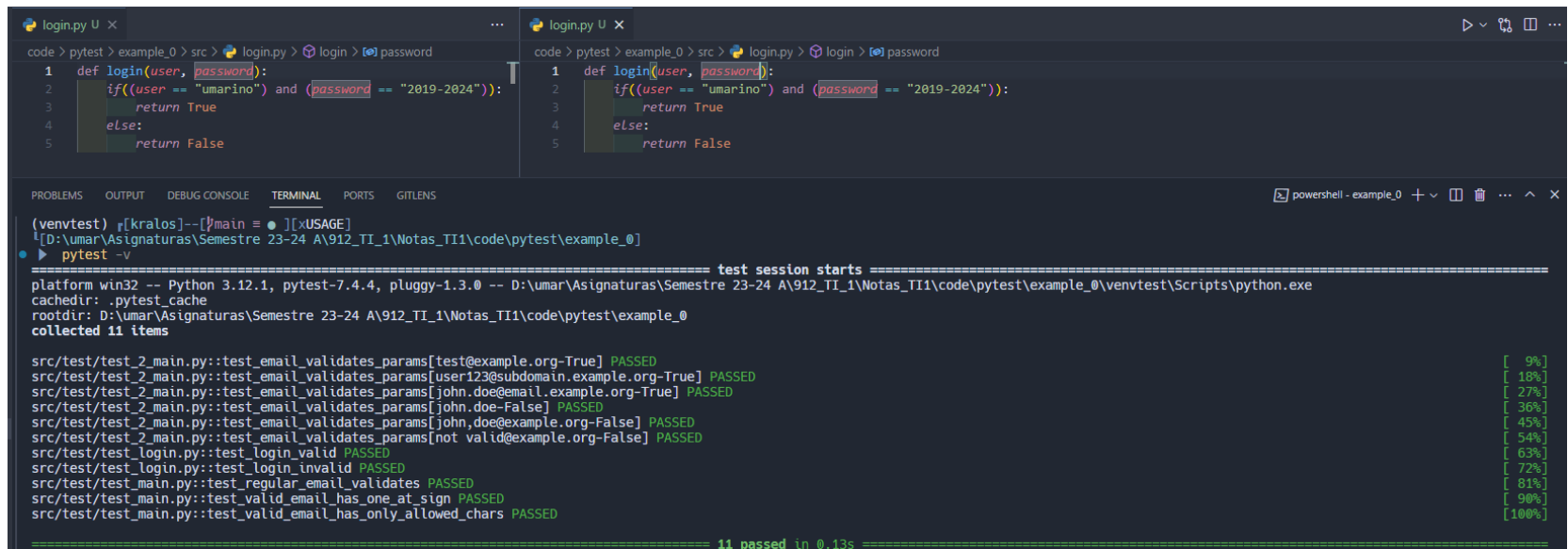
```
code > pytest > example_0 > src > test > test_2_main.py > ...
1 import pytest
2 from src.main import is_valid_email_address
3
4 @pytest.mark.parametrize(
5     "input_email, expected",
6     [
7         ('test@example.org', True),
8         ('user123@subdomain.example.org', True),
9         ('john.doe@email.example.org', True),
10         ('john.doe', False),
11         ('john.doe@example.org', False),
12         ('not valid@example.org', False)
13     ]
14 )
15 def test_email_validates_params(input_email, expected):
16     assert is_valid_email_address(input_email) == expected
```

```
(venvtest) [kralos]--[main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
● ▶ pytest -v
===== test session starts =====
platform win32 -- Python 3.12.1, pytest-7.4.4, pluggy-1.3.0 -- D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0\venvtest\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0
collected 9 items

src/test/test_2_main.py::test_email_validates_params[test@example.org-True] PASSED [ 11%]
src/test/test_2_main.py::test_email_validates_params[user123@subdomain.example.org-True] PASSED [ 22%]
src/test/test_2_main.py::test_email_validates_params[john.doe@email.example.org-True] PASSED [ 33%]
src/test/test_2_main.py::test_email_validates_params[john.doe-False] PASSED [ 44%]
src/test/test_2_main.py::test_email_validates_params[john.doe@example.org-False] PASSED [ 55%]
src/test/test_2_main.py::test_email_validates_params[not valid@example.org-False] PASSED [ 66%]
src/test/test_main.py::test_regular_email_validates PASSED [ 77%]
src/test/test_main.py::test_valid_email_has_one_at_sign PASSED [ 88%]
src/test/test_main.py::test_valid_email_has_only_allowed_chars PASSED [100%]

===== 9 passed in 0.05s =====
o (venvtest) [kralos]--[main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
● ▶
```

Ejemplo 1



The image shows a VS Code editor with two tabs for `login.py`. The left tab shows the source code of the `login` function, which checks if the user is "umarino" and the password is "2019-2024". The right tab shows the same code. Below the editor, the `TERMINAL` panel displays the output of running `pytest -v` in a virtual environment. The output shows that 11 tests passed in 0.13s, with a progress bar on the right indicating the completion percentage for each test.

```
code > pytest > example_0 > src > login.py > login > password
1 def login(user, password):
2     if((user == "umarino") and (password == "2019-2024")):
3         return True
4     else:
5         return False

code > pytest > example_0 > src > login.py > login > password
1 def login(user, password):
2     if((user == "umarino") and (password == "2019-2024")):
3         return True
4     else:
5         return False

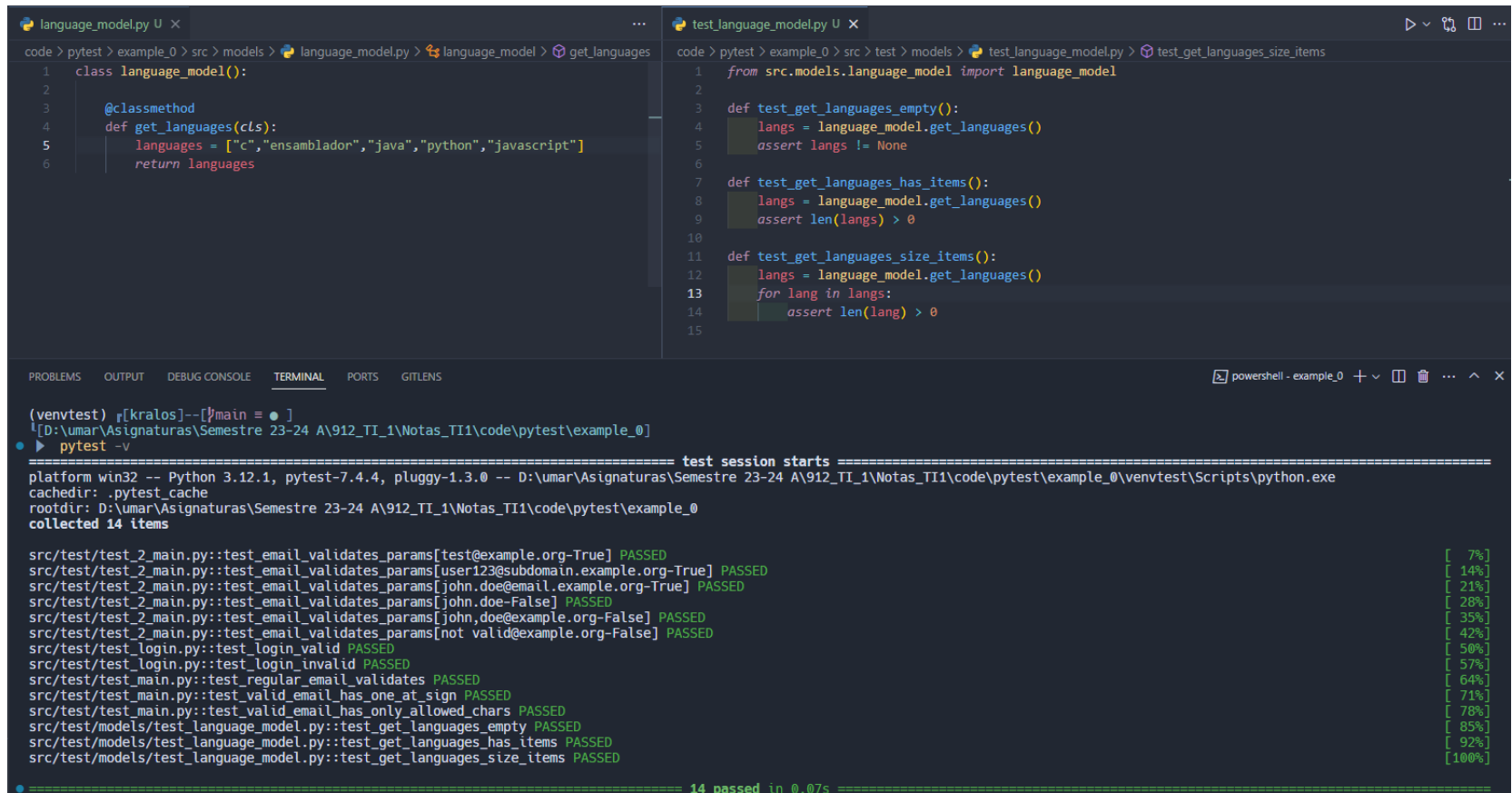
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
(venvtest) [kralos]--[main = •][xUSAGE]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
• ▶ pytest -v

===== test session starts =====
platform win32 -- Python 3.12.1, pytest-7.4.4, pluggy-1.3.0 -- D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0\venvtest\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0
collected 11 items

src/test/test_2_main.py::test_email_validates_params[test@example.org-True] PASSED [ 9%]
src/test/test_2_main.py::test_email_validates_params[user123@subdomain.example.org-True] PASSED [ 18%]
src/test/test_2_main.py::test_email_validates_params[john.doe@email.example.org-True] PASSED [ 27%]
src/test/test_2_main.py::test_email_validates_params[john.doe-False] PASSED [ 36%]
src/test/test_2_main.py::test_email_validates_params[john.doe@example.org-False] PASSED [ 45%]
src/test/test_2_main.py::test_email_validates_params[not valid@example.org-False] PASSED [ 54%]
src/test/test_login.py::test_login_valid PASSED [ 63%]
src/test/test_login.py::test_login_invalid PASSED [ 72%]
src/test/test_main.py::test_regular_email_validates PASSED [ 81%]
src/test/test_main.py::test_valid_email_has_one_at_sign PASSED [ 90%]
src/test/test_main.py::test_valid_email_has_only_allowed_chars PASSED [100%]

===== 11 passed in 0.13s =====
```

Ejemplo 1



```
language_model.py U x
code > pytest > example_0 > src > models > language_model.py > language_model > get_languages
1 class language_model():
2
3     @classmethod
4     def get_languages(cls):
5         languages = ["c", "ensamblador", "java", "python", "javascript"]
6         return languages

test_language_model.py U x
code > pytest > example_0 > src > test > models > test_language_model.py > test_get_languages_size_items
1 from src.models.language_model import language_model
2
3 def test_get_languages_empty():
4     langs = language_model.get_languages()
5     assert langs != None
6
7 def test_get_languages_has_items():
8     langs = language_model.get_languages()
9     assert len(langs) > 0
10
11 def test_get_languages_size_items():
12     langs = language_model.get_languages()
13     for lang in langs:
14         assert len(lang) > 0
15

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
powershell - example_0 + v [ ] ... ^ x

(venvtest) r[kralos]--[p]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0]
● ▶ pytest -v

===== test session starts =====
platform win32 -- Python 3.12.1, pytest-7.4.4, pluggy-1.3.0 -- D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0\venvtest\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\pytest\example_0
collected 14 items



src/test/test_2_main.py::test_email_validates_params[test@example.org-True] PASSED [ 7%]
src/test/test_2_main.py::test_email_validates_params[user123@subdomain.example.org-True] PASSED [ 14%]
src/test/test_2_main.py::test_email_validates_params[john.doe@email.example.org-True] PASSED [ 21%]
src/test/test_2_main.py::test_email_validates_params[john.doe-False] PASSED [ 28%]
src/test/test_2_main.py::test_email_validates_params[john.doe@example.org-False] PASSED [ 35%]
src/test/test_2_main.py::test_email_validates_params[not valid@example.org-False] PASSED [ 42%]
src/test/test_login.py::test_login_valid PASSED [ 50%]
src/test/test_login.py::test_login_invalid PASSED [ 57%]
src/test/test_main.py::test_regular_email_validates PASSED [ 64%]
src/test/test_main.py::test_valid_email_has_one_at_sign PASSED [ 71%]
src/test/test_main.py::test_valid_email_has_only_allowed_chars PASSED [ 78%]
src/test/models/test_language_model.py::test_get_languages_empty PASSED [ 85%]
src/test/models/test_language_model.py::test_get_languages_has_items PASSED [ 92%]
src/test/models/test_language_model.py::test_get_languages_size_items PASSED [100%]

===== 14 passed in 0.07s =====
```

Importancia de las pruebas

- Primero escribimos nuestra función y luego ideamos algunos casos de prueba para ella. Pero nos dimos cuenta de que estos casos de prueba no son en absoluto completos. Por el contrario, nos faltaban algunos aspectos esenciales.
- Desarrollo Dirigido por Pruebas (TDD): Conseguir que los requisitos sean correctos escribiendo primero los casos de prueba y no empezar a implementar una función antes de sentir que se han cubierto todos los casos de prueba.

Referencias

- De datos,  Tacos, & De datos en español, A. V. (2020, May 12). *Probando nuestro código con pytest*.  Tacos de Datos | Aprende Visualización de Datos En Español. <https://old.tacosdedatos.com/pruebas-unitarias-pytest>
- *Pytest: Helps you write better programs — pytest documentation*. (n.d.). Pytest.org. Retrieved January 15, 2024, from <https://docs.pytest.org/en/7.4.x/>
- (N.d.). Denshub.com. Retrieved January 15, 2024, from <https://denshub.com/es/intro-to-testing-pytest/#se-trata-de-assert>