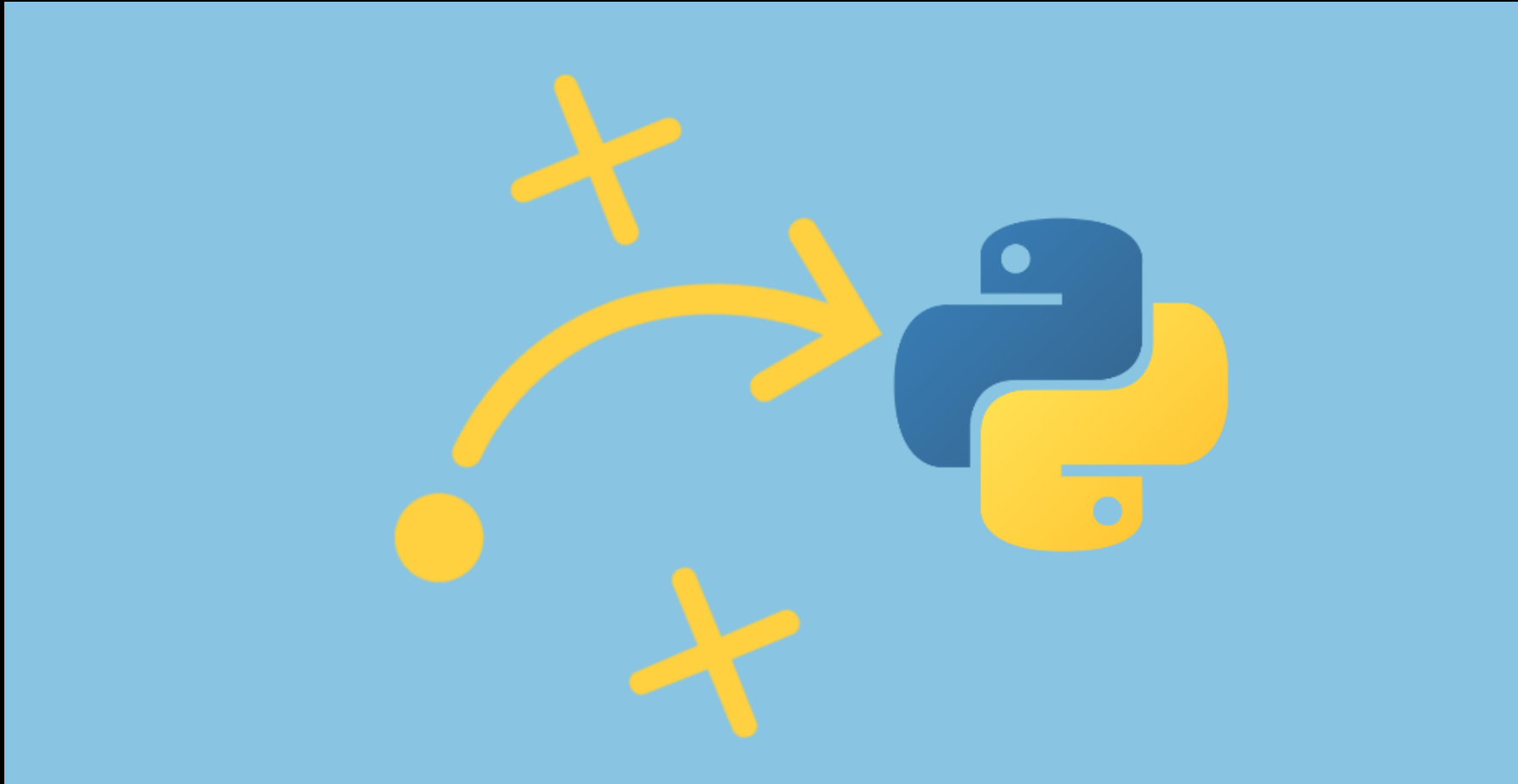
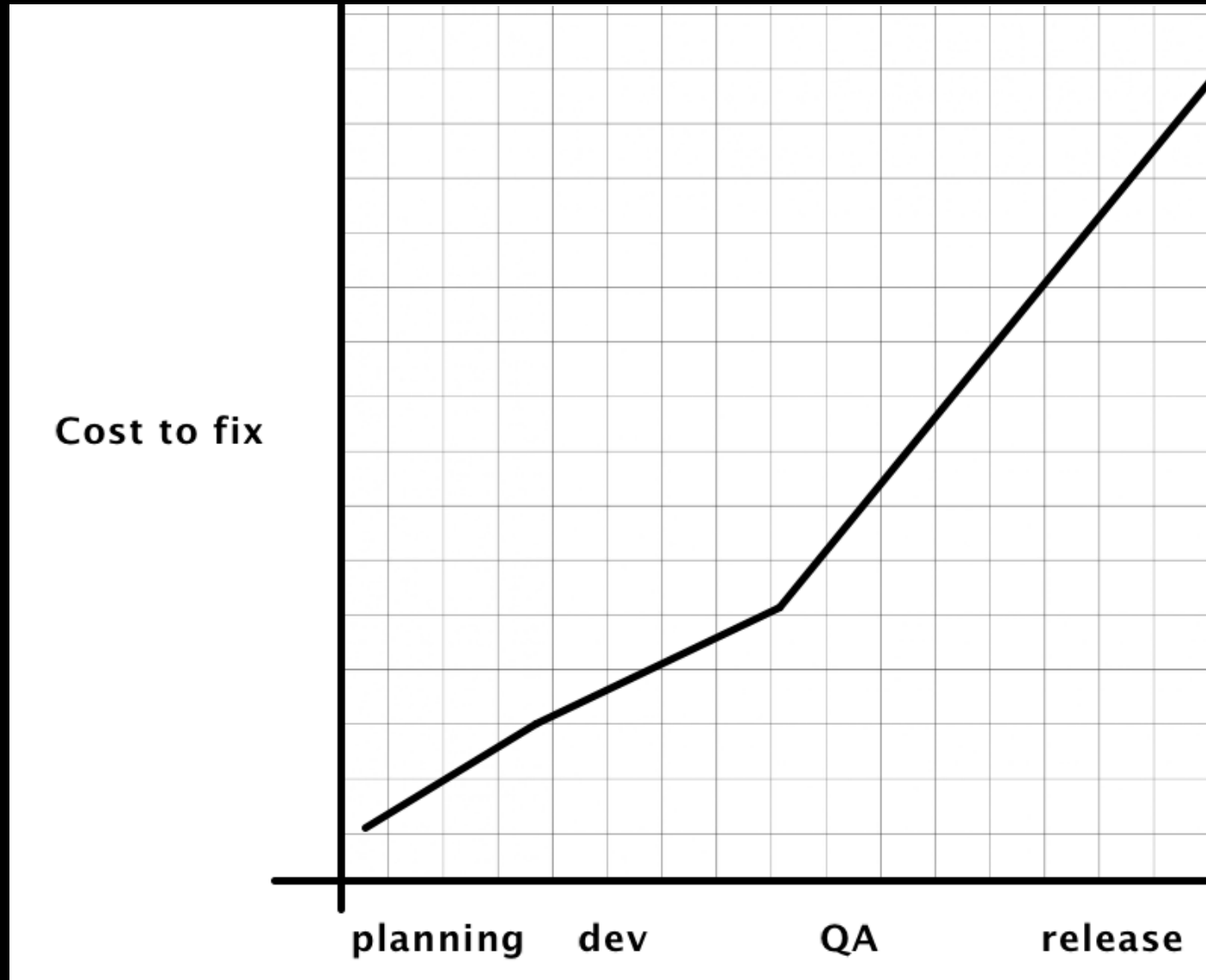


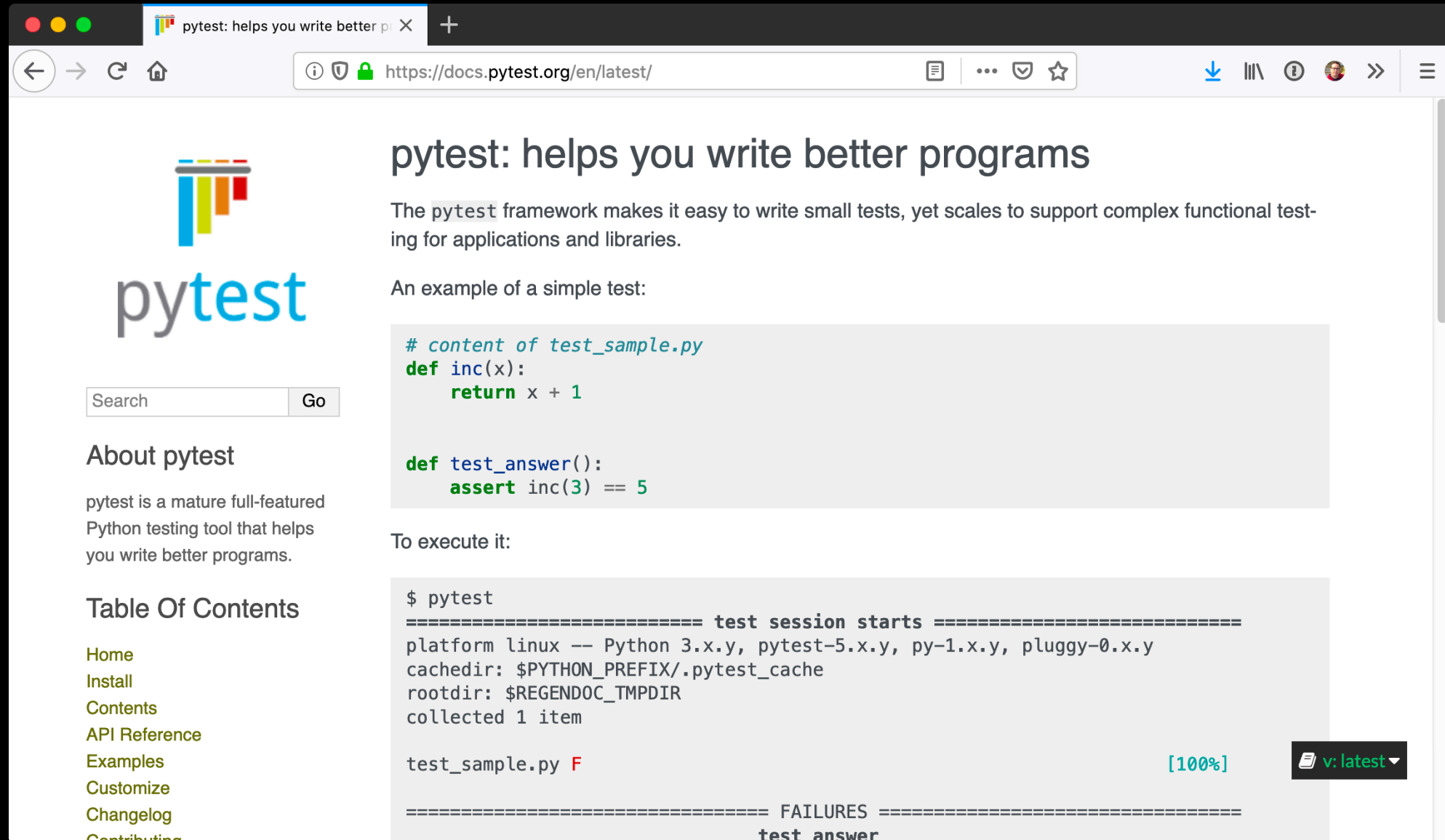
Testing



Testing lowers overall cost of systems



Python has a built-in version, but **pytest**



The image is a screenshot of a web browser displaying the official pytest website. The browser's address bar shows the URL `https://docs.pytest.org/en/latest/`. The website's header features the pytest logo, which consists of four vertical bars in blue, yellow, red, and green, followed by the word "pytest" in a blue sans-serif font. Below the logo is a search bar with the placeholder text "Search" and a "Go" button. To the right of the logo, the main heading reads "pytest: helps you write better programs". Below this heading is a paragraph: "The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries." This is followed by the text "An example of a simple test:". Below this text is a code block containing Python code for a simple test. The code defines a function `inc(x)` that returns `x + 1`, and a test function `test_answer()` that asserts `inc(3) == 5`. Below the code block is the text "To execute it:". Below this text is another code block showing the output of running the command `$ pytest`. The output shows the test session starting, the platform (linux), Python version (3.x.y), pytest version (5.x.y), and the number of items collected (1). The test result is shown as `test_sample.py F` with a red 'F' indicating a failure. The output also shows the test session ending with `[100%]` and a dropdown menu for the version (v: latest). The left sidebar of the website contains a "Table Of Contents" section with links to Home, Install, Contents, API Reference, Examples, Customize, Changelog, and Contributing.

pytest: helps you write better programs

The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries.

An example of a simple test:

```
# content of test_sample.py
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

To execute it:

```
$ pytest
===== test session starts =====
platform linux -- Python 3.x.y, pytest-5.x.y, py-1.x.y, pluggy-0.x.y
cachedir: $PYTHON_PREFIX/.pytest_cache
rootdir: $REGENDOC_TMPDIR
collected 1 item

test_sample.py F [100%]
===== FAILURES =====
test answer
```

A basic pytest test

Name starts with test_ has 'raw' assert statements



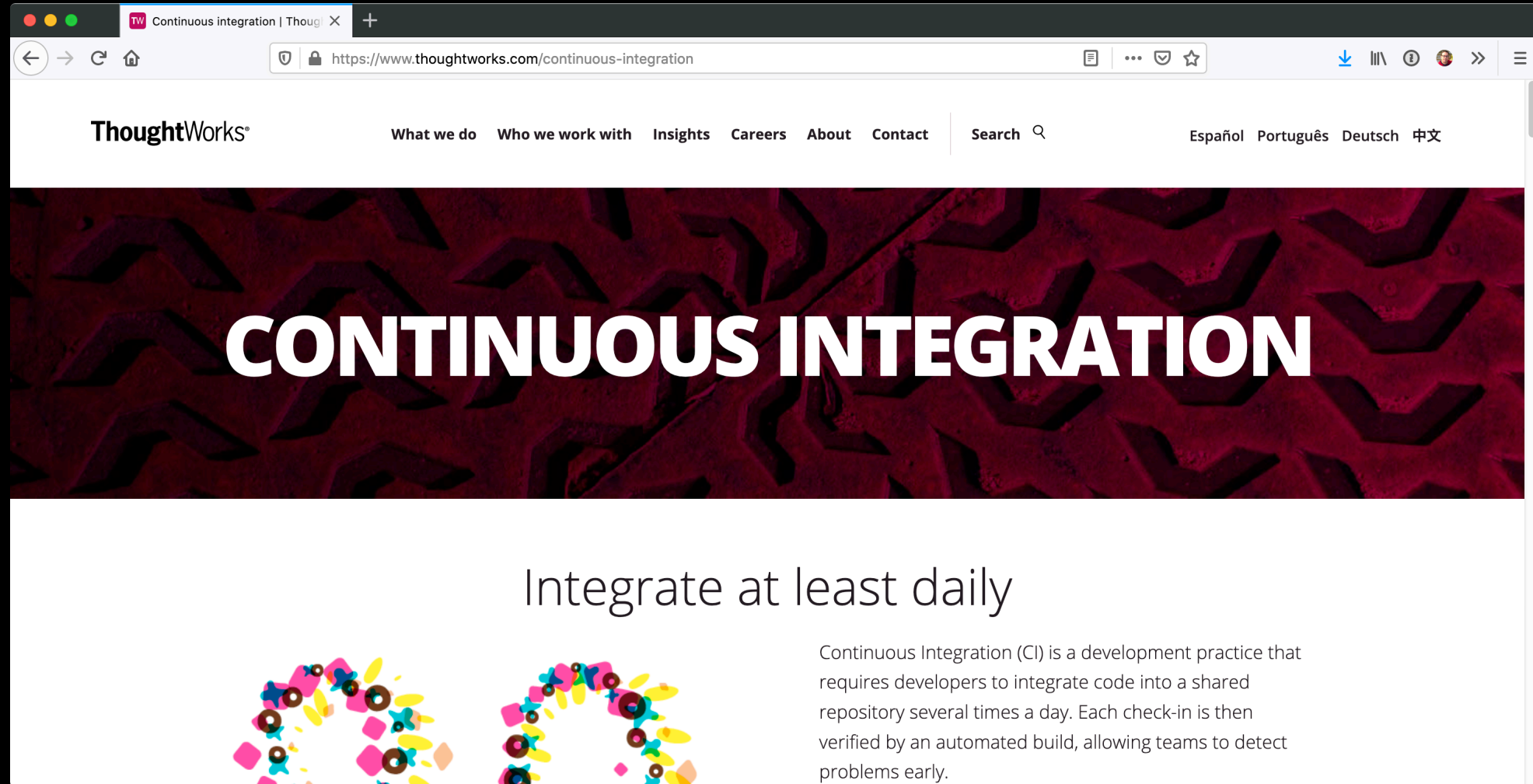
```
def test_electric_guitars():  
    guitars = lib.all_guitars('electric')  
  
    # Sweet little generator expression  
    assert all(g.style == 'electric' for g in guitars)
```

Testing for errors

```
def test_input_validation():  
    with pytest.raises(ValueError):  
        lib.all_guitars(None)
```

Test for errors (exceptions) with **pytest.raises(TYPE)**.

Pairs well with continuous integration



<https://www.thoughtworks.com/continuous-integration>