1. Escolha duas bibliotecas/frameworks de uma mesma linguagem de programação e: Python (Unittest e Pytest)  
  
a. Avalie os pontos positivos e negativos de cada uma delas.

**Unittest**  
  
**Pontos Positivos**

* É **padrão da linguagem Python**, não precisa instalar nada.
* Muito usada em empresas mais tradicionais (bom para compatibilidade).
* Estrutura bem definida e organizada (semelhante ao JUnit do Java).
* Suporta fixtures, mocks e integração com CI/CD.

**Pontos Negativos**

* Verboso: precisa de **classes** e **métodos específicos** (setUp, tearDown).
* A escrita de testes é menos intuitiva e exige mais boilerplate.
* Output menos amigável, comparado ao Pytest.

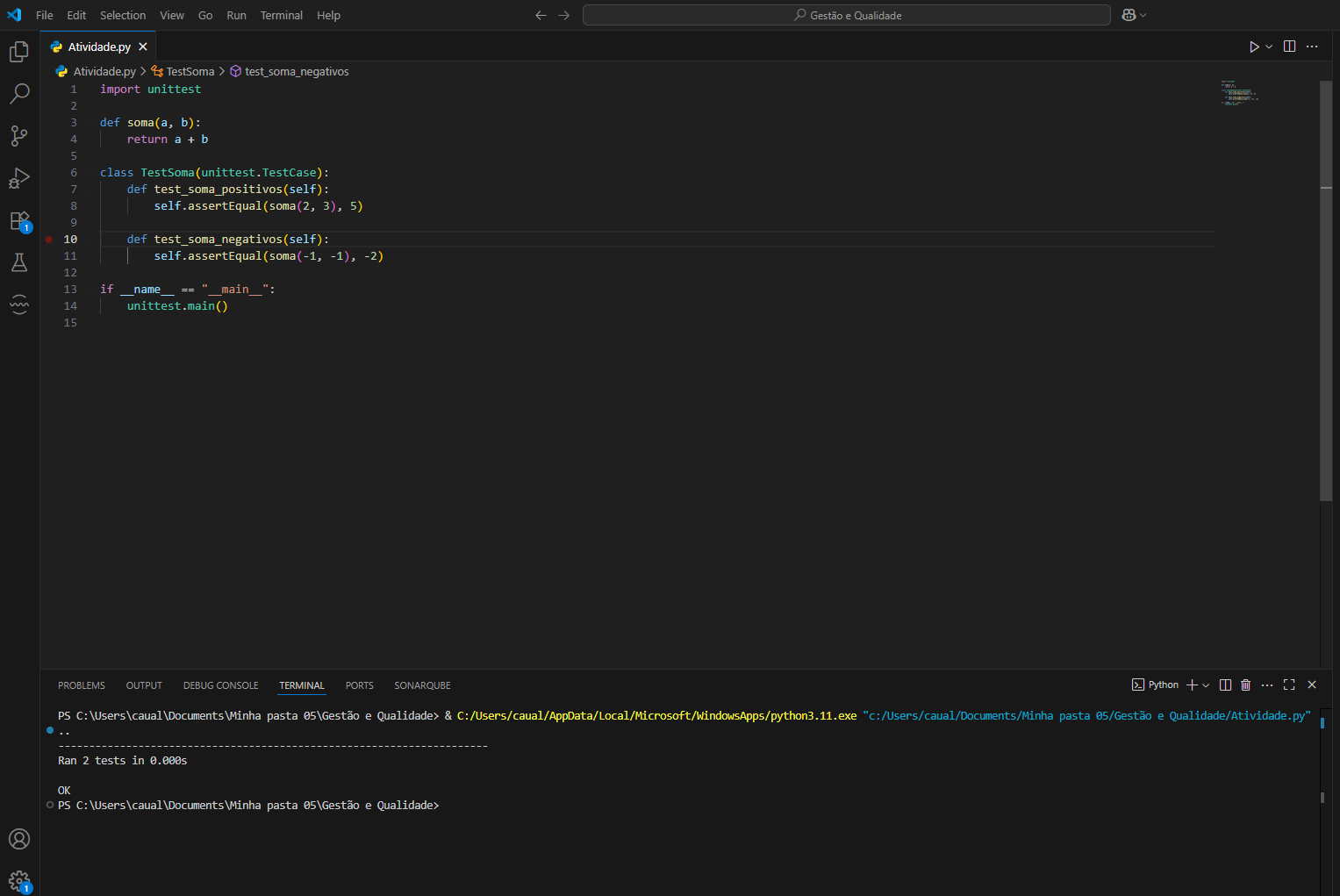
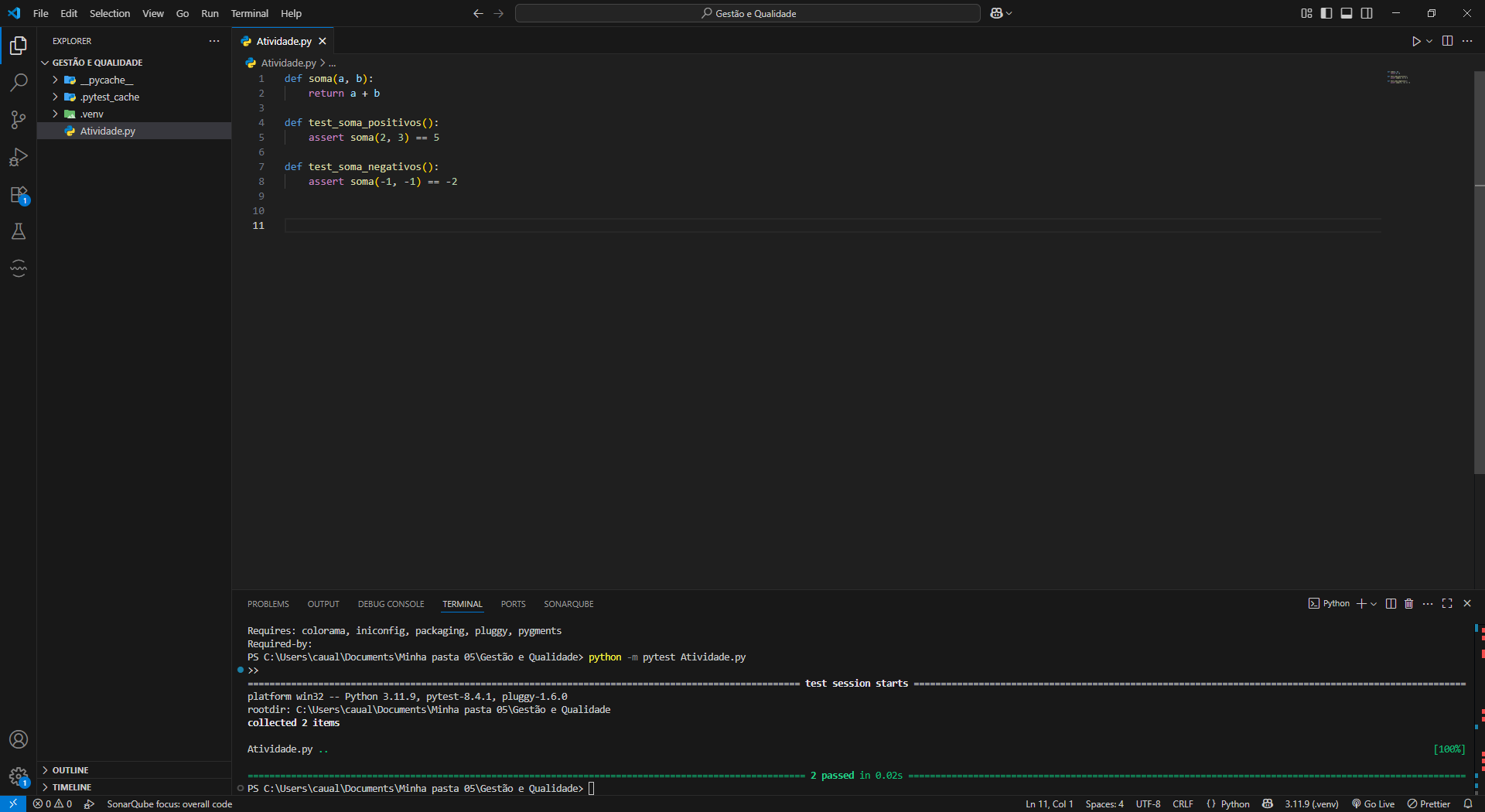
**Pytest**

Pontos Positivos

* Sintaxe simples e direta (não precisa criar classes).
* Suporte a fixtures avançadas, parametrização e plugins.
* Output detalhado e colorido (mostra falhas de forma clara).
* Muito popular em projetos modernos.

Pontos Negativos

* Precisa ser instalado via pip install pytest (não é padrão).
* Algumas empresas mais conservadoras ainda preferem unittest.
* Para quem vem do JUnit ou NUnit, pode parecer “menos formal”.

b. Demonstre exemplos de utilização das bibliotecas, indicando as diferenças de implementação e resposta (saída).  
  
Exemplo com Unittest:  
  
  
Exemplo com Pytest:  
  
  
Diferenças Principais

* Unittest: Mais verboso, orientado a classes e métodos.
* Pytest: Mais simples, direto, leitura mais fácil.
* Saída: Pytest mostra falhas de forma clara, enquanto Unittest mostra apenas “FAIL” ou “ERROR”.

2. Realize uma pesquisa bibliográfica demonstrando as diferenças entre essas duas abordagens de testes (Caixa Branca e Caixa Preta), como eles podem ser aplicados no processo de desenvolvimento de software e quais ferramentas (bibliotecas, frameworks, etc) existentes apoiam nessas abordagens.  
  
  
Testes de Caixa Preta (black-box)  
 Focam no comportamento observado do software (entradas → saídas), sem conhecer o código. Técnicas clássicas incluem Equivalence Partitioning (particionamento de equivalência), Boundary Value Analysis (análise de valores-limite) e Decision Table Testing (tabelas de decisão). Essas técnicas são formalizadas pelo ISTQB.  
  
Testes de Caixa Branca (white-box)  
 Focam na estrutura interna do código. O projeto de testes usa conhecimento de fluxo de controle e dados; metas comuns incluem *statement coverage* e *decision/branch coverage*.  
  
**Diferenças essenciais**

* **Visão interna**: Caixa Branca enxerga e usa o código; Caixa Preta ignora a implementação e valida apenas o comportamento.
* **Critérios de adequação**: Caixa Branca mede cobertura estrutural (ex.: *statement coverage*); Caixa Preta deriva casos por regras de negócio (ex.: particionamento, limites, tabelas de decisão).
* **Nível típico**: Caixa Branca é mais comum em **unit** e parte de **integração**; Caixa Preta aparece muito em **sistema**, **aceitação** e **E2E**, embora ambas possam existir em vários níveis.

Como aplicar no processo (SDLC)

1. Planejamento e requisitos

* Definir critérios de aceitação → base para testes de Caixa Preta (ex.: regras em tabelas de decisão).

1. Projeto e implementação

* Desenvolvedores escrevem unit tests visando cobertura estrutural (Caixa Branca) e monitoram métricas como *statement/branch coverage*.

1. Integração e sistema

* Caixa Preta valida fluxos de negócio, limites e combinações (partições, limites, tabelas).
* Caixa Branca ainda pode ser usada p/ assegurar que ramos e caminhos críticos são exercitados.

1. Teste de aceitação / E2E

* Predominância de Caixa Preta com automação de UI/API para simular uso real.

1. Manutenção/CI

* Executar suites automaticamente; acompanhar cobertura (Caixa Branca) e resultados funcionais (Caixa Preta). Ferramentas de cobertura integram com Maven/Gradle (Java) ou nyc/Istanbul (JS).

Ferramentas que apoiam cada abordagem

Principalmente Caixa Preta (funcional/E2E, comportamento observado)

* Web E2E (UI):
  + Selenium (WebDriver, multi-browser).
  + Cypress (teste de front-end com runner próprio).
  + Playwright (Chromium/Firefox/WebKit, multi-linguagem).
* API: Postman + Newman (executa coleções em CLI/CI).
* Mobile: Appium (automation multi-plataforma via WebDriver).
* Performance: Apache JMeter (carga e desempenho).
* Segurança (DAST): OWASP ZAP (scanner de apps web “black-box”).