DELL EMC

Функциональное тестирование продукта

Как начать?

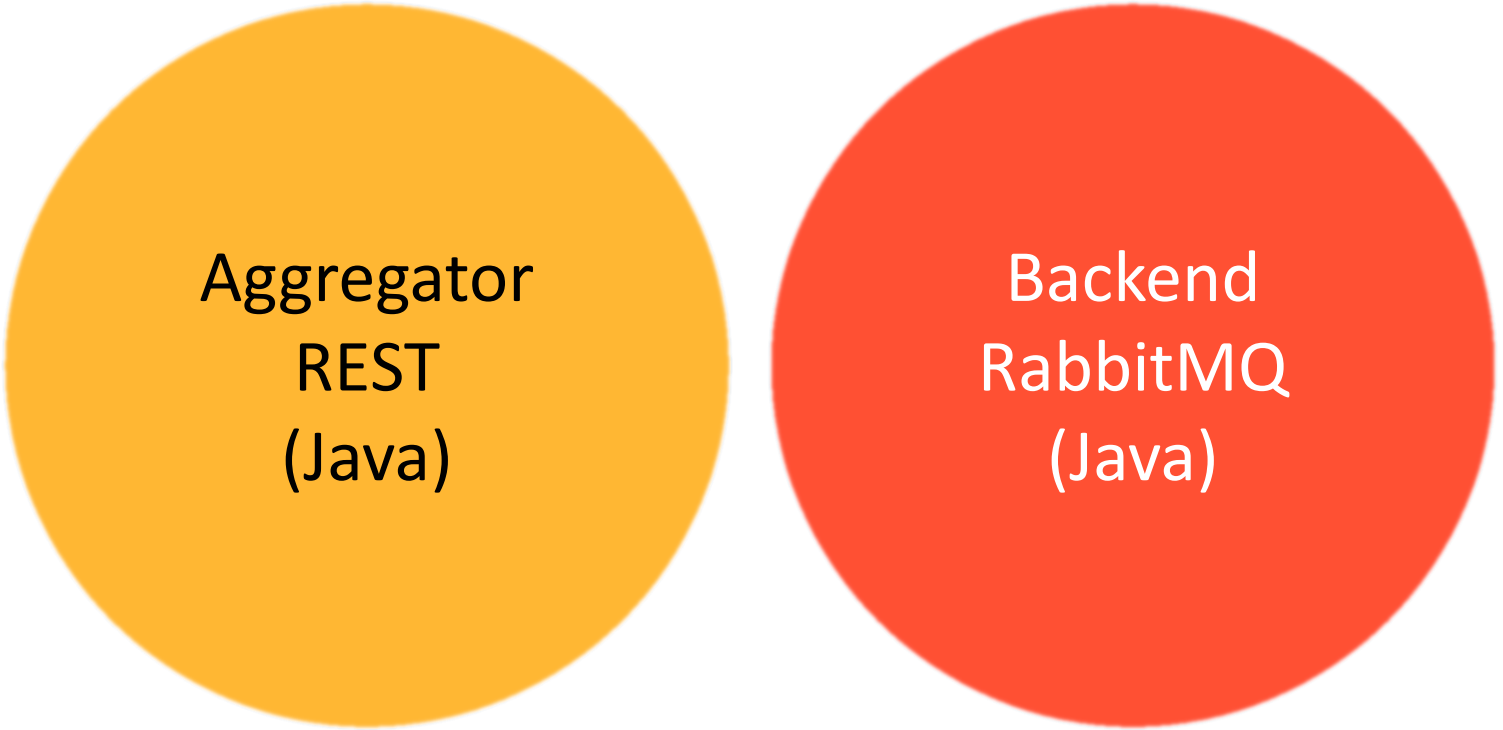
А потом…

сделать так, чтобы qa-инженеры быстро писали тесты и не задумывались (много)

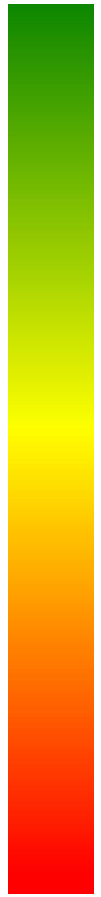
Никита Ковригин
Nikita.Kovrigin@dell.com

SECR

О продукте и начальный бэкграунд



Шкала покрытия тестами



Чуть-чуть о том, что было

- Aggregator:
  - Монолитные тесты
  - Сложная подготовка перед каждым классом
  - Неочевидные проверки
  - Неочевидные настройки
- Backend:
  - Тестов нет – казалось бы, одной проблемой меньше! *#нонет*

Pytest – какие преимущества?

1. Управление временем жизни объекта мы отдаем в pytest
2. Передача тестируемых объектов в объекты

Что теперь?

Одного Pytest’а недостаточно! *#жаль*

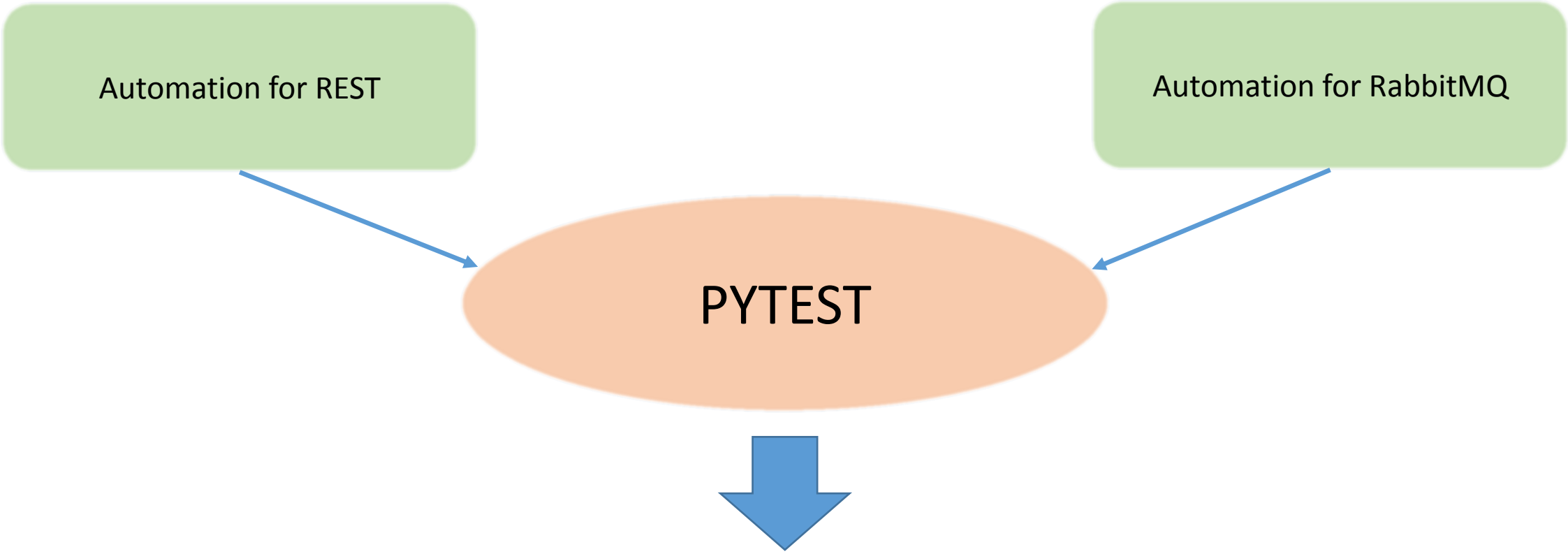
Тестируемые объекты то сложно описывать *#ситуациястарая*

Манипулировать объектами в тесте неудобно

РУТИНА –> ОШИБКИ

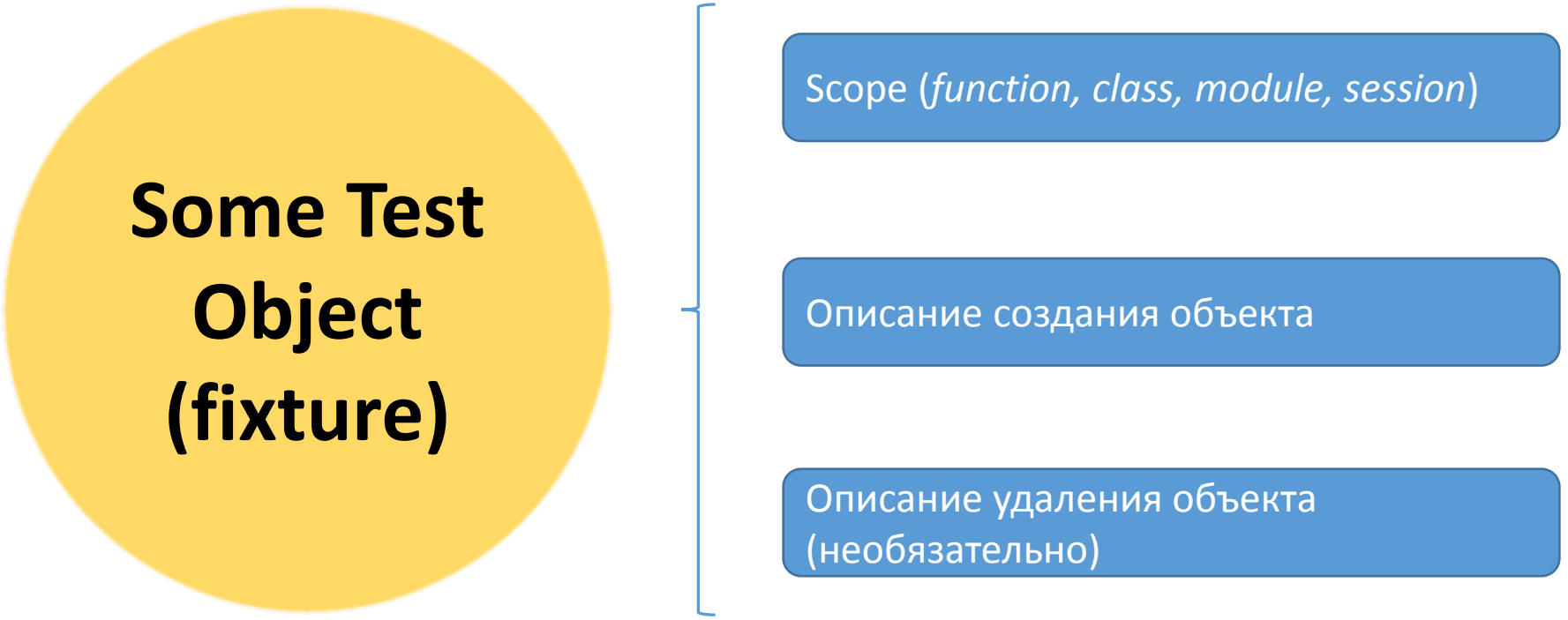
ВЫВОД –> УПРОСТИТЬ РАБОТУ ПО СОЗДАНИЮ ОБЪЕКТОВ

In automation we trust!

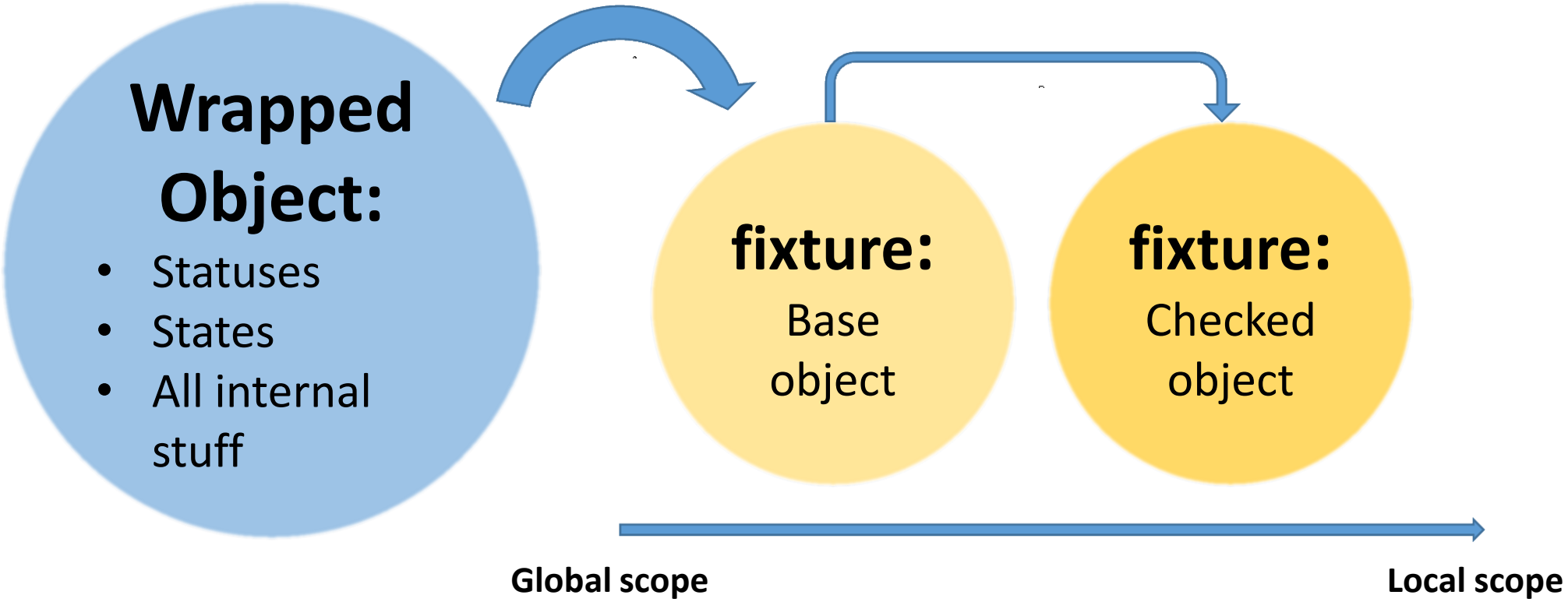


- Намного проще создавать тестовые объекты
- Намного проще управлять ими в тестах

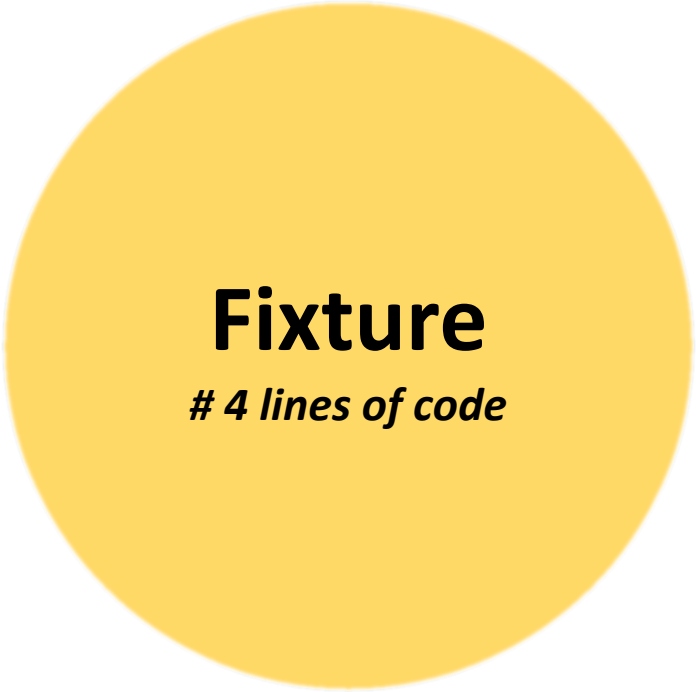
Удобное управление объектом



Что еще можно накрутить?



Пример 1

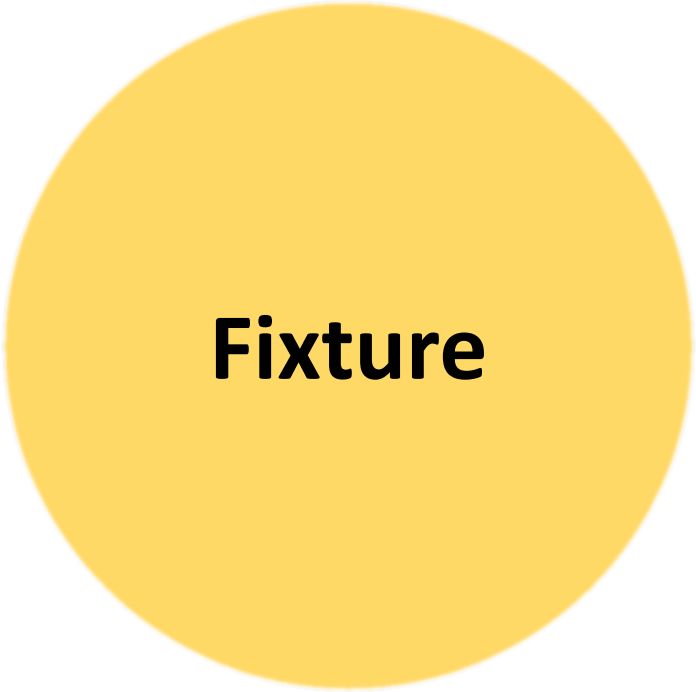



```
@pytest.fixture(scope="class")
def smoke_session_fixture(…)

    smoke_session = Session(params=…)
    smoke_session.create_session()
    smoke_session.remove_session()

    return smoke_session
```

1




```
class TestClass:

    def test_example(self,  ):
        # some test stuff


# our class scope fixture is the
# only one for whole test class
```

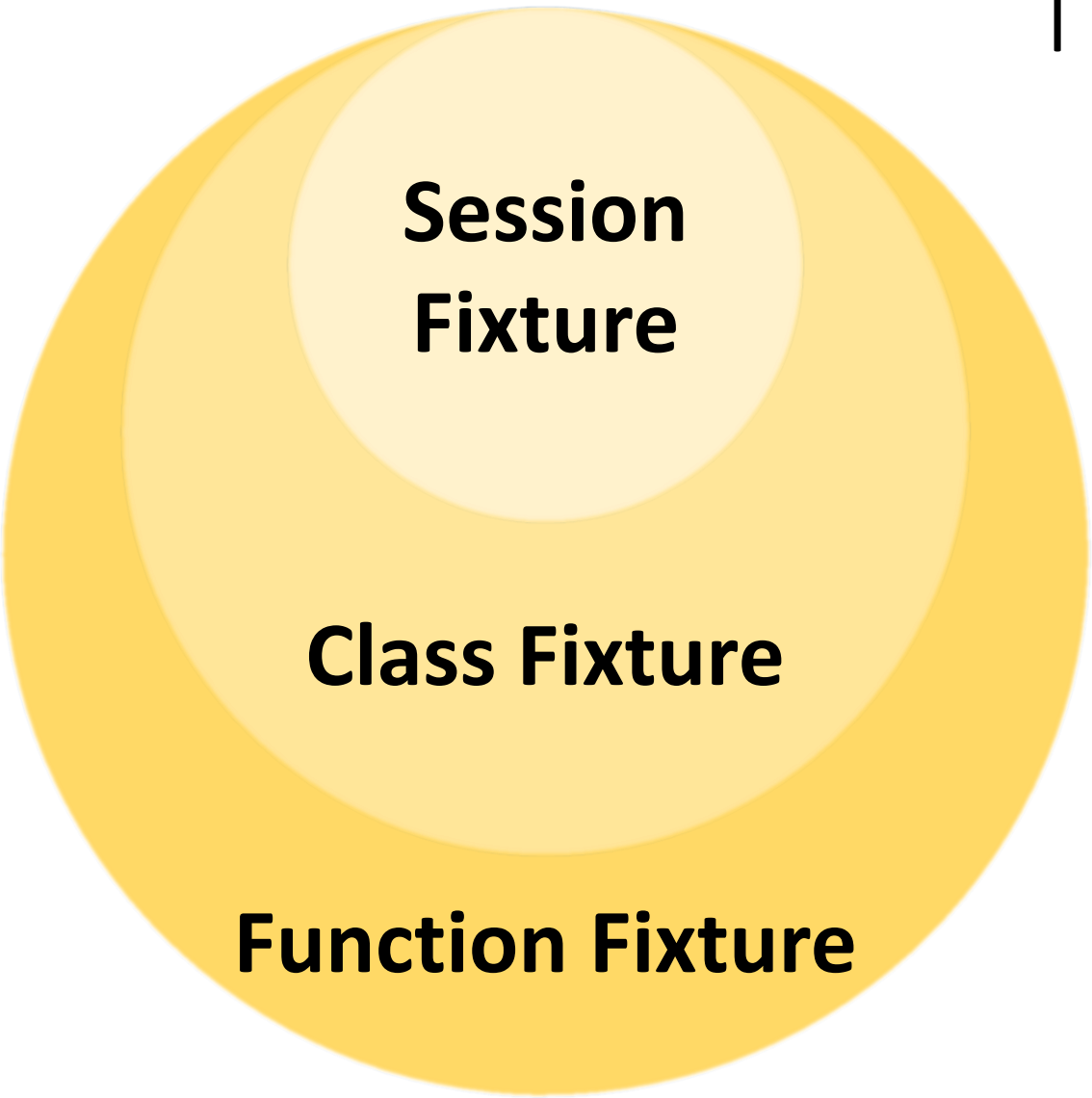
1

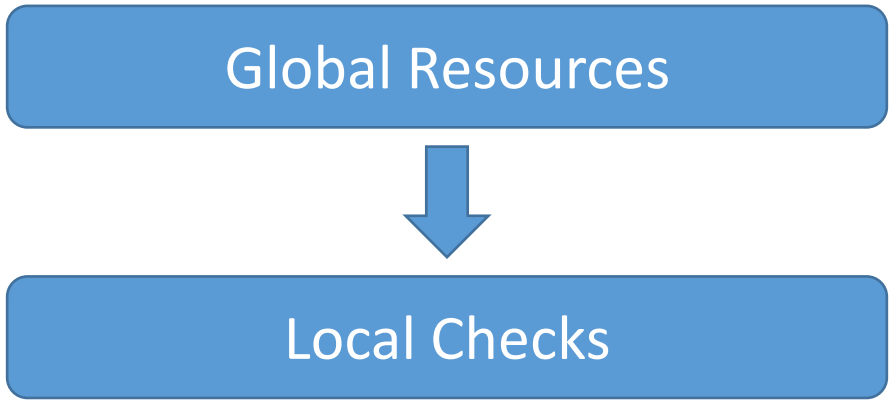
```
class TestSmokeSessionProcessing:

        def test_create_request_ok(self,  ):
                # one line check

        def test_creation_success(self,  )
        def test_failed_status_during_creation(self,  )
        def test_final_status(self,  )
        …
        def test_destroy_request_ok(self,  )
```

Пример 2


Session Fixture
Class Fixture
Function Fixture

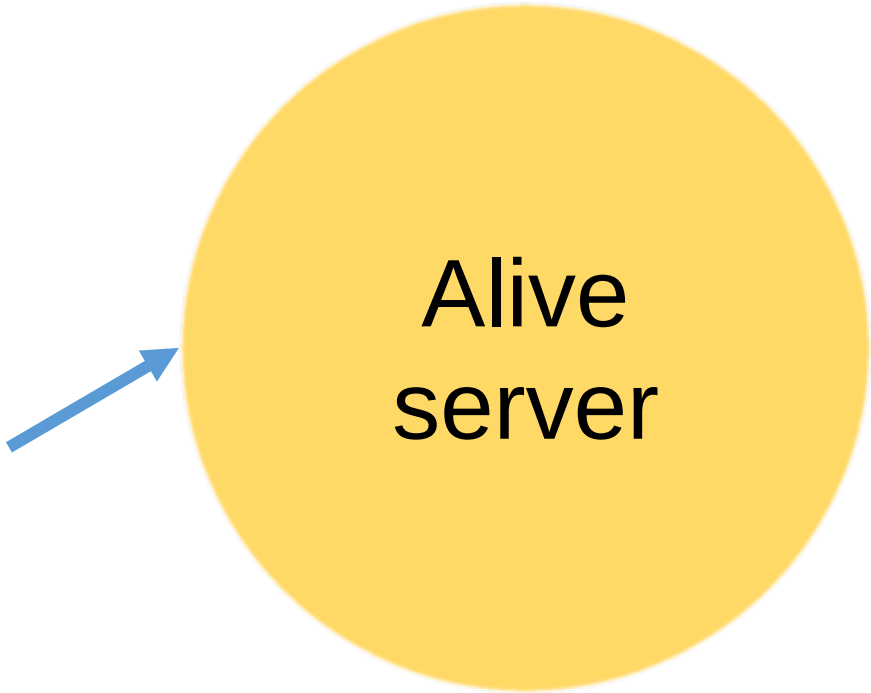

Global Resources
Local Checks

2

```
@pytest.fixture(scope="session")
def global_session():
    ses = Session(params={})
    ses.create_session()
    return ses

@pytest.fixture(scope="class")
def one_server(global_session):
    # get one
    return some_server

@pytest.fixture(scope="test")
def alive_server(one_server):
    # check it and turn on
    return one_server
```



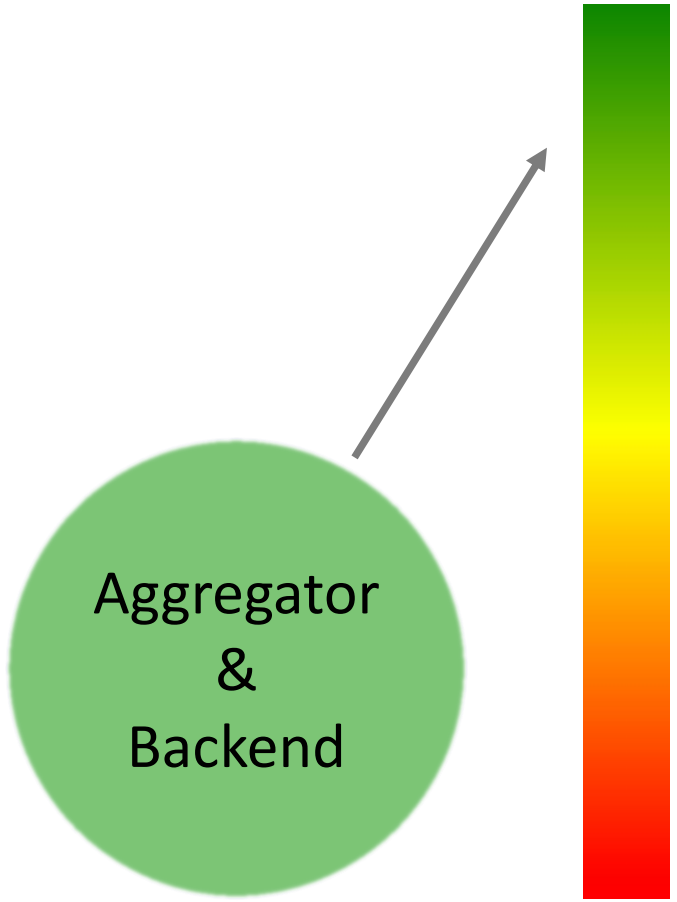
2

```
def test_ram(self,  ):
    result = alive_server.update(memory_mb=…)
    assert result.get_status()

def test_disk(self,  )
def test_network (self,  )
def test_cpu (self,  )
```

Результат незатейливого подхода

- ↗ Скорость написания тестов выросла
- ↗ Объем тестов увеличился в 15 раз
- ↘ Общее время прохождения тестов
- ↘ Размер тестов (1-15 строк)



К чему я это все говорю…

- Автоматизируй рутину
- Передавай обертки в Pytest
- Управляй ресурсами за счет скопуов
- Экономь на ресурсах за счет скоупов