Atividades – TDD

Grupo:

Guilherme Lima Machado – RA: 822162693

Arthur Salatine de Moraes – RA: 822151203

Atividade 1

Escreva testes TDD para uma função maior(a, b) que retorne o maior entre dois números.

maior.py

```
def maior(a, b):
    return a if a >= b else b
```

test_maior.py

```
import unittest
from maior import maior

class test_maior(unittest.TestCase):

    def test_maior_primeiro(self):
        self.assertEqual(maior(10, 5), 10)

    def test_maior_segundo(self):
        self.assertEqual(maior(3, 7), 7)

    def test_iguais(self):
        self.assertEqual(maior(4, 4), 4)
```

Atividade 2

Escreva testes TDD para uma função inverter_string(texto) que devolve a string invertida.

inverterString.py

```
def inverterString(texto):
    return texto[::-1]
```

test_inverterString.py

```
import unittest
from inverterString import inverterString

class test_inverterString(unittest.TestCase):
    def test_palavra(self):
        self.assertEqual(inverterString("Python"), "nohtyP")

    def test_vazia(self):
        self.assertEqual(inverterString(""), "")

    def test_espacos(self):
        self.assertEqual(inverterString("a b"), "b a")
```

Atividade 3

Criar função valida_cpf(cpf) que verifica se a string possui 11 dígitos numéricos.

(Não precisa validar matematicamente, apenas estrutura).

validaCPF.py

```
def validaCPF(cpf):
    return cpf.isdigit() and len(cpf) == 11
```

test_validaCPF.py

```
import unittest
from validaCPF import validaCPF

class test_validaCPF(unittest.TestCase):
    def test_cpf_valido(self):
        self.assertTrue(validaCPF("12345678901"))

    def test_cpf_curto(self):
        self.assertFalse(validaCPF("12345"))

    def test_cpf_com_letras(self):
        self.assertFalse(validaCPF("12345abc901"))

    def test_cpf_vazio(self):
        self.assertFalse(validaCPF(""))
```