

Testes e TDD

1 Configurações iniciais

As configurações contidas nesse arquivo definem as configurações do vs code para compilador e para code runner

settings.json

```
1 {
2     "editor.fontSize": 14,
3     "editor.fontFamily": "Cascadia code",
4     "editor.lineHeight": 20,
5     "editor.quickSuggestions": {
6         "other": true,
7         "comments": false,
8         "strings": true
9     },
10    "terminal.integrated.fontSize": 14,
11    "terminal.integrated.fontFamily": "Cascadia Code",
12    "[python]": {
13        "diffEditor.ignoreTrimWhitespace": false,
14        "editor.formatOnType": true,
15        "editor.wordBasedSuggestions": "off",
16        "editor.formatOnSave": true
17    },
18    "python.defaultInterpreterPath": "venv/bin/python3",
19    "python.linting.flake8Enabled": true,
20    "python.linting.mypyEnabled": true,
21    "code-runner.executorMap": {
22        "python": "clear && python3 -u"
23    },
24    "python.analysis.extraPaths": [
25        "./src"
26    ]
27 }
```

2 Assertions

Verifica se o caractere de entrada é válida.

assertions declaração e checagem

```
1 assert isinstance(x, (int, float)), 'x precisa ser int ou float'
```

Uso no código

```
1 def soma(x, y):
2     assert isinstance(x, (int, float)), 'x precisa ser int ou float'
3     assert isinstance(y, (int, float)), 'y precisa ser int ou float'
4     return x + y
```

2.1 desativação de assertions

desativação shell

```
1 python3 -O <nome_arquivo>
```

3 Testes via Doctests

3.1 Implementação

Listing:

```
1 def subtrai(x, y):
2     """
3     >>> subtrai(30 , 20)
4         10
5     >>> soma('a', 20)
6     TypeError
7
8     """
9     assert isinstance(x, (int, float)), 'x precisa ser int ou float'
10    assert isinstance(y, (int, float)), 'y precisa ser int ou float'
11    return x - y
12
13    if __name__ == "__main__":
14        import doctest
15        doctest.testmod(verbose= True)
```

4 Testes pelo unittest

teste para calculadora.py

```
1 try:
2     import sys
3     import os
4     sys.path.append(
5         os.path.abspath(
6             os.path.join(
7                 os.path.dirname(__file__),
8                 '../src'
9             )
10        )
11    )
12 except:
13     raise
14 import unittest
15 from calculadora import soma
16
17 class TestCalculadora(unittest.TestCase):
18     def test_soma_5_e_5_deve_retornar_10(self):
19         self.assertEqual(soma(5, 5), 10)
20
21     def test_soma_5_negativo_e_5_deve_retornar_0(self):
22         self.assertEqual(soma(-5, 5), 0)
23
24     def teste_soma_varias_entradas(self):
25         x_y_saidas =(
26             (10, 10, 20),
27             (5, 5, 10),
28             (1.5, 1.5, 3.0),
29             (10, 10, 20),
30             (10, 20, 30),
31
32         )
33         for x_y_saida in x_y_saidas:
34             with self.subTest(x_y_saida = x_y_saida):
35                 x, y, saida = x_y_saida
36                 self.assertEqual(soma(x, y), saida)
37
38     def soma_x_nao_int_ou_float_retorna_assertionerror(self):
39         with self.assertRaises(AssertionError):
40             soma(2, "a")
41
42     def soma_y_nao_int_ou_float_retorna_assertionerror(self):
43         with self.assertRaises(AssertionError):
44             soma("a", 2)
45
46
47 if __name__ == "__main__":
48     unittest.main(verbosity=2)
```

5 Código final

main.py

```
1 from calculadora import soma
2
3
4 if __name__ == "__main__":
5     x = 5
6     y = 6
7
8     """
9     print(soma(x, y))
10    print(soma(6, 7))
11    print(soma(80, 56))
12    print(soma(89, 90))
13    """
14    try:
15        soma('15', 10)
16    except TypeError as e:
17        print(f"0 erro foi {e}")
```

calculadora

```
1
2 def soma(x, y):
3     """
4     >>> soma(30 , 20)
5         50
6     >>> soma('a', 20)
7     """
8     assert isinstance(x, (int, float)), 'x precisa ser int ou float'
9     assert isinstance(y, (int, float)), 'y precisa ser int ou float'
10    return x + y
11
12 if __name__ == "__main__":
```

```
13     pass
14
15 def subtrai(x, y):
16     """
17     >>> subtrai(30 , 20)
18         10
19     >>> soma('a', 20)
20     TypeError
21
22     """
23     assert isinstance(x, (int, float)), 'x precisa ser int ou float'
24     assert isinstance(y, (int, float)), 'y precisa ser int ou float'
25     return x - y
26
27 if __name__ == "__main__":
28     import doctest
29     doctest.testmod(verbose= True)
```

6 TDD

Baconcomovos

```
1  """
2  1. receber um numero inteiro
3  2. saber se o numero e multiplo de 3 e 5
4  bacon com ovos
5  3. saver se o numero multiplio somente de 3
6  bacon
7  3. saver se o numero multiplio somente de 5
8  ovos
9  2. saber se onumero nao e multiplo de 3 e 5
10 passa fome
11
12 """
13
14 def bacon_com_ovos(n):
15     assert(isinstance(n, int))
16     if(n % 5 == 0 and n % 3 ==0):
17         return "Bacon com ovos"
18     elif(n%3 == 0):
19         return "Bacon"
20     elif(n%5==0):
21         return "Ovos"
22     return "Passa fome"
```

testeBaconcomovos

```
1  """
2  TDD - Test Driven Development
3
4  Red
5  Parte 1 - criar o teste e ver falhar
6
7  Green
8  Parte2 - criar o codigo e ver oteste passar
9
10 refactor
11 Parte3 - melhorar meu codigo
12
13 """
14 try:
15     import sys
16     import os
17     sys.path.append(
18         os.path.abspath(
19             os.path.join(
20                 os.path.dirname(__file__),
21                 '../src'
22             )
23         )
24     )
25 except:
26     raise
27 import unittest
28 from baconcomovos import bacon_com_ovos
29
30 class TesteBaconComOvos(unittest.TestCase):
31     def test_bacon_com_ovos_deve_levantar_assertion_error_se_nao_receber_int(self):
32         with self.assertRaises(AssertionError):
33             bacon_com_ovos(' ')
34
35     def test_bacon_com_ovos_deve_retornar_bacon_com_ovos_se_entrada_for_multiplo_de_3_e_5(self):
36         entradas = (15, 30, 45, 60)
37         saida = "Bacon com ovos"
38         for entrada in entradas:
39             self.assertEqual(
40                 bacon_com_ovos(entrada),
41                 saida,
42                 msg=f"{entrada} nao retornou {saida}"
```

```
43         )
44
45     def test_bacon_com_ovos_deve_retornar_passar_fome_se_entrada_nao_for_multiplo_de_3_e_5(self):
46         entradas = (1,2,4, 7, 8)
47         saida = "Passa fome"
48         for entrada in entradas:
49             self.assertEqual(
50                 bacon_com_ovos(entrada),
51                 saida,
52                 msg=f"{entrada} nao retornou {saida}"
53             )
54
55
56     def test_bacon_com_ovos_deve_retornar_bacon_se_entrada_nao_for_multiplo_de_3(self):
57         entradas = (3, 6, 9, 12, 18)
58         saida = "Bacon"
59         for entrada in entradas:
60             self.assertEqual(
61                 bacon_com_ovos(entrada),
62                 saida,
63                 msg=f"{entrada} nao retornou {saida}"
64             )
65
66     def test_bacon_com_ovos_deve_retornar_ovos_se_entrada_nao_for_multiplo_de_5(self):
67         entradas = (5, 10, 20, 25)
68         saida = "Ovos"
69         for entrada in entradas:
70             self.assertEqual(
71                 bacon_com_ovos(entrada),
72                 saida,
73                 msg=f"{entrada} nao retornou {saida}"
74             )
75
76
77
78
79
80 if __name__ == "__main__":
81     unittest.main(verbosity = 2)
```

7 TDD pessoa.py

link para a documenetção unittest documentação

testePessoa.py

```
1  """
2  class Pessoa:
3      __init__
4          nome str
5          sobrenome str
6          dados_obtidos bool
7
8      API:
9          obter_todos_os_dados -> method
10             ok
11             404
12             (dados_obtidos se torna True se dados_obtidos com sucesso)
13  """
14  try:
15      import sys
16      import os
17      sys.path.append(
18          os.path.abspath(
19              os.path.join(
20                  os.path.dirname(__file__),
21                  '../src'
22              )
23          )
24      )
25  except:
26      raise
27
28  import unittest
29  from unittest.mock import patch
30  from pessoa import Pessoa
31
32  class TestPessoa(unittest.TestCase):
33      def setUp(self):
34          self.p1 = Pessoa('Luiz','Otavio')
35          self.p2 = Pessoa('Marcos', 'Tome')
36
37      def test_pessoa_attr_nome_tem_valor_correto(self):
38          self.assertEqual(self.p1.nome , 'Luiz')
39          self.assertEqual(self.p2.nome , 'Marcos')
40
41      def test_pessoa_attr_nome_e_str(self):
42          self.assertIsInstance(self.p1.nome , str)
43          self.assertIsInstance(self.p2.nome , str)
44
```

```
45     def test_pessoa_attr_sobrenome_tem_valor_correto(self):
46         self.assertEqual(self.p1.sobrenome , 'Otavio')
47         self.assertEqual(self.p2.sobrenome , 'Tome')
48
49     def test_pessoa_attr_sobrenome_e_str(self):
50         self.assertIsInstance(self.p1.sobrenome , str)
51         self.assertIsInstance(self.p2.sobrenome , str)
52
53     def test_pessoa_attr_dados_obtidos_inicia_false(self):
54         self.assertFalse(self.p1.dados_obtidos)
55         self.assertFalse(self.p2.dados_obtidos)
56
57     def test_obter_todos_os_dados_sucesso_ok(self):
58         with patch('requests.get') as fake_request:
59             fake_request.return_value.ok = True
60             self.assertEqual(self.p1.obter_todos_os_dados(), 'CONECTADO')
61             self.assertTrue(self.p1.dados_obtidos)
62
63         with patch('requests.get') as fake_request:
64             fake_request.return_value.ok = True
65             self.assertEqual(self.p2.obter_todos_os_dados(), 'CONECTADO')
66             self.assertTrue(self.p2.dados_obtidos)
67
68     def test_obter_todos_os_dados_falha_404(self):
69         with patch('requests.get') as fake_request:
70             fake_request.return_value.ok = False
71             self.assertEqual(self.p1.obter_todos_os_dados(), 'ERRO 404')
72             self.assertFalse(self.p1.dados_obtidos)
73
74         with patch('requests.get') as fake_request:
75             fake_request.return_value.ok = False
76             self.assertEqual(self.p2.obter_todos_os_dados(), 'ERRO 404')
77             self.assertFalse(self.p2.dados_obtidos)
78
79
80
81     def test_obter_todos_os_dados_sucesso_e_falha_sequencial(self):
82         with patch('requests.get') as fake_request:
83             fake_request.return_value.ok = True
84
85             self.assertEqual(self.p1.obter_todos_os_dados(), 'CONECTADO')
86             self.assertTrue(self.p1.dados_obtidos)
87
88             fake_request.return_value.ok = False
89             self.assertEqual(self.p1.obter_todos_os_dados(), 'ERRO 404')
90             self.assertTrue(self.p1.dados_obtidos)
91         with patch('requests.get') as fake_request:
92             fake_request.return_value.ok = True
93
94             self.assertEqual(self.p2.obter_todos_os_dados(), 'CONECTADO')
95             self.assertTrue(self.p2.dados_obtidos)
96
97             fake_request.return_value.ok = False
98             self.assertEqual(self.p2.obter_todos_os_dados(), 'ERRO 404')
99             self.assertTrue(self.p2.dados_obtidos)
100
101
102 if __name__ == "__main__":
103     unittest.main(verbosity = 2)
```

8 importando módulos para dentro do codigo

Rodando todos os testes `python3 -m unittest -v`

importação

```
1     try:
2     import sys
3     import os
4     sys.path.append(
5         os.path.abspath(
6             os.path.join(
7                 os.path.dirname(__file__),
8                 '../src'
9             )
10        )
11    )
12 except:
13     raise
```

9 type hints e mypy

rodando testes no arquivo especificado `mypy nomeArquivo`

A aula referente mypy do curso se referia a pycharm, e portanto as tipagens pareciam não funcionar

typehints.py

```
1 def add_number(n1:int , n2:float) -> int:
```

```
2         if not n1:
3             return 0
4
5         return n1 + n2
6 initial: tuple[str, ...] = ("one", "two")
7
8 class Parent:
9     def capitalise(self, text:str) -> str:
10         return text.upper()
11
12 class child(Parent):
13     def capitalise(self, text: str) -> str:
14         return str(text).upper()
```