Unit Test (Python) Cheat Sheet
V2021.02.15
(Dr Yan Xu)

Unit Test Skeleton

- ➔ The skeleton is the same for function-based tests and class-based tests
- ➔ setUp() and tearDown() executes once per test
- ➔ Tests are ordered based on name

```
class TestXxx(unittest.TestCase):

    def setUp(self):
        ...

    def tearDown(self):
        ...

    def test_yyy_description1(self):
        ...

    def test_yyy_description2(self):
        ...
```

Unit Test Assertions

- ➔ You have to use unittest assertion functions
- ➔ Most frequently used assertion functions:
  - self.assertEqual(first, second, msg=None)
  - assertTrue(expr, msg=None)
  - assertFalse(expr, msg=None)
  - assertIn(member, container, msg=None)
  - assertIsInstance(obj, cls, msg=None)
  - assertRaises(exception, callable, *args, **kwds)
    - self.assertRaises(RuntimeError, lambda: func(a))
  - assertRaisesRegex(exception, regex, *, msg=None)
  - assertIsNone(expr, msg=None)
  - assertIsNotNone(expr, msg=None)

Unit Test Skip Decorator

- ➔ @unittest.skip(reason)
- ➔ @unittest.skipIf(condition, reason)
- ➔ @unittest.skipUnless(condition, reason)
- ➔ Example:

```
@unittest.skipIf(not TEST_CUDA, 'CUDA not available')
def test_pack_sequence_batch_sizes_throw(self):
    with self.assertRaisesRegex(ValueError, r"batch_sizes should always be on CPU"):
        m = nn.LSTM(3, 4, bidirectional=True, num_layers=2).to('cuda')
        a = torch.rand(5, 3, device='cuda')
        b = torch.tensor([1, 1, 1, 1, 1], device='cuda')
        input = nn.utils.rnn.PackedSequence(a, b)
```

Unit Test Scenarios

- ➔ Option 1: List scenarios in functions directly

```
# None indexing
self.assertEqual(reference[2, None], reference[2].unsqueeze(0))
self.assertEqual(reference[2, None, None], reference[2].unsqueeze(0).unsqueeze(0))
self.assertEqual(reference[2:4, None], reference[2:4].unsqueeze(1))
self.assertEqual(reference[None, 2, None, None], reference.unsqueeze(0)[:, 2].unsqueeze(0).unsqueeze(0))
self.assertEqual(reference[None, 2:5, None, None], reference.unsqueeze(0)[:, 2:5].unsqueeze(2).unsqueeze(2))

# indexing 0-length slice
self.assertEqual(torch.empty(0, 5, 5), reference[slice(0)])
self.assertEqual(torch.empty(0, 5), reference[slice(0), 2])
self.assertEqual(torch.empty(0, 5), reference[2, slice(0)])
self.assertEqual(torch.tensor([]), reference[2, 1:1, 2])
```

- ➔ Option 2: Define scenarios externally & for-loop

Run Unit Test

- ➔ python -m unittest tests
- ➔ python -m unittest tests/test_something.py

Unit Test or Not?

- ➔ surprisingly, many top ML projects have no tests
- ➔ manual common sense tests are used instead