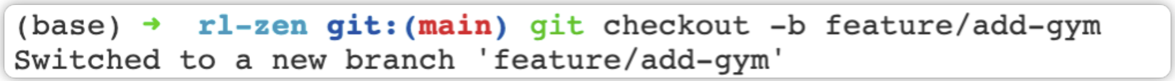
git 流程规范

1. **克隆** master 分支到自己的PC `*git clone xxx*`
2. **创建一个本地分支**，将分支名字叫做 *feature/some-short-title* 你所有开发的工作将要在这个分支上进行

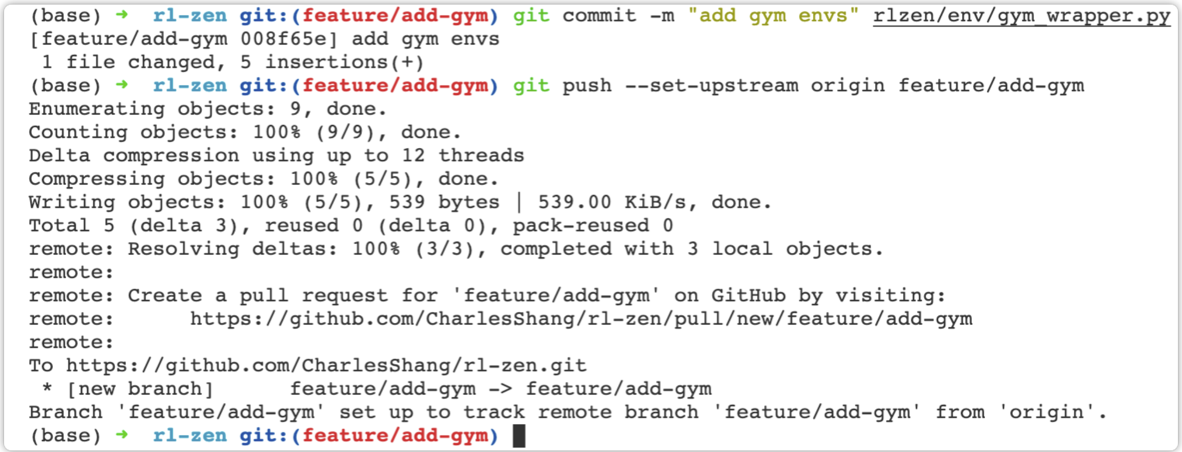
   例如：*git checkout -b feature/add-gym*

   ```
   (base) ➜  rl-zen git:(main) git checkout -b feature/add-gym
   Switched to a new branch 'feature/add-gym'
   ```

3. **在本地进行开发工作**，然后定期将本地修改 *commit* 并 *push* 到 git 服务器：*git commit -m "your modification summary" folder1/file1 folder2/file*

   例如：*git commit -m "add gym envs" rlzen/env/gym_wrapper.py*

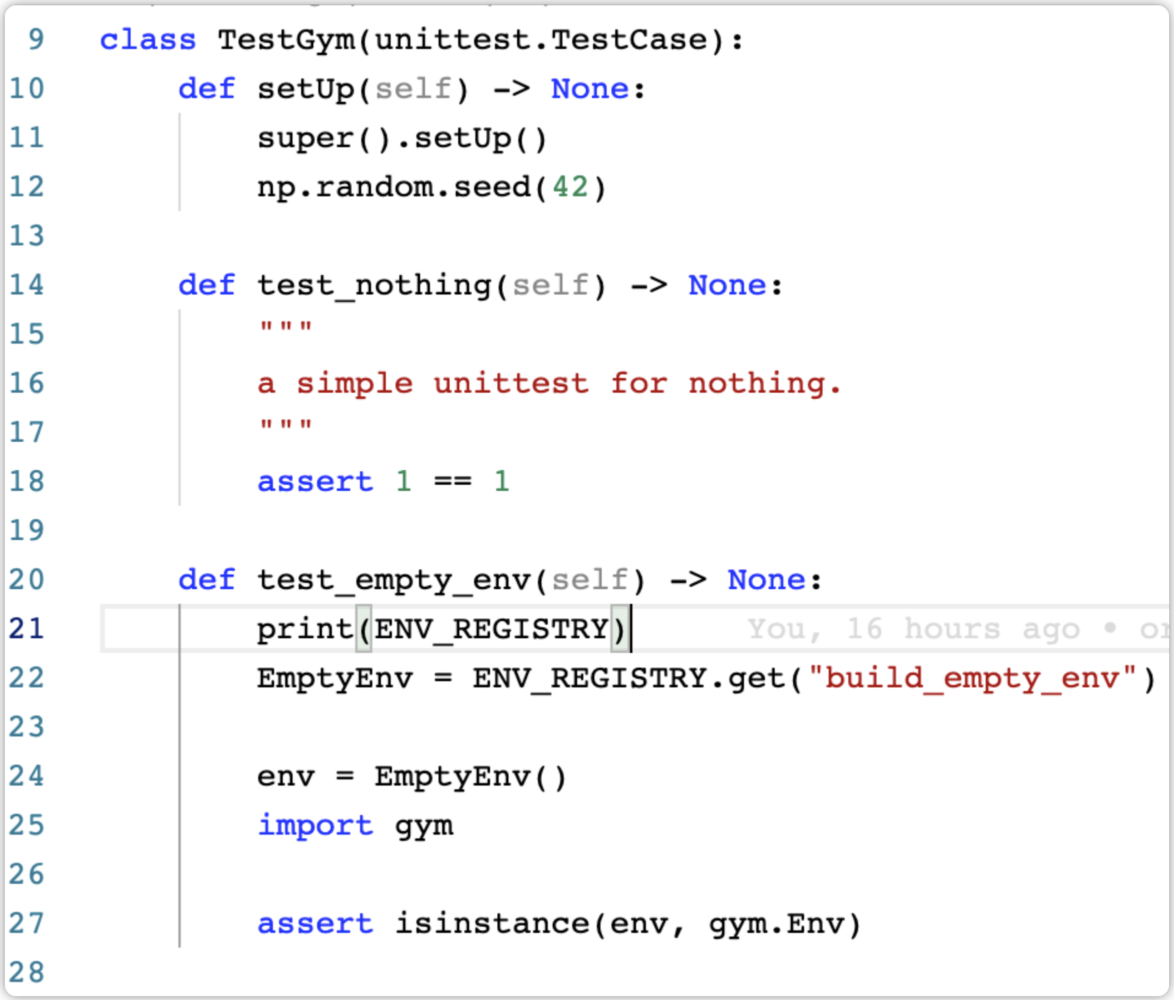
   

   ```
   (base) ➜  rl-zen git:(feature/add-gym) git commit -m "add gym envs" rlzen/env/gym_wrapper.py
   [feature/add-gym 008f65e] add gym envs
    1 file changed, 5 insertions(+)
   (base) ➜  rl-zen git:(feature/add-gym) git push --set-upstream origin feature/add-gym
   Enumerating objects: 9, done.
   Counting objects: 100% (9/9), done.
   Delta compression using up to 12 threads
   Compressing objects: 100% (5/5), done.
   Writing objects: 100% (5/5), 539 bytes | 539.00 KiB/s, done.
   Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
   remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
   remote:
   remote: Create a pull request for 'feature/add-gym' on GitHub by visiting:
   remote:      https://github.com/CharlesShang/rl-zen/pull/new/feature/add-gym
   remote:
   To https://github.com/CharlesShang/rl-zen.git
    * [new branch]      feature/add-gym -> feature/add-gym
   Branch 'feature/add-gym' set up to track remote branch 'feature/add-gym' from 'origin'.
   (base) ➜  rl-zen git:(feature/add-gym) █
   ```

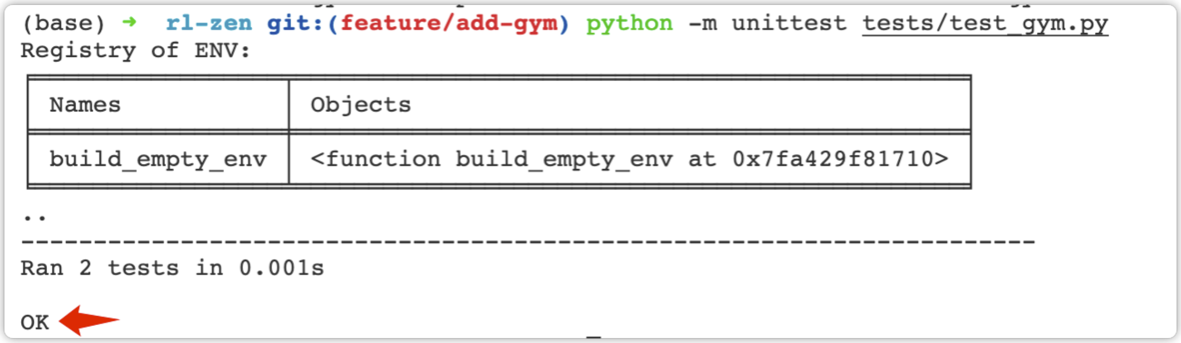
4. **测试你的代码**，一旦你完成了feature的功能，需要在本地测试，可以参考单元测试 *unittest*

   例如：

```
# 1. 继承 unittest.TestCase
# 2. setUp 函数中实现测试你的 code 的所需要的情景、数据、初始化等（如果不需要可以跳过）
# 3. 实现 test_xxx 函数，每个函数是一个测试情景，测试时会自动运行以 test_ 开头的函数
# 4. 运行 python -m unittest <your-unittest-code.py> 返回 OK 或者是 Failure
```



```python
class TestGym(unittest.TestCase):
    def setUp(self) -> None:
        super().setUp()
        np.random.seed(42)

    def test_nothing(self) -> None:
        """
        a simple unittest for nothing.
        """
        assert 1 == 1

    def test_empty_env(self) -> None:
        print(ENV_REGISTRY)        You, 16 hours ago • or
        EmptyEnv = ENV_REGISTRY.get("build_empty_env")

        env = EmptyEnv()
        import gym

        assert isinstance(env, gym.Env)

```



```
(base) ➜  rl-zen git:(feature/add-gym) python -m unittest tests/test_gym.py
Registry of ENV:
+-----------------+--------------------------------------------+
| Names           | Objects                                    |
+-----------------+--------------------------------------------+
| build_empty_env | <function build_empty_env at 0x7fa429f81710> |
+-----------------+--------------------------------------------+
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

参考后面一篇

1. **创建PR** Pull Request & Code Review：
2. **Complete the pull request** by merging into master.
3. **Checkout** the **master** branch on your local machine.
4. **Pull** the latest code, and you're ready for your next feature branch.