

AI 510 Artificial Intelligence of Cloud Computing

HOS10A Python Package

07/02/2024 Reviewed by Anh Nguyen

07/06/2024 Reviewed by Naveena Moddu

School of Technology and Computing (STC) @City University of Seattle (CityU)

Before You Start

- The directory path shown in the screenshots may be different from yours.
- Some steps are not explained in the tutorial. If you are not sure what to do:
 - Consult the resources listed below.
 - If you cannot solve the problem after a few tries, please contact the student worker through the MS Teams course channel.

Learning Outcomes

- Students will be able to learn:
 - Introduction to Python package structure
 - Introduction to Python test
 - Importing Modules

Resource

- Noah G., Alfredo D. (2021). Practical MLOps. O'Reilly Media, Inc.
pytest. (n.d.). pytest fixtures: <https://docs.pytest.org/en/6.2.x/fixture.html>

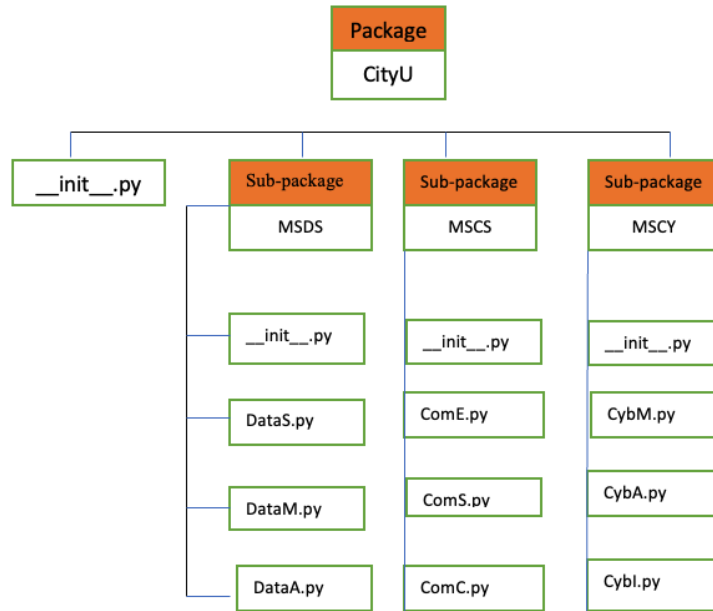
Introduction to Python package structure

Python packages are the organization of files in different folders and subfolders based on criteria so that they can be managed easily and efficiently. For example, if we create a CityU python package, we will keep all our department “app” in the CityU folder and subfolder according to the app requirements.

A Python module may contain several classes, functions, variables, and so on whereas a Python package can contain several modules. So, Python packages are folders that contain different modules as a file.

A Python package directory should contain a file named `__init__.py` for the Python interpreter to consider the folder as a package. It also specifies the resources to be imported from the module. If the `__init__.py` is empty this means that all the functions of the modules will be imported.

If we develop the game application, the structure of packages and modules might be like the following example structure.



In this module, we will look at how Python class or function base modules are created along with unit testing for each implemented function.

Introduction to pytest

Python also has unit tests like the junit test called pytest. The pytest package looks for the following criteria and executes the unit test.

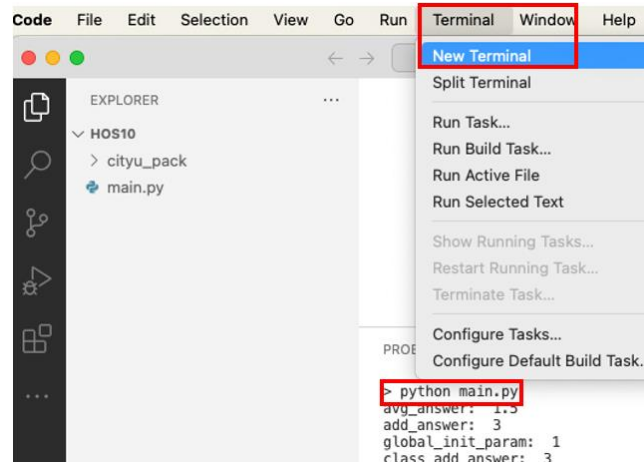
- “tests” folder which contains all of the test Python scripts
- Test Python scripts starts with “test”(ex. test_{module_name}.py)
- Test functions inside of Test Python scripts start with “test” (ex. test_{module_name}.py with `def test_some_function()`)
- pytest annotated functions(ex. `@pytest.fixture`)

Setup environment

Step 1) Open Terminal and execute the main.py:

```
python main.py
```

Take a **screenshot** of the executed output to be submitted for HOS10A submission.



Step 3) We will review the contents of the “class type” of the Python module.

- a. Open the “cityu_pack/sub_class_pack/class_module.py” file. We can see that the module contains a class name, initialization function, and add function.

```
class_module.py x
cityu_pack > sub_class_pack > class_module.py > ...
1 class ClassModule:
2     def __init__(self, global_init_param=None):
3         self.global_init_param = global_init_param
4         # we can also simply do a "pass" if we have
5         # pass
6
7     def add(self, x,y):
8         return x + y
```

Note that the “self” is the keyword that indicates the function (ex. __init__ or add) or variable(ex. self.global_init_param) belongs to the class. When other execution flow initializes this class, the “self” keyword specified functions or variables can be used by this class initialized object calls.

- b. Open “main.py” in the HOS10A folder. We can see that the class “ClassModule” is initialized (ex. ClassModule(global_init_param=1)) and the initialized class object variables and functions can be accessed due to the “self” keyword in the ClassModule.

```
main.py x
main.py > ...
1 #import class structured module
2 from cityu_pack.sub_class_pack.class_module import ClassModule
3
4 temp_object = ClassModule(global_init_param=1)
5 print("global_init_param: ", temp_object.global_init_param)
6
7 class_add_answer = temp_object.add(1,2)
8 print("class_add_answer: ", class_add_answer)
```

Step 4) We will review the contents of the “function type” of the Python module.

- a. Open the “cityu_pack/sub_func_pack/func_module.py” file. We can see that the module contains functions and is very simple.

```
func_module.py ×  
cityu_pack > sub_func_pack > func_module.py > ...  
1  def add(x,y):  
2      return x+y  
3  
4  def avg(x,y):  
5      return add(x,y)/2  
~
```

Note that when a function is a one-time call utility-like function, the “function type” module structure is used.

- b. Open “main.py” in the HOS10A folder. The second part of the “main.py” script contains two ways of importing the function module. One explicitly imports just the “avg” function and the other one imports the whole (ex. *) module. Note that the first importing style is recommended as engineers can follow through the code more easily. The latter case is often used in framework automation where it gives the capability to external users to create custom functions.

```
func_module.py  main.py ×  
main.py > ...  
12  # explicit import such as average  
13  from cityu_pack.sub_func_pack.func_module import avg  
14  avg_answer = avg(1,2)  
15  print("avg_answer: ", avg_answer)  
16  
17  #import all in function structured module  
18  from cityu_pack.sub_func_pack.func_module import *  
19  add_answer = add(1,2)  
20  print("add_answer: ",add_answer)
```

Step 5) In the terminal run the following command to install the pytest package.

```
pip3 install pytest
```

Step 6) After installing the package, navigate into the “cityu_pack” folder and execute pytest with the following command.

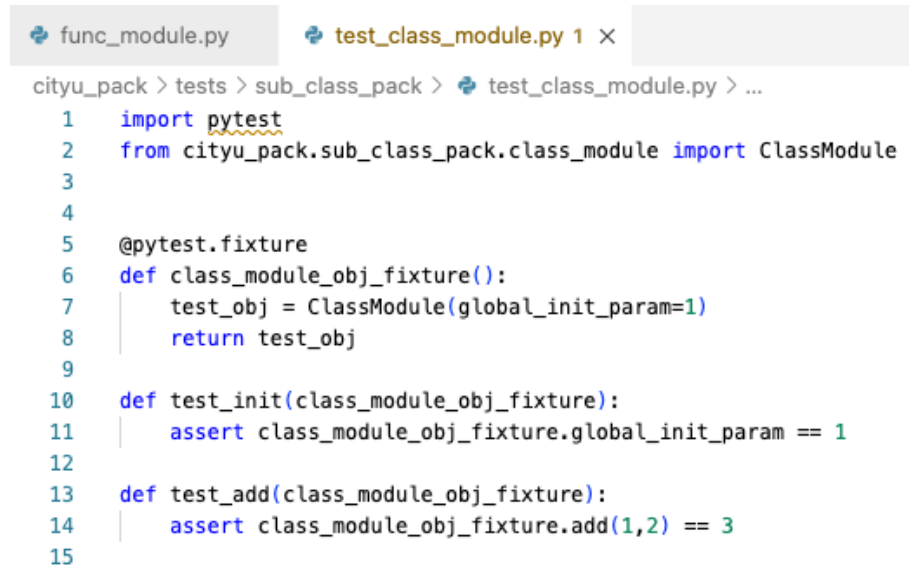
```
cd cityu_pack
```

```
pytest -s
```

```
plugins: anyio-4.2.0  
collected 3 items  
  
tests/sub_class_pack/test_class_module.py ..  
tests/sub_func_pack/test_func_module.py .  
  
===== 3 passed in 0.00s =====
```

We can see that there were 3 tests executed

Step 7) Open “cityu_pack/tests/sub_class_pack/test_class_module.py” to review test functions.



```
cityu_pack > tests > sub_class_pack > test_class_module.py > ...
1  import pytest
2  from cityu_pack.sub_class_pack.class_module import ClassModule
3
4
5  @pytest.fixture
6  def class_module_obj_fixture():
7      test_obj = ClassModule(global_init_param=1)
8      return test_obj
9
10 def test_init(class_module_obj_fixture):
11     assert class_module_obj_fixture.global_init_param == 1
12
13 def test_add(class_module_obj_fixture):
14     assert class_module_obj_fixture.add(1,2) == 3
15
```

Note that the pytest fixture is used to create and initialize ClassModule. The fixture is also used when we need to create dummy mock objects. The dummy mock fixtures may simulate communicating external resources such as servers. In this test, the fixture is used for both “__init__” and “add” functions test from “class_module.py”. The “assert” is used in general Python programming to test and throw an error if the condition is false.

Step 8) Save the executed folder, and screenshots and submit the entire HOS10A folder for this HOS.

HOS submission instructions:

1. Please install the GitHub Desktop: https://cityuseattle.github.io/docs/git/github_desktop/
2. Clone, organize, and submit your work through GitHub Desktop:
<https://cityuseattle.github.io/docs/hoporhos>