For the small program, I wrote a test code. In the test code, I used the unittest unit test framework that comes with python. (For the code, see test\_DesktopPet.py)

When testing the code, I first run DesktopPet.py. I modified the original basic code so that it can generate specified pets as required to facilitate testing. In its method LoadPetImages(), I set up the function of obtaining the iconpath according to the entered pet serial number, which can store the iconpath in a txt file. Take number 39 as an example. After running DesktopPet.py, fill in 39 and summon the number 39 pet. At this time, the path has been stored in the txt file. I will run test\_DesktopPet.py again. In this code, the estimated path number 39 has been set. Through the code, compare the expected path and the obtained path in txt to see if they are the same to determine whether the path is read correctly.

At the same time, I conducted a manual test. This program can successfully summon different pets according to the number entered by the user, and the position of the pet on the desktop is random. As time changes, pets will have different actions and show different postures. Press the left button of the mouse on the position of the pet, and the pet will be bound to the position of the mouse. When the mouse moves, the pet will also move; when the mouse is released, the binding between the mouse and the pet is cancelled. When the program is running, there will be a tray icon corresponding to the pet, right-click to achieve the exit function. Through manual testing, we can see that the functions are basically correct.