Analysis Report for Assignment 1 - Agents

- Name: Tsang Kwong Ngan
- Student ID: 301328893
- Class: COMP237004_2023F
- Date: Sep 17, 2023

Objectives

This report describes how the Python program "blinddog_simple_reflex.py" was modified to meet the requirements for Assignment 1, and shows the execution results of the finished program. It also includes an UML class digram showing all the classes defined in the program and the relationships between them.

Descriptions

All modifications made to the "blinddog_simple_reflex.py" file are marked with comments showing their related requirement numbers as stated in the assignment instructions document. The changes are as follows:

1. Add a new food item called **chicken** at location 9 in the **Park**.

```
chicken = Food()
park.add_thing(chicken, 9)
```

2. Add a new class called **Person** which is a subclass of **Thing**.

```
class Person(Thing):
    def __init__(self, name=None):
        if name is not None:
            self.__name__ = name
```

3. Create two instances of **Person** with their names as my first name and last name respectively. They are added at location 3 and 9 in the **Park**.

```
person1 = Person("Vincent")
person2 = Person("Ngan")
park.add_thing(person1, 3)
park.add_thing(person2, 12)
```

4. Add code to model the behavior a new receipt that represents how a **BlindDog** would bark at a **Persion**. The changes can be found in the program marked with comments as 4a, 4b and 4c.

   (a) Add a condition in **execute_action()** of class **Park** to handle the "bark" action:

```
#
# 4a - Add a condition to handle the action "bark".
# When the agent (BlindDog) encounters a thing which is of
# type Person, it calls the bark() method to check
# if the bark() supports this action. If yes, the program
# goes on to move the person up one location, and move
# the agent down one location.
#
elif action == "bark":
    items = self.list_things_at(agent.location, tclass=Person)
    if len(items) != 0:
        person = items[0]
        if agent.bark(person):  # Have the dog bark at the first person
            print('{} barked at {} at location: {}'
                  .format(str(agent)[1:-1], str(person)[1:-1], agent.location))
            # Delete person from the Park after:
            self.delete_thing(person)
            # Add the person back to Park at (agent.location - 1):
            self.add_thing(person, agent.location - 1)
            print('{} was scared by {} and moved to location: {}'
                  .format(str(person)[1:-1], str(agent)[1:-1], agent.location - 1))
            print('{} moved down to location: {}'
                  .format(str(agent)[1:-1], agent.location + 1))
    agent.movedown()
```

   (b) Add a function called **bark()** in the **BlindDog** class:

```
#
# 4b - Add a new method called bark() for BindDog to represent
# the bark behavior of BlindDog.
#
def bark(self, thing):
    """
    This method models the 'bark' behavior of BlindDog.
    It returns True if the input parameter thing is an instance
    of Person; otherwise it returns False.
    """
    if isinstance(thing, Person):
        return True
    return False
```

   (c) Add a condition in function **program()** to handle a percept as Person:

```
def program(percepts):
    """Returns an action based on the dog's percepts"""
    for p in percepts:
        if isinstance(p, Food):
            return 'eat'
        elif isinstance(p, Water):
            return 'drink'
        #
        # 4c - Add a percept that a Person can be barked at by the BlindDog
        #
        elif isinstance(p, Person):
            return 'bark'
    return 'move down'
```

5. The program is run by setting the number of steps to 18. The status of the **Park** and the **BlindDog** are printed out.

```
    #
    # 5 - Run the park for 18 steps
    #
    park.run(18)

    print("Park is done: {}".format(park.is_done()))

    no_edibles = not any(
        isinstance(thing, Food) or isinstance(thing, Water)
        for thing in park.things
    )
    no_agents_alive = not any(
        agent.is_alive() for agent in park.agents
    )
    if no_edibles:
        print("{} starved at location: {}".format(str(dog)[1:-1], dog.location))
    if no_agents_alive:
        print("{} died at location: {}".format(str(dog)[1:-1], dog.location))
```

Additional Details

1. The main logic of the program is moved to a new function called **main()**. This is to avoid the main logic being run when the program is used as a library instead of a main program.

```
def main():
    park = Park()
    dog = BlindDog(program)
    dogfood = Food()
    ...
```

2. Add a condition to check if this program is being used as a main program before it is run; otherwise, no execution of the main logic will be carried out.

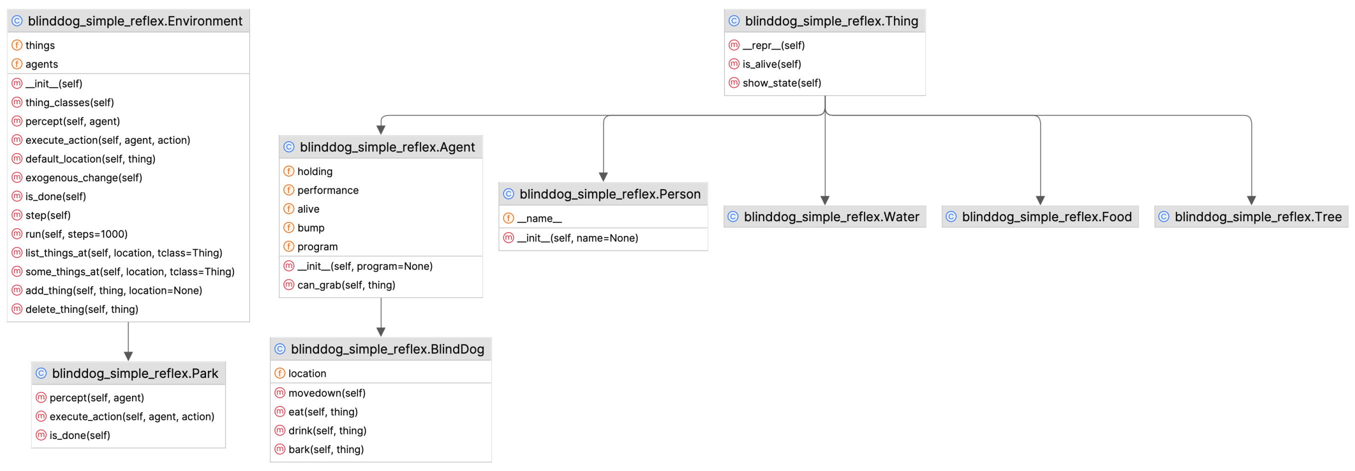
```python
if __name__ == "__main__":
    main()
```

3. Add a unit test program called "blinddog_simple_reflex_test.py" to provide unit tests for program to make sure the program meets the requirements from unit test point of view. The unit test program includes the following tests:

```python
def test_that_Person_can_be_instantiated_with_a_name(self):

def test_that_BlindDog_barks_at_Person(self):

def test_that_BlindDog_does_not_bark_at_things_other_than_Person(self):

def test_that_BlindDog_eats_food_if_there_is_food_at_its_current_location(self):

def test_that_BlindDog_drinks_water_if_there_is_water_at_its_current_location(self):

def test_that_BlindDog_moves_to_next_location_after_it_drinks_water_or_eats_a_food_ite

def test_that_as_long_as_there_is_food_in_the_park_the_BlindDog_will_continue_move_dow

def test_that_BlindDog_does_not_eat_a_Person(self):

def test_that_when_a_Person_meets_a_dog_he_will_move_up_one_location(self):
```

These tests may not be exhaustive, but they pretty much cover all the majority parts of the program. The coverage test shows that the unit tests cover 75% of the code.

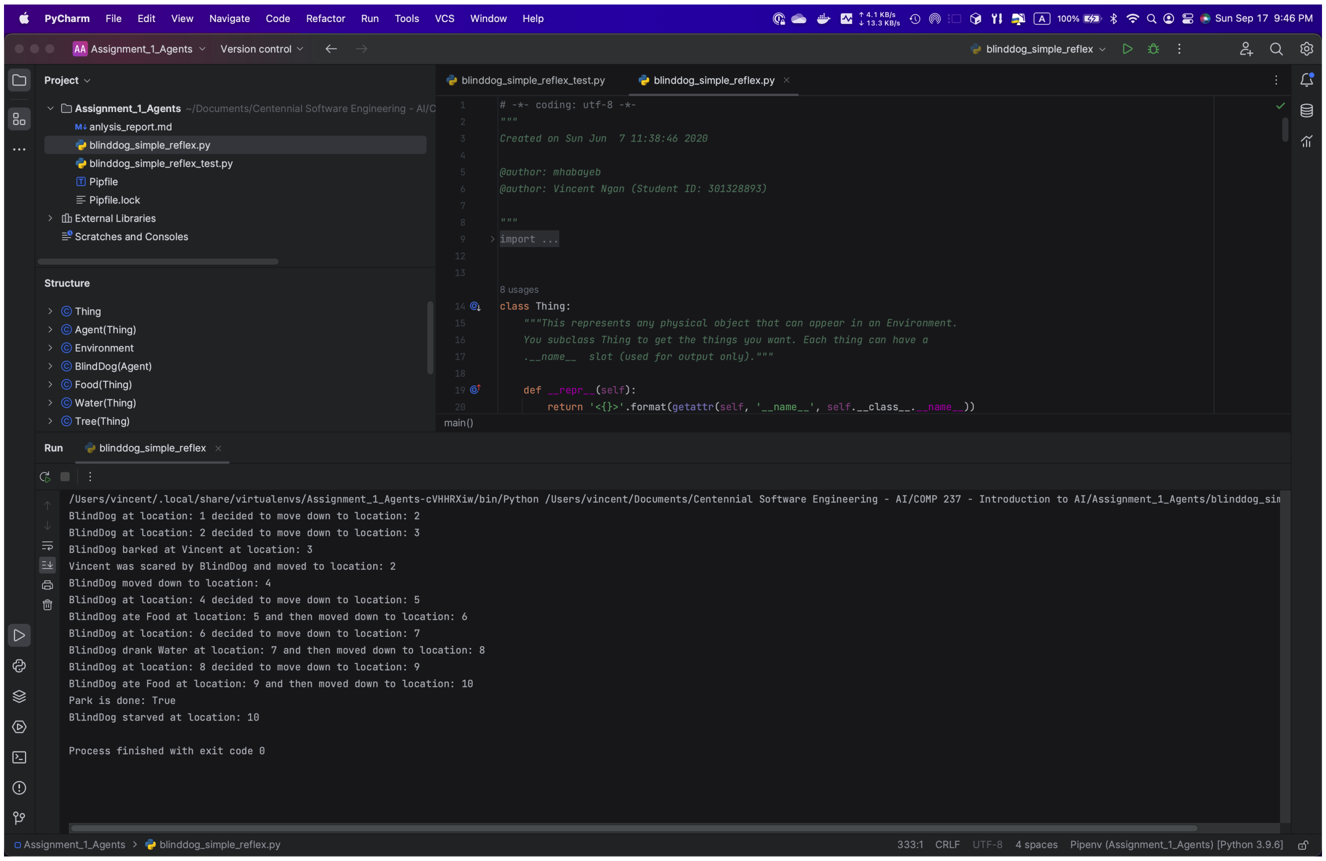
Class Diagram



blinddog_simple_reflex.Environment
things
agents
__init__(self)
thing_classes(self)
percept(self, agent)
execute_action(self, agent, action)
default_location(self, thing)
exogenous_change(self)
is_done(self)
step(self)
run(self, steps=1000)
list_things_at(self, location, tclass=Thing)
some_things_at(self, location, tclass=Thing)
add_thing(self, thing, location=None)
delete_thing(self, thing)
blinddog_simple_reflex.Park
percept(self, agent)
execute_action(self, agent, action)
is_done(self)
blinddog_simple_reflex.Agent
holding
performance
alive
bump
program
__init__(self, program=None)
can_grab(self, thing)
blinddog_simple_reflex.BlindDog
location
movedown(self)
eat(self, thing)
drink(self, thing)
bark(self, thing)
blinddog_simple_reflex.Person
__name__
__init__(self, name=None)
blinddog_simple_reflex.Thing
__repr__(self)
is_alive(self)
show_state(self)
blinddog_simple_reflex.Water
blinddog_simple_reflex.Food
blinddog_simple_reflex.Tree

Execution Results

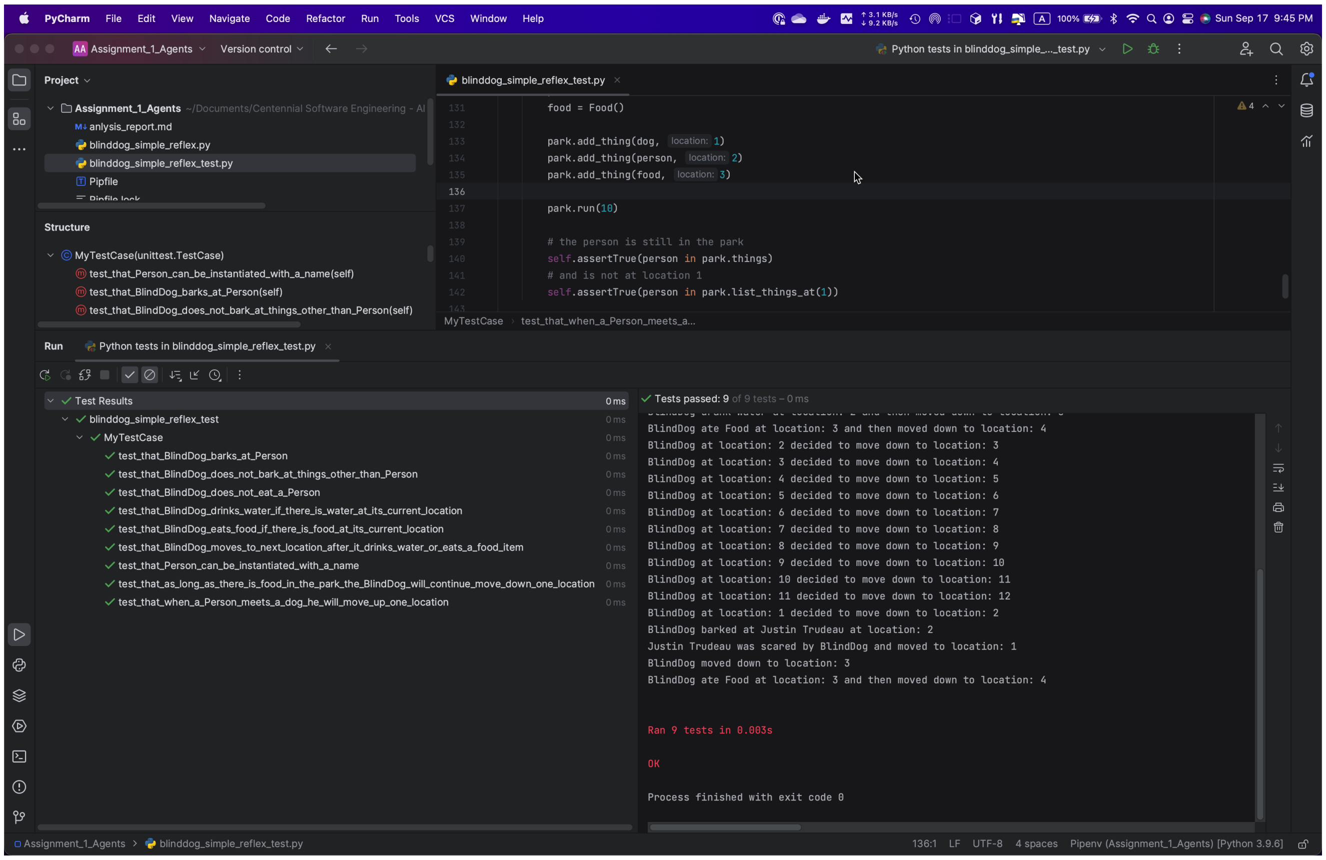
1. Program Execution



```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  7 11:38:46 2020

@author: mhabayeb
@author: Vincent Ngan (Student ID: 301328893)

"""
import ...


class Thing:
    """This represents any physical object that can appear in an Environment.
    You subclass Thing to get the things you want. Each thing can have a
    .__name__  slot (used for output only)."""

    def __repr__(self):
        return '<{}>'.format(getattr(self, '__name__', self.__class__.__name__))
```

```
/Users/vincent/.local/share/virtualenvs/Assignment_1_Agents-cVHHRXiw/bin/Python /Users/vincent/Documents/Centennial Software Engineering - AI/COMP 237 - Introduction to AI/Assignment_1_Agents/blinddog_sim
BlindDog at location: 1 decided to move down to location: 2
BlindDog at location: 2 decided to move down to location: 3
BlindDog barked at Vincent at location: 3
Vincent was scared by BlindDog and moved to location: 2
BlindDog moved down to location: 4
BlindDog at location: 4 decided to move down to location: 5
BlindDog ate Food at location: 5 and then moved down to location: 6
BlindDog at location: 6 decided to move down to location: 7
BlindDog drank Water at location: 7 and then moved down to location: 8
BlindDog at location: 8 decided to move down to location: 9
BlindDog ate Food at location: 9 and then moved down to location: 10
Park is done: True
BlindDog starved at location: 10

Process finished with exit code 0
```

2. Unit Test Execution



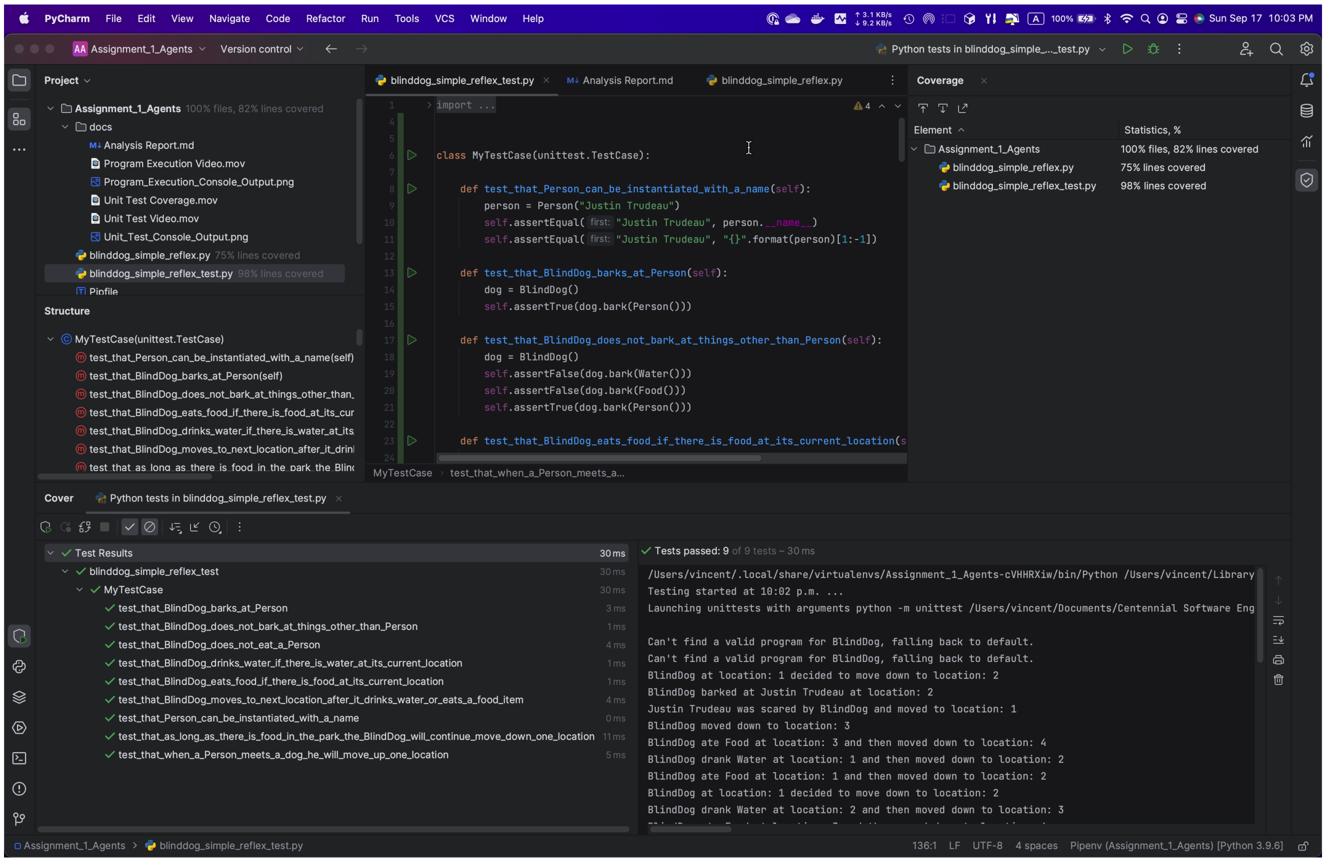
```
        food = Food()

        park.add_thing(dog, location: 1)
        park.add_thing(person, location: 2)
        park.add_thing(food, location: 3)

        park.run(10)

        # the person is still in the park
        self.assertTrue(person in park.things)
        # and is not at location 1
        self.assertTrue(person in park.list_things_at(1))
```

```
Test Results                                                                             0 ms
  blinddog_simple_reflex_test                                                            0 ms
    MyTestCase                                                                           0 ms
      ✓ test_that_BlindDog_barks_at_Person                                               0 ms
      ✓ test_that_BlindDog_does_not_bark_at_things_other_than_Person                     0 ms
      ✓ test_that_BlindDog_does_not_eat_a_Person                                         0 ms
      ✓ test_that_BlindDog_drinks_water_if_there_is_water_at_its_current_location        0 ms
      ✓ test_that_BlindDog_eats_food_if_there_is_food_at_its_current_location            0 ms
      ✓ test_that_BlindDog_moves_to_next_location_after_it_drinks_water_or_eats_a_food_item  0 ms
      ✓ test_that_Person_can_be_instantiated_with_a_name                                 0 ms
      ✓ test_that_as_long_as_there_is_food_in_the_park_the_BlindDog_will_continue_move_down_one_location  0 ms
      ✓ test_that_when_a_Person_meets_a_dog_he_will_move_up_one_location                 0 ms
```

✓ Tests passed: 9 of 9 tests – 0 ms

```
BlindDog ate Food at location: 3 and then moved down to location: 4
BlindDog at location: 2 decided to move down to location: 3
BlindDog at location: 3 decided to move down to location: 4
BlindDog at location: 4 decided to move down to location: 5
BlindDog at location: 5 decided to move down to location: 6
BlindDog at location: 6 decided to move down to location: 7
BlindDog at location: 7 decided to move down to location: 8
BlindDog at location: 8 decided to move down to location: 9
BlindDog at location: 9 decided to move down to location: 10
BlindDog at location: 10 decided to move down to location: 11
BlindDog at location: 11 decided to move down to location: 12
BlindDog at location: 1 decided to move down to location: 2
BlindDog barked at Justin Trudeau at location: 2
Justin Trudeau was scared by BlindDog and moved to location: 1
BlindDog moved down to location: 3
BlindDog ate Food at location: 3 and then moved down to location: 4


Ran 9 tests in 0.003s

OK

Process finished with exit code 0
```

3. Unit Test Code Coverage



Project

Assignment_1_Agents	100% files, 82% lines covered
docs	
Analysis Report.md	
Program Execution Video.mov	
Program_Execution_Console_Output.png	
Unit Test Coverage.mov	
Unit Test Video.mov	
Unit_Test_Console_Output.png	
blinddog_simple_reflex.py	75% lines covered
blinddog_simple_reflex_test.py	98% lines covered
Pipfile	

Structure

- MyTestCase(unittest.TestCase)
  - test_that_Person_can_be_instantiated_with_a_name(self)
  - test_that_BlindDog_barks_at_Person(self)
  - test_that_BlindDog_does_not_bark_at_things_other_than_
  - test_that_BlindDog_eats_food_if_there_is_food_at_its_cur
  - test_that_BlindDog_drinks_water_if_there_is_water_at_its
  - test_that_BlindDog_moves_to_next_location_after_it_drinl
  - test_that_as_long_as_there_is_food_in_the_park_the_Blinc

blinddog_simple_reflex_test.py

```python
import ...


class MyTestCase(unittest.TestCase):

    def test_that_Person_can_be_instantiated_with_a_name(self):
        person = Person("Justin Trudeau")
        self.assertEqual( first: "Justin Trudeau", person.__name__)
        self.assertEqual( first: "Justin Trudeau", "{}".format(person)[1:-1])

    def test_that_BlindDog_barks_at_Person(self):
        dog = BlindDog()
        self.assertTrue(dog.bark(Person()))

    def test_that_BlindDog_does_not_bark_at_things_other_than_Person(self):
        dog = BlindDog()
        self.assertFalse(dog.bark(Water()))
        self.assertFalse(dog.bark(Food()))
        self.assertTrue(dog.bark(Person()))

    def test_that_BlindDog_eats_food_if_there_is_food_at_its_current_location(s
```

Coverage

Element	Statistics, %
Assignment_1_Agents	100% files, 82% lines covered
blinddog_simple_reflex.py	75% lines covered
blinddog_simple_reflex_test.py	98% lines covered

Cover — Python tests in blinddog_simple_reflex_test.py

✓ Test Results	30 ms
✓ blinddog_simple_reflex_test	30 ms
✓ MyTestCase	30 ms
✓ test_that_BlindDog_barks_at_Person	3 ms
✓ test_that_BlindDog_does_not_bark_at_things_other_than_Person	1 ms
✓ test_that_BlindDog_does_not_eat_a_Person	4 ms
✓ test_that_BlindDog_drinks_water_if_there_is_water_at_its_current_location	1 ms
✓ test_that_BlindDog_eats_food_if_there_is_food_at_its_current_location	1 ms
✓ test_that_BlindDog_moves_to_next_location_after_it_drinks_water_or_eats_a_food_item	4 ms
✓ test_that_Person_can_be_instantiated_with_a_name	0 ms
✓ test_that_as_long_as_there_is_food_in_the_park_the_BlindDog_will_continue_move_down_one_location	11 ms
✓ test_that_when_a_Person_meets_a_dog_he_will_move_up_one_location	5 ms

✓ Tests passed: 9 of 9 tests – 30 ms

```
/Users/vincent/.local/share/virtualenvs/Assignment_1_Agents-cVHHRXiw/bin/Python /Users/vincent/Library
Testing started at 10:02 p.m. ...
Launching unittests with arguments python -m unittest /Users/vincent/Documents/Centennial Software Eng

Can't find a valid program for BlindDog, falling back to default.
Can't find a valid program for BlindDog, falling back to default.
BlindDog at location: 1 decided to move down to location: 2
BlindDog barked at Justin Trudeau at location: 2
Justin Trudeau was scared by BlindDog and moved to location: 1
BlindDog moved down to location: 3
BlindDog ate Food at location: 3 and then moved down to location: 4
BlindDog drank Water at location: 1 and then moved down to location: 2
BlindDog ate Food at location: 1 and then moved down to location: 2
BlindDog at location: 1 decided to move down to location: 2
BlindDog drank Water at location: 2 and then moved down to location: 3
```

Assignment_1_Agents > blinddog_simple_reflex_test.py — 136:1 LF UTF-8 4 spaces Pipenv (Assignment_1_Agents) [Python 3.9.6]