TRÍ TUỆ NHÂN TẠO

# LAB 02 – AGENTS

Trong bài thực hành này, chúng ta muốn xây dựng một tác nhân (Agent) là một con Chó, tuy nhiên con Chó này bị mù.

---

Định nghĩa lớp **Thing** – để biểu diễn cho bất cứ thứ gì trên thế giới này.

```
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__))

    def is_alive(self):
        """Things that are 'alive' should return true."""
        return hasattr(self, 'alive') and self.alive

    def show_state(self):
        """Display the agent's internal state. Subclasses should
override."""
        print("I don't know how to show_state.")

    def display(self, canvas, x, y, width, height):
        """Display an image of this Thing on the canvas."""
        # Do we need this?
        pass
```

Test đầu tiên:

```
if __name__ == "__main__":
    t = Thing()
    print(repr(t))
```

Hãy cho biết ý nghĩa của phương thức **__repr__** trong lớp Thing ở trên.

Chúng ta có thể tạo ra thức ăn (Food) và nước (Water) cho chó.

```
class Food(Thing):
    pass


class Water(Thing):
    pass
```

Một tác tử (Agent) là một lớp con của Thing.

```
class Agent(Thing):
    """An Agent is a subclass of Thing with one required instance
attribute."""

    def __init__(self, program=None):
        self.alive = True
        self.bump = False
        self.holding = []
        self.performance = 0
        if program is None or not isinstance(program,
collections.abc.Callable):
            print("Can't find a valid program for {}, falling back to
default.".format(self.__class__.__name__))

            def program(percept):
                return eval(input('Percept={}; action?
'.format(percept)))

        self.program = program

    def can_grab(self, thing):
        """Return True if this agent can grab this thing.
        Override for appropriate subclasses of Agent and Thing."""
        return False
```

Chúng ta xây dựng lớp BlindDog kế thừa lớp Agent:

```
class BlindDog(Agent):
    location = 1

    def movedown(self):
        self.location += 1

    def eat(self, thing):
        '''returns True upon success or False otherwise'''
        if isinstance(thing, Food):
            return True
```

```
        return False

    def drink(self, thing):
        ''' returns True upon success or False otherwise'''
        if isinstance(thing, Water):
            return True
        return False
```

Phương thức program sẽ nhận một percept (nhận thức) và trả về một action (hành động):

```
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'
    return 'move down'
```

Bây giờ chúng ta có thể tạo ra chú chó mù

```
if __name__ == "__main__":
    # t = Thing()
    # print(repr(t))
    dogfood = Food()
    water = Water()
    dog = BlindDog(program)
```

Tiếp theo, chúng ta cần tạo ra môi trường (Environment):

```
class Environment:
    """Abstract class representing an Environment. 'Real' Environment
classes
    inherit from this. Your Environment will typically need to
implement:
        percept:           Define the percept that an agent sees.
        execute_action:    Define the effects of executing an action.
                           Also update the agent.performance slot.
    The environment keeps a list of .things and .agents (which is a
subset
```

```
    of .things). Each agent has a .performance slot, initialized to 0.
    Each thing has a .location slot, even though some environments may
not
    need this."""

    def __init__(self):
        self.things = []
        self.agents = []

    def thing_classes(self):
        return []  # List of classes that can go into environment

    def percept(self, agent):
        """Return the percept that the agent sees at this point.
(Implement this.)"""
        raise NotImplementedError

    def execute_action(self, agent, action):
        """Change the world to reflect this action. (Implement this.)"""
        raise NotImplementedError

    def default_location(self, thing):
        """Default location to place a new thing with unspecified
location."""
        return None

    def exogenous_change(self):
        """If there is spontaneous change in the world, override
this."""
        pass

    def is_done(self):
        """By default, we're done when we can't find a live agent."""
        return not any(agent.is_alive() for agent in self.agents)

    def step(self):
        """Run the environment for one time step. If the
        actions and exogenous changes are independent, this method will
        do. If there are interactions between them, you'll need to
        override this method."""
        if not self.is_done():
            actions = []
            for agent in self.agents:
                if agent.alive:
                    actions.append(agent.program(self.percept(agent)))
                else:
                    actions.append("")
            for (agent, action) in zip(self.agents, actions):
                self.execute_action(agent, action)
            self.exogenous_change()

    def run(self, steps=1000):
        """Run the Environment for given number of time steps."""
        for step in range(steps):
```

```
            if self.is_done():
                return
            self.step()

    def list_things_at(self, location, tclass=Thing):
        """Return all things exactly at a given location."""
        if isinstance(location, numbers.Number):
            return [thing for thing in self.things
                    if thing.location == location and isinstance(thing,
tclass)]
        return [thing for thing in self.things
                if all(x == y for x, y in zip(thing.location, location))
and isinstance(thing, tclass)]

    def some_things_at(self, location, tclass=Thing):
        """Return true if at least one of the things at location
        is an instance of class tclass (or a subclass)."""
        return self.list_things_at(location, tclass) != []

    def add_thing(self, thing, location=None):
        """Add a thing to the environment, setting its location. For
        convenience, if thing is an agent program we make a new agent
        for it. (Shouldn't need to override this.)"""
        if not isinstance(thing, Thing):
            thing = Agent(thing)
        if thing in self.things:
            print("Can't add the same thing twice")
        else:
            thing.location = location if location is not None else
self.default_location(thing)
            self.things.append(thing)
            if isinstance(thing, Agent):
                thing.performance = 0
                self.agents.append(thing)

    def delete_thing(self, thing):
        """Remove a thing from the environment."""
        try:
            self.things.remove(thing)
        except ValueError as e:
            print(e)
            print("  in Environment delete_thing")
            print("  Thing to be removed: {} at {}".format(thing,
thing.location))
            print("  from list: {}".format([(thing, thing.location) for
thing in self.things]))
        if thing in self.agents:
            self.agents.remove(thing)
```

Hãy tạo một file blind_dog.py. Hãy định nghĩa lớp BlindDog sau:

```
class BlindDog(Agent):
    location = 1

    def movedown(self):
        ##########################################################################
        # TODO: increase location (our solution is 1 line of code, but don't worry if
        #  you deviate from this)
        ##########################################################################
        #                               BEGIN OF YOUR CODE                       #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                END OF YOUR CODE                        #
        ##########################################################################

    def eat(self, thing):
        '''returns True upon success or False otherwise'''
        ##########################################################################
        # TODO: return True if thing is food for dog (our solution is 3 lines of code,
        #  but don't worry if you deviate from this)
        ##########################################################################
        #                               BEGIN OF YOUR CODE                       #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                END OF YOUR CODE                        #
        ##########################################################################

    def drink(self, thing):
        ''' returns True upon success or False otherwise'''
        ##########################################################################
        # TODO: return True if thing is water (our solution is 3 lines of code,
        #  but don't worry if you deviate from this)
        ##########################################################################
        #                               BEGIN OF YOUR CODE                       #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                END OF YOUR CODE                        #
        ##########################################################################
```

Định nghĩa Định nghĩa (Park) kế thừa từ Environment

```
class Park(Environment):
    def percept(self, agent):
        ##########################################################################
        # TODO: return a list of things that are in our agent's location (our solution
        #  is 2 lines of code, but don't worry if you deviate from this)
        ##########################################################################
        #                               BEGIN OF YOUR CODE                       #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                END OF YOUR CODE                        #
        ##########################################################################

    def execute_action(self, agent, action):
        ##########################################################################
        # TODO: changes the state of the environment based on what the agent does.
```

```
        #  (our solution is 15 lines of code, but don't worry if you deviate from this)
        ##########################################################################
        #                                BEGIN OF YOUR CODE                      #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                 END OF YOUR CODE                       #
        ##########################################################################

    def is_done(self):
        '''By default, we're done when we can't find a live agent,
        but to prevent killing our cute dog, we will stop before itself - when there is
no more food or water'''
        ##########################################################################
        # TODO: By default, we're done when we can't find a live agent, but to prevent
        #  killing our cute dog, we will stop before itself - when there is no more
        #  food or water. (our solution is 3 lines of code, but don't worry if you
        #  deviate from this)
        ##########################################################################
        #                                BEGIN OF YOUR CODE                      #
        ##########################################################################
        raise Exception("Not implemented yet")
        ##########################################################################
        #                                 END OF YOUR CODE                       #
        ##########################################################################
```

Bây giờ, hãy tạo một công viên (Park) vài thức ăn (Food), nước (Water) và con chó

```
if __name__ == "__main__":
    # t = Thing()
    # print(repr(t))

    park = Park()
    dog = BlindDog(program)
    dogfood = Food()
    water = Water()

    park.add_thing(dog, 1)
    park.add_thing(dogfood, 5)
    park.add_thing(water, 7)

    park.run(5)
```

Chúng ta sẽ thấy con cho di chuyển 4 bước và ăn thức ăn ở bước thứ 5.

```
BlindDog decided to move down at location: 1
BlindDog decided to move down at location: 2
BlindDog decided to move down at location: 3
BlindDog decided to move down at location: 4
BlindDog ate Food at location: 5
```

```
BlindDog decided to move down at location: 5
BlindDog decided to move down at location: 6
BlindDog drank Water at location: 7
```

Nếu chúng ta tăng lên 10 steps, thì con chó di chuyển, và ăn thức ăn ở bước thứ 5, uống nước ở step thứ 7 và dừng. Tại sao chú cho không di chuyển 3 bước còn lại?

```
if __name__ == "__main__":
    # t = Thing()
    # print(repr(t))

    park = Park()
    dog = BlindDog(program)
    dogfood = Food()
    water = Water()

    park.add_thing(dog, 1)
    park.add_thing(dogfood, 5)
    park.add_thing(water, 7)

    park.run(10)
```

```
BlindDog decided to move down at location: 1
BlindDog decided to move down at location: 2
BlindDog decided to move down at location: 3
BlindDog decided to move down at location: 4
BlindDog ate Food at location: 5
BlindDog decided to move down at location: 5
BlindDog decided to move down at location: 6
BlindDog drank Water at location: 7
```