



Intelligent Agents Workshop

AIMA Chapter 2: Intelligent Agents

เป้าหมายการเรียนรู้:

- เข้าใจแนวคิดพื้นฐานของ Agent และ Environment
- เรียนรู้การออกแบบและ implement agent programs
- ลงมือปฏิบัติสร้าง agents ในสภาพแวดล้อมต่างๆ

สิ่งที่ต้องเตรียม:

- Python environment พร้อม agents.py
- Jupyter Notebook หรือ Python IDE



สารบัญ

1. **Overview - Agent คืออะไร?**
2. **Agent Architecture**
3. **Environment Types**
4. **Hands-on: BlindDog Agent**
5. **2D Environment & EnergeticBlindDog**
6. **Wumpus World Challenge**
7. **การประเมินผลและแนวคิดขั้นสูง**

Agent คืออะไร?

Agent = สิ่งที่สามารถ:

- **รับรู้** สภาพแวดล้อม (perceive) ผ่าน sensors
- **กระทำ** ต่อสภาพแวดล้อม (act) ผ่าน actuators

ตัวอย่าง Agents:

- 🐕 สุนัข (sensors: ตา, หู, จมูก | actuators: ขา, เสียง)
- 🤖 หุ่นยนต์ (sensors: กล้อง, เซ็นเซอร์ | actuators: มอเตอร์, แขนกล)
- 💻 Software Agent (sensors: ข้อมูลอินพุต | actuators: ข้อมูลเอาต์พุต)
- 👤 มนุษย์ (sensors: ประสาทสัมผัส | actuators: มือ, เท้า, เสียง)

Agent คืออะไร?

Agent = สิ่งที่สามารถ:

- **รับรู้** สภาพแวดล้อม (perceive) ผ่าน **sensors**
- **กระทำ** ต่อสภาพแวดล้อม (act) ผ่าน **actuators**

แผนภาพการทำงาน:



ตัวอย่าง: มนุษย์, หุ่นยนต์, Software Agent



Agent Architecture

```
class Agent(Thing):
    def __init__(self, program=None):
        self.alive = True           # สถานะการมีชีวิต
        self.bump = False          # การชนกำแพง
        self.holding = []          # สิ่งถือ
        self.performance = 0        # คะแนนประสิทธิภาพ
        self.program = program     # โปรแกรมควบคุม

    def can_grab(self, thing):
        """Return True if this agent can grab this thing."""
        return False
```

องค์ประกอบสำคัญ:

- **Program:** ฟังก์ชันที่แปลง percept → action
- **Performance:** การวัดประสิทธิภาพ
- **State:** สถานะภายในของ agent



Environment Architecture

```
class Environment:
    def __init__(self):
        self.things = []    # สิ่งของทั้งหมดใน environment
        self.agents = []    # agents ทั้งหมด

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

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

    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()
```






สร้าง BlindDog Agent แรก

จากไฟล์ agents.ipynb:

```
class BlindDog(Agent):  
    def eat(self, thing):  
        print("Dog: Ate food at {}".format(self.location))  
  
    def drink(self, thing):  
        print("Dog: Drank water at {}".format(self.location))  
  
# สร้าง dog instance  
dog = BlindDog()  
print(dog.alive)  # True
```

สุนัขตาบอดของเรา:

-  มองไม่เห็น แต่สัมผัสได้ในตำแหน่งปัจจุบัน
-  กินอาหารได้
-  ดื่มน้ำได้



สร้าง Park Environment

```
class Food(Thing):
    pass

class Water(Thing):
    pass

class Park(Environment):
    def percept(self, agent):
        '''return a list of things that are in our agent's location'''
        things = self.list_things_at(agent.location)
        return things

    def execute_action(self, agent, action):
        '''changes the state of the environment based on what the agent does.'''
        if action == "move down":
            print('{} decided to {} at location: {}'.format(
                str(agent)[1:-1], action, agent.location))
            agent.movedown()
        elif action == "eat":
            items = self.list_things_at(agent.location, tclass=Food)
            if len(items) != 0:
                if agent.eat(items[0]):
                    self.delete_thing(items[0])
            # ... drink logic
```



BlindDog Program Implementation

```
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

    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'
```

Demo: BlindDog Simulation

```
# สร้าง environment และ agent
park = Park()
dog = BlindDog(program)
dogfood = Food()
water = Water()

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

# รันการจำลอง
park.run(5)
```

ผลลัพธ์ที่คาดหวัง:

- สุนัขเดินจากตำแหน่ง 1 → 5 และกินอาหาร
- เดินต่อไปยังตำแหน่ง 7 และดื่มน้ำ

Percept-Action Table

จากตัวอย่างใน notebook:

Percept	Action
Feel Food	eat
Feel Water	drink
Feel Nothing	move down

```
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'
```

Simple Reflex Agent: ตอบสนองทันทีต่อ percept



GraphicEnvironment - 2D Upgrade!

```
class Park2D(GraphicEnvironment):
    def percept(self, agent):
        '''return a list of things that are in our agent's location'''
        things = self.list_things_at(agent.location)
        return things

    def execute_action(self, agent, action):
        if action == "move down":
            agent.movedown()
        elif action == "eat":
            items = self.list_things_at(agent.location, tclass=Food)
            if len(items) != 0:
                if agent.eat(items[0]):
                    self.delete_thing(items[0])
            # ... similar for drink

# สร้างพร้อมสี
park = Park2D(5, 20, color={
    'BlindDog': (200, 0, 0),      # แดง
    'Water': (0, 200, 200),      # น้ำ
    'Food': (230, 115, 40)       # ส้ม
})
```



EnergeticBlindDog - 2D Movement

```
class EnergeticBlindDog(Agent):
    location = [0,1]
    direction = Direction("down")
    def moveforward(self, success=True):
        if not success:
            return
        if self.direction.direction == Direction.R:
            self.location[0] += 1
        elif self.direction.direction == Direction.L:
            self.location[0] -= 1
        elif self.direction.direction == Direction.D:
            self.location[1] += 1
        elif self.direction.direction == Direction.U:
            self.location[1] -= 1
    def turn(self, d):
        self.direction = self.direction + d
```



EnergeticBlindDog Program

```
from random import choice
import random

def program(percepts):
    '''Returns an action based on it's percepts'''

    for p in percepts: # first eat or drink – you're a dog!
        if isinstance(p, Food):
            return 'eat'
        elif isinstance(p, Water):
            return 'drink'
        if isinstance(p, Bump): # then check if you are at an edge
            choice = random.choice((1,2)) # turn only
        else:
            choice = random.choice((1,2,3,4)) # 1-right, 2-left, others-forward

    if choice == 1:
        return 'turnright'
    elif choice == 2:
        return 'turnleft'
    else:
        return 'moveforward'
```


กลยุทธ์: กิน/ดื่ม → หลีกเลี้ยงกำแพง → เคลื่อนไหวสุม



Enhanced Park2D Environment

```
class Park2D(GraphicEnvironment):
    def percept(self, agent):
        things = self.list_things_at(agent.location)
        loc = copy.deepcopy(agent.location) # find target location

        # Check if agent is about to bump into a wall
        if agent.direction.direction == Direction.R:
            loc[0] += 1
        elif agent.direction.direction == Direction.L:
            loc[0] -= 1
        elif agent.direction.direction == Direction.D:
            loc[1] += 1
        elif agent.direction.direction == Direction.U:
            loc[1] -= 1

        if not self.is_inbounds(loc):
            things.append(Bump())
        return things

    def execute_action(self, agent, action):
        if action == 'turnright':
            agent.turn(Direction.R)
        elif action == 'turnleft':
            agent.turn(Direction.L)
        elif action == 'moveforward':
            agent.moveforward()
        # ... eat/drink logic
```

Demo: 2D EnergeticBlindDog

```
park = Park2D(5, 5, color={
    'EnergeticBlindDog': (200,0,0),
    'Water': (0, 200, 200),
    'Food': (230, 115, 40)
})

dog = EnergeticBlindDog(program)
park.add_thing(dog, [0,0])
park.add_thing(Food(), [1,2])
park.add_thing(Water(), [0,1])
park.add_thing(Water(), [2,4])
park.add_thing(Food(), [4,3])

print("dog started at [0,0], facing down. Let's see if he found any food!")
park.run(20)
```

ผลลัพธ์: สุนัขจะสำรวจพื้นที่ 2D แบบสุ่มและหาอาหาร/น้ำ!

ลองรันดู!

Wumpus World - The Ultimate Challenge!

จากไฟล์ notebook:

```
color = {"Breeze": (225, 225, 225),  
         "Pit": (0, 0, 0),  
         "Gold": (253, 208, 23),  
         "Glitter": (253, 208, 23),  
         "Wumpus": (43, 27, 23),  
         "Stench": (128, 128, 128),  
         "Explorer": (0, 0, 255),  
         "Wall": (44, 53, 57)}  
  
def program(percepts):  
    '''Returns an action based on it's percepts'''  
    print(percepts)  
    return input() # Manual control!  
  
w = WumpusEnvironment(program, 7, 7)
```

เป้าหมาย:

- 🏆 หาทองคำและออกจากถ้ำ
- 💀 หลีกเลียง Wumpus และ Pits
- 🏹 ใช้ลูกศรอย่างชาญฉลาด

Wumpus World Interactive Demo

```
from ipythonblocks import BlockGrid
w = WumpusEnvironment(program, 7, 7)
grid = BlockGrid(w.width, w.height, fill=(123, 234, 123))

def draw_grid(world):
    global grid
    grid[:] = (123, 234, 123)
    for x in range(0, len(world)):
        for y in range(0, len(world[x])):
            if len(world[x][y]):
                grid[y, x] = color[world[x][y][-1].__class__.__name__]

def step():
    global grid, w
    draw_grid(w.get_world())
    grid.show()
    w.step()

# ใช้ step() เพื่อเล่นทีละขั้น
step()
```

Actions ที่ใช้ได้: Forward, TurnLeft, TurnRight, Grab, Shoot, Climb



Wumpus World Percepts

Format: [Left, Right, Up, Down, Center]

ตัวอย่าง Percepts:

- Glitter : ทองคำอยู่ในช่องเดียวกัน
- Stench : Wumpus อยู่ในช่องข้างเคียง
- Breeze : หลุมอยู่ในช่องข้างเคียง
- Bump : ชนกำแพง
- Scream : Wumpus ถูกยิงแล้ว

```
# ตัวอย่าง percepts ที่อาจได้รับ  
[[None], [Stench], [None], [None], [Glitter]]  
# หมายความว่า: ทางขวามี Stench, ตำแหน่งปัจจุบันมี Glitter
```




Hands-on Exercise Time!



ภารกิจ 1: ปรับปรุง BlindDog (15 นาที)

```
class SmartBlindDog(Agent):  
    def __init__(self, program):  
        super().__init__(program)  
        self.visited = set()    # จำตำแหน่งที่เคยไป  
  
    def smart_program(percepts):  
        # TODO: หลีกเลียงไม่ให้กลับไปที่เคย  
        # TODO: เพิ่มกลยุทธ์การค้นหา  
        pass
```

ภารกิจ 2: Wumpus Agent Prototype (15 นาที)

```
def safe_wumpus_program(percepts):  
    # TODO: หลีกเลียงพื้นที่อันตราย  
    # TODO: หาทองคำและกลับ  
    pass
```

เริ่มเลย!

Agent Performance Comparison




จากไฟล์ agents.py:

```
def compare_agents(EnvFactory, AgentFactories, n=10, steps=1000):
    """เปรียบเทียบ agents หลายตัวใน environments"""
    envs = [EnvFactory() for i in range(n)]
    return [(A, test_agent(A, steps, copy.deepcopy(envs)))
            for A in AgentFactories]

def test_agent(AgentFactory, steps, envs):
    """Return the mean score of running an agent"""
    def score(env):
        agent = AgentFactory()
        env.add_thing(agent)
        env.run(steps)
        return agent.performance

    return mean(map(score, envs))
```

เกณฑ์การประเมิน:

-  คะแนนเฉลี่ย
-  เวลาที่ใช้
-  อัตราความสำเร็จ



Types of Agent Programs

1. Simple Reflex Agent

```
def simple_reflex_program(percept):  
    if percept == 'Dirty':  
        return 'Suck'  
    elif location == loc_A:  
        return 'Right'  
    else:  
        return 'Left'
```

2. Model-Based Reflex Agent

```
model = {loc_A: None, loc_B: None}

def model_based_program(percept):
    location, status = percept
    model[location] = status # Update the model
    if model[loc_A] == model[loc_B] == 'Clean':
        return 'NoOp'
    elif status == 'Dirty':
        return 'Suck'
    # ...
```



Vacuum World Example

จากไฟล์ agents.py:

```
def ReflexVacuumAgent():  
    def program(percept):  
        location, status = percept  
        if status == 'Dirty':  
            return 'Suck'  
        elif location == loc_A:  
            return 'Right'  
        elif location == loc_B:  
            return 'Left'  
    return Agent(program)
```



Vacuum World Example

จากไฟล์ agents.py:

```
def ModelBasedVacuumAgent():
    model = {loc_A: None, loc_B: None}
    def program(percept):
        location, status = percept
        model[location] = status
        if model[loc_A] == model[loc_B] == 'Clean':
            return 'NoOp'
        elif status == 'Dirty':
            return 'Suck'
        elif location == loc_A:
            return 'Right'
        elif location == loc_B:
            return 'Left'
    return Agent(program)
```


Design Principles

PEAS Framework:

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors

Agent Architecture:

- **Rationality:** ทำสิ่งที่ถูกต้องในสถานการณ์
- **Autonomy:** ตัดสินใจด้วยตนเองจาก percepts
- **Adaptation:** ปรับตัวและเรียนรู้

Environment Properties:

- **Observable:** มองเห็นได้เต็มหรือบางส่วน
- **Deterministic:** ผลลัพธ์แน่นอนหรือสุ่ม
- **Episodic:** แต่ละ action เป็นอิสระ
- **Static:** เปลี่ยนแปลงหรือคงที่

Advanced Agent Types

Goal-Based Agent

- มีเป้าหมายชัดเจน
- วางแผนการกระทำ
- ใช้ Search algorithms

Utility-Based Agent

- ประเมินความต้องการ (utility function)
- เลือกทางที่ดีที่สุด
- จัดการ trade-offs

Learning Agent

- เรียนรู้จากประสบการณ์
- ปรับปรุงประสิทธิภาพ
- Performance element + Learning element

Implementation Tips

การเก็บ State:

```
class SmartAgent(Agent):  
    def __init__(self, program):  
        super().__init__(program)  
        self.knowledge = {}      # ความรู้เกี่ยวกับโลก  
        self.visited = set()    # ตำแหน่งที่เคยไป  
        self.goals = []         # เป้าหมาย
```

การวางแผน:

```
def planning_program(percepts):  
    # 1. อัปเดตความรู้  
    update_knowledge(percepts)  
  
    # 2. ประเมินสถานการณ์  
    if immediate_action_needed():  
        return urgent_action()  
  
    # 3. วางแผนเคลื่อนไหว  
    return plan_next_move()
```




สรุป Workshop



สิ่งที่เราได้เรียนรู้:

1. แนวคิด **Agent-Environment**

- Agent รับรู้และกระทำต่อสิ่งแวดล้อม
- Environment ให้ percepts และตอบสนอง actions

2. การ **Implement** จริง

- BlindDog: Simple 1D agent
- EnergeticBlindDog: 2D movement with graphics
- WumpusWorld: Complex reasoning environment



สรุป Workshop



สิ่งที่เราได้เรียนรู้:

3. Agent Program Patterns

- Simple Reflex vs Model-Based
- Performance measurement
- การจัดการ state และ knowledge

Code Examples Summary

BlindDog (1D)

```
def program(percepts):  
    for p in percepts:  
        if isinstance(p, Food): return 'eat'  
        elif isinstance(p, Water): return 'drink'  
    return 'move down'
```

EnergeticBlindDog (2D)

```
def program(percepts):  
    for p in percepts:  
        if isinstance(p, Food): return 'eat'  
        elif isinstance(p, Water): return 'drink'  
        if isinstance(p, Bump):  
            return random.choice(['turnright', 'turnleft'])  
    return random.choice(['turnright', 'turnleft', 'moveforward', 'moveforward'])
```

Wumpus World

```
def program(percepts):  
    print(percepts) # [Left, Right, Up, Down, Center]  
    return input()  # Manual control for learning
```



Next Steps & Applications







หัวข้อต่อไป:

- **Search Algorithms** (Chapter 3-4): การค้นหาเส้นทาง
- **Knowledge Representation** (Chapter 7-9): การใช้ logic
- **Machine Learning** (Chapter 18-21): agents ที่เรียนรู้



การประยุกต์ใช้จริง:

-  **Robotics**: หุ่นยนต์ทำความสะอาด, โดรน
-  **Game AI**: NPCs ที่ฉลาด, ปฏิกริยาตอบสนอง
-  **Software Agents**: Chatbots, recommendation systems
-  **Automation**: ระบบควบคุมอุตสาหกรรม

Resources:

- AIMA Python Code
- Jupyter Notebooks



Homework & Projects



การบ้าน (เลือก 1 ข้อ):

1. **Smart BlindDog:** ให้จำตำแหน่งและหลีกเลี่ยงไม่กลับ
2. **Wumpus Solver:** สร้าง agent ที่ใช้ logical reasoning
3. **Multi-Agent Park:** สุนัขหลายตัวแข่งขันหาอาหาร

Project Ideas:

- **Pac-Man AI:** Agent กินจุดหลีกผี
- **Trading Bot:** Agent ซื้อขายหุ้นจำลอง
- **Smart Home Controller:** ควบคุมอุปกรณ์บ้าน
- **Maze Solver:** หาทางออกจากเขาวงกต

กำหนดส่ง: x สัปดาห์

ส่งเป็น: Jupyter Notebook + วิดีโอสาธิต

? Q&A Session

ถาม-ตอบ และ Discussion

คำถามที่น่าสนใจ:

- ทำไม BlindDog ถึงใช้ Simple Reflex ได้?
- Agent แบบไหนเหมาะกับปัญหาแบบไหน?
- จะทำให้ Agent เรียนรู้ได้อย่างไร?
- ความแตกต่างระหว่าง AI Agent กับ Program ทั่วไป?

แบ่งปันประสบการณ์:

- ปัญหาที่เจอในการ implement
- ไอเดียการปรับปรุง agents
- การประยุกต์ใช้ในงานจริง





มาคุยกัน! 

 **Thank You!**

Workshop: Intelligent Agents

AIMA Chapter 2 - Hands-on Experience

สิ่งที่ได้เรียนรู้วันนี้:

-  Agent & Environment concepts
-  Implementation patterns
-  จาก BlindDog → EnergeticBlindDog → Wumpus World
-  Performance evaluation

ติดต่อ:

-  Email: [ittipon@g.sut.ac.th]

Happy Coding!  

