Artificial Intelligence

Algorithms and Applications with Python

Lectorial 01

Dr. Dominik Jung
dominik.jung42@gmail.com

</> course material: ↗ https://github.com/dominikjung42

Image source: ↗ Pixabay (2019) / ↗ CC0

Agenda

1.1 Basics and Repetition

1.2 Vaccum Cleaner as Simple Reflex-Agents

1.3 Playground: Implement other Agents

Note

- This is a lectorial: I will explain/repeat the most important concepts and then you try to solve the programming task by your own

- You are explicitly encouraged to solve this task in groups. And I will help you and give suggestions. However, there is no perfect solution, you will get a possible solution.

- If the task is too hard for you at the moment – relaxe ☺. Just look at the task again at a later point in the course.

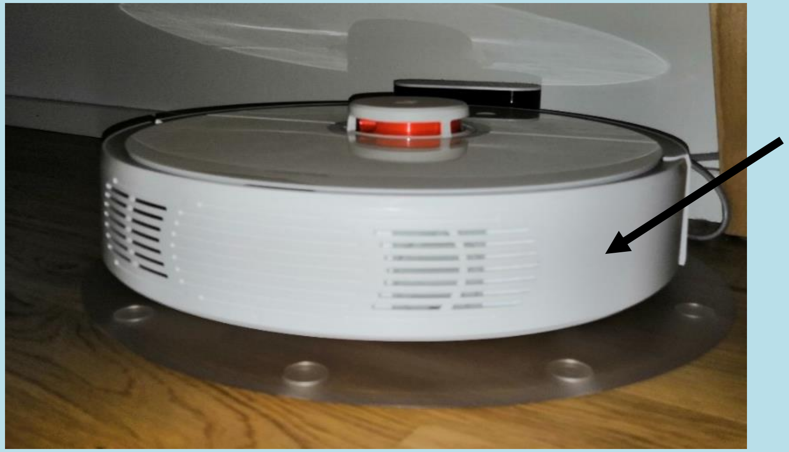
Recapitulation: Structure of Simple Reflex-Agents

Algorithm: Reflex-Vacuum Agent

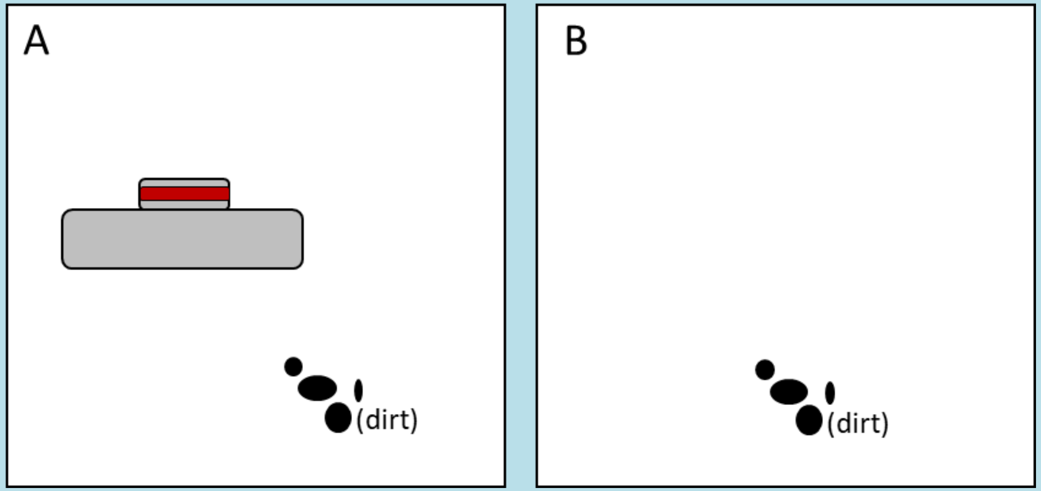
```
$\textbf{if } status = dirty \textbf{ then}$
    $return\ suck$
$\textbf{end}$

$\textbf{else if } location = A \textbf{ then}$
    $return\ right$
$\textbf{end}$

$\textbf{else if } location = B \textbf{ then}$
    $return\ left$
$\textbf{end}$
```

? What would you change if there are multiple rooms?

Adapted from Rusell, S., & Norvig, P. (2016)



Btw his name is dobby and I use him with my AI assistant Alex



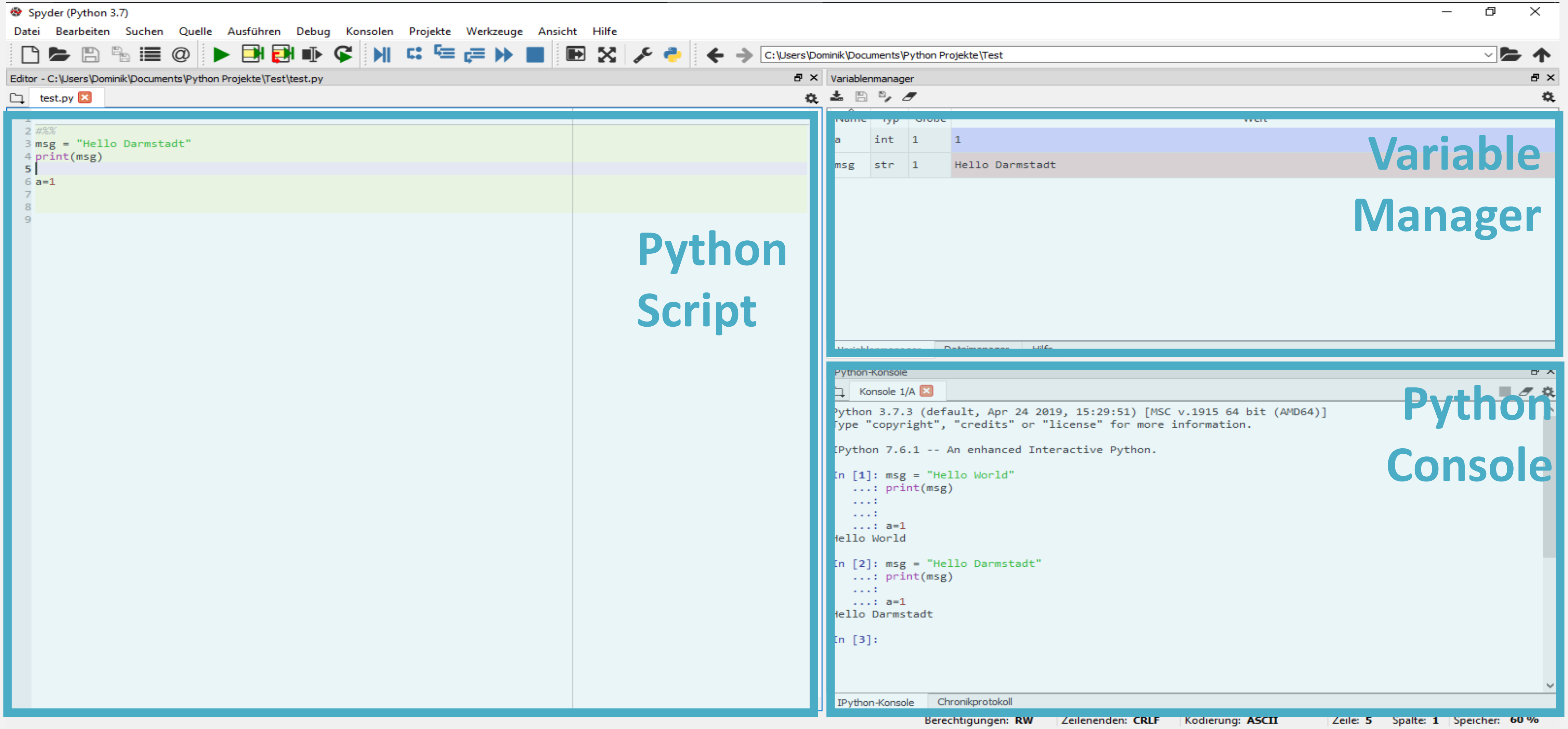
Implementing Randomness in Python

- You can use `numpy` to generate random integers in the intervall from `low` to `high`

```
import numpy

numpy.random.randint(low = 0, high = num_rooms)
```

Adapted from Ernesti, J. & Kaiser, P. (2017)

Spyder IDE


Spyder (Python 3.7)
Datei Bearbeiten Suchen Quelle Ausführen Debug Konsolen Projekte Werkzeuge Ansicht Hilfe
C:\Users\Dominik\Documents\Python Projekte\Test
Editor - C:\Users\Dominik\Documents\Python Projekte\Test\test.py
test.py
Variablenmanager
Python-Konsole
Konsole 1/A
IPython-Konsole
Chronikprotokoll

```

#%%
msg = "Hello Darmstadt"
print(msg)

a=1
```

Python Script

Name	Typ	Größe	Wert
a	int	1	1
msg	str	1	Hello Darmstadt

Variable Manager

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]: msg = "Hello World"
   ...: print(msg)
   ...:
   ...:
   ...: a=1
Hello World

In [2]: msg = "Hello Darmstadt"
   ...: print(msg)
   ...:
   ...: a=1
Hello Darmstadt

In [3]:
```

Python Console

Berechtigungen: RW Zeilenenden: CRLF Kodierung: ASCII Zeile: 5 Spalte: 1 Speicher: 60 %

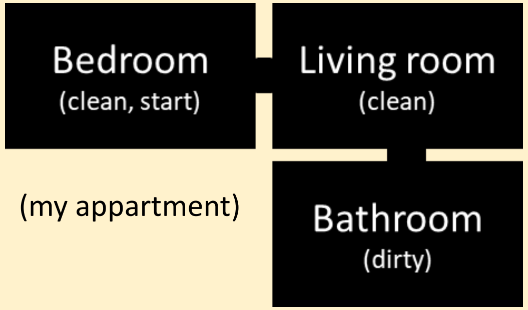
Classroom Case

You can find the solution of each lectorial online in the git repository if you need some starting help!

Case

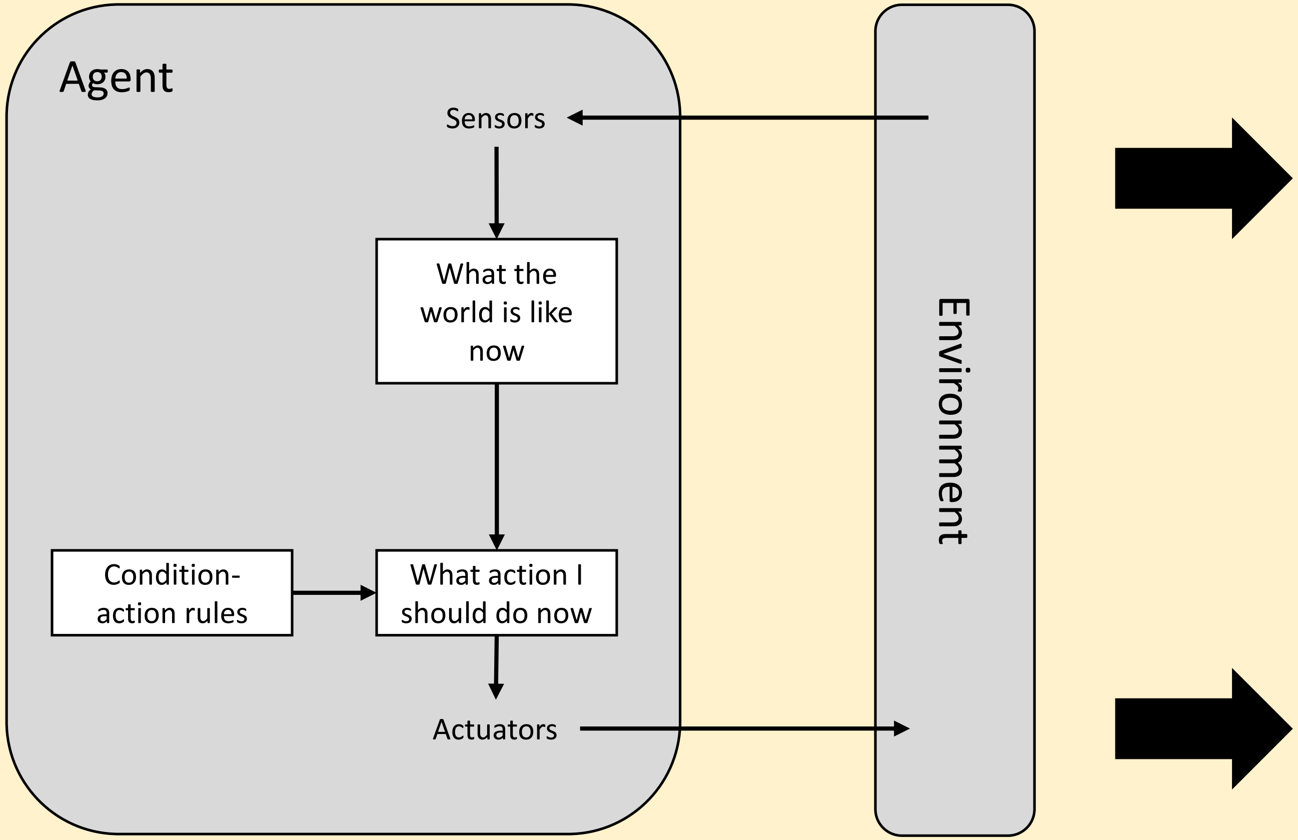
Implement the simple reflex-vacuum agent from the lecture to clean my apartment. Use the following steps as orientation for coding:

- Define a dictionary `vacuum_world` and store the `room names`, `neighbor rooms`, and `cleaning status`. Set “`Bedroom`” as starting point. Use the following map to initialize the `vacuum_cleaner_world`:



- Use the `agent class` on the next slide as template to build your own agent implementing the simple reflex logic from the lecture with e.g. random room selection. Decide for yourself which action will consume energy and reduce energy by one for each of these actions.
- Use a `while` function to setup an simulation that stops, when the whole world is cleaned up. Use print statements to print the status in the console.

Classroom Case



```
class Cleaner:
    energy = 5

    def __init__(self, room):
        pass

    # cost function
    def power_consumption(self, world):
        pass

    # perception
    def percepts(self, world):
        pass

    # actions
    def suck(self, world):
        pass

    def drive(self, world):
        pass

    def act(self, world):
        pass
```

Adapted from Rusell, S., & Norvig, P. (2016)

Further questions?

→ Coding

Artificial Intelligence

Algorithms and Applications with Python

Lectorial 01

Dr. Dominik Jung
dominik.jung42@gmail.com

Image source: ↗ Pixabay (2019) / ↗ CC0

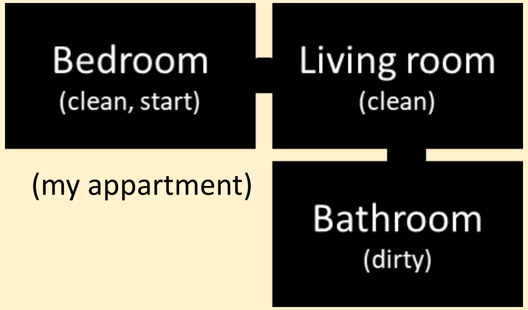
Classroom Case

You can find the solution of each lectorial online in the git repository if you need some starting help!

Case

Implement the simple reflex-vacuum agent from the lecture to clean my apartment. Use the following steps as orientation for coding:

- Define a dictionary `vacuum_world` and store the `room names`, `neighbor rooms`, and `cleaning status`. Set “`Bedroom`” as starting point. Use the following map to initialize the `vacuum_cleaner_world`:

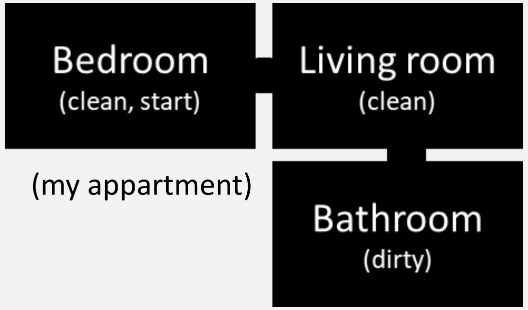


- Use the `agent class` on the right as template to build your own agent implementing the simple reflex logic from the lecture with e.g. random room selection. Decide for yourself which action will consume energy and reduce energy by one for each of these actions.
- Use a `while` function to setup an simulation that stops, when the whole world is cleaned up. Use print statements to print the status in the console.

```
class Cleaner:
    energy = 5

    def __init__(self, room):
        pass

    # cost function
    def power_consumption(self, world):
        pass

    # perception
    def percepts(self, world):
        pass

    # actions
    def suck(self, world):
        pass

    def drive(self, world):
        pass

    def act(self, world):
        pass
```

1. Setup world status

```
#%% 1. Define variables
vacuum_world = { "Bedroom" : [["Living room"], False],
      "Living room" : [["Bedroom","Bathroom"], False],
      "Bathroom": [["Living room"], True]}

start_room = "Bedroom"

print("Cleaning need of the start room is:", vacuum_world[start_room][1])
```



Key	Values	
Bedroom	Living room	False
Living room	Bedroom, Bathroom	False
Bathroom	Living room	True

Nested list

! If you store a list in a list (and even further lists in lists) this is a data structure termed "nested list"

2. Use Cleaner Class Template to Setup Your Agent

```
class Cleaner:
    energy = 5

    def __init__(self, room , vacuum_world):
        Set start room and world

    # cost function
    def power_consumption(self):
        We will reduce the energy one step for
        each activation of the agent

    # perception
    def percepts(self, vacuum_world):
        We will check if the current room is
        dirty or not and return the value

    # actions
    def suck(self):
        We will clean the room (update the
        cleaning status of room in the world)

    def drive(self):
        We will drive to the next room (update
        our position in the world)

    def act(self, status):
        Decide what to do based on percepts
```

Algorithm: Reflex-Vacuum Agent

```
$\textbf{\textit{if}}\ status = dirty\ \textbf{\textit{then}}$
    $return\ suck$
$\textbf{\textit{end}}$

$\textbf{\textit{else if}}\ location = A\ \textbf{\textit{then}}$
    $return\ right$
$\textbf{\textit{end}}$

$\textbf{\textit{else if}}\ location = B\ \textbf{\textit{then}}$
    $return\ left$
$\textbf{\textit{end}}$
```

2. Build Agent Logic: Initialization, Cost Function and Perception

```
class Cleaner:
    energy = 5

    def __init__(self, room , vacuum_world):
        Set start room and world

    # cost function
    def power_consumption(self):
        We will reduce the energy one step for
        each activation of the agent

    # perception
    def percepts(self, vacuum_world):
        We will check if the current room is
        dirty or not and return the value

    # actions
    def suck(self):
        We will clean the room (update the
        cleaning status of room in the world)

    def drive(self):
        We will drive to the next room (update
        our position in the world)

    def act(self, status):
        Decide what to do based on percepts
```

```
def __init__(self, room, vacuum_world):
        self.location = room
        self.world = vacuum_world

    # cost function
    def power_consumption(self):
        self.energy = self.energy-1

    # perception
    def percepts(self, vacuum_world):
        self.world = vacuum_world
        status = self.world[self.location][1]
        self.act(status)
```

2. Build Agent Logic: Actions and Decision Logic

```
class Cleaner:
    energy = 5

    def __init__(self, room , vacuum_world):
```

Set start room and world

```
    # cost function
    def power_consumption(self):
```

We will reduce the energy one step for each activation of the agent

```
    # perception
    def percepts(self, vacuum_world):
```

We will check if the current room is dirty or not and return the value

```
    # actions
    def suck(self):
```

We will clean the room (update the cleaning status of room in the world)

```
    def drive(self):
```

We will drive to the next room (update our position in the world)

```
    def act(self, status):
```

Decide what to do based on percepts

```
# actions
   def suck(self):
       self.world[self.location][1] = False
       self.power_consumption()

   def drive(self):
       neigbor_rooms = self.world[self.location][0]
       num_rooms = len(neigbor_rooms)
       r = numpy.random.randint(low = 0, high = num_rooms)

       self.location = neigbor_rooms[r]
       print("Drive to next room: {}".format(self.location))
       self.power_consumption()

   def act(self, status):
       room_status = status
       if(room_status == True):
           self.suck()
           print("Room {} is dirty, clean room".format(self.location))
       else:
           print("Room {} is clean".format(self.location))
           self.drive()

       print("Energy left: {}".format(self.energy))
       if(self.energy <= 1):
           self.location = "Bedroom"
           print("Return to docking station.")
```

3. Start Simulation

```
dobby = Cleaner("Bedroom", vacuum_world)
stop = False

while stop != True:
    world = dobby.world
    world_status = world.values()

    cleaning_status=[]
    for room in world_status:
        cleaning_status.append(room[1])

    if(True in cleaning_status):
        dobby.percepts(world)
    else:
        print("Finished Cleaning")
        stop=True
```

```
Room Bedroom is clean
Drive to next room: Living room
Energy left: 4
Room Living room is clean
Drive to next room: Bathroom
Energy left: 3
Room Bathroom is dirty, clean room
Energy left: 2
Finished Cleaning
```

Classroom Case

Case

Now implement other agent types, for that purpose expand the simple reflex agent by your own:

- Use another `cost_function`
- Expand the cleaning map and energy costs
- Implement an other type of agent e.g. an utility-based agent or model-based agent use an specific cleaning strategy (instead of random room selection) and the `cost_function` to model utility based behaviour
- …

{ Just Keep Coding }