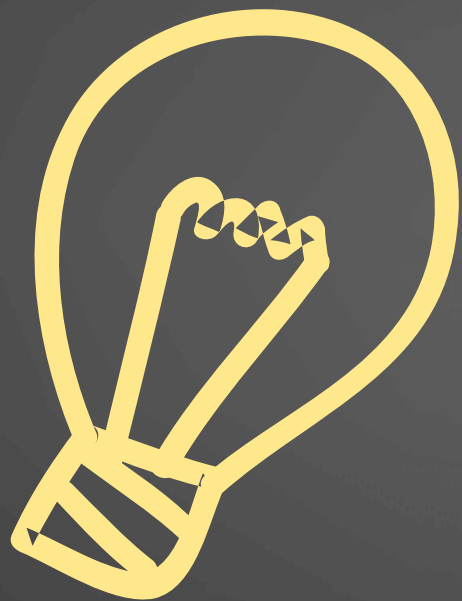


Duckiebot介紹

Python與Jupyter Notebook

主講：呂霽欣
助教：李聖誠、張博凱
技術指導：王學誠 教授

CONTENTS



01.VirtualBox介紹

02.Linux基本指令教學

03.Duckietown與ROS

04.Python語法教學和應用

05.Car Command函數



01.VirtualBox介紹

VirtualBox是什麼？



虛擬電腦介面

模擬個人電腦的硬體環境。
可以在虛擬的電腦硬體中安裝
作業系統、測試軟體或病毒

獨立作業環境

可以跟現在的電腦系統同時運作，而且兩個系統的資料與程式不會互相干擾或影響。



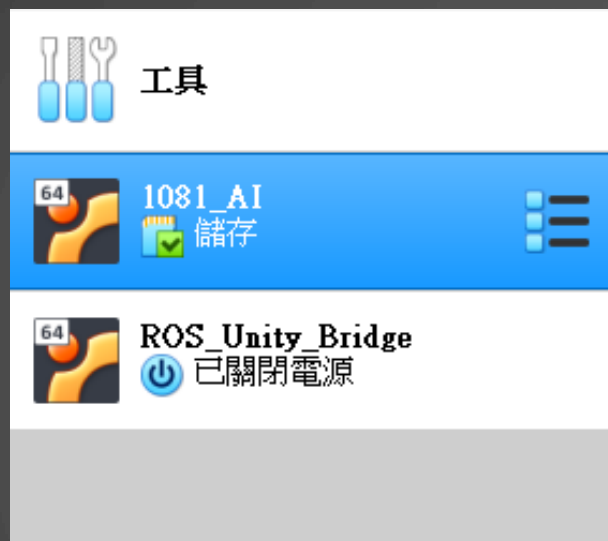
支援多個作業系統

可以安裝並且執行Solaris、DOS、Windows、Linux、OS/2 Warp、OpenBSD及FreeBSD等系統。

實用便利低風險

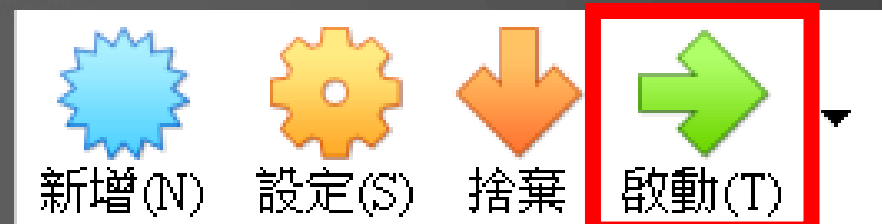
可與主機端電腦共用剪貼簿與資料夾，也不必擔心安裝錯誤或系統中毒。

Let's Start!



選擇虛擬電腦

於VirtualBox左側工具列選取
要開啟的虛擬電腦



啟動虛擬電腦

點擊紅框處的「啟動」鍵，
啟動虛擬電腦



02.Linux基本指令教學

Linux是什麼?



Linux是一種自由開放原始碼的類 Unix 的作業系統，其廣泛運用於伺服器 and 嵌入式系統中。目前主流的 Linux 發佈版本(指可完整安裝使用的套件版本)包括：Fedora、CentOS、**Ubuntu** 等。



Linux常用指令



`$ ls`

list, 查看檔案及
子目錄。

`$ cd [資料夾名稱]`

change directory,
移動進該資料夾

`$ mkdir`

make directory,
創建新資料夾

`$ cp [檔案名稱]`

copy, 複製檔案

`$ mv`

move(rename) files,
移動檔案或是重新
命名檔案

`$ rm [檔案名稱]`

remove file, 刪
除檔案



03.Duckietown與ROS

機器人的基本結構



核心運算

機器人的**大腦**，我們會在裡面寫下程式碼，讓機器人可以照所想的進行感測、規劃、執行。



感測器

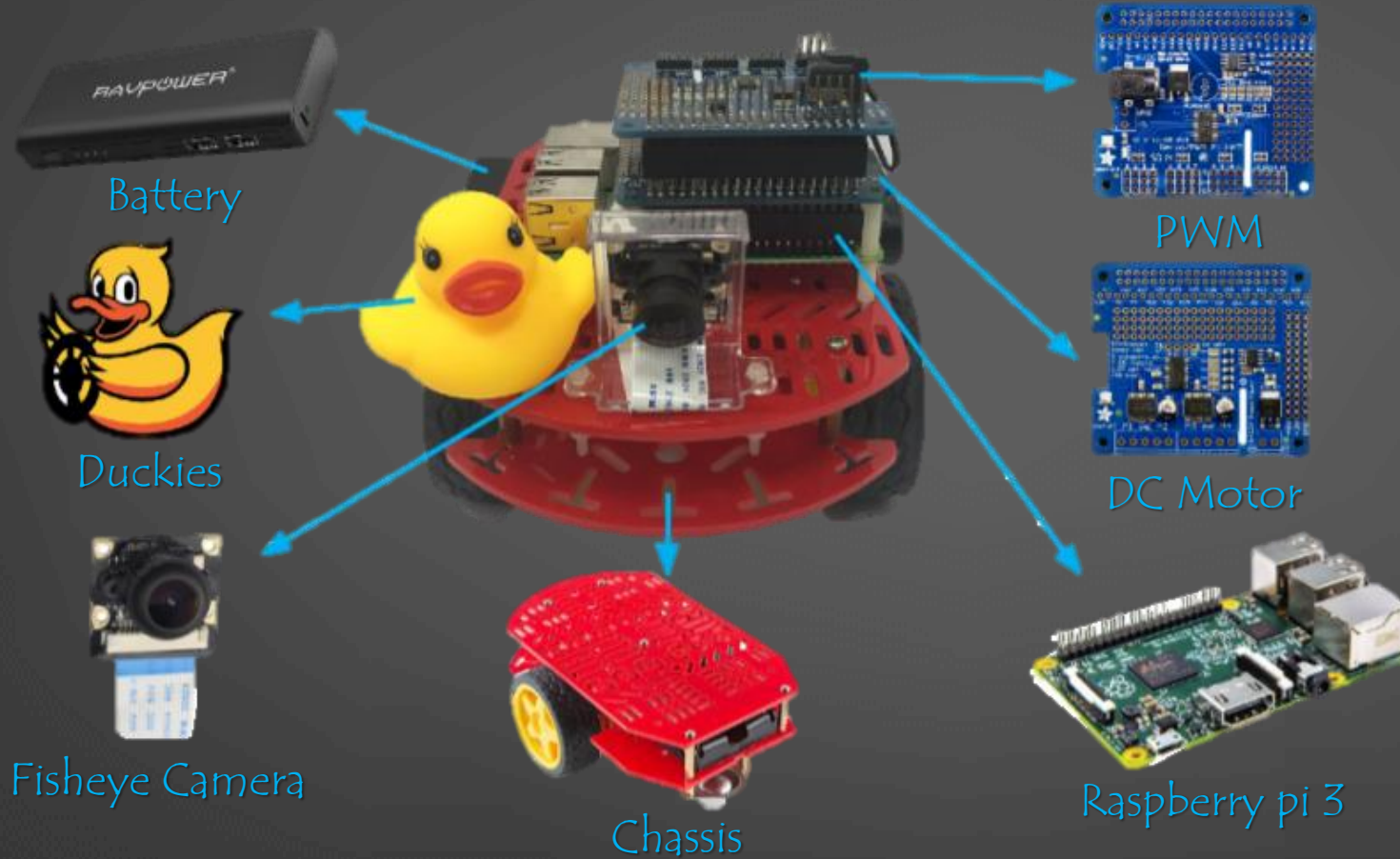
就像是人類的**眼睛**，賦予機器人「視覺」，使機器人知道環境的變化，進而讓核心運算做出正確的決定。



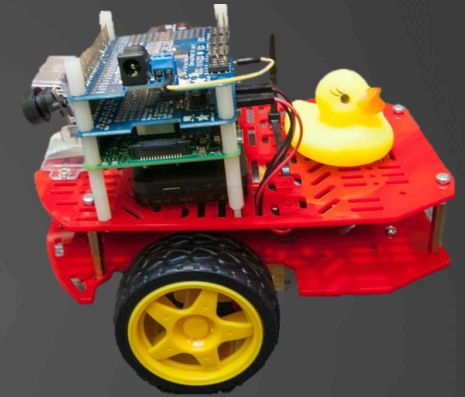
運動

機器人根據感測器及核心運算後所進行的**反應**，例如：馬達運轉、機器手臂運動。廣義來說，用語音做表達、螢幕顯示訊息，都可以算是「運動」。

Duckietown and Duckiebot



Duckiebot



Joystick



Duckiebot小鴨車主架構



核心運算：

樹莓派 Raspberry Pi 3 Model B +

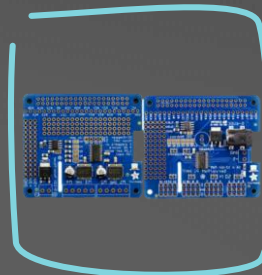
可以把它看作是一台**小電腦**，上面一樣有中央處理器（CPU）、記憶體（RAM）。



感測器：

RPi 鏡頭 (G)

支援所有版本的樹莓派，其魚眼鏡頭的設計可以讓視角寬度達**160 度**，最高解析度達**2592x1944 或 1080p**。



運動：

PWM板(左)及馬達驅動板(右)

PWM 板的目的是要用PWM的技術控制馬達驅動。而馬達、馬達控制器，就是小鴨車「運動」的部分。

Duckiebot自動駕駛原理

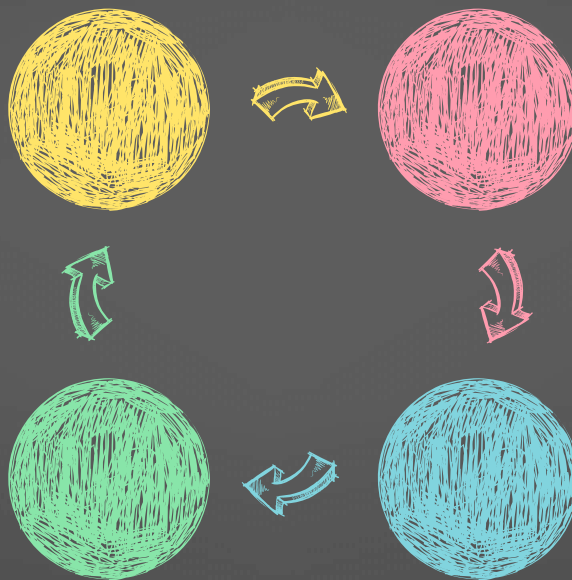
接收圖片

鏡頭得到圖片之後，調整光線對比，讓小鴨車可以更容易區分黃線、白線、紅線。



方向修正

經過運算，決定左轉、右轉、直走後，再給馬達相對應的動力。



辨識線段

偵測圖中的黃、白線邊緣(藍、綠線段)，幫助定位小鴨車。



推算位置

將線段在照片中的**像素**位置，轉換成真實世界中以相機為原點的三維座標位置(x, y, z)，並回推小鴨車的位置。

Demo

- 打開終端Terminal

```
$ ssh admin@小鴨車名稱(ex.aiXX) (密碼 : aidriving)
```

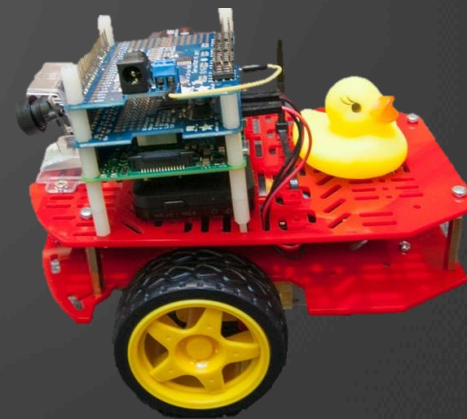
```
$ cd duckietown
```

```
$ source environment.sh
```

```
$ source set_ros_master.sh aiXX
```

```
$ source set_vehicle_name.sh aiXX
```

```
$ roslaunch duckietown_demos lane_following.launch veh:=aiXX
```



ROS基本介紹



- 全名為Robot Operating System(機器人作業系統)
- 提供類似於作業系統的服務
- 提高了機器人領域的代碼複用率，實現其設計目標「**共享與協作**」
- 適合於大型實時系統與大型的系統開發項目

The ROS logo, consisting of a dark blue rectangle. On the left side of the rectangle are three vertical columns of three white dots each. To the right of the dots, the letters "ROS" are written in a large, white, sans-serif font.

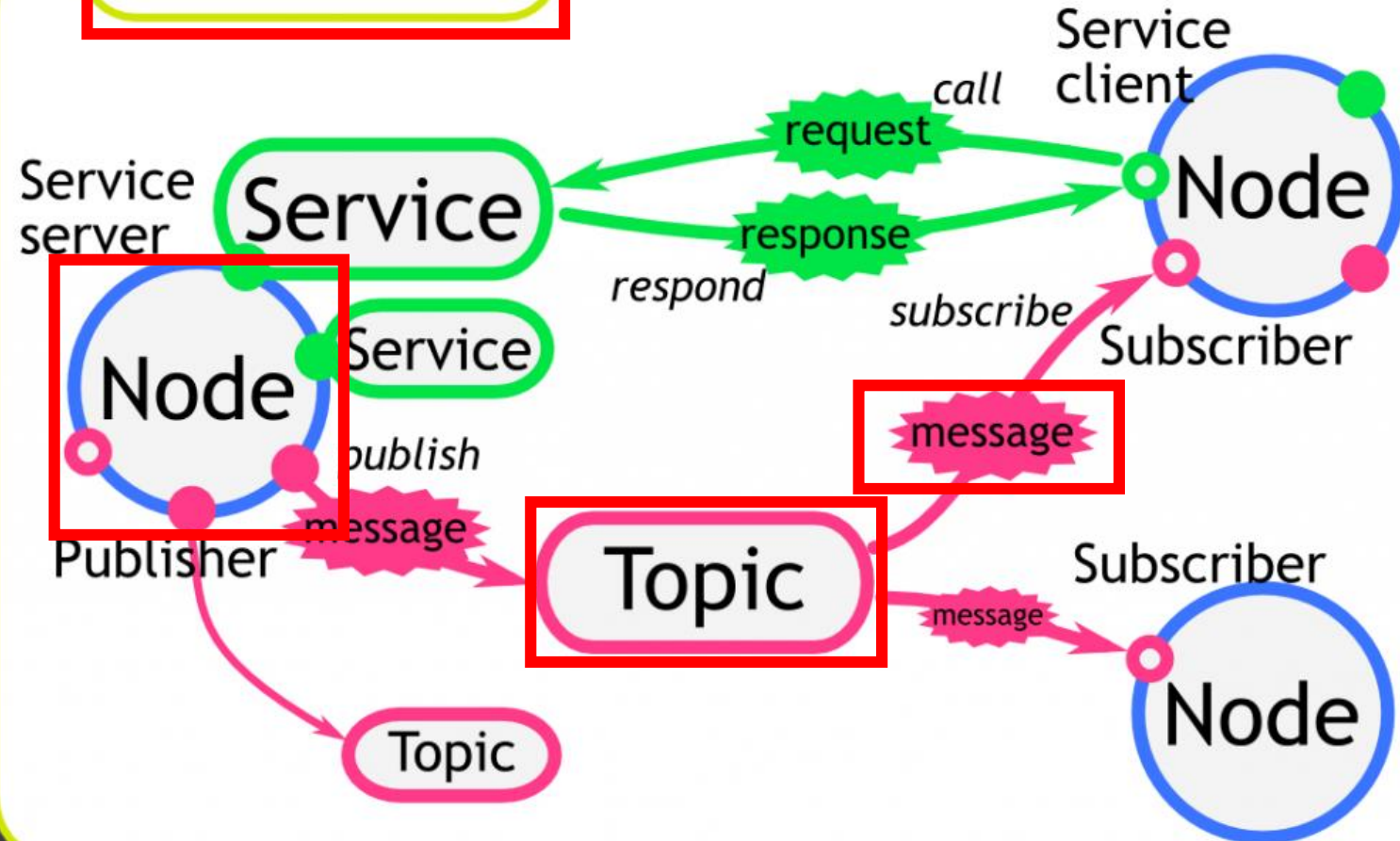
ROS系統架構



Master:
管理ROS中
個Node，
負責Node之
信息傳遞

ROS Master

192.168.1.100:11311





04. Python語法教學和應用

Python



廣泛運用的程式語言

現在非常廣用的程式語言之一，不管是在資料科學、機器學習、機器人領域都能看見它的蹤跡。

易上手易閱讀

Python語言特別強調其可讀性，讓程式碼可以輕易地被他人理解。

直譯式語言

Python語法不需先編譯(Compile)再執行，而是一邊執行一邊編譯，與C、C++等語言很不同。

Jupyter Notebook



- 類似於Google doc，為網路上的文字編輯軟體。
- 使用者可以在裡面直接**書寫程式碼**，並且**執行**。
- 能將程式碼分成一個一個的區塊 (Cell)，並且分別觀看輸出結果，方便檢查每段的程式是否有期望的輸出抑或是挑出其中的 Bug。
- 只能使用**直譯式**語言，例如 **Python**。

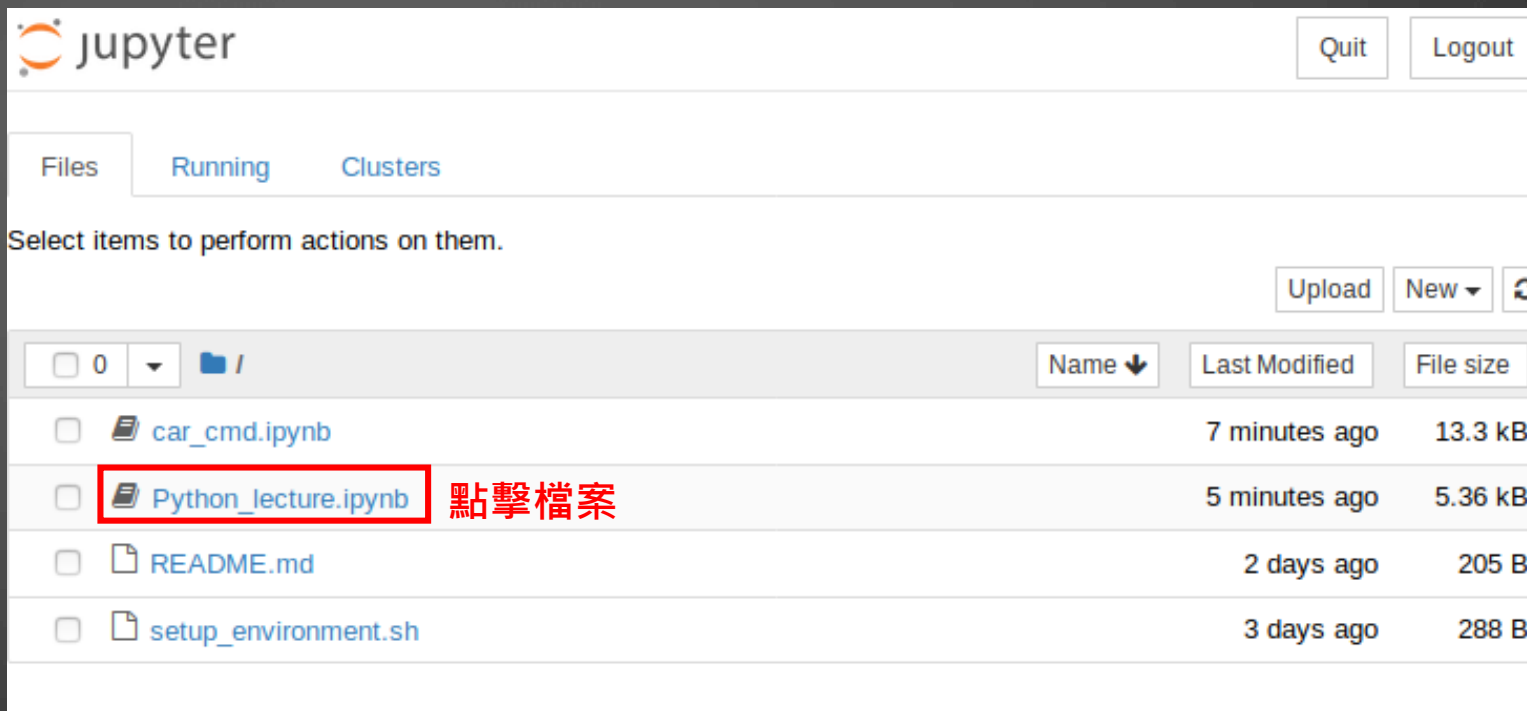
進入Jupyter Notebook

- 打開終端Terminal

```
$ cd hs_winter_lecture
```

```
$ jupyter notebook
```

開啟後畫面→



The screenshot shows the Jupyter Notebook web interface. At the top, there's a 'jupyter' logo and 'Quit' and 'Logout' buttons. Below that are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' followed by 'Upload', 'New', and a refresh icon. A file browser shows the current directory as '/'. A table lists files:

	Name ↓	Last Modified	File size
<input type="checkbox"/>	car_cmd.ipynb	7 minutes ago	13.3 kB
<input type="checkbox"/>	Python_lecture.ipynb	5 minutes ago	5.36 kB
<input type="checkbox"/>	README.md	2 days ago	205 B
<input type="checkbox"/>	setup_environment.sh	3 days ago	288 B

The file 'Python_lecture.ipynb' is highlighted with a red box, and the text '點擊檔案' (Click the file) is written next to it.

簡單Jupyter Notebook指令

- Enter/esc
進入/跳出該cell
的編輯模式

- shift+Enter
執行該cell

- ↑ ↓
選擇不同的cell

- a
向上加一個cell

- b
向下加一個cell

- x
刪除當前cell

Python-1 印出句子

Python Basic

print a sentence

```
In [1]: print ("Hello world and Duckietown!")
```

```
Hello world and Duckietown!
```

Python-2 變數與加減法

Initialize a variable and addition

```
In [2]: duckiebot = 0  
a = 1  
total = duckiebot + a  
  
print ("The total is:", total)  
  
The total is: 1
```

Practice

A small task - 1

```
In [3]: ### copy and paste the above cell and make it to subtraction
```

```
In [ ]:
```

Python-3 乘法與除法



Multiplication and division

```
In [4]: duckies = 5
students = 10
total_amount = duckies*students

print ("How many duckies do we need?", total_amount)

duckies = 60
students = 12
duckie_per_student = duckies/students

print ("A student can get this amount of duckies.", duckie_per_studen
```

```
How many duckies do we need? 50
A student can get this amount of duckies. 5.0
```


Python-4 定義函數

Function in Python

In [5]: *### A function is a part of code that can be called and reused whenever*

```
In [6]: def addition(a, b):  
        result = a + b  
        return result
```

In [7]: *### How do we used a defined function?*

```
In [8]: duckies = 5  
        chickens = 10  
        total_birds = addition(duckies, chickens)  
  
        print ("Here are the total birds we have.", total_birds)
```

Here are the total birds we have. 15

Python-5 for 迴圈



For loop

In [9]: `### A for loop is a "loop" which you can defined how many times you w`

In [10]: `for duckie in range(5): # i = 0, 1, 2, 3, 4 in each loop
 print ("Now the 'duckie' is", duckie)`

Now the 'duckie' is 0
Now the 'duckie' is 1
Now the 'duckie' is 2
Now the 'duckie' is 3
Now the 'duckie' is 4

In [11]: `for duckie in range(2, 5): # i = 2, 3, 4 in each loop
 print ("Now the 'duckie' is", duckie)`

Now the 'duckie' is 2
Now the 'duckie' is 3
Now the 'duckie' is 4

Python-6 if-else 條件

if-else

```
In [12]: duckies = 5  
  
if duckies > 0:  
    print ("There are duckies.")  
else:  
    print ("There are no duckies.")
```

There are duckies.



05.Car Command函數

開啟car command模式-1

- 打開終端Terminal 1

```
$ ssh admin@aiXX (密碼 : aidriving)
```

```
$ cd duckietown
```

```
$ source environment.sh
```

```
$ source set_ros_master.sh aiXX
```

```
$ roslaunch duckietown car_command_control.launch
```

開啟car command模式-2

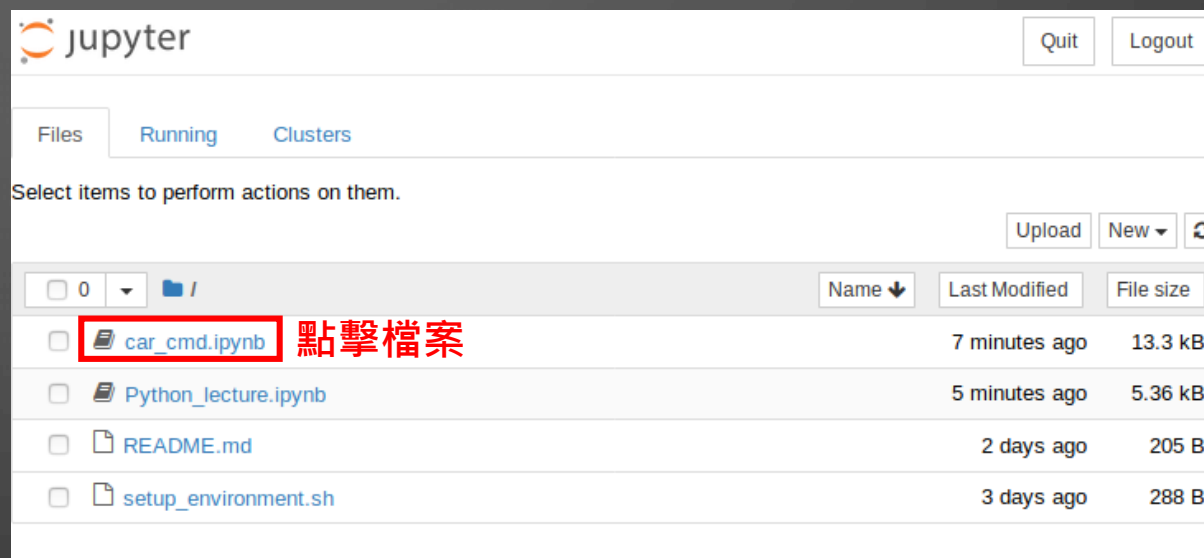
- ctrl+t 打開另一個終端Terminal 2

```
$ source ~/duckietown/environment.sh
```

```
$ source ./setup_environment.sh aiXX
```

```
$ jupyter notebook
```

開啟後畫面→



Car commands and ROS

Import ROS

```
In [13]: # Environment set up
import sys
# rospy
sys.path.insert(0, '/opt/ros/kinetic/lib/python2.7/dist-packages')
# rospkg
sys.path.insert(0, '/usr/lib/python2.7/dist-packages/')

# duckietown_msgs
duckietown_root = './duckietown/' # set this path to your duckietown directory
sys.path.insert(0, duckietown_root + 'catkin_ws/devel/lib/python2.7/dist-packages')

import rospy
from duckietown_msgs.msg import Twist2DStamped
from sensor_msgs.msg import Joy
```

Initialize Node and Publisher

```
In [14]: #please replace "ai19" with your duckiebot name
#if rosgraph error occurs, close jupyter notebook and run setup.bash first
pub_car_cmd = rospy.Publisher("/ai19/joy_mapper_node/car_cmd", Twist2DStamped, queue_size=1)
rospy.init_node("joy_mapper_node", anonymous=True)
```

Car commands

Examples of Using car_command

Ex1: Forward velocity = 1 for 0.75 seconds

In [16]: `#car_command(1, 0, 0.75)`

EX2: Turn with angular velocity = 4 for 1.0 seconds

In [17]: `#car_command(0, 4, 1.0)`

Define Function

Define car_command

```
In [15]: def car_command(v, omega, duration):  
# Send stop command  
    car_control_msg = Twist2DStamped()  
    car_control_msg.v = v  
    car_control_msg.omega = omega  
    pub_car_cmd.publish(car_control_msg)  
    rospy.sleep(duration)  
    #rospy.loginfo("Shutdown")  
    car_control_msg.v = 0.0  
    car_control_msg.omega = 0.0  
    pub_car_cmd.publish(car_control_msg)  
    rospy.sleep(0.5)
```

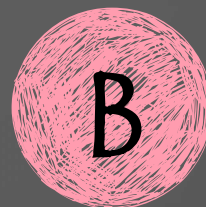
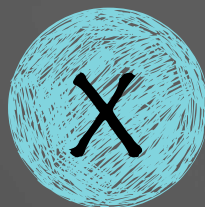
定義按鈕指令



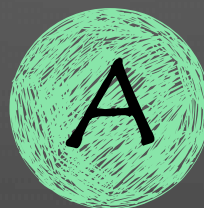
```
#button Y
def button_Y():
    # put your creation!!
    pass
```



```
# button B
def button_B():
    # turn around
    car_command(1, 10, 1)
```



```
# button X
def button_X():
    # for loop
    for i in range(4):
        car_command(1, 0, 0.5)
        car_command(0, 6, 0.5)
```



```
# button A
def button_A():
    # forward
    car_command(1, 0, 1)
```

Challenges



定義
按鈕

改變car command函數中的變數，定義屬於自己的按鈕Y。

完成指
定動作

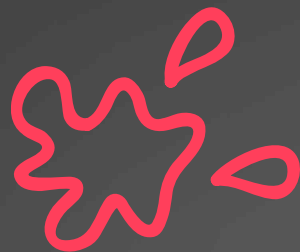
利用所學的Python語法，設計程式完成指定的動作。

到達
目標

挑戰使用「最少次」按鈕指令到達指定的位置。

自由
發揮

利用今天所學，設計理想的小鴨車！



Thank you

