Python 程式設計

Department of Computer Science and Information Engineering
Chaoyang University of Technology
Taichung, Taiwan, Republic of China

Instructor: De-Yu Wang (王德譽)
E-mail: dywang@csie.cyut.edu.tw
Homepage: http://dywang.csie.cyut.edu.tw
Phone: (04)23323000 ext 4538
Office: E738



Monday 15th June, 2026

- 教材宗旨：提供同學上網預習、複習及自我測驗。
- 學習目標：
  1. 瞭解 Python 語言
  2. 使用 Linux 作業系統之 Python 開發環境。
  3. 使用 Python 開發應用程式。
- 先修科目或先備能力：
- Instructor: De-Yu Wang
  1. Homepage: http://dywang.csie.cyut.edu.tw
  2. 本文件 pdf 檔
  3. 本文件網頁：https://dywang.csie.cyut.edu.tw/dywang/pythonProgram/
  4. Email: dywang@csie.cyut.edu.tw
  5. Phone: (04)23323000 ext 4538
  6. Office: E738
- 參考資料
  1. vim 官方網站亦稱 vim 爲程式開發工具。
  2. 大家來學 VIM（一個歷久彌新的編輯器）
  3. Python
  4. Python Tutorial
  5. Glade - A User Interface Designer
  6. PyGTK: GTK+ for Python
  7. PyGTK tutorial
  8. Python 線上程式練習
  9. 洪錦魁著，「Python 入門遇向高手之路王者歸來」，深石數位科技，2018 年 1 月。

Contents

1 實機練習系統 **1**

1.1 動機 1
1.2 系統設計 1
1.3 練習系統製作-VirtualBox 2
1.4 練習系統製作-Raw Image 5
1.5 NAT 設計 6
1.6 虛擬機管理界面 7
1.7 虛擬機管理界面 V2 8
1.8 系統管理與評分 9
1.9 系統註冊–上課前先確定三件事 11
1.10 虛擬機設定 12
1.11 系統伺服器管理 13
1.11.1 DNS server 13
1.11.2 LDAP server 14
1.11.3 NFS server 15

2 GIT 版本控制 **16**

2.1 Git 簡介 16
2.2 Git 命令 17
2.3 * 建立 Git Repository 17
2.4 *Http Git Server 19
2.5 *http 存取 Git Repository 20
2.6 Git Clone 21
2.7 Git Config 22
2.8 Git Push 22
2.9 實機練習 23

3 vi 編輯器 **25**

3.1 vi 與 vim 25

3.2 vi 的使用 25
3.3 vim 的額外功能 28
3.4 vi 實機練習題 30
3.4.1 練習一 30
3.4.2 練習二 31
3.4.3 練習三 32

4 認識 Python **33**
4.1 簡介 33
4.2 python 執行腳本 36
4.3 獨立執行 python 腳本 37
4.4 內縮語法 38
4.5 括號、引號、換行、註解 39
4.6 實機練習題 40

5 變數 **42**
5.1 變數指定 42
5.2 字串處理 43
5.3 名稱前單底線 43
5.4 名稱後單底線 45
5.5 名稱前後雙底線 45
5.6 名稱前雙底線 47
5.7 實機練習題 48
5.7.1 字串變數 48
5.7.2 單底線及雙底線 48
5.7.3 名稱前雙底線 49

6 資料型態 **50**
6.1 type function 50
6.2 進位轉換 50
6.3 強制轉換 51
6.4 數值運算常用函式 52
6.5 布林 Boolean 52
6.6 字串 String 53
6.7 字元 54
6.8 實機練習題 55
6.8.1 type 函式 55
6.8.2 進位轉換 55

6.8.3 強制轉換 56
6.8.4 數學函式 56
6.8.5 Boolean 57
6.8.6 字串 57
6.8.7 字元 58

7 運算子 **59**
7.1 算術運算子 59
7.2 比較運算子 60
7.3 指派運算子 60
7.4 邏輯運算子 61
7.5 身份運算子 62
7.6 成員運算子 62
7.7 Bitwise 運算子 63
7.8 實機練習題 64
7.8.1 算術運算子 64
7.8.2 比較運算子 65
7.8.3 指派運算子 65
7.8.4 邏輯運算子 66
7.8.5 身份運算子 66
7.8.6 成員運算子 67
7.8.7 Bitwise 運算子 67

8 Print 格式化輸出 **69**
8.1 Python3 內建 print 函式 69
8.2 字串模數運算子 (%) 70
8.3 format() 方法 70
8.4 f-字串 71
8.5 格式轉換規則 72
8.6 實機練習題 73

9 Input 輸入 **75**
9.1 input 函式 75
9.2 Type 轉換 75
9.3 實機練習題 76

10 判斷 **78**
10.1 if statement 78

10.2 if...else statement 79
10.3 if...elif...else Statement 80
10.4 nested if statements 81
10.5 and or 82
10.6 實機練習題 84
10.6.1 if 84
10.6.2 if..else 84
10.6.3 if..elif..else 84
10.6.4 nested if 85
10.6.5 and or 86

11 串列 **87**
11.1 簡介 87
11.2 串列統計資料 87
11.3 串列長度 88
11.4 更改元素內容 89
11.5 串列相加 89
11.6 串列乘一數字 90
11.7 刪除串列或元素 90
11.8 實機練習題 91

12 元組 **94**
12.1 簡介 94
12.2 定義 94
12.3 list 與 tuple 比較 95
12.4 修改元組 95
12.5 元組的 max min sum 96
12.6 串列元組互換 96
12.7 實機練習題 97

13 迴圈 **99**
13.1 while loop 99
13.2 for loop 100
13.3 range function 101
13.4 nested loops 102
13.5 for - break 103
13.6 for - continue 104
13.7 for - else 105

13.8 實機練習題 106
13.8.1 while 106
13.8.2 for 106
13.8.3 range 107
13.8.4 nested for 107
13.8.5 for break 108
13.8.6 for continue 108
13.8.7 for else 108

14 字典 **109**
14.1 簡介 109
14.2 定義 109
14.3 增加元素 110
14.4 變更元素內容 110
14.5 刪除元素 111
14.6 驗證元素是否存在 111
14.7 遍歷字典鍵值 112
14.8 遍歷字典鍵 114
14.9 遍歷字典值 115
14.10 實機練習題 115
14.10.1 字典定義、增加、變更、驗證 115
14.10.2 遍歷字典 116

15 函式 **118**
15.1 簡介 118
15.2 定義 118
15.3 呼叫函式 119
15.4 傳入多個參數 119
15.5 參數預設值 120
15.6 回傳值 121
15.7 任意數量參數 122
15.8 Global vs. Local variables 123
15.9 匿名函式 lambda 124
15.10 匿名函式與 map 函式 124
15.11 實機練習題 125
15.11.1 average function 125
15.11.2 lambda function 126

16 模組 **127**
16.1 簡介 127
16.2 自建模組 127
16.3 導入模組 127
16.4 導入部分函式 128
16.5 導入所有函式 129
16.6 as 替代名稱 130
16.7 系統 sys 模組 131
16.8 實機練習題 132
16.8.1 自建模組 132
16.8.2 sys 模組 133

17 檔案讀取寫入 **135**
17.1 簡介 135
17.2 open 開檔 135
17.3 file 物件屬性 136
17.4 close method 137
17.5 write method 137
17.6 read method 138
17.7 File Positions 139
17.8 os 模組 139
17.9 實機練習題 141

18 re 模組之正規表示式 **143**
18.1 前言 143
18.2 re 模組函式 143
18.3 匹配模式 pattern 145
18.4 選項 flag 147
18.5 match 函式 147
18.6 search 函式 148
18.7 findall 函式 149
18.8 sub 函式 150
18.9 匹配數量範例 151
18.10 實機練習題 151

19 除錯與異常處理 **155**
19.1 簡介 155
19.2 標準異常 156

19.3 try-except . . . 157
19.4 try-except-else . . . 158
19.5 try-finally . . . 159
19.6 Exception 參數 . . . 160
19.7 raise 觸發異常 . . . 161
19.8 自定義異常類型 . . . 162
19.9 實機練習題 . . . 163

20 網路爬蟲 165
20.1 前言 . . . 165
20.2 簡單上網 . . . 165
20.3 Google 地圖 . . . 166
20.4 requests 模組 . . . 166
20.5 網頁存取失敗 . . . 167
20.6 偽裝瀏覽器 . . . 168
20.7 儲存網頁 . . . 169
20.8 BeautifulSoup 安裝啓用 . . . 170
20.9 bs4 find . . . 171
20.10 bs4 getText . . . 172
20.11 bs4 select . . . 173

21 PyGTK 176
21.1 PyGTK 簡介與開始 . . . 176
21.2 PyGTK 除錯 . . . 177
21.3 Buttons 與 Layout . . . 179
21.4 Signals, Events and Functions . . . 180
21.5 Label and Entry . . . 182
21.6 實機練習題 . . . 185
21.6.1 練習一 . . . 185

22 Glade 187
22.1 簡介 . . . 187
22.2 Calculator Example . . . 189
22.3 實機練習題 . . . 190
22.3.1 練習一 . . . 190

Chapter 1

實機練習系統

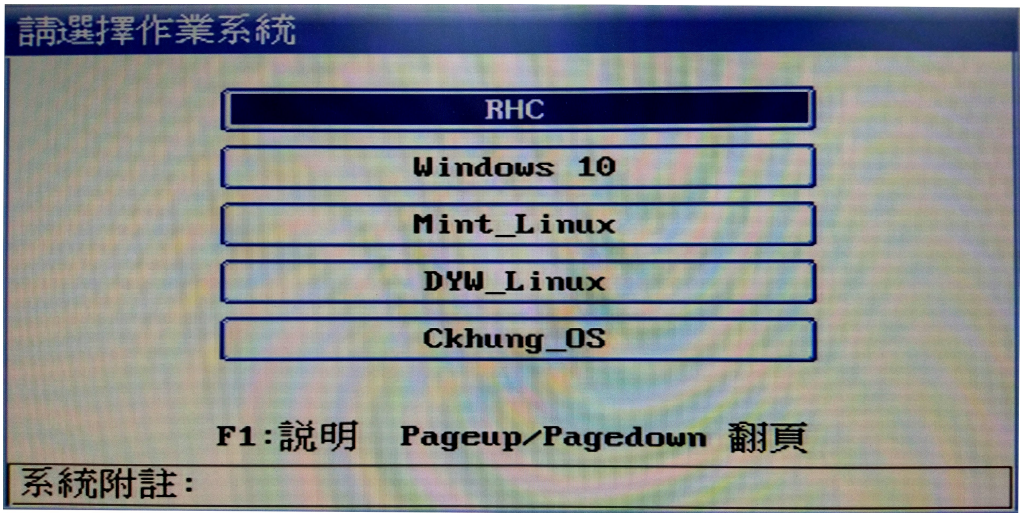
1.1 動機

1. 教學若有實際環境，讓學生實機操作練習，可以大大提昇學習成效。
2. Linux 實機練習，很容易因學生操作錯誤，而造成系統損壞。例如：硬碟分割、格式化及管理的練習。
3. 實機練習結束或操作不當造成當機，系統必須可以迅速還原，才能重複練習。
4. 系統必須可以批次快速修改或更新。
5. 系統必須可以依據學生實際操作狀況，批次進行檢查或評分。
6. 虛擬機練習所需伺服器資源，例如：LDAP 網路帳號、YUM repository 套件庫伺服器、NFS 分享、NTP 校時伺服器、ISCSI 硬碟分享、內部網域 DNS 伺服器、FTP 資源下載... 等環境必須架設好，才能提供虛擬機做各種狀況的練習。

1.2 系統設計

1. Linux 教學系統設計
   (a) 客製化 Linux 系統。
   (b) Linux 伺服器架設。
   (c) VNC 遠端桌面與 KVM 虛擬機。
2. Linux 教學系統安裝

(a) E517 電腦教室：方便學生上課使用。



(b) 個人隨身碟：插上隨身碟，以安裝光碟開機，不需要任何設定，自動安裝於隨身碟，不會影響電腦中的硬碟。

(c) LiveCD：以 LiveCD 光碟開機，開機後即可使用，只是所有操作與設定都是暫存，重新開機後即回到原始狀態。

1.3 練習系統製作-VirtualBox

1. 下載 DYW Linux 練習系統 vdi 檔，下載網址公告於本平台課程內，請登入查看。

```
[root@dywmsi ~]# unxz crt9.vdi.xz
```

2. 依自己使用的 OS (Mac Windows Linux) 下載安裝 VirtualBox。

```
https://www.virtualbox.org/wiki/Downloads
```

3. DYW Linux 練習系統，Host 主機以 virsh 管理 VM，最低硬體因所需 VM 個數不同：

(a) CPU 4 核心以上，若修的課不用內部虛擬機，設定 2 核心也可以。

(b) RAM 8G 以上，若修的課不用內部虛擬機，設定 4G RAM 也可以。

(c) RHCSA, RHCE 請設定 4 核心，8G RAM 以上

(d) CPU 支援虛擬化技術 (virtualization technology)，BIOS 啓用 VT。

4. VirtualBox 新增 VM

```
New(Ctrl+N)
    Name: 自訂，例如 crt9
    Type: Linux
    Version: 只要是 64-bit 就可以，例如 Fedora (64-bit)
    Next >
Memory size
    8192 MB
    Next >
Hard disk
    Use an existing virtual hard disk file
    新增解壓縮的 crt9.vdi，並選用這個硬碟檔。
    Create
```

5. 若要使用內部虛擬機，必須保證巢式虛擬化有開啓，假設匯入的 VM 名稱爲 crt9

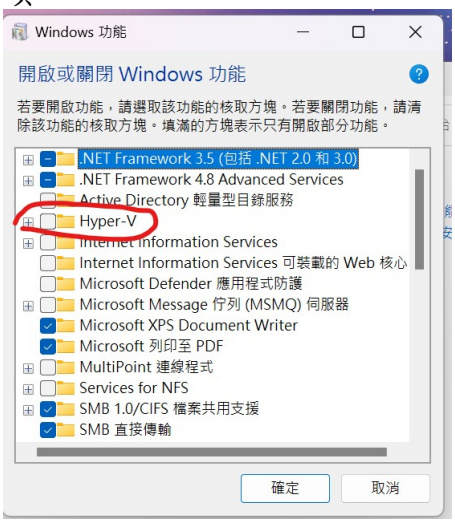
   (a) Host 是 Linux，在終端執行以下命令：

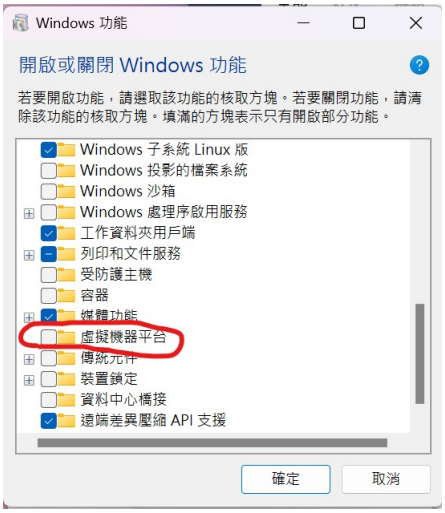
   ```
   [dywang@dywmsi ~]$ VBoxManage modifyvm crt9 --nested-hw-virt on
   ```

   (b) Host 是 window，在命令提示 cmd 執行以下命令：

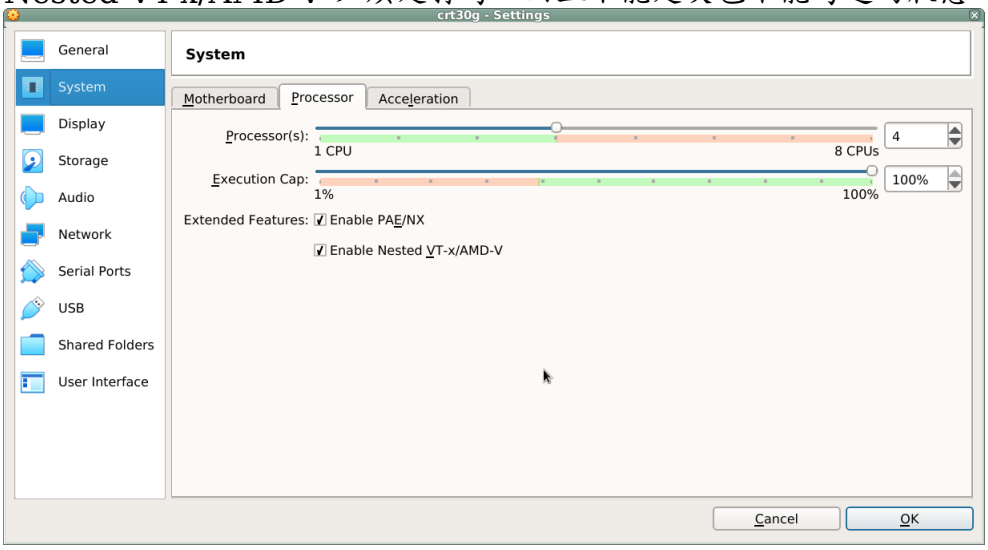
   ```
   cd "c:\Program Files\Oracle\VirtualBox"
   VBoxManage.exe modifyvm crt9 --nested-hw-virt on
   ```

   (c) Host 是 window，還必須關閉「Hyper-V」及「虛擬機器平台」，兩個選項。

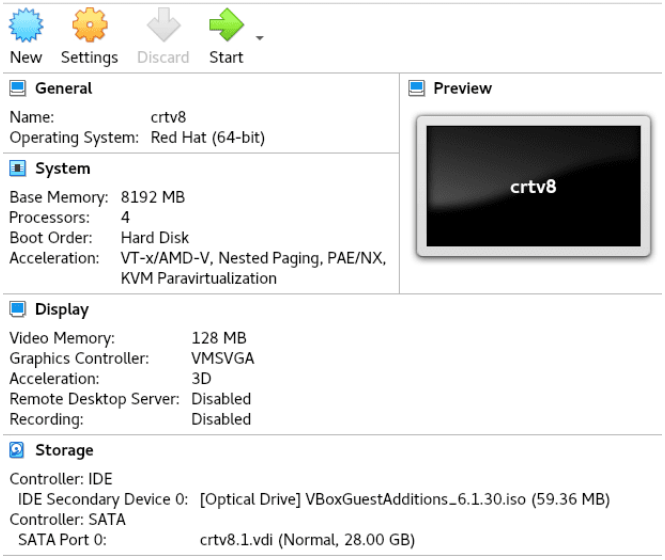




6. 進 VirtualBox 查看 VM crt9 的設定，system -> Processor -> Enable Nested VT-x/AMD-V 必須是打勾，而且不能是灰色不能勾選的狀態。



7. 使用 VirtualBox 或其他虛擬化工具，都要提供 VM crt9 足夠的 CPU 及 RAM，且要開啓巢式虛擬化，只要這三個要求達到，應該不會有□題，其他設定不懂可以先用預設值。若進不了圖形界面，可以打開 Display 的 3D 加速看看，VirtualBox VM 設定 (Settings) 參考值如下：



8. * 爲方便 VirtualBox 使用，建議下載附加套件 Oracle_VM_VirtualBox_Extension_Pack-6.1.30.vbox-extpack，放到你的 VirtualBox 目錄中，並於 VirtualBox 選單中加入附加 CD image。

```
Devices --> Insert Guest Additions CD image...
```

9. 開啓 DYW Linux 後，若虛擬機沒安裝，請先 rebuild 安裝虛擬機，練習後只需要 revert 還原，不用再要重新 rebuild，rebuild 要從學校 server 下載 image，要花較多時間。

1.4 練習系統製作-Raw Image

1. 下載 DYW Linux 練習系統 image 檔，或由 VDI 檔轉換格式，下載網址公告於本平台課程內，請登入查看。

2. 下載的 image 檔是 xz 壓縮檔，先用 unxz 解壓縮，windows 可用 winzip 解壓縮。

```
1 [root@dywmsi ~]# unxz crt9.img.xz
```

3. DYW Linux 練習系統，Host 主機以 virsh 管理 VM，最低硬體因所需 VM 個數不同：

   (a) CPU 4 核心以上，若修的課不用內部虛擬機，設定 2 核心也可以。

   (b) RAM 8G 以上，若修的課不用內部虛擬機，設定 4G RAM 也可以。

(c) RHCSA, RHCE 請設定 4 核心，8G RAM 以上

(d) CPU 支援虛擬化技術 (virtualization technology)，BIOS 啓用 VT。

4. 將 image 檔 dd 寫到 32G 以上硬碟或隨身碟，其中 /dev/sdX 是您要寫入的硬碟或隨身碟。

```
1 [root@dywmsi ~]# dd if=crt9.img of=/dev/sdX bs=100M
```

5. /dev/sdX 整顆硬碟的資料會被覆蓋，請確認要寫入的硬碟無誤再按 Enter 執行。

6. dd 完成後以 /dev/sdX 開機。

7. 開啓 DYW Linux 後，若虛擬機沒安裝，請先 rebuild 安裝虛擬機，練習後只需要 revert 還原，不用再要重新 rebuild，rebuild 要從學校 server 下載 image，要花較多時間。

1.5 NAT 設計

1. NAT 架設：電腦教室網路設置、管控學生上課上網。

2. 在 Linux，一行命令即可完成設定；在 Windows，Honda 老師亦幫忙撰寫批次檔。控制方式：

3. 任何設定皆不管控老師機，老師機可自由上網。

4. 學生機上網方式：

(a) 自由上網

(b) 不能上網

(c) 限制只能上學內網

(d) 限制只能上某一網站

(e) 限制只能上「依老師設定」的部分網站

(f) 只有某一學生機可自由上網

5. 不管老師做任何網路設定，下課後系統自動恢復成所有電腦皆可自由上網，以避免因老師忘了復原，而影響下節上課的老師。

1.6 虛擬機管理界面

1. 設計虛擬機管理 GUI 程式，方便學生管理虛擬機。程式包含虛擬機重安裝、還原、重新啓動及終端界面等四個功能。

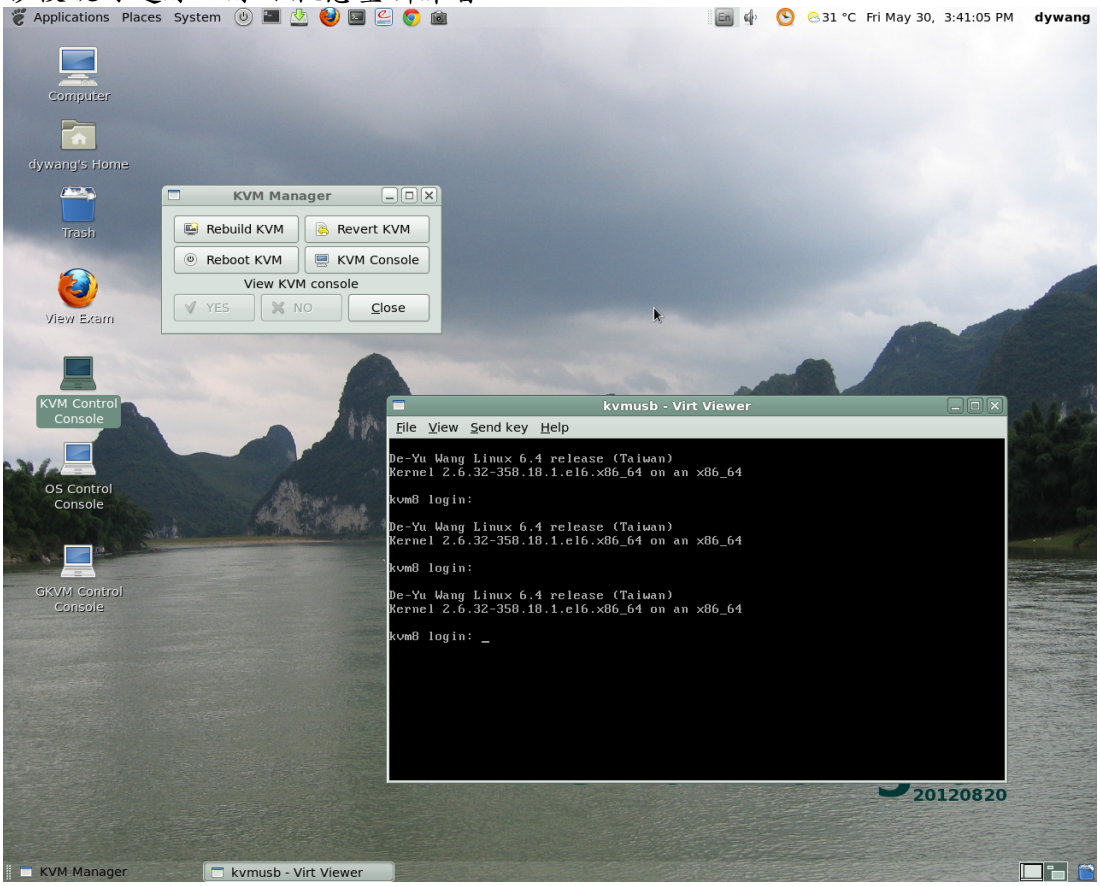
   (a) Rebuild KVM：重新安裝 KVM 虛擬機，約 10 幾分鐘。

   (b) Revert KVM：還原 KVM 虛擬機，約幾秒鐘到幾分鐘，請耐心等侯，不要在還原過程按管理界面的其他功能，以免造成虛擬機損壞。

   (c) Reboot KVM：重新啓動 KVM 虛擬機。

   (d) KVM Console：打開 KVM 虛擬機終端畫面。

2. 系統桌面中黑色背景視窗爲 KVM 虛擬機操作界面。原始 KVM 虛擬機主機名稱爲 111111，因已練習過，主機也已被設定成 kvm8，點擊 Revert KVM，幾秒後就可還原回原始狀態重新練習。



3. 在終端機以指令 virt-view 直接開啓虛擬機，出現錯誤訊息。

```
[dywang@dywH ~]$ sudo virt-viewer kvmusb
No protocol specified
Cannot open display:
Run 'virt-viewer --help' to see a full list of available command line
    options
```

4. 先檢查 Xwindow 認證變數 $XAUTHORITY 存不存在？

```
[dywang@dywH ~]$ echo $XAUTHORITY
```

5. 變數 XAUTHORITY 不存在，必須先設定 XAUTHORITY=~/.Xauthority 後再使用 sudo 啓動虛擬機，才能正常啓動。

```
[dywang@dywH ~]$ XAUTHORITY=~/.Xauthority sudo virt-viewer kvmusb
```

1.7 虛擬機管理界面 V2

1. 設計虛擬機管理 GUI 程式，方便學生管理虛擬機。程式包含虛擬機重安裝、還原、重新啓動、終端界面、啓用實體 USB 隨身碟、解除實體 USB 隨身碟的使用等六個功能。



(a) Rebuild KVM：重新安裝 KVM 虛擬機，直接下載 qcow2 映像檔，產生虛擬機，校內網路約 1 分多鐘。

(b) Revert KVM：還原 KVM 虛擬機，約幾秒鐘到幾分鐘，請耐心等侯，務必等黑色終端出現登入畫面，表示還原完成。練習前先關閉黑色終端畫面讓還原程式拍照，記下還原狀態，以免造成虛擬機損壞。

(c) Reboot KVM：重新啓動 KVM 虛擬機。

(d) KVM Console：打開 KVM 虛擬機終端畫面。

(e) Attach USB：插上 USB 隨身碟，按下此按鈕，KVM 虛擬機啓動實體 USB 隨身碟。

(f) Detach USB：KVM 虛擬機解除實體 USB 隨身碟的使用。

1.8 系統管理與評分

1. 系統管理腳本程式自動掃瞄目前開機的電腦，選擇要進行的工作，輸入密碼後即可自動完成。

```
[root@dywH ~]# e517sshpass.sh
myip=104
hosts=103,105
q(quit) iscsi halt reboot c(clear) l(lvm) rhcsa rhce
>  t(mounttest) k(kvmreboot) v(kvmrevert) s(servicerestart)
>  vi1 c1 lpvar c999 sh999 hp(hostpatch)? c999
keyin remote host root password:
```

2. 預設工作項目：

   (a) q(quit)：退出，無動作。

   (b) iscsi：學生機 ISCSI 硬碟分享架設。

   (c) halt：學生機關機。

   (d) reboot：學生機重新啓動。

   (e) c(clear)：登入學生機後清除部分資料，此部分保留修改彈性，目前入作爲限制學生機互相登入之限制，以防代作答情況。

   (f) l(lvm)：硬碟分割、格式化、自動掛載評分。

   (g) rhcsa：RHCSA 考照練習總評分。

   (h) rhce：RHCE 考照練習總評分。

   (i) t(mounttest)：測試學生機是否有掛載隨身碟或光碟。

   (j) k(kvmreboot)：學生機 KVM 虛擬機重新啓動。

   (k) v(kvmrevert)：學生機 KVM 虛擬機還原。

   (l) s(servicerestart)：啓動學生機主機服務。

   (m) vi1：vi 編輯器實機練習評分。

   (n) c1：C 語言第一支程式練習評分。

   (o) lpvar：shell 變數變化練習評分。

   (p) c999：C 語言程式實機練習評分。

   (q) sh999：Shell script 程式練習評分。

   (r) hp(hostpatch)：學生機網路修正後重新啓動網路。

3. 遠距教學，在家練習，自己評分上傳：

(a) 登入本平台，下載評分腳本及參數檔，以「資料庫」第一次評分爲例。

(b) 切換身分爲 root，root 密碼在平台部落格，請登入。

```
[dywang@deyu ~]$ su -
Password:
[root@deyu ~]#
```

(c) 切換工作目錄到評分程式的下載目錄，用瀏覽器下載預設是放在家目錄的 Downloads。

```
[root@deyu ~]# cd /home/dywang/Downloads
```

(d) 下載的評分程式沒有可執行的權限，先增加評分程式 mariadbeval1.sh.x 可執行屬性，使評分腳本 mariadbeval1.sh.x 成爲可執行檔。

```
[root@deyu Downloads]# chmod +x mariadbeval1.sh.x
```

(e) 查看目前有兩個檔案，評分程式爲 mariadbeval1.sh.x，參數檔爲 mariadbpara1.sh。

```
[root@deyu Downloads]# ll
-rwxrwxr-x. 1 root root 58656 May 21 13:29 mariadbeval1.sh.x
```

(f) 執行評分腳本 mariadbeval1.sh.x

```
[root@deyu Downloads]# ./mariadbeval1.sh.x
20210521-13:46:36
./mariadbeval1.sh.x
163.17.10.123
kvm=kvm19
163.17.10.123 10927000 王大呆
mariadbpara1.sh
mariadbeval1.sh.x_kvm19_1.mysqlversion:0
mariadbeval1.sh.x_kvm19_2.mariadb_enabled:0
mariadbeval1.sh.x_kvm19_3.mariadb_active:0
mariadbeval1.sh.x_kvm19_4.rootpw:0
```

(g) 如果評分參數檔不在同目錄，評分腳本 mariadbeval1.sh.x 無法執行。

```
[root@deyu Downloads]# mv mariadbpara1.sh /tmp/
```

```
[root@deyu Downloads]# ./mariadbeval1.sh
20210521-13:48:51
./mariadbeval1.sh
163.17.10.123
kvm=kvm19
163.17.10.123 10927000 王大呆
ls: cannot access mariadbpara1: No such file or directory
NO mariadbpara1 file
```

1.9 系統註冊-上課前先確定三件事

1. E517 電腦教室

   (a) E517 教室位置，靠窗爲第一排，確認自己電腦位置爲?-?。例如：1-1。

   (b) 在伺服器 (也就是開機進入圖形界面中有虛擬機管理界面的這台主機)，不是虛擬機，確認自己的電腦 IP。以下例子爲 192.168.1.140，IP 尾數爲 140。

```
[dywang@deyu ~]$ ifconfig | grep 192
          inet addr:192.168.1.140  Bcast:192.168.1.255  Mask
              :255.255.255.0
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask
              :255.255.255.0
```

   (c) 在伺服器 deyu.wang 主機一般帳號 dywang 家目錄，產生學號姓名檔 sid，檔案資訊格式爲「電腦位置 IP 尾數學號姓名」，每一欄位中間以一個空格隔開。以下例子電腦位置 1-11、IP 尾數 199、學號 123456，姓名王大呆。

```
[dywang@deyu ~]$ echo '1-11 199 123456 王大呆' > ~/sid
[dywang@deyu ~]$ cat ~/sid
1-11 199 123456 王大呆
```

2. 在家練習

   (a) 查詢練習系統對外 public ip 有三種方式：

```
[dywang@deyu ~]$ curl -4 ipinfo.io/ip
163.17.10.122[dywang@deyu ~]$
[dywang@deyu ~]$ curl -4 icanhazip.com
163.17.10.122
```

```
[dywang@deyu ~]$ curl -4  ifconfig.me
163.17.10.122[dywang@deyu ~]$
```

(b) 將 ip 學號姓名寫入 dywang 家目錄的 sid 檔，如下範例：

```
[dywang@deyu ~]$ echo '163.17.10.123 10827000 王大呆' > sid
```

(c) 查看 sid，ip 學號姓名務必正確，以免評分錯誤或無法匯入成績。

```
[dywang@deyu ~]$ cat sid
163.17.10.123 10827000 王大呆
```

1.10 虛擬機設定

1. 伺服器(也就是開機進入圖形界面中有虛擬機管理界面的這台主機)，不確定的話，先看命令列提示符號是不是 [dywang@deyu ~]$，或是 ifconfig 查目前所在主機的 IP：

   (a) Server 主機有兩個名稱 deyu.wang 及 server.deyu.wang

   (b) Server 與虛擬機網域 deyu.wang，使用網段 192.168.122.0/24。

   (c) Server IP 爲 192.168.122.1，虛擬機 IP 爲 192.168.122.X ，其中 X 依題目要求設定。

2. 虛擬機環境說明：

   (a) 虛擬機主機名稱 kvmX.deyu.wang IP 192.168.122.X，X 爲虛擬機編號，例如 5 或 7。

   (b) 虛擬機 root 密碼無法取得，統一重新設定爲 123qwe。

   (c) RHCE 練習環境增加
   網域 my111.wang，使用網段 192.168.111.0/24，
   server 主機增加此網域 IP 192.168.111.1。

   (d) 虛擬機網路設定：

```
IP Address:   192.168.122.X
Netmask:      255.255.255.0
Gateway:      192.168.122.1
Name server:  192.168.122.1
```

(e) RHCSA/RHCE 6,7 repository 套件庫網址：
http://dywang.csie.cyut.edu.tw/centos7.

(f) RHCSA 8 repository 套件庫網址：

i. BaseOS repo server:
http://dywang.csie.cyut.edu.tw/centos8/BaseOS.

ii. AppStream repo server:
http://dywang.csie.cyut.edu.tw/centos8/AppStream.

(g) kernel packages: 網路下載核心 RPM 檔
http://dywang.csie.cyut.edu.tw/kernel/centos7.

(h) LDAP server 設定：

i. 系統 deyu.wang

ii. base DN 爲 dc=deyu,dc=wang

iii. 證書 ftp://deyu.wang/pub/cacert.pem.

1.11 系統伺服器管理

1. 使用虛擬機練習時，伺服器 (也就是開機進入圖形界面中有虛擬機管理界面的這台主機) 必須正常運作，否則有些驗證無法進行。

2. 練習用到的 server 都已寫成腳本，如果運作有問題，可以執行腳本重啓。

1.11.1 DNS server

(a) 如果確定你的練習虛擬機 DNS 設定沒問題，但 ssh 卻無法使用主機名稱連線到虛擬機，可以在 server 機使用 host 命令看是否可以查到虛擬機的 IP？

(b) 沒有 DNS server 可查。

```
[dywang@deyu ~]$ host kvm7.deyu.wang
;; connection timed out; trying next origin
;; connection timed out; no servers could be reached
```

(c) 在 DNS server 查不到 kvm7.deyu.wang 的 IP。

```
[dywang@deyu ~]$ host kvm7.deyu.wang
Host kvm7.deyu.wang not found: 3(NXDOMAIN)
```

(d) 在伺服器 deyu.wang 主機切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(e) 執行 setdnsmasq.sh 腳本重啓 dnsmasq DNS 服務。

```
[root@deyu ~]# setdnsmasq.sh restart
```

(f) 再使用 host 查詢 kvm7.deyu.wang，已可查到其 IP 爲 192.168.122.7。

```
[root@deyu ~]# host kvm7.deyu.wang
kvm7.deyu.wang has address 192.168.122.7
kvm7.deyu.wang mail is handled by 1 20.
```

(g) 如果執行 setdnsmasq.sh 腳本重啓 dnsmasq DNS 服務未能解決問題，還可以使用 setdns.sh 腳本啓動 named DNS 服務。

```
[root@deyu ~]# setdns.sh restart
```

1.11.2 LDAP server

(a) 如果確認你的虛擬機 LDAP 設定沒問題，但還是無法以 ldapuser1 登入。在伺服器 deyu.wang 主機 (不是你的練習虛擬機)，先切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(b) 在伺服器執行 setldap1.sh restart 重新產生 LDAP 憑證。因爲憑證是新產生的，所以虛擬機必須再重新執行認證設定更新。

```
[root@deyu ~]# setldap1.sh restart
```

1.11.3 NFS server

(a) 虛擬機設定自動掛載必須查得到 server 有分享目錄，server 的 NFS 不像 LDAP 有憑證過期問題，所以一般都不需要重啓，但如果要重啓 NFS server，先在伺服器 deyu.wang 主機切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(b) 執行 setnfs.sh 腳本重新啓動。

```
[root@deyu ~]# setnfs.sh restart
```

Chapter 2

GIT 版本控制

2.1 Git 簡介

1. Git 是 Linux 之父 linus Trovalds 開發的版本控制系統，用來管理 Linux kernel 的 source code。
2. 採用快照 (snapshot) 方式記錄版本之間的更動、新增、刪除。
3. 快照方式，只會紀錄版本與版本之間的相異之處，可提升存取速度，也減少儲存的空間。
4. github, gitlab 等網站是儲存空間的 remote repository。
5. Git 順向流程：從工作區存到遠端儲存庫。
   (a) 工作區 (workspace)：git add 後檔案送到暫存區。
   (b) 暫存區 (staging area)：檔案處於 staged 狀態，git commit 後，檔案存入本地儲存庫。
   (c) 本地儲存庫 (local repository)：git push 後存入遠端儲存庫。
   (d) 遠端儲存庫 (remote repository)：github, gitlab 等網站。
6. Git 逆向流程：從遠端儲存庫取回最新資料到工作區
   (a) 遠端儲存庫 (remote repository)：github, gitlab 等網站，存有順向流程的檔案。
   (b) 工作區 (workspace)：git pull 後，工作區的檔案會被更新到與遠端儲存庫同步。
7. 新手也能懂的 Git 教學

2.2 Git 命令

1. host 已安裝 git 套件，提供 git 開頭的命令：

```
[pyuser@deyu ~]# git[TAB][TAB]
git                  git-shell            git-upload-pack
git-receive-pack     git-upload-archive
```

2. git 工具的命令：

```
[pyuser@deyu ~]# git [TAB][TAB]
add               describe          mergetool         revert
am                diff              mv                rm
apply             difftool          notes             scalar
archive           fetch             prune             send-email
bisect            format-patch      pull              shortlog
blame             fsck              push              show
branch            gc                range-diff        show-branch
bundle            gitk              rebase            sparse-checkout
checkout          grep              reflog            stage
cherry            gui               refs              stash
cherry-pick       help              remote            status
citool            init              repack            submodule
clean             instaweb          replace           switch
clone             log               request-pull      tag
commit            maintenance       reset             whatchanged
config            merge             restore           worktree
```

2.3 * 建立 Git Repository

1. 建立 git repository 資料儲存目錄

```
root@deyu:~# mkdir -p /srv/git
```

2. 進入 git repository 資料儲存目錄

```
root@deyu:~# cd /srv/git
```

3. 建立一個空的 git repository `python_pg.git`，一般副檔名設定爲.git。

```
root@deyu:/srv/git# git init --bare python_pg.git
hint: Using 'master' as the name for the initial branch. This default
    branch name
hint: is subject to change. To configure the initial branch name to use
    in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this
    command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /srv/git/python_pg.git/
```

4. git repository python_pg.git 目錄產生的資料。

```
root@deyu:/srv/git# ll python_pg.git/
total 16
drwxr-xr-x. 2 root root    6 Feb  3 07:38 branches
-rw-r--r--. 1 root root   66 Feb  3 07:38 config
-rw-r--r--. 1 root root   73 Feb  3 07:38 description
-rw-r--r--. 1 root root   23 Feb  3 07:38 HEAD
drwxr-xr-x. 2 root root 4096 Feb  3 07:38 hooks
drwxr-xr-x. 2 root root   21 Feb  3 07:38 info
drwxr-xr-x. 4 root root   30 Feb  3 07:38 objects
drwxr-xr-x. 4 root root   31 Feb  3 07:38 refs
```

5. Git 提供掛鉤 (hook) 事件觸發功能，在某些時間點執行客製化的工作，掛鉤檔都在 hooks 目錄下，範例檔都以.sample 爲副檔名，先使用範例 post-update.sample。

```
root@deyu:/srv/git# mv python_pg.git/hooks/post-update.sample python_pg.
    git/hooks/post-update
root@deyu:/srv/git# cat python_pg.git/hooks/post-update
#!/usr/bin/sh
#
# An example hook script to prepare a packed repository for use over
# dumb transports.
#
# To enable this hook, rename this file to "post-update".

exec git update-server-info
```

6. 變更掛鉤檔 post-updates 屬性爲可執行。

```
root@deyu:/srv/git# chmod +x python_pg.git/hooks/post-update
```

7. 爲允許用户經由 http push 到 git repository，必須在 config 配置檔中加入 http 章節，設定「receivepack = true」。

```
root@deyu:/srv/git# vi python_pg.git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = true
[http]
    receivepack = true
```

2.4 *Http Git Server

1. Git server 支援 SSH 及 HTTP 兩種協定，本系統採用 http 協定存取 git repository，先安裝 httpd 套件。

```
root@deyu:/srv/git# dnf install httpd
```

2. vi 編輯 /etc/httpd/conf.d/git.conf，限制用户存取 python_pg.git。

```
root@deyu:/srv/git# vi /etc/httpd/conf.d/git.conf
<VirtualHost *:80>
    ServerName deyu.wang

    # Set environment variables for git-http-backend
    SetEnv GIT_PROJECT_ROOT /srv/git
    SetEnv GIT_HTTP_EXPORT_ALL
    ScriptAliasMatch "(?x)^/(.*/(HEAD | \
        info/refs | \
        objects/(info/[^/]+ | \
        [0-9a-f]{2}/[0-9a-f]{38} | \
        pack/pack-[0-9a-f]{40}\.(pack|idx)) | \
        git-(upload|receive)-pack))$" \
        "/usr/libexec/git-core/git-http-backend/$1"

    # Directory for the git-http-backend script
    <Directory "/usr/libexec/git-core">
        Options +ExecCGI
        Require all granted
    </Directory>
```

```
21
    <Location "^/python_pg.git">
23      AuthType Basic
        AuthName "Git Access"
25      AuthUserFile "/etc/httpd/conf.d/git.pyuser.htpasswd"
        Require valid-user
27  </Location>
</VirtualHost>
```

3. 使用 htpasswd 建立可存取 python_pg.git 的用户檔。

```
root@deyu:/srv/git# htpasswd -c /etc/httpd/conf.d/git.pyuser.htpasswd
    pyuser
2 New password:
Re-type new password:
4 Adding password for user pyuser
```

4. 設定 httpd 開機啓動且現在啓動。

```
root@deyu:/srv/git# systemctl enable --now httpd
```

2.5 *http 存取 Git Repository

1. 本系統架設的 git repository python_pg.git 爲 http 協定存取，所以其用户及群組變更爲 apache。

```
1 root@deyu:/srv/git# chown -R apache:apache python_pg.git
root@deyu:/srv/git# ll python_pg.git/
3 total 16
drwxr-xr-x. 2 apache apache    6 Feb  3 07:38 branches
5 -rw-r--r--. 1 apache apache   66 Feb  3 07:38 config
-rw-r--r--. 1 apache apache   73 Feb  3 07:38 description
7 -rw-r--r--. 1 apache apache   23 Feb  3 07:38 HEAD
drwxr-xr-x. 2 apache apache 4096 Feb  3 07:38 hooks
9 drwxr-xr-x. 2 apache apache   21 Feb  3 07:38 info
drwxr-xr-x. 4 apache apache   30 Feb  3 07:38 objects
11 drwxr-xr-x. 4 apache apache   31 Feb  3 07:38 refs
```

2. 爲確保 SELinux enforcing 情況下，可以存取 git repository python_pg.git，所以其用户及群組變更爲 apache。

```
root@deyu:/srv/git# semanage fcontext -a -t httpd_sys_rw_content_t "/srv
    /git/python_pg.git(/.*)?"
root@deyu:/srv/git# restorecon -rv /srv/git/python_pg.git
Relabeled /srv/git/python_pg.git from unconfined_u:object_r:var_t:s0
to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /srv/git/python_pg.git/branches from unconfined_u:object_r:
    var_t:s0
to unconfined_u:object_r:httpd_sys_rw_content_t:s0
..........以下省略.............
```

3. 查看 git repository python_pg.git 的 SELinux fcontext type，已變更爲 httpd_sys_rw_content_t。

```
root@deyu:/srv/git# ls -Z python_pg.git/
unconfined_u:object_r:httpd_sys_rw_content_t:s0 branches
unconfined_u:object_r:httpd_sys_rw_content_t:s0 config
unconfined_u:object_r:httpd_sys_rw_content_t:s0 description
unconfined_u:object_r:httpd_sys_rw_content_t:s0 HEAD
unconfined_u:object_r:httpd_sys_rw_content_t:s0 hooks
unconfined_u:object_r:httpd_sys_rw_content_t:s0 info
unconfined_u:object_r:httpd_sys_rw_content_t:s0 objects
unconfined_u:object_r:httpd_sys_rw_content_t:s0 refs
```

4. 重新啓動 httpd 服務。

```
root@deyu:/srv/git# systemctl restart httpd
```

2.6 Git Clone

1. 用户 pyuser 在其家目錄，以帳號 pyuser 密碼 123qwe，從遠端伺服器 deyu.wang，clone python_pg 遠端儲存庫 python_pg，clone 的 repository 是空的。

```
pyuser@deyu:~$ git clone http://pyuser:123qwe@deyu.wang/python_pg.git
Cloning into 'python_pg'...
warning: You appear to have cloned an empty repository.
```

2. 沒有設定目錄，clone 建立的目錄名稱爲遠端儲存庫名稱 python_pg。

```
pyuser@deyu:~$ ll
total 4
drwxr-xr-x. 3 pyuser pyuser 18 Feb 14 17:43 python_pg
-rw-r--r--. 1 pyuser pyuser 33 Feb  9 11:25 sid
```

3. clone 遠端儲存庫 python_pg，並設定本地端的目錄爲學號。

```
pyuser@deyu:~$ git clone http://pyuser:123qwe@deyu.wang/python_pg.git
    10927000
Cloning into '10927000'...
warning: You appear to have cloned an empty repository.
```

4. 設定本地端目錄爲 10927000，clone 建立的目錄名稱爲 10927000。

```
pyuser@deyu:~$ ll
total 4
drwxr-xr-x. 3 pyuser pyuser 18 Feb 14 17:53 10927000
drwxr-xr-x. 3 pyuser pyuser 18 Feb 14 17:43 python_pg
-rw-r--r--. 1 pyuser pyuser 33 Feb  9 11:25 sid
```

2.7 Git Config

1. 進入 python_pg 目錄。

```
pyuser@deyu:~$ cd python_pg
```

2. 使用 python_pg repository 前最重要的是設定用户 email 及 name。

```
pyuser@deyu:~$ cd python_pg
pyuser@deyu:~/python_pg$ git config user.email 'pyuser@deyu.wang'
pyuser@deyu:~/python_pg$ git config user.name 'pyuser'
```

2.8 Git Push

1. 建立一個測試檔案 test.txt。

```
pyuser@deyu:~/python_pg$ echo 'git test' > test.txt
```

2. git add 將 test.txt 送到暫存區。

```
pyuser@deyu:~/python_pg$ git add test.txt
```

3. git commit 將 test.txt 存入本地 repository。

```
pyuser@deyu:~/python_pg$ git commit -m "test.txt"
[master (root-commit) f99a587] test.txt
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

4. git push 將本地 repository 存入遠端的 git repository。

```
pyuser@deyu:~/python_pg$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 211 bytes | 211.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To http://deyu.wang/python_pg.git
 * [new branch]      master -> master
```

2.9 實機練習

1. 練習系統的 GIT Server 架設在 http://deyu.wang/python_pg.git，用户 pyuser，密碼 123qwe。
2. 以 pyuser 登入 host 主機。
3. 在 pyuser 家目錄 git clone 遠端 deyu.wang 的 python_pg repository，並設定本地端的目錄爲 $mysid，變數 $mysid 爲學號不加 s。
4. git config 設定 email 爲 $mysid@deyu.wang 及 name 爲 $mysid。
5. 編輯一個測試檔案 test.txt，內容爲 $mysid git test。
6. git add 將 test.txt 送到本地暫存區。

7. git commit 將 test.txt 存入本地 git repository。
8. git push 本地 git repository 同步到遠端 git repository。

Chapter 3

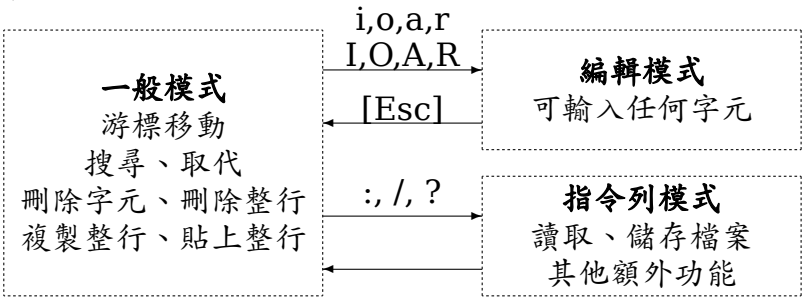
vi 編輯器

3.1 vi 與 vim

1. Linux 的系統中使用文字編輯器來編輯您的 Linux 參數設定檔。
2. Linux 與 Unix 系統中的參數檔幾乎都是 ASCII 碼的『純文字』檔。
3. vi 爲文書編輯器，vim 是 vi 的進階軟體。
4. vim 加入了支援正規表示法的搜尋架構、多檔編輯、區塊複製等功能。
5. vim 官方網站亦稱 vim 爲程式開發工具。
6. 大家來學 VIM（一個歷久彌新的編輯器）

3.2 vi 的使用

1. vi 共分爲三種模式，分別是『一般模式』、『編輯模式』與『指令列命令模式』三種。



2. 簡易範例

   (a) 使用 vi 進入一般模式

```
1 [dywang@deyu zzz]$ vim test.txt
```

(b) 按下 i 進入編輯模式，開始編輯文字。

(c) 按下 [Esc] 按鈕回到一般模式。

(d) 在一般模式中按下:wq 儲存後離開 vi。

3. 常用指令列表

一般模式	移動游標
h 或向左方向鍵	游標向左移動一個字元
j 或向下方向鍵	游標向下移動一個字元
k 或向上方向鍵	游標向上移動一個字元
l 或向右方向鍵	游標向右移動一個字元
[Ctrl]+f	螢幕『向前』移動一頁
[Ctrl]+b	螢幕『向後』移動一頁
[Ctrl]+d	螢幕『向前』移動半頁
[Ctrl]+u	螢幕『向後』移動半頁
+	游標移動到非空白字元的下一行
-	游標移動到非空白字元的上一行
n<space>	按下數字後再按空白鍵，游標會向右移動這一行的 n 個字元。 例如 20<space> 則游標會向後面移動 20 個字元距離。
0	這是數字『0 』: 移動到這一行的最前面字元處。
$	移動到這一行的最後面字元處
H	游標移動到這個螢幕的最上方那一行
M	游標移動到這個螢幕的中央那一行
L	游標移動到這個螢幕的最下方那一行
G	游標移動到這個檔案的最後一行
nG	游標移動到這個檔案的第 n 行。 例如 20G 則會移動到檔案的第 20 行 (可配合:set nu)
n<Enter>	游標向下移動 n 行

一般模式	搜尋與取代
/word	向游標之後尋找一個字串名稱爲 word 的字串。
?word	向游標之前尋找一個字串名稱爲 word 的字串。
n	重複前一個搜尋。
N	反向進行前一個搜尋動作。

:n1,n2s/word1/word2/g	在第 n1 與 n2 行之間尋找 word1 這個字串，並將該字串取代爲 word2 。加 g 參數取代該行所有匹配字串，否則只取代第一個字串。
:1,$s/word1/word2/g	從第一行到最後一行尋找 word1 字串，並將該字串取代爲 word2 。
:1,$s/word1/word2/gc	從第一行到最後一行尋找 word1 字串，並將該字串取代爲 word2 且在取代前顯示提示字元給使用者確認是否需要取代。

一般模式	**刪除、複製與貼上**
x, X	x 爲向後刪除一個字元，X 爲向前刪除一個字元
nx	向後刪除 n 個字元
dd	刪除游標所在的那一整行
ndd	刪除游標所在的向下 n 行，例如 20dd 則是刪除 20 行
d1G	刪除游標所在到第一行的所有資料
dG	刪除游標所在到最後一行的所有資料
dw	刪除一個字 (delete word)。不適用於中文。
yy	複製游標所在的那一行（常用）
nyy	複製游標所在的向下 n 行，例如 20yy 則是複製 20 行
y1G	複製游標所在行到第一行的所有資料
yG	複製游標所在行到最後一行的所有資料
p, P	p 爲複製的資料在游標下一行貼上，P 則爲貼在游標上一行。
J	將游標所在行與下一行的資料結合成同一行
u	復原前一個動作。
[Ctrl]+r	反復原前一個動作。
.	重做上一個動作。

進入編輯模式	
i, I	插入：i(I) 在目前的游標所在處 (所在行第一個非空白字元處) 插入輸入文字；
a, A	增加：a(A) 由目前游標所在的下一個 (最後一個) 字元處開始輸入：
o, O	插入新的一行：o(O) 從游標所在的下 (上) 一行插入新的一行；
r, R	取代：r 會取代游標所在的那一個字元；R 會一直取代游標所在的文字，直到按下 ESC 爲止；
Esc	退出編輯模式，回到一般模式中。

指令列命令模式	
:w	將編輯的資料寫入硬碟檔案中
:w!	若檔案屬性爲『唯讀』時，強制寫入該檔案

:q	離開 vi
:q!	若曾修改過檔案，又不想儲存，使用! 爲強制離開不儲存檔案。
:wq	儲存後離開，若爲:wq! 則爲強制儲存後離開
:x	與 wq 相同
:e!	將檔案還原到最原始的狀態。
ZZ	若檔案沒有更動，則不儲存離開，若檔案已經經過更動，則儲存後離開。
:w [filename]	將編輯的資料儲存成另一個檔案（類似另存新檔）
:r [filename]	在編輯的資料中，讀入另一個檔案的資料。亦即將『filename』這個檔案內容加到游標所在行後面。
:set nu	顯示行號，設定之後，會在每一行的字首顯示該行的行號
:set nonu	與 set nu 相反，爲取消行號。
:n1,n2 w [filename]	將 n1 到 n2 的內容儲存成 filename 這個檔案。
:! command	暫時離開 vi 到指令列模式下執行 command 的顯示結果。例如 [:! ls /home]

4. vi 編輯中回復與暫存檔

   (a) 如果編輯檔案爲 /home/csie/vitest.txt，則同時會有一個暫存檔 /home/csie/.vitest.txt.swp 產生，其爲隱藏檔。

   (b) 結束 vi 編輯時暫存檔也會刪除。

   (c) 若不知因素導致 vi 程式中斷，則在下次編輯該檔案時會出現是否回復的訊息。

3.3 vim 的額外功能

1. 如果使用 vi，在畫面右下角有目前游標所在行列號碼，則 vi 已被 vim 取代了。

```
[dywang@deyu zzz]$ ll /bin/vi*
lrwxrwxrwx  1 root root 20 Oct 17 13:11 /bin/vi -> /etc/alternatives/vi*
lrwxrwxrwx  1 root root 21 Oct 17 13:11 /bin/vim -> /etc/alternatives/
    vim*
[dywang@deyu zzz]$ ll /etc/alternatives/vi*
lrwxrwxrwx  1 root root 21 Oct 17 13:11 /etc/alternatives/vi ->
/usr/bin/vim-enhanced*
lrwxrwxrwx  1 root root 21 Oct 17 13:11 /etc/alternatives/vim ->
/usr/bin/vim-enhanced*
```

2. vim 具有顏色顯示的功能，並且還支援許多的程式語法。

3. vim /etc/man.config 出現訊息說明：

```
"/etc/man.config" [readonly] 150L, 4900C                    1,1
   Top
```

(a) 編輯檔案名稱爲 /etc/man.config；

(b) 檔案爲唯讀檔；

(c) 檔案共有 150 行，4900 字元；

(d) 目前游標所在位置爲第一行, 第一列；

(e) 目前頁面在最前頁。

4. 區塊選擇

	區塊選擇的按鍵意義
v	字元選擇，會將游標經過的地方反白選擇。
V	行選擇，會將游標經過的反白選擇。
[Ctrl]+v	區塊選擇，可以用長方形的方式選擇資料。
y	將反白的地方複製起來。
d	將反白的地方刪除掉。

5. 多檔案編輯

(a) vi 內使用:r filename 可將檔案 filename 的內容在游標處插入。

(b) 可於 vim 後接多個檔案來同時開啓多個檔案，例如：vim filename1 filename2 filename3。其相關按鍵有：

	多檔案編輯的按鍵
:n	編輯下一個檔案。
:N	編輯上一個檔案。
:files	列出目前開啓的所有檔案。

6. 多視窗功能

(a) 在指令列模式輸入:sp filename。

(b) 如果省略 filename 則兩視窗爲同一檔案。

	多視窗下的按鍵功能
:sp	開啓同一檔案於新視窗。
:sp filename	開啓檔案 filename 於新視窗。
[Ctrl]+wj	游標移動到下方的視窗。按法爲：先按下 [Ctrl] 不放，再下 w 後放開所有的按鍵，然後再按下 j。
[Ctrl]+wk	游標移動到上方的視窗。按法爲同上。
[Ctrl]+wq	結束下方視窗，與 [Ctrl]+w 移動到下方視窗後，再按下:q 離開相同。

7. vim 環境設定

   (a) 個人動作記錄檔案：~/.viminfo。例如：編輯同一檔案時，游標會在上次退出時的位置。

   (b) 整體 vim 的設定值放在 /etc/vimrc。

   (c) 若要更改 vim 設定，建議自行建立 ~/.vimrc。

	vim 的環境設定參數
:set nu	設定行號。
:set nonu	取消定行號。
:set hlsearch	將搜尋的字串反白。
:set autoindent	自動縮排。
:set noautoindent	不自動縮排。
:set backup	自動儲存備份。備份檔名爲 filename~。
:set rule	顯示右下角的狀態說明。
:set showmode	顯示 –insert– 等字眼在左下角的狀態列。
:set backspace=(012)	2 利用 backspace 例退鍵除任意字元；0 或 1 僅可刪除剛剛輸入的字元。
:set all	顯示目前所有的環境參數設定值。
:syntax (off|on)	是否依據程式相關語法顯示不同顏色。

3.4 vi 實機練習題

3.4.1 練習一

1. 在自己的家目錄建立一個新的目錄 zzz。

2. 進入目錄 zzz。

3. 下載檔案 viex1.txt
4. 將 viex1.txt 重新命名爲 vi1.txt
5. 使用 vi 或 vim 編輯 vi1.txt，執行以下動作：
   (a) 先到第 n 行，n 爲學號除 37 取餘數 +10。
   (b) 複製 3 行。
   (c) 再到第 20 行，往下貼上。
   (d) 再到最後一行，往下產生一行空白行方式進入”插入”模式進行編輯。
   (e) 加入文字’ABCDEabcde’。
   (f) 換行再加入’1234567890’。
   (g) 搜尋字串”MANPATH”，並將其取代爲”manpath”。
   (h) 存檔後退出。

3.4.2 練習二

1. 在自己的家目錄建立一個新的目錄 zzz。
2. 進入目錄 zzz。
3. 下載檔案 viex2.txt
4. 將 viex2.txt 重新命名爲 vi2.txt
5. 使用 vi 或 vim 編輯 vi2.txt，執行以下動作：
   (a) 先到第 n 行，n 爲學號除 27 取餘數 +8。
   (b) 刪除 3 行。
   (c) 再到第 30 行，往上貼上。
   (d) 再到第一行，往上產生一行空白行方式進入”插入”模式進行編輯。
   (e) 加入文字’ABCDEabcde’。
   (f) 換行再加入’1234567890’。
   (g) 從第 30 行至最後一行搜尋字串「MANPATH」，並將其取代爲「manpath」。
   (h) 存檔後退出。

3.4.3 練習三

1. 在自己的家目錄建立一個新的目錄 zzz。
2. 進入目錄 zzz。
3. 下載檔案 viex3.txt
4. 將 viex3.txt 重新命名爲 vi3.txt
5. 使用 vi 或 vim 編輯 vi3.txt，執行以下動作：
   (a) 先到第 n 行，n 爲學號除 44 取餘數 +11。
   (b) 將下一行合併至這一行
   (c) 再到第 20 行，刪除這一行到第一行的資料。
   (d) 再到最後一行，往下貼上。
   (e) 再到第 13 行，在這行的最後加入文字'ABCDEabcde'。
   (f) 換行再加入'1234567890'。
   (g) 從第 30 行至第 55 行搜尋字串"man"，並將其取代爲"MAN"。
   (h) 存檔後退出。

Chapter 4

認識 Python

4.1 簡介

1. Python 特色
   (a) 與 shell scripts 相似，是直譯式程式。
   (b) 互動式程式，執行過程中可與程式互動。
   (c) 物件導向程式。
   (d) 跨平台，未經修改下可在 Linux，Windows，MacOS X 和其他平台上使用。
   (e) 易讀、易學、易維護，適合初學者。
   (f) 可設計各種應用程式，從文字處理、www 網頁瀏覽器至遊戲都可完成。
2. Python 歷史
   (a) Python 構想起於 1980 年代後期，1989 年 12 月 Guido van Rossum 於荷蘭國家數學及計算機科學研究所發展出來。
   (b) Python 由許多程式語言發展出來，包含 ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell 及其他腳本語言。
   (c) Python 由 GNU General Public License (GPL) 授權，原始碼可自由獲得。
   (d) 1994 年 1 月釋出 1.0 版。
3. python 發展前景 參考資料來源
   (a) Python 對於 Linux 管理員是否必要？
   LinuxCareer.com 在大約 20% 的職位中將 Python 能力作爲關鍵性技術加以考量。

(b) Linux 社群如何看待 Python
Linux Journal 在 2011 年 9 月，網路所進行的調查，24% 將 Python 列爲自己最喜歡的程式語言。

Language	Number of votes	%
Python	2,025	24%
C	1,661	19%
C++	1,488	17%
Java	1,118	13%
Perl	674	8%
Other	670	8%
C#	399	5%
Ruby	361	4%
Haskell	126	1%
Ocaml	47	1%
Totals	8,569	100%

(c) Python 目前已經由 Google 大範圍使用。

(d) Python 是一種强大而全面的程式語言，根據 TIOBE 的索引，截至 2011 年 8 月止，Python 排在十大程式語言榜中的第八位。

(e) Python 能够在跨平台的系統中使用，且具備良好的擴展性，相信 Python 的前景是成長。

4. IEEE Spectrum: The 2018 Top Programming Languages

Rank	Language	Spectrum Ranking
1	Python	100.0
2	C++	99.7
3	Java	97.5
4	C	96.7
5	C#	89.4
6	PHP	84.9
7	R	82.9
8	JavaScript	82.6
9	Go	76.4
10	Assembly	74.1

5. Go vs Python: Select the Best One For Your Business

Worldwide, Jan 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	25.95 %	+5.2 %
2	↓	Java	21.42 %	-1.3 %
3	↑	Javascript	8.26 %	-0.2 %
4	↑	C#	7.62 %	-0.4 %
5	↓↓	PHP	7.37 %	-1.3 %
6		C/C++	6.31 %	-0.3 %
7		R	4.04 %	-0.2 %
8		Objective-C	3.15 %	-0.8 %
9		Swift	2.56 %	-0.7 %
10		Matlab	2.04 %	-0.3 %

6. Most Popular Programming Languages 1965 - 2019

4.2 python 執行腳本

1. 查看系統中的 python 程式，輸入 python 接兩次 [TAB]，提示所有 python 開頭的命令，

```
pyuser@deyu:~/zzz$ python[TAB][TAB]
python
python3
python3.12
python-argcomplete-check-easy-install-script
```

2. 查詢命令 python 命令在 /usr/bin 資料夾。

```
pyuser@deyu:~/zzz$ which python
/usr/bin/python
```

3. 命令 python 及 python3 都是 python3.12 的連結檔，也就是三個執行檔是同一個檔。

```
pyuser@deyu:~/zzz$ ll /usr/bin/python*
lrwxrwxrwx. 1 root root     9 Aug 14  2025 /usr/bin/python -> ./python3
lrwxrwxrwx. 1 root root    10 Aug 14  2025 /usr/bin/python3 -> python3
    .12
-rwxr-xr-x. 1 root root 15800 Aug 14  2025 /usr/bin/python3.12
-rwxr-xr-x. 1 root root  2610 Oct 29  2024 /usr/bin/python-argcomplete-
    check-easy-install-script
```

4. 執行 python

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> b=4
>>>
>>> a+b
7
>>> quit()
```

5. 寫成 python 腳本 abc.py

```
pyuser@deyu:~/zzz$ vi abc.py
pyuser@deyu:~/zzz$ cat abc.py
a=4
b=3
print (a+b)
```

6. 腳本 abc.py 中沒有宣告要用什麼程式執行，所以必須用 python2 或 python3 命令執行 abc.py 腳本。

```
pyuser@deyu:~/zzz$ python abc.py
7
```

4.3 獨立執行 python 腳本

1. 要獨立執行 python 腳本，必須在腳本的第一行宣告使用 python 執行 abc.py，先使用 type 查詢 python 命令儲存位置爲 /usr/bin/python：

```
pyuser@deyu:~/zzz$ type python
python is hashed (/usr/bin/python)
```

2. 在腳本的第一行宣告使用 python 執行 abc.py，由井號及驚嘆號#! 組成的字元組是 shebang (hashbang)，其後面接的內容是直譯器指令。

3. 原先 abc.py 沒有指定直譯器指令，無法獨立執行，要讓其獨立執行必須先宣告 abc.py 是使用 /usr/bin/python 直譯器執行：

```
pyuser@deyu:~/zzz$ vi abc.py
pyuser@deyu:~/zzz$ cat abc.py
#!/usr/bin/python
a=4
b=3
print(a+b)
```

4. 執行 abc.py，回應命令找不到，原因爲 abc.py 不在 $PATH 變數中的路徑中。

```
pyuser@deyu:~/zzz$ abc.py
bash: abc.py: command not found...
```

5. 執行 abc.py，回應命令找不到，原因爲 abc.py 不在 $PATH 變數中的路徑中。

```
pyuser@deyu:~/zzz$ pwd
```

6. 查看目前的工作路徑

```
1 /home/pyuser/zzz
```

7. 以完整路徑執行 abc.py，回應拒絕存取。

```
1 pyuser@deyu:~/zzz$ /home/pyuser/zzz/abc.py
-bash: /home/pyuser/zzz/abc.py: Permission denied
```

8. 查看 abc.py，沒有可執行的屬性。

```
2 pyuser@deyu:~/zzz$ ll abc.py
-rw-r--r--. 1 pyuser pyuser 37 Mar 23 10:49 abc.py
```

9. abc.py 加上可執行屬性。

```
2 pyuser@deyu:~/zzz$ chmod +x abc.py
pyuser@deyu:~/zzz$ ll abc.py
-rwxr-xr-x. 1 pyuser pyuser 37 Mar 23 10:49 abc.py
```

10. 可獨立執行 abc.py 了。

```
1 pyuser@deyu:~/zzz$ /home/pyuser/zzz/abc.py
7
3 pyuser@deyu:~/zzz$ ./abc.py
7
```

4.4 內縮語法

1. 一般程式語言對於控制流程、函式、類別等區塊程式碼，開始與結束會以指定的碼來對應。例如：shell script 的 if 判斷式，結束程式碼爲 fi。但對於程式碼的內縮排列則不規定。

2. python 最特別的設計是以內縮來判斷控制流程、函式等區塊程式碼的開始與結束，因此每段區塊程式碼都沒有結束程式碼的設計。

3. 這種設計不但可減少程式碼，更可強迫程式撰寫者以階層式撰寫，讓程式更容易解讀。

4. 要對齊才算是同區塊程式碼，若用 4 個空白內縮，則同區塊程式都要用 4 個空白對齊，即使 tab 設定爲 4 個空白位置，也無法執行。

5. 錯誤的 python 程式碼

```
[pyuser@deyu zzz]$ vi error.py
[pyuser@deyu zzz]$ cat error.py
if True:
print("Yes")
else:
print("No")
pyuser@deyu:~/zzz$ python error.py
  File "/home/pyuser/zzz/error.py", line 2
    print("Yes")
    ^^^^^
IndentationError: expected an indented block after 'if' statement on
    line 1
```

6. 正確的 python 程式：

```
pyuser@deyu:~/zzz$ vi correct.py
pyuser@deyu:~/zzz$ cat correct.py
if True:
    print("Yes")
else:
    print("No")
```

7. 執行結果：

```
pyuser@deyu:~/zzz$ python correct.py
Yes
```

4.5 括號、引號、換行、註解

1. 與 shell script 一樣，可使用倒斜線 \ 沿續下一行。

```
pyuser@deyu:~/zzz$ vi correct.py
pyuser@deyu:~/zzz$ cat correct.py
if True:
    print("Yes,\
 new line.")
else:
    print("No")
```

2. 大中小括號 {} [] () 可以不在同一行。

```
days = ['Monday', 'Tuesday', 'Wednesday',
             'Thursday', 'Friday']
```

3. 註解：

```
# 空白行視爲註解
# 符號 # 後的文字爲註解
'這是註解'
"這是註解"
''這是註解''
""這是註解""
"""這是註解
這是也是註解
""" 與 """ 之間文字都爲註解
"""
# 成對的單引號雙引號之間爲字串，且可換行。因不執行故都爲註解。

days = ['Monday', 'Tuesday', 'Wednesday',
             'Thursday', 'Friday']
```

4. 與 shell script 一樣，可以使用分號; 將多個命令放在同一行。

```
a=4; b=3; print (a+b)
```

4.6 實機練習題

1. 在家目錄下建立 zzz 目錄。

2. 切換工作目錄到 zzz。

3. python 環境建立

   (a) 在此目錄下寫一 python 程式 my1.py。

   (b) 設定變數 a=2 b=7，輸出 a+b 的結果。

4. 內縮語法

   (a) 下載 mypy2.py。

   (b) mypy2.py 重新命名爲 my2.py。

   (c) 除錯 my2.py，讓 my2.py 可以獨立執行。

Chapter 5

變數

5.1 變數指定

1. 變數直接使用，不需宣告：

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "Devid"      # A string
```

2. 多個變數一起指定：

```
a=b=c=1
a,b,c=1,2,"Devid"
```

3. 變數指定後，變數型態就固定，重新指定新型態的值，等同重新宣告：

```
a=1
a="Devid"
```

4. 變數的刪除：

```
a=b=c=1
del a
```

5.2 字串處理

1. 字串變數以單引號' 或雙引號" 括起來。

```
name="Devid"            # A string
```

2. 字串變數爲字元陣列，測試程式如下 str.py：

```
pyuser@deyu:~/zzz$ vi str.py
pyuser@deyu:~/zzz$ cat str.py
#!/usr/bin/python

name="David"
print(name)             # 印出整個字串
print(name[0])          # 印出第 1 個字元
print(name[1:3])        # 印出第 2 至 3 個字元
print(name[1:])         # 印出第 2 至最後一個字元
print(name*3)           # 印出字串三次
print(name+" csie")     # 印出連續字串
```

3. str.py 執行結果：

```
pyuser@deyu:~/zzz$ python str.py
David
D
av
avid
DavidDavidDavid
David csie
```

5.3 名稱前單底線

1. Python 中的函式 (function) 與方法 (method) 簡單的區分是：function 是一段區塊程式碼，透過呼叫及傳入參數後將結果輸出或回傳；method 不同於 function 的是 method 定義在類別 (class) 中，透過執行該類別的物件 (object) 中的 method 得到輸出。

2. `_single_leading_underscore`: Python 沒有眞正的 private methods (只能同一個程式內使用的 methods)，所以使用一個底線 _ 開頭的 method 表示其他程式不能存取這個 method。例如：mypy.py 有兩個 classes `myPy`, `_myPy1`，在 mypy.py 的程式中直接呼叫都可以使用。

```
pyuser@deyu:~/zzz$ vi mypy.py
pyuser@deyu:~/zzz$ python mypy.py
my first python myPy
my first python myPy1
pyuser@deyu:~/zzz$ vi test.py
pyuser@deyu:~/zzz$ python test.py
my first python myPy
my first python myPy1
my first python myPy
Traceback (most recent call last):
  File "/home/pyuser/zzz/test.py", line 6, in <module>
    _myPy1()
    ^^^^^^
NameError: name '_myPy1' is not defined. Did you mean: 'myPy'?

[dywang@deyu zzz]$ vim mypy.py
[dywang@deyu zzz]$ cat mypy.py
#!/usr/bin/python3

class _myPy1:
    def __init__(self):
        print ("my first python myPy1")
class myPy:
    def __init__(self):
        print ("my first python myPy")

myPy()
_myPy1()
```

3. 直接執行 mypy.py，不論有無前底線的 method 都可以執行。

```
[dywang@deyu zzz]$ ./mypy.py
my first python myPy
my first python myPy1
```

4. 撰寫 test.py 從 mypy 匯入所有模組，執行 myPy() 及 _myPy()。

```
[dywang@dywIssd zzz]$ vim test.py
[dywang@dywIssd zzz]$ cat test.py
#!/usr/bin/python3

from mypy import *

myPy()
_myPy1()
```

5. 執行 test.py，myPy() 印出"my first python myPy"，但 _myPy() 則出現錯誤，因爲 _myPy() 是 mypy.py 中的 private method，只能在 mypy.py 中執行，無法在 test.py 執行。

```
[dywang@dywIssd zzz]$ ./test.py
my first python myPy
my first python myPy1
my first python myPy
Traceback (most recent call last):
  File "./test.py", line 7, in <module>
    _myPy1()
NameError: name '_myPy1' is not defined
```

5.4 名稱後單底線

1. single_trailing_underscore_: 使用單底線在名稱後面，用來避免與 Python 的 keyword 搞混。例如：class_

```
  Tkinter.Toplevel(master, class_='ClassName')
```

5.5 名稱前後雙底線

1. __double_leading_and_trailing_underscore__: 名稱前後都有雙底線爲特殊物件 (magic object)，例如：__init__, __name__，不要自己產生這樣的變數。

2. mypy.py 程式判斷變數 __name__ 是否爲 __main__，也就是執行主程式，不是由其他程式呼叫。

```
pyuser@deyu:~/zzz$ vi mypy1.py
pyuser@deyu:~/zzz$ cat mypy1.py
#!/usr/bin/python

class _myPy1:
    def __init__(self):
        print("my first python myPy1")
class myPy:
    def __init__(self):
        print("my first python myPy")

if __name__ == "__main__":
```

```
13      print(__name__)
        myPy()
15      _myPy1()
else:
17      print(__name__)
```

3. 直接執行 mypy.py，印出變數 __name__ 爲 __main__，myPy() 及 _myPy1() 也都成功執行。

```
1  pyuser@deyu:~/zzz$ python mypy1.py
   __main__
3  my first python myPy
   my first python myPy1
```

4. 撰寫 test1.py，使用 form mypy import * 從檔案 mypy.py 匯入所有 classes，印出 myPy 的變數 __name__，執行 myPy 及 _myPy1。

```
   pyuser@deyu:~/zzz$ vi test1.py
2  pyuser@deyu:~/zzz$ cat test1.py
   #!/usr/bin/python
4
   from mypy1 import *
6
   myPy.__name__
8  myPy()
   _myPy1()
```

5. 執行 test1.py，印出 myPy 的變數 __name__ 爲 mypy1，執行 myPy() 正常輸出，但執行 _myPy1() 則出現錯誤。

```
1  pyuser@deyu:~/zzz$ python test1.py
   mypy1
3  my first python myPy
   Traceback (most recent call last):
5    File "/home/pyuser/zzz/test1.py", line 7, in <module>
       _myPy1()
7      ^^^^^^
   NameError: name '_myPy1' is not defined. Did you mean: 'myPy'?
```

5.6 名稱前雙底線

1. __double_leading_underscore: 在一個 class 中 method 使用雙底線在名稱前面，且最後沒有底線。例如：class myPy 中的 method __method，則其全名爲 _myPy__method。

```
pyuser@deyu:~/zzz$ vi mypy2.py
pyuser@deyu:~/zzz$ cat mypy2.py
#!/usr/bin/python

class _myPy1:
    def __init__(self):
        print("my first python myPy1")
class myPy:
    def __init__(self):
        print("my first python myPy")
    def __method(self):
        print("method in class myPy")

if __name__ == "__main__":
    print(__name__)
    myPy()._myPy__method()
    _myPy1()
else:
    print(__name__)
```

2. 執行 mypy2.py，印出變數 __name__ 爲 __main__，執行 myPy()._myPy__method() 就是執行 class myPy 中的 method __method()。

```
pyuser@deyu:~/zzz$ python mypy2.py
__main__
my first python myPy
method in class myPy
my first python myPy1
```

3. 撰寫 test2.py，使用 form mypy2 import * 從檔案 mypy2.py 匯入所有 classes，印出 myPy 的變數 __name__，執行 myPy._myPy__method() 及 _myPy1。

```
pyuser@deyu:~/zzz$ vi test2.py
pyuser@deyu:~/zzz$ cat test2.py
#!/usr/bin/python

from mypy2 import *

myPy.__name__
myPy()._myPy__method()
```

```
9 _myPy1()
```

4. 執行 test2.py，印出 myPy 的變數 __name__ 爲 mypy2，執行 myPy()._myPy__method() 正常輸出，但執行 _myPy1() 則出現錯誤。

```
1 pyuser@deyu:~/zzz$ python test2.py
mypy2
3 my first python myPy
method in class myPy
5 Traceback (most recent call last):
  File "/home/pyuser/zzz/test2.py", line 7, in <module>
7   _myPy1()
    ^^^^^^
9 NameError: name '_myPy1' is not defined. Did you mean: 'myPy'?
```

5.7 實機練習題

5.7.1 字串變數

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在此目錄下寫一 python 程式 str1.py。
   (a) 設定變數 var1 = "abc123cde"，後面加中文註解「我的第一個變數」。
   (b) 印出以下內容：
      i. 整個字串
      ii. 第 4 個字元
      iii. 第 3 至 6 個字元
      iv. 第 4 到最後一個字元
      v. 字串 2 次加上字串"CYUT"

5.7.2 單底線及雙底線

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。

3. 在此目錄下寫一 python 程式 underscore1.py。

   (a) 設定 myc1 及 _myc2 兩個 classes，都是定義 __init__(self) 印出 class 名稱。

   (b) 判斷變數 __name__ 是否等於 __main__，是則印出變數 __name__，並執行 myc1 及 _myc2；否則印出變數 __name__

4. 在此目錄下寫一 python 程式 underscore1test.py。

   (a) 從 underscore1.py import 所有 classes。

   (b) 輸出 myc1.__name__，觀察其與直接執行 underscore1.py 時的 __name__ 有何不同？

   (c) 執行 myc1 及 _myc2，觀察其結果。

5.7.3 名稱前雙底線

1. 在家目錄下建立 zzz 目錄。

2. 切換工作目錄到 zzz。

3. 在此目錄下寫一 python 程式 underscore2.py。

   (a) 設定 myc1 class，定義 __init__(self) 印出 class 名稱，定義 __method1(self) 印出"method1 in class myc1"。

   (b) 判斷變數 __name__ 是否等於 __main__，是則印出變數 __name__，並執行 myc1 及 myc1 中的 method __method1；否則印出變數 __name__

4. 在此目錄下寫一 python 程式 underscore2test.py。

   (a) 從 underscore2.py import 所有 classes。

   (b) 執行 myc1.__name__，觀察其與直接執行 underscore1.py 時的 __name__ 有何不同？

   (c) 執行 myc1 中的 method __method1，觀察其結果。

Chapter 6

資料型態

6.1 type function

1. typy 函式列出變數的資料型態，python3 除法會自動轉換成浮點數。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=10
>>> print(x,type(x))
10 <class 'int'>
>>> y=x/2
>>> print(y,type(y))
5.0 <class 'float'>
>>> quit()
```

6.2 進位轉換

1. bin 函式將數字轉換爲 2 進位，0b 開頭的數字代表 2 進位整數。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=0b0111
>>> print(x)
7
>>> y=12
>>> print(bin(y))
0b1100
```

2. oct 函式將數字轉換爲 8 進位，0o 開頭的數字代表 8 進位整數。

```
>>> x=0o037
>>> print(x)
31
>>> y=31
>>> print(oct(y))
0o37
```

3. hex 函式將數字轉換爲 16 進位，0x 開頭的數字代表 16 進位整數。

```
>>> x=0x3F
>>> print(x)
63
>>> y=63
>>> print(hex(y))
0x3f
>>> quit()
```

6.3 強制轉換

1. int 函式強制轉換爲整數。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=5.0
>>> y=int(x)+2
>>> print(y,type(y))
7 <class 'int'>
```

2. float 函式強制轉換爲浮點數。

```
>>> x=5
>>> y=float(x)+2
>>> print(y,type(y))
7.0 <class 'float'>
>>> quit()
```

6.4 數值運算常用函式

1. abs 函式計算絕對值。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=-5.2
>>> y=2.3
>>> print(abs(x),abs(y))
5.2 2.3
```

2. pow(x,y) 函式計算 x 的 y 次方。

```
>>> x=2
>>> y=3.1
>>> print(pow(x,y))
8.574187700290345
```

3. round(x) 函式取 x 的「四捨五入」值，回傳整數。

```
>>> x=3.5
>>> y=2.4
>>> print(round(x),round(y))
4 2
>>> print(type(round(x)),type(round(y)))
<class 'int'> <class 'int'>
>>> quit()
```

6.5 布林 Boolean

1. Boolean 資料型態值只有 True 及 False 兩種。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=True
>> y=False
>>> print(type(x),type(y))
<class 'bool'> <class 'bool'>
```

2. Boolean 值強制轉換成數字，True 爲 1，False 爲 0。

```
>>> print(int(x),int(y))
1 0
>>> quit()
```

6.6 字串 String

1. String 字串資料型態，一般是使用單引號或雙引號括起來的資料。

```
pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x='Just a test'
>>> print(type(x))
<class 'str'>
```

2. 字串有單引號可用倒斜線 跳脫，或外括號改用雙引號。

```
>>> x='David\'s friend'
>>> print(x,type(x))
David's friend <class 'str'>
>>> x="David's friend"
>>> print(x,type(x))
David's friend <class 'str'>
```

3. 使用運算子 + 連接字串。

```
>>> str1="abc def"
>>> str2="edc rfv"
>>> str3=str1+str2
>>> print(str3)
abc defedc rfv
```

4. str 函式強制將數字轉換成字串。

```
>>> num1=111
>>> num2=222
>>> str3=str(num1)+str(num2)
>>> print(str3)
```

```
5 111222
```

5. 字串與整數相乘。

```
1 >>> c1='qwe'
>>> c2=c1*4
3 >>> print(c2)
qweqweqweqwe
5 >>> quit()
```

6.7 字元

1. chr 函式回傳字元的 ASCII 碼值。

```
1 pyuser@deyu:~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
3 Type "help", "copyright", "credits" or "license" for more information.
>>> c1=100
5 >>> c2=50
>>> print(chr(c1),chr(c2))
7 d 2
```

2. ord 函式回傳字元的 Unicode 碼值。

```
1 >>> c1='d'
>>> c2='2'
3 >>> print(ord(c1),ord(c2))
100 50
```

3. 使用 '\n' 字元換行輸出。

```
>>> str1='abc def gef'
2 >>> str2='ert dfg cvb'
>>> str3=str1+'\n'+str2
4 >>> print(str3)
abc def gef
6 ert dfg cvb
```

4. 字串前加 r，可以防止逸出字元 (例如換行 \n) 被轉譯。

```
>>> str1='abcde\nABCDE'
>>> str2=r'abcde\nABCDE'
>>> print(str1)
abcde
ABCDE
>>> print(str2)
abcde\nABCDE
>>> quit()
```

6.8 實機練習題

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下依講義說明的方式撰寫以下 python 程式：

6.8.1 type 函式

1. dtype-type1.py
   (a) 設定變數 x 等於 55，後面加中文註解「變數 x」。
   (b) 設定變數 y 等於 x 除以 11。
   (c) 同行輸出 y 及 y 的 type
2. dtype-type2.py
   (a) 設定變數 x 等於 55，後面加中文註解「變數 x」。
   (b) 設定變數 y 等於 x 除以 11.0。
   (c) 同行輸出 y 及 y 的 type

6.8.2 進位轉換

1. dtype-dec.py
   (a) 設定二進位變數 x 等於 1011，印出十進位結果。
   (b) 設定八進位變數 x 等於 7215，印出十進位結果。
   (c) 設定十六進位變數 x 等於 5FE2，印出十進位結果。

2. dtype-bin.py

   (a) 設定十進位變數 x 等於 3285

   (b) 印出 x 二進位結果。

   (c) 印出 x 八進位結果。

   (d) 印出 x 十六進位結果。

6.8.3 強制轉換

1. dtype-ft1.py

   (a) 設定變數 x 等於 16。

   (b) 設定變數 y 等於 x 強制轉成浮點數再加 351。

   (c) 同行輸出 y 及 y 的 type

2. dtype-ft2.py

   (a) 設定變數 x 等於 16.7。

   (b) 設定變數 y 等於 x 強制轉成整數再加 218。

   (c) 同行輸出 y 及 y 的 type

6.8.4 數學函式

1. dtype-math1.py

   (a) 設定變數 x 等於 -7.6。

   (b) 設定變數 y 等於 2.4。

   (c) 同行輸出 x 及 y 的絕對值。

2. dtype-math2.py

   (a) 設定變數 x 等於 -7.6。

   (b) 設定變數 y 等於 2.4。

   (c) 輸出 y 的 x 次方。

3. dtype-math3.py

   (a) 設定變數 x 等於 -7.6。

   (b) 設定變數 y 等於 2.4。

(c) 輸出 x 的「四捨五入」值及其 type。

(d) 輸出 y 的「四捨五入」值及其 type。

6.8.5 Boolean

1. dtype-bool1.py

   (a) 設定變數 x 等於 1。

   (b) 設定變數 y 等於 0。

   (c) 同行輸出 x 及 y 的布林值。

2. dtype-bool2.py

   (a) 設定變數 x 等於 True。

   (b) 設定變數 y 等於 False。

   (c) 同行輸出 x 及 y 的整數值。

6.8.6 字串

1. dtype-str1.py

   (a) 設定變數 x 等於字串”Linda’s friend”。

   (b) 同行輸出 x 及 x 的 type。

2. dtype-str2.py

   (a) 設定變數 x 等於字串”Linda’s friend”。

   (b) 設定變數 y 等於字串”CSIE CYUT”。

   (c) 輸出 x 及 y 連接的字串。

3. dtype-str3.py

   (a) 設定變數 x 等於整數 123。

   (b) 設定變數 y 等於整數 456。

   (c) 將整數 x 及 y 轉換成字串，並連接起來輸出。

4. dtype-str4.py

   (a) 設定變數 x 等於整數 123。

   (b) 設定變數 y 等於整數 456。

   (c) 將整數 x 及 y 轉換成字串，輸出 x 連續 3 次，連接 y 連續 2 次。

6.8.7 字元

1. dtype-c1.py
   (a) 設定變數 x 等於 21。
   (b) 設定變數 y 等於 85。
   (c) 同行輸出 x 及 y 的字元 ASCII 碼值。
2. dtype-c2.py
   (a) 設定變數 x 等於字元 G。
   (b) 設定變數 y 等於字元 9。
   (c) 同行輸出 x 及 y 字元的 Unicode 碼值。
3. dtype-c3.py
   (a) 設定變數 x 等於字串 ABC。
   (b) 設定變數 y 等於字串 123。
   (c) 一個 print 函式輸出字串 x + 換行 + y。
4. dtype-c4.py
   (a) 設定變數 x 等於字串 ABC 加 123，但 ABC 與 123 必須換行。
   (b) 設定變數 y 跟 x 一樣，但不解譯換行字元。
   (c) 輸出 x。
   (d) 輸出 y。

Chapter 7

運算子

7.1 算術運算子

1. 假設 a=15 b=10

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=15
>>> b=10
>>> a+b
25
>>> a-b
5
>>> a*b
150
>>> a/b
1.5
>>> a%b
5
>>> a**5      ## a的5次方
759375
>>> a//b      ## 除法只留整數
1
>>> 32.8/9
3.644444444444444
>>> 32//9
3
>>> -32//9    ## 結果是負的時，取地板值， -3.64... ==> -4
-4
>>> -32.8//9
-4.0
>>> quit()
```

2. 算術運算子優先序

```
1. **
2. * / % //
3. + -
```

7.2 比較運算子

1. 假設 a=15 b=10

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=15
>>> b=10
>>> a==b
False
>>> a!=b
True
>>> a<>b     ## python2 與 != 相同，python3 不適用。
  File "<stdin>", line 1
    a<>b
      ^
SyntaxError: invalid syntax
>>> a>b
True
>>> a>=b
True
>>> a<b
False
>>> a<=b
False
>>> quit()
```

7.3 指派運算子

1. 運算結果指派回原變數

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=10; a+=5; print(a)
```

```
15
>>> a=10; a-=5; print(a)
5
>>> a=10; a/=5; print(a)
2.0
>>> a=10; a%=5; print(a)
0
>>> a=10; a//=5; print(a)
2
>>> a=10; a**=5; print(a)
100000
```

2. 多重指定

```
>>> a=b=c=10
>>> print(a,b,c)
10 10 10
>>> a,b,c=10,20,30
>>> print(a,b,c)
10 20 30
```

3. 變數交換

```
>>> a,b=10,20
>>> print(a,b)
10 20
>>> a,b=b,a
>>> print(a,b)
20 10
```

4. 刪除變數

```
>>> del a
>>> print(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> quit()
```

7.4 邏輯運算子

1. 邏輯 (logical) 運算子，主要用在組合判斷狀況。

```
pyuser@deyu ~/zzz$ python
Type "help", "copyright", "credits" or "license" for more information.
>>> x=7
>>> x<10 and x>5
True
>>> x<5 or x>10
False
>>> not(x<5 or x>10)
True
>>> quit()
```

7.5 身份運算子

1. 身份 (identity) 運算子，身份運算子用於比較是否相同物件 (objects)，而不是比較物件是否相等。x 跟 y 兩個串列內容都一樣，但不是同一個串列，所以 x is y 回傳 False，x == y 回傳 True，z 跟 x 是同一個串列，所以 x is z 回傳 True。

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=["abc",123]
>>> y=["abc",123]
>>> x is y
False
>>> x == y
True
>>> z=x
>>> x is z
True
>>> quit()
```

7.6 成員運算子

1. 成員 (membership) 運算子，成員運算子用於測試序列是不是物件 (objects) 中的成員，串列 x 包含"abc" 及 123 兩元素，所以"abc" in x 及 123 in x 都回傳 True，但 12 in x 回傳 False，12 not in x 回傳 True。

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> x=["abc",123]
>>> "abc" in x
True
>>> 123 in x
True
>>> 12 in x
False
>>> 12 not in x
True
>>> quit()
```

7.7 Bitwise 運算子

1. 整數變數的二進位值的運算，先設定變數 a b，並列出其二進位值。

```
pyuser@deyu ~/zzz$ python
Python 3.12.9 (main, Aug 14 2025, 00:00:00) [GCC 14.2.1 20250110 (Red
    Hat 14.2.1-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=54
>>> b=41
>>> print(bin(a),bin(b))
0b110110 0b101001
```

2. Binary AND

```
>>> print(bin(a),bin(b))
0b110110 0b101001
>>> c=a&b
>>> print(c,bin(c))
32 0b100000
```

3. Binary OR

```
>>> print(bin(a),bin(b))
0b110110 0b101001
>>> c=a|b
>>> print(c,bin(c))
63 0b111111
```

4. Binary XOR

```
>>> print(bin(a),bin(b))
0b110110 0b101001
>>> c=a^b
>>> print(c,bin(c))
31 0b11111
```

5. Binary Ones Complement

```
>>> print(bin(a))
0b110110
>>> print(bin(a),bin(~a))
0b110110 -0b110111
```

6. Binary Left Shift

```
>>> print(bin(a))
0b110110
>>> print(bin(a),bin(a<<2))
0b110110 0b11011000
```

7. Binary Right Shift

```
>>> print(bin(a))
0b110110
>>> print(bin(a),bin(a>>2))
0b110110 0b1101
>>> quit()
```

7.8 實機練習題

1. 在家目錄下建立 zzz 目錄。

2. 切換工作目錄到 zzz。

3. 在 zzz 目錄下依講義說明的方式撰寫以下 python 程式：

7.8.1 算術運算子

1. op-arith1.py

(a) 設定變數 x 等於 21，y 等於 85。

(b) 同行輸出 x 與 y 的「加、減、乘、除、x/y 取餘數」。

2. op-arith2.py

(a) 設定變數 x 等於 21，y 等於 2.1。

(b) 不使用 pow 數學函式輸出 x 的 y 次方。

3. op-arith3.py

(a) 設定變數 x 等於 21.8，y 等於 7，z 等於 -52.9。

(b) 輸出 x 除 y 只留整數，z 除 y 只留整數。

4. op-arith4.py

(a) 設定變數 x 等於 121，y 等於 43，z 等於 2.1。

(b) 輸出 x 除 y 後的 z 次方。

5. 註：輸出各數字間以空白隔開，不加任何其他字元，例如：

```
12 -44 217 10 -2
```

7.8.2 比較運算子

1. op-cmp1.py

(a) 設定變數 x 等於 21，y 等於 85。

(b) 同行輸出 x 與 y 的比較「相等、不等!=。

2. op-cmp2.py

(a) 設定變數 x 等於 41，y 等於 -58。

(b) 同行輸出 x 與 y 的比較「大於、小於、大於等於、小於等於。

3. 註：輸出各數字間以空白隔開，不加任何其他字元，例如：

```
12 -44 217 10 -2
```

7.8.3 指派運算子

1. op-assign1.py

(a) 設定變數 x 等於 14，y 等於 5。

(b) 使用指派運算子將 x 與 y 的「加、減、乘、除」結果指定回 x。

(c) 輸出每次計算的 x。

2. op-assign2.py

(a) 設定變數 x 等於 8721，y 等於 485。

(b) 使用指派運算子「x/y 取餘數、x/y 留整數」結果指定回 x。

(c) 輸出每次計算的 x。

3. op-assign3.py

(a) 使用多重指定方式設定變數 x 等於 291，y 等於 12.1。

(b) 依序輸出 x,y。

(c) x,y 變數交換

(d) 再依序輸出 x,y。

4. 註：輸出各數字間以空白隔開，不加任何其他字元，例如：

```
12 -44 217 10 -2
```

7.8.4 邏輯運算子

1. op-logical1.py

(a) 設定變數 x 等於 5。

(b) 輸出 x > 3 而且 x< 10。

2. op-logical2.py

(a) 設定變數 x 等於 5。

(b) 輸出 x > 13 或者 x< 9。

3. op-logical3.py

(a) 設定變數 x 等於 5。

(b) 輸出 x > 13 或者 x< 9 的反面結果。

7.8.5 身份運算子

1. op-identity1.py

(a) 設定變數 x 是包含 1 至 10 中所有的奇數。

(b) 設定變數 y 也是包含 1 至 10 中所有的奇數，但不是由 x 指派給的。

(c) 輸出 x 與 y 是否爲相同物件。

2. op-identity2.py

(a) 設定變數 x 是包含 1 至 10 中所有的奇數。

(b) 使用指派運算子，將 x 指派給 y。

(c) 輸出 x 與 y 是否爲相同物件。

7.8.6 成員運算子

1. op-membership1.py

(a) 設定變數 x 是包含 1 至 10 中所有的奇數。

(b) 輸出 7 是否爲 x 的成員。

(c) 輸出 13 是否爲 x 的成員。

2. op-membership2.py

(a) 設定變數 x 是包含 1 至 10 中所有的偶數。

(b) 輸出 8 是否不是 x 的成員。

(c) 輸出 14 是否不是 x 的成員。

7.8.7 Bitwise 運算子

1. op-bitwise1.py

(a) 設定變數 x 等於 27，y 等於 69。

(b) 同行輸出 x 二進位值與 y 的二進位值。

2. op-bitwise2.py

(a) 設定變數 x 等於 27，y 等於 69。

(b) 同行以二進位型式輸出 x 與 y 的 binary 「AND, OR, XOR」值。

3. op-bitwise3.py

(a) 設定變數 x 等於 27。

(b) 同行輸出 x 二進位值與 x 補數的二進位值。

4. op-bitwise4.py

(a) 設定變數 x 等於 237。

(b) 同行輸出 x 二進位值與 x left shift 2 的二進位值。

5. op-bitwise5.py

(a) 設定變數 x 等於 237。

(b) 同行輸出 x 二進位值與 x right shift 3 的二進位值。

6. 註：輸出各數字間以空白隔開，不加任何其他字元，例如：

```
12 -44 217 10 -2
```

Chapter 8

Print 格式化輸出

8.1 Python3 內建 print 函式

1. 文字輸出函式 print 語法

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

   (a) object 爲要印出的內容

   (b) sep= 指定印出多項時的分隔，sep=' ' 不指定時預設爲「一個空白」。

   (c) end= 指定印出結束的符號，end='\n' 不指定時預設爲「換行」

   (d) file= 指定輸出的目標，file=sys.stdout 不指定時預設爲「標準輸出，也就是螢幕輸出」

   (e) flush= 指定要不要清除緩衝區，flush=False 不指定時預設爲「不清除緩衝器」

2. print 分隔符號與結束符號心範例

```
pyuser@deyu:~/zzz$ vi print1.py
pyuser@deyu:~/zzz$ cat print1.py
#!/usr/bin/python

print('default:')
print('abc', 'cde', 'efg')
print('sep=\'.\':')
print('abc', 'cde', 'efg' , sep='.')
print('end=\'===\\n\':')
print('abc', 'cde', 'efg' , end='===\n')
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./print1.py
default:
abc cde efg
sep='.':
abc.cde.efg
end='===\n':
abc cde efg===
```

8.2 字串模數運算子 (%)

1. string 中所有的 % 開頭的轉換符號，都會被 values 中的元素取代，轉換符號例如：%c %d %s 分別代表字元、整數、字串，轉換符號與 C 語言的 printf 函式定義幾乎一樣。

```
print('string' % values)
```

2. 範例：

```
pyuser@deyu:~/zzz$ vi print2.py
pyuser@deyu:~/zzz$ cat print2.py
#!/usr/bin/python

a=8347
b='A'
c='qweasd'
print("a=%d b=%c c=%s" % (a,b,c))
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./print2.py
a=8347 b=A c=qweasd
```

8.3 format() 方法

1. format() 是 string 類別內建的方法 (method)，允許變數替換和值的格式化。相較於模數 % 運算子，是一種更彈性的方式。

```
str.format(var1, var2,...)
```

2. 範例：字串中的大括號 {}，會插入 format() 傳入的變數。

```
pyuser@deyu:~/zzz$ vi print3.py
pyuser@deyu:~/zzz$ cat print3.py
#!/usr/bin/python

a=8347
b='A'
c='qweasd'
print("a={} b={} c={}".format(a,b,c))
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./print3.py
a=8347 b=A c=qweasd
```

4. 範例：也可以將變數作爲 format() 方法的關鍵字參數，並使用變數名稱作爲字符串中的佔位符 (placeholder)。

```
pyuser@deyu:~/zzz$ vi print4.py
pyuser@deyu:~/zzz$ cat print4.py
#!/usr/bin/python

print("a={a} b={b} c={c}".format(a=8347,b='A',c='qweasd'))
```

5. 執行結果：

```
pyuser@deyu:~/zzz$ ./print4.py
a=8347 b=A c=qweasd
```

8.4 f-字串

1. F-strings 在字符串開頭引號之前添加 f，是一種簡單、快速且易於閱讀的 Python 格式字符串的方式。

```
f'... {..} .... {..}..'
```

2. 範例：字串中的大括號 {}，會插入 format() 傳入的變數。

```
pyuser@deyu:~/zzz$ vi print5.py
pyuser@deyu:~/zzz$ cat print5.py
#!/usr/bin/python

a=8347
b='A'
c='qweasd'
print(f"a={a} b={b} c={c}")
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./print5.py
a=8347 b=A c=qweasd
```

8.5 格式轉換規則

1. 指定 C 格式化輸出符號，唯一要更改是使用冒號 : 而不是百分比符號 %。例如，字串格式化輸出 %s，改用 {:s}、整數輸出 %d 改用 {:d}，

```
```

2. 範例：也可以將變數作爲 format() 方法的關鍵字參數，並使用變數名稱作爲字符串中的佔位符 (placeholder)。

```
pyuser@deyu:~/zzz$ vi print6.py
pyuser@deyu:~/zzz$ cat print6.py
#!/usr/bin/python

a=8347
b=65
c='qweasd'
print("a={:f} b={:c} c={:s}".format(a,b,c))
```

3. 執行結果：變數 a 轉換成浮點數輸出，變數 b 整數轉換成 char 字元輸出。

```
pyuser@deyu:~/zzz$ ./print6.py
a=8347.000000 b=A c=qweasd
```

8.6 實機練習題

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. myprint1.py 使用 print 函式的 sep 參數，印出與註冊的 sid 檔一樣的資料，但分隔符號改成等號"="。
4. myprint2.py 使用 print 函式的 end 參數，印出與註冊的 sid 檔一樣的資料，但結束符號改成"#mysid#\n"。
5. myprint3.py 使用%開頭的轉換符號，設定 a,b,c 三個變數，a 是日期，格式 20211111，b 是學號，c 是評分程式名稱，例如 py3。a,b 以整數輸出，c 以字串輸出，a,b,c 三個變數以空白隔開。輸出範例如下：

```
[dywang@deyu 10827000]$ ./myprint3.py
20240501 10827000 pyeval5
```

6. myprint4.py 使用%開頭的轉換符號，設定 a,b,c 三個變數，a 是字元'3'，b 是整數 4，c 字串'py3'。以正確的轉換符號格式化輸出 a,b,c 三個變數，使用預設分隔符號，變數中間以等號"=" 隔開。
7. myprint5.py 定義今天日期 date，學號 sid，評分項目 eval 三個變數，使用 string 內建 format() 方法，印出「日期：{} 學號：{} 評分項目：{}」，三個大括號佔位符以 date, sid, eval 三個變數取代。輸出範例如下：

```
pyuser@deyu:~/zzz$ ./myprint5.py
日期：20240501 學號：10827000 評分項目：pyeval5
```

8. myprint6.py 印出與 myprint5.py 一樣的內容，但不用先定義 date, sid, eval 三個變數，直接用 date, sid, eval 做爲 format() 的關鍵字參數，並於字串中以變數名稱做爲佔位符。

9. myprint7.py 定義 date, sid, eval 三個變數，將這三個變數插入「f-字串」的佔位符，印出與 myprint5.py 相同的內容。

10. myprint8.py 定義四個變數

    (a) date ：今天日期，例如 20260427

    (b) sid：學號，例如 10927000

    (c) eval：評分項目，例如 pyeval6

    (d) ch：sid 除 90 取餘數加 32，例如 10927000%90+32=42 是一個整數

    使用轉換格式方式輸出：

    (a) 以浮點數印出 date

    (b) 以 10 個位數整數印出 sid

    (c) 以字串印出 eval 的後 w 個字元，其中 w 星期幾，星期一至星期日 w 爲 1-7

    (d) 以字元印出 ch

    (e) 以 date=20260427 sid=10927000 eval=pyeval6 爲例，輸出範例如下：

```
pyuser@deyu:~/zzz$ ./myprint8.py
date=20260427.000000 sid=  10927000 eval=6 ch=*
```

Chapter 9

Input 輸入

9.1 input 函式

1. Python3 使用 input 函式讓使用者輸入資料，語法如下，傳入的參數 prompt 爲輸入提示用，會顯示在螢幕上。

```
input(prompt)
```

2. input 輸入一個字串，輸入的內容會儲存在變數 string。

```
pyuser@deyu:~/zzz$ vi input1.py
pyuser@deyu:~/zzz$ cat input1.py
#!/usr/bin/python

string=input("keyin a string: ")
print(string)
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./input1.py
keyin a string: qweasd ads
qweasd ads
```

9.2 Type 轉換

1. input 函式預設將輸入的資料存成字串，若輸入的是整數，必須使用 int 函式轉換。

```
pyuser@deyu:~/zzz$ vi ./input2.py
pyuser@deyu:~/zzz$ cat ./input2.py
#!/usr/bin/python

num=int(input("Enter a number: "))
print(type(num))
add=num+10
print("add=%d" % add )
print("num+10=%d" % (num+10))
```

2. 執行結果：

```
pyuser@deyu:~/zzz$ ./input2.py
Enter a number: 12
<class 'int'>
add=22
num+10=22
```

9.3 實機練習題

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. myinput1.py 使用 input 函式，提示「輸入一字串 :」，使用 print 的 sep 參數，輸出「輸入的字串 ==sting」，其中 string 是輸入的字串，== 是分隔符號。執行範例如下：

```
pyuser@deyu:~/zzz$ ./myinput1.py
輸入一字串：ssss
輸入的字串==ssss
```

4. myinput2.py 使用 input 函式，提示「輸入一數字 :」，輸入的數字放在變數 n，使用 print 的格式化輸出「n+10=X AND n-10=Y」，其中變數 X,Y 爲 n+10 及 n-10 的結果。執行範例如下：

```
pyuser@deyu:~/zzz$ ./myinput2.py
輸入一數字：4
n+10=14 AND n-10=-6
```

5. myinput3.py 使用 input 函式，提示「輸入一個 9 至 14 的數字 :」，輸入的數字存在變數 w，提示「輸入一個 2 至 6 的數字 :」，輸入的數字存在變數 v。定義今天日期 date，學號 sid，評分項目 eval 三個變數，將這三個變數插入「f-字串」的佔位符，並使用格式轉換規則，以 w 個位數 v 個小數點位數的浮點數元印出 date，以 w+v 個位數整數印出 sid，以字串印出 eval 第 v 至 w 字元。以 date=20250113 sid=10827000 eval=pythoneval1 爲例，輸出範例如下：

```
pyuser@deyu:~/zzz$ ./myinput3.py
輸入一個9至14的數字：12
輸入一個2至6的數字：3
date=20250113.000 sid=       10927000 eval=thoneval1
```

6. myinput4.py 使用 input 函式，提示「輸入一個 2 至 7 的數字 :」，輸入的數字存在變數 w，提示「輸入一個 33 至 126 的數字 :」，輸入的數字存在變數 ch。定義今天日期 date，學號 sid，評分項目 eval 三個變數：使用這 date, sid, eval, ch 四個變數做爲字串 format() 的關鍵字參數，並使用格式轉換規則，以 15 個位元整數印出 date，以 13 個位元，小數點 1 個位元的浮點數印出 sid，以字串印出 eval 的後 w 個字元，以字元印出 ch。以 date=20250113 sid=10827000 eval=pythoneval1 爲例，輸出範例如下：

```
pyuser@deyu:~/zzz$ ./myinput4.py
輸入一個2至7的數字：4
輸入一個33至126的數字：44
date=       20250113 sid=   10927000.0 eval=val1 ch=,
```

7. 輸出到檔案：等檔案 INPUT/OUTPUT 進度上完再做。

   (a) myio1.py 使用 input 函式，提示「輸入一檔名 :」。

   (b) 使用 print 函式內建的功能，將註冊檔 sid 儲存在輸入指定的檔名，例如輸入 abc，則儲存在 /tmp/abc。

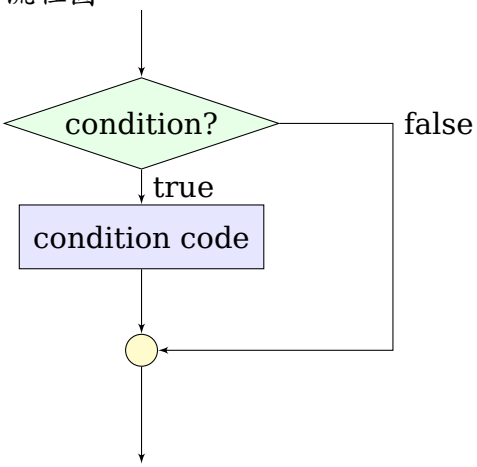
Chapter 10

判斷

10.1 if statement

1. Python 程式語言認定任何 non-zero, non-null 的值爲 true，zero 或 null 則爲 false。

2. 流程圖



3. 語法

```
if expression:
    statement(s)
```

4. 範例：輸入一個數字或字串，如果 TRUE 則印出結果。

```
pyuser@deyu:~/zzz$ cat if0.py
#!/usr/bin/env python

var = input("Enter a number or string: ")
if var:
    print("Got a true expression value", var)
```

5. 執行程式，輸入 123 或 asd 都是 TRUE，印出結果。

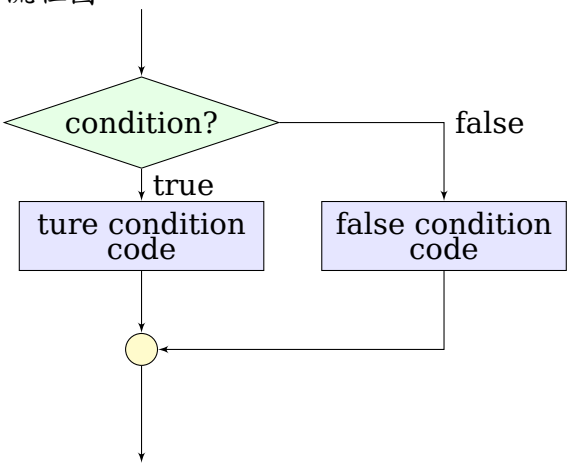
```
pyuser@deyu:~/zzz$ ./if0.py
Enter a number or string: 123
Got a true expression value 123
pyuser@deyu:~/zzz$ ./if0.py
Enter a number or string: asd
Got a true expression value asd
```

6. 執行程式，不輸入任何字元爲 FALSE，直接退出。

```
pyuser@deyu:~/zzz$ ./if0.py
Enter a number or string:
```

10.2 if...else statement

1. 流程圖



2. 語法

```
if expression:
    statement(s)
else:
    statement(s)
```

3. 範例：輸入一個數字或字串，如果 TRUE 則印出結果，否則印出"Good bye!"。

```
pyuser@deyu:~/zzz$ cat if1.py
#!/usr/bin/env python

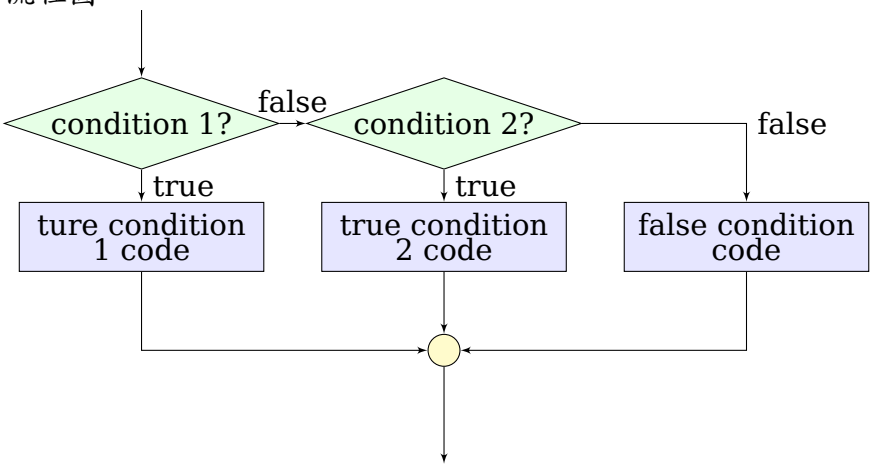
var = input("Enter a number or string: ")
if var:
    print("Got a true expression value", var)
else:
    print("Good bye!")
```

4. 執行程式，輸入 0 ，變數 var 是字串 0，不是數字 0，所以還是 TRUE，印出結果，若要視爲 FALSE，則必須將變數 var 轉換爲數字。不輸入任何字元則爲 FALSE，印出"Good bye!"。

```
pyuser@deyu:~/zzz$ ./if1.py
Enter a number or string: 0
Got a true expression value 0
pyuser@deyu:~/zzz$ ./if1.py
Enter a number or string:
Good bye!
```

10.3 if...elif...else Statement

1. 流程圖



2. 語法

```
if expression1:
    statement(s)
elif expression2:
    statement(s)
else:
```

```
6     statement(s)
```

3. 範例：輸入一個數字，判斷是否等於 100，或者小於 100，其他則是大於 100。

```
pyuser@deyu:~/zzz$ cat if2.py
#!/usr/bin/env python

var = input("Enter a number: ")
num = int(var)
if num == 100:
    print("Which is 100")
elif num < 100:
    print("%d is less than 100" % num)
else:
    print("%d is greater than 100" % num)
```

4. 執行程式，輸入 100 , 80, 200 等數字進行測試。

```
pyuser@deyu:~/zzz$ ./if2.py
Enter a number: 100
Which is 100
pyuser@deyu:~/zzz$ ./if2.py
Enter a number: 80
80 is less than 100
pyuser@deyu:~/zzz$ ./if2.py
Enter a number: 200
200 is greater than 100
```

10.4 nested if statements

1. 語法

```
if expression1:
    statement(s)
    if expression2:
        statement(s)
    elif expression3:
        statement(s)
    else:
        statement(s)
else:
    statement(s)
```

2. 範例：輸入一個數字，如果小於 100，再判斷是否等於 50？等於 10？小於 10？。

```
pyuser@deyu:~/zzz$ cat if3.py
#!/usr/bin/env python

var = input("Enter a number: ")
num = int(var)
if num == 100:
    print("Expression value is equal to 100")
elif num < 100:
    print("Expression value is less than 100")
    if num == 50:
        print("Which is 50")
    elif num < 50:
        print("Expression value is less than 50")
else:
    print("Expression value is larger than 100")
```

3. 執行程式，輸入 100 , 50, 10, 2, 150 等數字進行測試。

```
pyuser@deyu:~/zzz$ chmod +x if3.py
pyuser@deyu:~/zzz$ ./if3.py
Enter a number: 100
Expression value is equal to 100
pyuser@deyu:~/zzz$ ./if3.py
Enter a number: 50
Expression value is less than 100
Which is 50
pyuser@deyu:~/zzz$ ./if3.py
Enter a number: 10
Expression value is less than 100
Expression value is less than 50
pyuser@deyu:~/zzz$ ./if3.py
Enter a number: 2
Expression value is less than 100
Expression value is less than 50
pyuser@deyu:~/zzz$ ./if3.py
Enter a number: 150
Expression value is larger than 100
```

10.5 and or

1. if 有多項判斷時，可用 and 及 or 邏輯運算子組合。

2. 範例：and 關鍵字組合兩個判斷

```
pyuser@deyu:~/zzz$ cat if4.py
#!/usr/bin/env python

a=100
b=50
var = input("Enter a number: ")
num = int(var)
if num > b and num < a:
    print("50 < %d < 100" % num)
else:
    print("%d <= 50 or %d >= 100" % (num,num))
```

3. 執行程式，輸入 110,50,65 等數字進行測試。

```
pyuser@deyu:~/zzz$ ./if4.py
Enter a number: 110
110 <= 50 or 110 >= 100
pyuser@deyu:~/zzz$ ./if4.py
Enter a number: 50
50 <= 50 or 50 >= 100
pyuser@deyu:~/zzz$ ./if4.py
Enter a number: 65
50 < 65 < 100
```

4. 範例：or 關鍵字組合兩個判斷

```
pyuser@deyu:~/zzz$ cat if5.py
#!/usr/bin/env python

a=100
b=50
var = input("Enter a number: ")
num = int(var)
if num > a or num < b:
    print("%d > 100 or %d < 50" % (num,num))
else:
    print("50 <= %d <= 100" % num)
```

5. 執行程式，輸入 58,50,35,120 等數字進行測試。

```
pyuser@deyu:~/zzz$ ./if5.py
Enter a number: 58
50 <= 58 <= 100
pyuser@deyu:~/zzz$ ./if5.py
Enter a number: 50
50 <= 50 <= 100
```

```
pyuser@deyu:~/zzz$ ./if5.py
Enter a number: 35
35 > 100 or 35 < 50
pyuser@deyu:~/zzz$ ./if5.py
Enter a number: 120
120 > 100 or 120 < 50
```

10.6 實機練習題

10.6.1 if

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 if1.py。
   (a) 執行程式時，提示「請輸入一字串 :」。
   (b) if 判斷是否輸入字串，有則輸出「輸入的字串是：...」，其中... 是您輸入的字串。

10.6.2 if..else

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 if-else1.py。
   (a) 執行程式時，提示「請輸入一字串 :」。
   (b) if..else 判斷是否輸入字串，有則輸出「輸入的字串是：...」，其中... 是您輸入的字串，否則輸出「沒有輸出任何字串」。

10.6.3 if..elif..else

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 if-elif-else1.py。

(a) 執行程式時，提示「請輸入一字元：」。

(b) if..elif..else 比較輸入的字元及字元'D'，結果輸出格式如下，其中 var 爲您輸入的字元。

i. 大於 D，輸出「var > D」。

ii. 等於 D，輸出「var = D」。

iii. 其他，輸出「var < D」。

輸出範例如下：

```
pyuser@deyu:~/zzz$ ./if-elif-else1.py
請輸入一字元 D
D = D
pyuser@deyu:~/zzz$ ./if-elif-else1.py
請輸入一字元 E
E > D
pyuser@deyu:~/zzz$ ./if-elif-else1.py
請輸入一字元 A
A < D
```

10.6.4 nested if

1. 在家目錄下建立 zzz 目錄。

2. 切換工作目錄到 zzz。

3. 在 zzz 目錄下寫一 python 程式 nested-if1.py。

(a) 執行程式時，提示「請輸入一數字：」。

(b) nested if 判斷先判斷數字是否小於 1000，是則進一步判斷並輸出結果如下格式；否則輸出「var >= 1000」，其中 var 爲您輸入的數字。

i. 大於 500，輸出「var > 500」。

ii. 小於 500，輸出「var < 500」。

iii. 等於 500，輸出「var = 500」。

輸出範例如下：

```
pyuser@deyu:~/zzz$ ./nested-if1.py
請輸入一數字：100
100 < 500
pyuser@deyu:~/zzz$ ./nested-if1.py
請輸入一數字：600
600 > 500
pyuser@deyu:~/zzz$ ./nested-if1.py
```

```
請輸入一數字：500
500 = 500
pyuser@deyu:~/zzz$ ./nested-if1.py
請輸入一數字：2000
2000 >= 1000
```

10.6.5 and or

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫 python 程式。
   (a) and-or1.py。
      i. 執行程式時，提示「請輸入一數字：」。
      ii. if 判斷判斷數字是否小於 1000，而且大於 500，是則輸出「500 < var < 1000」，否則輸出「var >= 1000 或者 var <= 500」，其中 var 爲您輸入的數字。
   (b) and-or2.py。
      i. 執行程式時，提示「請輸入一數字：」。
      ii. if 判斷判斷數字是否小於 500，或者大於 1000，是則再用 nested if 判斷是是小於 500，則輸出「var < 500」，否則輸出「var > 1000」，不是小於 500 或大於 1000 則輸出「500 <= var <= 1000」，其中 var 爲您輸入的數字。

Chapter 11

串列

11.1 簡介

1. Python 串列 (list) 類似其他語言的陣列 (array)。
2. 不同於其他語言的陣列，串列除儲存相同資料型態的資料外，還可以儲存不同的資料型態，例如：串列同時含有整數、浮點數及字串，甚至串列中還可以有其他串列。
3. 串列定義：每一筆資料稱爲元素，元素放於中括號中，每一元素以逗號隔開，例如：

```
list1 = [ "David", 123, 23.4, 'abc', 4.321 ]
```

11.2 串列統計資料

1. 如果串列所有元素皆爲數字，可以用內建的 max, min, sum 函式，找到最大值、最小值及總和。

```
pyuser@deyu:~/zzz$ cat list2.py
#!/usr/bin/python

list2 = [ 32, 12, 73, 92, 43, 65, 17, 25 ]
print("max =", max(list2))        # 印出串列最大值
print("min =", min(list2))        # 印出串列最小值
print("sum =", sum(list2))        # 印出串列元素總和
```

2. 執行結果：最大值 92、最小值 12、總和 359。

```
pyuser@deyu:~/zzz$ ./list2.py
max = 92
min = 12
sum = 359
```

3. 如果串列元素全部是字元一樣可以使用 max, min 函式找到 ascii 最大及最小的字元元素，但 sum 則不適用。

```
pyuser@deyu:~/zzz$ cat list3.py
#!/usr/bin/python

list3 = [ 'a', 'b', 'y', 'H', 'Z', 'U' ]
print("max=", max(list3))        # 印出串列最大值
print("min=", min(list3))        # 印出串列最小值
print("sum=", sum(list3))        # 印出串列元素總和
```

4. 執行結果：ASCII 最大值字元爲 y、最小值 H、sum 總和函式不適用字元元素。

```
pyuser@deyu:~/zzz$ ./list3.py
max = y
min = H
sum =
Traceback (most recent call last):
  File "./list3.py", line 7, in <module>
    print("sum = ", sum(list3)        )# 印出串列元素總和
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

11.3 串列長度

1. 程式中有時需要知道串列的元素個數，例如 for 迴圈，此時可以使用 len 函式取得串列長度，當串列長度爲 0 表示空串列，所以可以用此來判斷是否爲空串列。

```
pyuser@deyu:~/zzz$ cat list4.py
#!/usr/bin/python

listx = [ 'a', 'b', 'y', 'H', 'Z', 'U' ]
listy = [ ]
print("length of list x =", len(listx))
print("length of list y =", len(listy))
```

2. 執行結果：串列 listx 總共有 6 個元素，串列 listy 長度爲 0，也就是 listy 爲空串列。

```
pyuser@deyu:~/zzz$ ./list4.py
length of list x = 6
length of list y = 0
```

11.4 更改元素內容

1. 利用索引值將第 4 個元素內容由'H' 改爲'T'。

```
pyuser@deyu:~/zzz$ cat list5.py
#!/usr/bin/python

list5 = [ 'a', 'b', 'y', 'H', 'Z', 'U' ]
list5[3] = 'T'
print("list5 =", list5)
```

2. 執行結果

```
pyuser@deyu:~/zzz$ ./list5.py
list5 = ['a', 'b', 'y', 'T', 'Z', 'U']
```

11.5 串列相加

1. 兩串列相加。

```
pyuser@deyu:~/zzz$ cat list6.py
#!/usr/bin/python

listx = [ 'a', 'b', 'c' ]
listy = [ 'A', 'B', 'C' ]
listx += listy
print("listx =", listx)
```

2. 執行結果

```
pyuser@deyu:~/zzz$ ./list6.py
listx = ['a', 'b', 'c', 'A', 'B', 'C']
```

11.6 串列乘一數字

1. 串列乘 2。

```
pyuser@deyu:~/zzz$ cat list7.py
#!/usr/bin/python

listx = [ 'a', 'b', 'c' ]
listx *= 2
print("listx =", listx)
```

2. 執行結果

```
pyuser@deyu:~/zzz$ ./list7.py
listx = ['a', 'b', 'c', 'a', 'b', 'c']
```

11.7 刪除串列或元素

1. 刪除第 i+1 個元素、每隔 step，第 start+1 到 end 個元素，step 不指定則預設爲 1。

```
del listx[i]
del listx[start:end]
del listx[start:end:step]
```

2. 範例：刪除串列 listx 第 4 個元素、再刪除第 6 到 6 個元素，再每隔 2，刪除第 3 到 8 個元素。

```
pyuser@deyu:~/zzz$ cat list8.py
#!/usr/bin/python

listx = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ]
print("listx =", listx)
del listx[3]
```

```
print("listx =", listx)
del listx[5:6]
print("listx =", listx)
del listx[2:8:2]
print("listx =", listx)
```

3. 執行結果

```
pyuser@deyu:~/zzz$ ./list8.py
listx = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
listx = [1, 2, 3, 5, 6, 7, 8, 9, 0]
listx = [1, 2, 3, 5, 6, 8, 9, 0]
listx = [1, 2, 5, 8, 0]
```

4. 範例：刪除整個串列 listx。

```
pyuser@deyu:~/zzz$ cat list9.py
#!/usr/bin/python

listx = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ]
print("listx =", listx)
del listx
print("listx =", listx)
```

5. 執行結果：刪除後 listx 不存在，不是變成空串列。

```
pyuser@deyu:~/zzz$ ./list9.py
listx = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
listx =
Traceback (most recent call last):
  File "./list9.py", line 7, in <module>
    print("listx = ", listx)
NameError: name 'listx' is not defined
```

11.8 實機練習題

1. 在家目錄下建立 zzz 目錄。

2. 切換工作目錄到 zzz，撰寫以下 python 程式。

3. 請 下載 list.txt。

4. list-sum1.py

   (a) 使用 list1 串列。

   (b) 執行程式時，提示「請輸入一數字：」，假設是 nu。

   (c) 輸出「串列最小值, 串列最大值, 串列總和, 串列總和除 nu 取餘數, 串列長度乘 nu」。輸出範例如下：

```
pyuser@deyu:~/zzz$ ./list-sum1.py
請輸入一數字：11
10,992,28212,8,539
```

5. list-change1.py

   (a) 使用 list2 串列。

   (b) 執行程式時，提示「請輸入一數字：」，假設是 nu。

   (c) 變更串列 list2 的第 nu 個元素爲 nu，變更後以串列格式輸出 list2 的第 nu-1 至 nu+1 個元素。輸出範例如下：

```
pyuser@deyu:~/zzz$ ./list-change1.py
請輸入一數字：23
[516, 23, 506]
```

6. list-add1.py

   (a) 使用 list7 list8 list9 串列。

   (b) 執行程式時，提示「請輸入一數字：」，輸入大於 1 的數字，假設是 nu。

   (c) 如果 nu 大於 3 輸出 list7 與 list8 相加後的串列。

   (d) 否則輸出 list9 乘 nu 後的串列。輸出範例如下：

```
pyuser@deyu:~/zzz$ ./list-add1.py
請輸入一數字：5
['1', '9', 'N', 'h', 'e', '5', 'V', 'l', '2', 'N', 'I', 'i', 'E']
pyuser@deyu:~/zzz$ ./list-add1.py
請輸入一數字：2
['b', 'd', 'u', '3']
```

7. list-del1.py

   (a) 使用 list3 串列。

(b) 執行程式時，提示「請輸入一數字：」，輸入 1 至 4 的數字，假設是 nu。

(c) 刪除第 nu 個元素。

(d) 再從第 10 到 40 個元素，每隔 nu 刪除元素。

(e) 以串列格式輸出 list3 的第 nu+2 至 nu+5 個元素。輸出範例如下：

```
pyuser@deyu:~/zzz$ ./list-del1.py
請輸入一數字：4
[351, 261, 968, 697]
pyuser@deyu:~/zzz$ ./list-del1.py
請輸入一數字：2
[997, 120, 351, 261]
```

Chapter 12

元組

12.1 簡介

1. 元組 (tuple) 資料型態與結構與串列 (list) 完全相同。
2. 元組與串列最大的不同在於，元組的元素值與個數不可更動。
3. 元組又稱「唯讀的串列」。

12.2 定義

1. 串列將元素放在中括號”[]” 中，元組則將元素放在小括號”()” 中，當元組中只有一個元素時，元素右邊必須加上逗號”,”，以免與函式混淆。

```
pyuser@deyu:~/zzz$ cat tuple1.py
#!/usr/bin/python

tuplex = ( 32, 12, 'abc' )
tupley = ( 18, )
print(tuplex[1],type(tuplex))
print(tupley,type(tupley))
```

2. 執行結果，元組的元素讀取方式與串列一樣。

```
pyuser@deyu:~/zzz$ ./tuple1.py
12 <class 'tuple'>
(18,) <class 'tuple'>
```

12.3 list 與 tuple 比較

1. list 與 tuple 變更元素內容比較。

```
pyuser@deyu:~/zzz$ cat tuple2.py
#!/usr/bin/python

listx = [ 32, 12, 'abc' ]
tuplex = ( 32, 12, 'abc' )
listx[1] = 10
tuplex[1] = 10
```

2. 執行結果：list 可以變更元素內容，tuple 變更元素內容會出現錯誤。

```
pyuser@deyu:~/zzz$ ./tuple2.py
Traceback (most recent call last):
  File "./tuple2.py", line 7, in <module>
    tuplex[1] = 10
TypeError: 'tuple' object does not support item assignment
```

12.4 修改元組

1. 元組無法修改元素內容，若要變更只能重新定義。

```
pyuser@deyu:~/zzz$ cat tuple3.py
#!/usr/bin/python

tuplex = ( 32, 12, 'abc' )
print("tuplex=", tuplex)
tuplex = ( 'cde', 65, 54, 21, 'abc' )
print("tuplex=", tuplex)
```

2. 執行結果：重新定義元組後，tuplex 元組正常輸出。

```
pyuser@deyu:~/zzz$ ./tuple3.py
tuplex= (32, 12, 'abc')
tuplex= ('cde', 65, 54, 21, 'abc')
```

12.5 元組的 max min sum

1. 串列可以修改，元組不能變更，所以應用在串列的 method 或 function，只要不會變更元組內容的都可以使用。

```
pyuser@deyu:~/zzz$ cat tuple4.py
#!/usr/bin/python

tuplex = ( 32, 12, -98 )
print("len(tuplex) =", len(tuplex))
print("max(tuplex) =", max(tuplex))
print("min(tuplex) =", min(tuplex))
print("sum(tuplex) =", sum(tuplex))
tupley = ( '9', 65, 54, 21, '7' )
print("len(tupley) =", len(tupley))
print("max(tupley) =", max(tupley))
print("min(tupley) =", min(tupley))
print("sum(tupley) =", sum(tupley))
```

2. 執行結果：len, max, min sum 對所有元素都是整數的元組可用，但 max min sum 在有字元及整數的元組不可用。

```
pyuser@deyu:~/zzz$ ./tuple4.py
len(tuplex) = 3
max(tuplex) = 32
min(tuplex) = -98
sum(tuplex) = -54
max(tupley) = 5
Traceback (most recent call last):
  File "./tuple4.py", line 11, in <module>
    print("max(tupley) =", max(tupley))
TypeError: '>' not supported between instances of 'int' and 'str'
```

12.6 串列元組互換

1. 需要將元組改成串列時，可用 list() 函式，反之使用 tuple() 函式。將元組 tuplex 轉成串列 listx，內容就可變動了。

```
pyuser@deyu:~/zzz$ cat tuple5.py
#!/usr/bin/python

tuplex = ( 32, 12, 'abc' )
listx = list(tuplex)
print("listx =", listx)
```

```
listx[1] = 99
print("listx =", listx)
```

2. 執行結果：元組轉成串列後，第 2 個元素由 12 變更成 99。

```
pyuser@deyu:~/zzz$ ./tuple5.py
listx = [32, 12, 'abc']
listx = [32, 99, 'abc']
```

12.7 實機練習題

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 請 下載 tuple.txt。
4. tuple-change1.py
   (a) 使用 tuple1 元組。
   (b) 變更 tuple1 的第 2 元素爲'abc'。
   (c) 執行並觀察其結果。
5. tuple-change2.py
   (a) 使用 tuple2 元組。
   (b) 變更 tuple2 增加一個元素'abc' 爲第一個元素，輸出變更後的元組。
6. tuple-method1.py
   (a) 使用 tuple8 元組。
   (b) 執行程式時，提示「請輸入一數字 :」，假設是 nu。
   (c) 輸出「元組最小值, 元組最大值, 元組總和, 元組總和除 nu 取餘數, 元組長度乘 nu」。
7. tuple-list1.py
   (a) 使用 tuple8 元組。
   (b) 執行程式時，提示「請輸入一數字 :」，輸入 1 至 4 的數字，假設是 nu。

(c) 將 tuple8 改成串列。

(d) 再從第 4 到 45 個元素，每隔 nu 刪除元素。

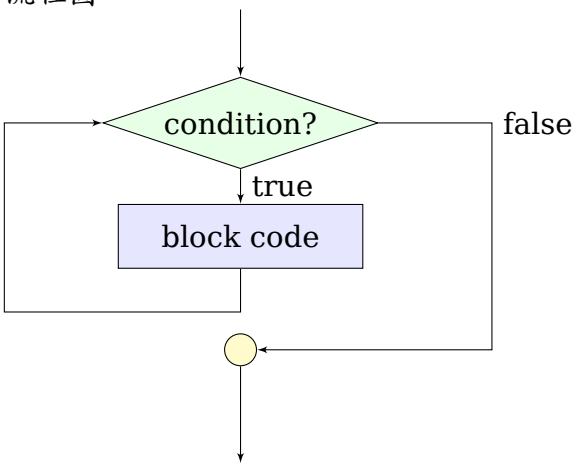
(e) 以串列格式輸出第 nu+2 至 nu+5 個元素。

Chapter 13

迴圈

13.1 while loop

1. 流程圖



2. 語法

```
while expression:
    statement(s)
```

3. 範例：while 迴圈直接輸入 q 才退出。

```
pyuser@deyu:~/zzz$ cat loop1.py
#!/usr/bin/env python

var = 'start'
while var != 'q':
    var = input("Enter a string: ")
    print("You entered:", var)
print("Good bye!")
```

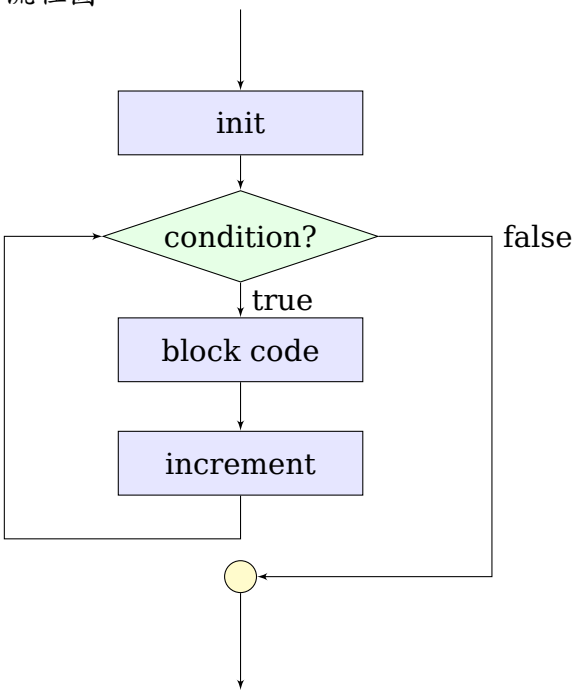
4. 執行程式，直到輸入 q 程式才退出。

```
pyuser@deyu:~/zzz$ ./loop1.py
Enter a string: 123
You entered:  123
Enter a string: qwe
You entered:  qwe
Enter a string: q
You entered:  q
Good bye!
```

5. 練習使用 while 從 1 累加到輸入的數字，印出結果。

13.2 for loop

1. 流程圖



2. 語法

```
for iterating_var in sequence:
    statement(s)
```

3. 範例：列出前二位及後兩位球員。

```
pyuser@deyu:~/zzz$ cat loop2.py
#!/usr/bin/env python
```

```
4 players = ['dywang', 'david', 'linda', 'peter', 'rita']
print("top 2 players")
6 for player in players[:2]: print(player)
print("last 2 players")
8 for player in players[-2:]: print(player)
```

4. 執行結果。

```
pyuser@deyu:~/zzz$ ./loop2.py
2 top 2 players
dywang
4 david
last 2 players
6 peter
rita
```

13.3 range function

1. range 函式可輸入如下三個參數，第一參數 start 爲起始值，第二個參數 end 爲終止值，第三個 step 爲間隔值。range(start, end, step) 從起始值 start 到終止值 end-1 以間隔 step 產生串列，例如：range(2,11,2) 產生串列 [2,4,6,8,10]；range(7,11) 產生串列 [7,8,9,10]。start, end 參數爲必要參數，step 不指定時預設爲 1。

```
1 range(start,end,step)
```

2. range 函式經常用於 for 迴圈，例如：計算上下限值中的所有偶數和。

```
1 pyuser@deyu:~/zzz$ cat loop3.py
#!/usr/bin/env python
3
lower = int(input("Enter a lower number: "))
5 upper = int(input("Enter an upper number: "))
print("Even sum between %d and %d is: " %(lower,upper))
7 lower += (lower%2)
total = 0
9 for num in range(lower,upper+1, 2):
    total += num
11 print('%d is the sum' % total)
```

3. 執行程式，上下限值分別輸入 13 及 17，所有偶數和爲 30。

```
pyuser@deyu:~/zzz$ ./loop3.py
Enter a lower number: 13
Enter an upper number: 17
Even sum between 13 and 17 is:
30 is the sum
```

13.4 nested loops

1. 巢狀 for 迴圈語法。

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statement1(s)
    statement2(s)
```

2. 範例：列印直角三角型。

```
pyuser@deyu:~/zzz$ cat loop4.py
#!/usr/bin/env python

num = int(input("Enter a number: "))
for i in range(1, num+1):
    for j in range(1, num+1):
        if j<=i: print("* ", end="")
    print()
```

3. 列印直角三角型程式執行結果。

```
pyuser@deyu:~/zzz$ ./loop4.py
Enter a number: 5
*
* *
* * *
* * * *
* * * * *
```

4. 巢狀 while 迴圈語法。

```
while expression1:
    while expression2:
        statement1(s)
    statement2(s)
```

5. 範例：列印直角三角型。

```
pyuser@deyu:~/zzz$ cat loop5.py
#!/usr/bin/env python

num = int(input("Enter a number: "))
i = j = 1
while(i<num+1):
    while(j<num+1):
        if j<=i: print("* ",end="")
        j += 1
    print()
    i += 1
    j = 1
```

6. 列印直角三角型程式執行結果。

```
pyuser@deyu:~/zzz$ ./loop5.py
Enter a number: 5
*
* *
* * *
* * * *
* * * * *
```

13.5 for - break

1. for 迴圈在某條件發生時使用 break 離開迴圈，語法如下：

```
for iterating_var in sequence:
    statement1(s)
    if expression:
        statement2(s)
        break
    statement3(s)
```

2. 範例：for 迴圈列印 1 到 10 數字，輸入一個中斷數字，碰到此數字時終止迴圈。

```
pyuser@deyu:~/zzz$ cat loop6.py
#!/usr/bin/env python

num = int(input("Enter a number: "))
for i in range(1, 11):
    print(i, end=" ")
    if i == num: break
print()
```

3. 執行結果，輸入 5 列印 1 2 3 4 5，輸入大於 10 的數字 100，列印 1 2 3 4 5 6 7 8 9 10。

```
pyuser@deyu:~/zzz$ ./loop6.py
Enter a number: 5
1 2 3 4 5
pyuser@deyu:~/zzz$ ./loop6.py
Enter a number: 100
1 2 3 4 5 6 7 8 9 10
```

13.6 for - continue

1. for 迴圈在某條件發生時，不執行後面的程式 (statement3)，可以使用 coutinue 繼續迴圈，語法如下：

```
for iterating_var in sequence:
    statement1(s)
    if expression:
        statement2(s)
        coutinue
    statement3(s)
```

2. 範例：輸入一分數，統計分數串列中大於此分數的個數。

```
pyuser@deyu:~/zzz$ cat loop7.py
#!/usr/bin/env python

num = int(input("Enter a number: "))
scores = [ 15, 93, 47, 62, 45, 88, 20, 79, 31, 65, 18, 27 ]
count = 0
```

```
for i in scores:
    if i < num: continue
    count += 1
print("total=%d" % count)
```

3. 執行結果，大於 60 分有五位，大於 80 分有二位。

```
pyuser@deyu:~/zzz$ ./loop7.py
Enter a number: 60
total=5
pyuser@deyu:~/zzz$ ./loop7.py
Enter a number: 80
total=2
```

13.7 for - else

1. 以下語法如果符合 if 條件會離開 for 迴圈，當最後一次迴圈 if 條件判斷還是 False，才會執行 else 的程式 (statement4)。

```
for iterating_var in sequence:
    statement1(s)
    if expression:
        statement2(s)
        break
    statement3(s)
else:
    statement4(s)
```

2. 範例：判斷輸入的數字是否爲質因數，當沒有任何數字可以整除輸入的數字時，印出這個數字是質因數。

```
pyuser@deyu:~/zzz$ cat loop8.py
#!/usr/bin/env python

num = int(input("Enter a number: "))
count = 0
for n in range(2, int(num/2+1)):
    if num % n == 0:
        print("%d is not prime" % num)
        break
else:
    print("%d is prime" % num)
```

3. 執行結果，2, 13 是質因數，39 不是質因數。

```
pyuser@deyu:~/zzz$ ./loop8.py
Enter a number: 2
2 is prime
pyuser@deyu:~/zzz$ ./loop8.py
Enter a number: 13
13 is prime
pyuser@deyu:~/zzz$ ./loop8.py
Enter a number: 39
39 is not prime
```

13.8 實機練習題

13.8.1 while

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 while1.py。
   (a) 執行程式時，提示「請輸入一數字：」。
   (b) while 迴圈從 1 累加到輸入的數字。
   (c) 輸出「1+2+...+nu=sum」，其中 nu 是輸入的數字，sum 是累加的結果。例如：

```
zzz$ ./while1.py
請輸入一數字：7
1+2+3+4+5+6+7=28
```

13.8.2 for

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 for1.py。
   (a) 執行程式時，提示「請輸入一字串：」。

(b) 使用 for 迴圈，輸出字串第二個字元到倒數第二個字元，且中間都加入等號"="。

13.8.3 range

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 range1.py。
   (a) 執行程式時，提示「請輸入一下限值 :」，輸入後再提示「請輸入一上限值 :」
   (b) 使用 for 迴圈，下限至上限中所有可以被 3 整除的數之總和。
   (c) 輸出「總和 =sum」，其中 sum 是算出的結果。
4. 在 zzz 目錄下寫一 python 程式 range2.py。
   (a) 執行程式時，提示「請輸入一下限值 :」，輸入後再提示「請輸入一上限值 :」
   (b) 計算從「下限值」累加到「上限值」的結果
   (c) 輸出：學號 <Tab> 姓名 <Tab> 結果
   (d) 輸出結果範例：10227000 王大明 49

13.8.4 nested for

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 nested-for1.py。
   (a) 執行程式時，提示「請輸入一數字 :」，假設爲 4。
   (b) 輸出：學號 <TAB> 姓名 <TAB>
   (c) 畫出以下圖形

```
---1---
--222--
-33333-
4444444
```

13.8.5 for break

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 for-break1.py。
   (a) 執行程式時，提示「請輸入一字串 :」。
   (b) 使用 for – break 依序印出字串中的每個字元，且在每個字元後面加等號'='，直到 h (包含) 字元。輸出範例如下：

```
[dywang@dgfu zzz]$ ./for-break1.py
輸入一字串duehfj
d=u=e=h=
```

13.8.6 for continue

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 for-continue1.py。
   (a) 執行程式時，提示「請輸入一字串 :」。
   (b) 使用 for – continue 刪除字串中的 h 字元，並在每個字元後面加等號"="。

13.8.7 for else

1. 在家目錄下建立 zzz 目錄。
2. 切換工作目錄到 zzz。
3. 在 zzz 目錄下寫一 python 程式 for-else1.py。
   (a) 執行程式時，提示「請輸入一字串 :」。
   (b) 使用 for – else 找到 h 字元，印出「第 nu 字元」，其中 nu 是找到的 h 字元位置，例如：字串 asdfghjkhrha 輸出「第 6 字元」，數字 6 前後有空白。
   (c) 如果找不到 h 字元，則輸出「找不到 h 字元」，字元 h 前後有空白。

Chapter 14

字典

14.1 簡介

1. 串列 (list) 與元組 (tuple) 是依序排列元素，只要知道元素的位置，就可用數值索引 (index) 取得元素內容。
2. 字典 (dictionary) 的索引不是使用 (0,1,2,3,...) 順序數字，每一元素必須自訂索引鍵 (key)。
3. 字典可說是鍵 (key)-值 (value) 配對的串列。

14.2 定義

1. 字典定義將所有配對 key:value 放在大括號”” 中，語法如下：

```
name_dict = { key1:vaule1, key2:vaule2, ....}
```

2. 範例：以字典建立用户 uid 及 name。

```
pyuser@deyu:~/zzz$ cat dict1.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
print("uid =", uid)
print("uid[%s] = %d" % ('david', uid['david']))
```

3. 執行結果，取得用户 david 的 uid 是 2346。

```
pyuser@deyu:~/zzz$ ./dict1.py
uid = {'dywang': 1000, 'david': 2346, 'linda': 1324}
uid[david] = 2346
```

14.3 增加元素

1. 字典 uid 增加用户 peter。

```
pyuser@deyu:~/zzz$ cat dict2.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
print("uid =", uid)
uid['peter'] = 1202
print("uid =", uid)
```

2. 執行結果：uid 字典從 3 個用户增加爲 4 個用户。

```
pyuser@deyu:~/zzz$ ./dict2.py
uid = {'dywang': 1000, 'david': 2346, 'linda': 1324}
uid = {'dywang': 1000, 'david': 2346, 'linda': 1324, 'peter': 1202}
```

14.4 變更元素內容

1. 將字典中用户 dywang 的 uid 由 1000 改爲 1002。

```
pyuser@deyu:~/zzz$ cat dict3.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
print("uid[%s] = %d" %('dywang', uid['dywang']))
uid['dywang'] = 1002
print("uid[%s] = %d" %('dywang', uid['dywang']))
```

2. 執行結果：用户 dywang 的 uid 由 1000 改爲 1002。

```
pyuser@deyu:~/zzz$ ./dict3.py
uid[dywang] = 1000
```

```
3 uid[dywang] = 1002
```

14.5 刪除元素

1. 刪除字典中某的特定元素用 del，但要刪除所有元素則要利用字典物件導向成員 clear。

```
1 pyuser@deyu:~/zzz$ cat dict4.py
  #!/usr/bin/python
3
  uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
5 print("uid =", uid)
  del uid['dywang']
7 print("uid =", uid)
  uid.clear()
9 print("uid =", uid)
```

2. 執行結果：del 刪除用户 dywang，clear() 清除所有元素，uid 變成一個空字典。

```
1 pyuser@deyu:~/zzz$ ./dict4.py
  uid = {'dywang': 1000, 'david': 2346, 'linda': 1324}
3 uid = {'david': 2346, 'linda': 1324}
  uid = {}
```

14.6 驗證元素是否存在

1. 字典的索引並不是序列數字，不能依序列出，要判斷字典中是否存在某鍵可用下列語法：

```
key in dict_name
```

2. 範例：輸入一鍵，判斷字典 uid 是否存在此鍵？不存在則輸入其值並新增。

```
1 pyuser@deyu:~/zzz$ cat dict5.py
  #!/usr/bin/python
3
  uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
```

```
key = input("Enter a key: ")
if key in uid:
    print("%s in uid, uid[%s] = %d" % (key, key, uid[key]))
else:
    value = int(input("Enter a value: "))
    uid[key] = value
    print("add %s to uid" % key)
    print("uid[%s] = %d" % (key, value))
```

3. 執行結果：輸入 dywang 時，程式判斷此鍵已存在於字典 uid；輸入 rita 則要求輸入 value，並新增此元素於字典中。

```
pyuser@deyu:~/zzz$ ./dict5.py
Enter a key: dywang
dywang in uid, uid[dywang] = 1000
pyuser@deyu:~/zzz$ ./dict5.py
Enter a key: perter
Enter a value: 1234
add perter to uid
uid[perter] = 1234
```

14.7 遍歷字典鍵值

1. 字典的索引並不是序列數字，不能依序列出，要遍歷 (traverse) 字典的鍵及值可用下列 for 迴圈取出所有鍵值配對：

```
for key,value in dict_name.items():
```

2. 範例：遍歷字典 uid，並列出其值及鍵。

```
pyuser@deyu:~/zzz$ cat dict6.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
for key,value in uid.items():
    print("uid = %d" % value)
    print("key = %s" % key)
```

3. 執行結果：for 迴圈逐一輸出字典所有元素。

```
pyuser@deyu:~/zzz$ ./dict6.py
uid = 1000
key = dywang
uid = 2346
key = david
uid = 1324
key = linda
```

4. 修改 dict6.py 爲 dict6a.py，增加 for 迴圈中判斷 key 是 david 時，刪除該元素。

```
pyuser@deyu:~/zzz$ cat ./dict6a.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
for key,value in uid.items():
    print("uid = %d" % value)
    print("key = %s" % key)
    if key == 'david':
        del uid[key]
```

5. 執行結果：出現錯誤，python 無法在 for 迴圈改變字典大小。

```
pyuser@deyu:~/zzz$ ./dict6a.py
uid = 1000
key = dywang
uid = 2346
key = david
Traceback (most recent call last):
  File "./dict6a.py", line 5, in <module>
    for key,value in uid.items():
RuntimeError: dictionary changed size during iteration
```

6. 爲解決上述問題，uid.items() 必須轉換成 list。

```
pyuser@deyu:~/zzz$ vi dict6b.py
pyuser@deyu:~/zzz$ cat dict6b.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
for key,value in list(uid.items()):
    print("uid = %d" % value)
    print("key = %s" % key)
    if key == 'david':
        del uid[key]
```

```
11 print(uid)
```

7. 執行結果：uid.items() 轉成 list 後，迴圈中成功刪除 devid。

```
1  pyuser@deyu:~/zzz$ ./dict6b.py
   uid = 1000
3  key = dywang
   uid = 2346
5  key = david
   uid = 1324
7  key = linda
   {'dywang': 1000, 'linda': 1324}
```

14.8 遍歷字典鍵

1. 使用 items() 可以遍歷 (traverse) 字典的鍵及值，若只想遍歷字典的鍵，可用如下 keys()：

```
for key in dict_name.keys():
```

2. 範例：遍歷並列出字典 uid 的鍵。

```
1  pyuser@deyu:~/zzz$ cat dict7.py
   #!/usr/bin/python
3
   uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
5  for key in uid.keys():
       print("key = %s" % key)
```

3. 執行結果

```
   pyuser@deyu:~/zzz$ ./dict7.py
2  key = dywang
   key = david
4  key = linda
```

14.9 遍歷字典值

1. 使用 items() 可以遍歷 (traverse) 字典的鍵及值，若只想遍歷字典的值，可用如下 values()：

```
for value in dict_name.values():
```

2. 範例：遍歷並列出字典 uid 的值。

```
pyuser@deyu:~/zzz$ cat dict8.py
#!/usr/bin/python

uid = { 'dywang':1000, 'david':2346, 'linda':1324 }
for value in uid.values():
    print("uid = %d" % value)
```

3. 執行結果

```
pyuser@deyu:~/zzz$ ./dict8.py
uid = 1000
uid = 2346
uid = 1324
```

14.10 實機練習題

14.10.1 字典定義、增加、變更、驗證

1. 前置作業
   (a) 在家目錄下建立 zzz 目錄。
   (b) 切換工作目錄到 zzz。
   (c) 請下載 dict-passwd.txt，dict-passwd.txt 是從 linux 系統的 /etc/passwd 帳號檔截錄。
   (d) 本節建立的字典，都以字典「值」uid 由小到大排序。
   (e) 在 zzz 目錄下寫 python 程式。
2. dict-def1.py

(a) 將 dict-passwd.txt 中 uid 大於 50 的帳號，uid 由小到大定義成字典 passwd_dict1，key 是帳號名稱、value 是 uid。

(b) 輸出字典 passwd_dict1。

3. dict-add1.py

(a) 將 dict-passwd.txt 中 uid 小於 5 的帳號，uid 由小到大定義成字典 passwd_dict2，key 是帳號名稱、value 是 uid。

(b) 字典 passwd_dict2 增加一元素，帳號爲 deyu1，uid 1000。

(c) 輸出字典 passwd_dict2。

4. dict-change1.py

(a) 變更字典 passwd_dict2 元素 deyu1 的 uid 爲 1001。

(b) 輸出字典 passwd_dict2。

5. dict-verify1.py

(a) 將 dict-passwd.txt 中 uid 小於 10 的帳號，uid 由小到大定義成字典 passwd_dict3，key 是帳號名稱、value 是 uid。

(b) 執行程式時，提示「請輸入一字串 :」，假設是 key。

(c) 判斷字典 passwd_dict3 中是否存在此鍵 key？

(d) 存在則輸出其「key value」

(e) 否則新增此鍵，value 爲 1122，並輸出字典 passwd_dict3。

14.10.2 遍歷字典

1. 前置作業

(a) 在家目錄下建立 zzz 目錄。

(b) 切換工作目錄到 zzz。

(c) 請 下載 dict-passwd.txt，dict-passwd.txt 是從 linux 系統的 /etc/passwd 帳號檔截錄。

(d) 本節建立的字典，都以字典「值」uid 由小到大排序。

(e) 在 zzz 目錄下寫 python 程式。

2. dict-traverse1.py

(a) 將 dict-passwd.txt 中 uid 介於 60 到 100 的帳號，uid 由小到大定義成字典 `passwd_dict4`，key 是帳號名稱、value 是 uid。

(b) 使用 for 迴圈遍歷字典 `passwd_dict4`，將所有鍵字串不含空白，串接成變數 keys，所有的值 (value) 則累加儲存於 valuesum。

(c) 輸出「keys<TAB>valuesum」

3. dict-traverse2.py

(a) 將 dict-passwd.txt 中 uid 介於 80 到 1000 的帳號，uid 由小到大定義成字典 `passwd_dict5`，key 是帳號名稱、value 是 uid。

(b) 使用 for 迴圈遍歷 `passwd_dict5` 字典鍵，將鍵 (key) 字串長度小於 (含) 4 個字元的所有鍵字串不含空白，串接成變數 keys，其值 (value) 則累加儲存於 valuesum。

(c) 輸出「keys<TAB>valuesum」

4. dict-del1.py

(a) 將 dict-passwd.txt 中 uid 介於 30 到 120 的帳號，uid 由小到大定義成字典 `passwd_dict6`，key 是帳號名稱、value 是 uid。

(b) 使用 for 迴圈遍歷 `passwd_dict6` 字典值，刪除值 (value) 是奇數元素。

(c) 輸出字典 `passwd_dict6`。

Chapter 15

函式

15.1 簡介

1. 大程式可依功能分割成多個部分，每一部分寫成函式 (function)，可以簡化程式也容易除錯。

2. 程式中經常要重複一部分程式時，可將此部分程式寫成函式，需要時再呼叫，不但可避免重複撰寫程式，更可使程式簡潔、清晰，易讀。

15.2 定義

1. 函式定義語法如下：

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

2. `def functionname( parameters )`: keyword def 後面接函式名稱、小括號，小括號爲呼叫時傳入的參數，參數可有可無，多個參數則以逗號隔開。

3. `function_docstring` 註解可有可無。

4. `function_suite` 函式要執行的程式。

5. `return [expression]`：return 及回傳值都可有可無，多個回傳值則以逗號隔開。

6. 簡單範例：列印呼叫時傳入的字串參數

```
def printme( str1 ):
    "This prints a passed string into this function"
    print(str1)
    return
```

15.3 呼叫函式

1. 呼叫 printme 函式，傳入字串，printme 函式將傳入的字串印出。

```
pyuser@deyu:~/zzz$ cat function1.py
#!/usr/bin/python

# Function definition is here
def printme( str1 ):
    "This prints a passed string into this function"
    print(str1)
    return;

# Now you can call printme function
printme("第一次呼叫！")
printme("再次呼叫！")
```

2. 執行結果：printme 函式印出傳入的字串。

```
pyuser@deyu:~/zzz$ ./function1.py
第一次呼叫！
再次呼叫！
```

15.4 傳入多個參數

1. 範例：函式 add 將傳入的兩個數字相加並列印。

```
pyuser@deyu:~/zzz$ cat function2.py
#!/usr/bin/python

def add( x1, x2 ):
    result = x1 +x2
    print(result)
    return

a = int(input("a = "))
```

```
b = int(input("b = "))
print("a + b =", end=' ')
add(a,b)
```

2. 執行結果：輸入 a=13 b=2，輸出 a+b=15。

```
pyuser@deyu:~/zzz$ ./function2.py
a = 13
b = 2
a + b = 15
```

15.5 參數預設值

1. 範例：設計函式時可以給參數預設值，呼叫時若沒有傳遞參數時，會使用參數預設值。add 函式將兩參數相加後回傳，兩參數預設值皆爲 0。

```
pyuser@deyu:~/zzz$ vi function3.py
#!/usr/bin/python

def add( x1=0, x2=0 ):
    result = x1 + x2
    return result

a = int(input("a = "))
b = int(input("b = "))
print("add() = %d" % add())
print("add(x1=a) = %d" % add(x1=a))
print("add(x2=b) = %d" % add(x2=b))
print("add(x1=a,x2=b) = %d" % add(x1=a,x2=b))
print("add(a,b) = %d" % add(a,b))
print("add(a,) = %d" % add(a,))
print("add(a) = %d" % add(a))
```

2. 執行結果：輸入 a=12, b=6，輸出都不傳參數、只傳參數 x1 或 x1、兩參數都傳等四種狀況。

```
pyuser@deyu:~/zzz$ ./function3.py
a = 12
b = 6
add() = 0
add(x1=a) = 12
add(x2=b) = 6
add(x1=a,x2=b) = 18
```

```
add(a,b) = 18
add(a,) = 12
add(a) = 12
```

15.6 回傳值

1. 範例：沒有 retrun 時，python 自動回傳 None。

```
pyuser@deyu:~/zzz$ cat function4.py
#!/usr/bin/python

def add( x1, x2 ):
    result = x1 + x2
    return

return_value = add(1,2)
print("return_type = ", type(return_value))
print("return_value = ", return_value)
```

2. 執行結果：回傳值是 None，type 是 NoneType。

```
pyuser@deyu:~/zzz$ ./function4.py
return_type =  <class 'NoneType'>
return_value =  None
```

3. 範例：arithmetic 函式回傳加減乘除四則運算四個值。

```
pyuser@deyu:~/zzz$ cat function5.py
#!/usr/bin/python

def arithmetic( x1, x2 ):
    add = x1 + x2
    sub = x1 - x2
    mul = x1 * x2
    div = x1 / x2
    return add, sub, mul, div

a,s,m,d = arithmetic(20,10)
print("x1 + x2 = %d" % a)
print("x1 - x2 = %d" % s)
print("x1 * x2 = %d" % m)
print("x1 / x2 = %d" % d)
```

4. 執行結果：印出 arithmetic 函式回傳加減乘除四則運算四個值。

```
pyuser@deyu:~/zzz$ ./function5.py
x1 + x2 = 30
x1 - x2 = 10
x1 * x2 = 200
x1 / x2 = 2
```

15.7 任意數量參數

1. 星號"*" 的參數代表可以有 1 到多個參數傳入函式，任意數量參數必須放在最右邊，如下 *mulpar：

```
def function_name(par1, par2, *mulpar)
```

2. 範例：函式列出學生學號、姓名、選課，選課科目不固定。

```
pyuser@deyu:~/zzz$ cat function6.py
#!/usr/bin/python

def course( sid, name, *subjects ):
    print("學號", sid, "姓名", name, "選課：")
    for subject in subjects:
        print("\t", subject, sep="")

course(100, 'linda', 'python')
course(101, 'peter', 'python', 'linux', 'c++')
```

3. 執行結果：linda 只選一門課，peter 選三門課。

```
pyuser@deyu:~/zzz$ ./function6.py
學號 100 姓名 linda 選課：
    python
學號 101 姓名 peter 選課：
    python
    linux
    c++
```

15.8 Global vs. Local variables

1. 定義在函式內的變數區域 (local) 變數，只限函式內存取；定義在函式外的全域 (global) 變數，則整個程式都可存取。

```
pyuser@deyu:~/zzz$ cat function7.py
#!/usr/bin/python

total = 0; # This is global variable.
# Function definition is here
def sum( arg1, arg2 ):
    # Add both the parameters and return them."
    total = arg1 + arg2; # Here total is local variable.
    print("Inside the function local total :", total)
    return total;

# Now you can call sum function
sum( 10, 20 );
print("Outside the function global total :", total)
```

2. 執行結果：函式內的區域變數 total=30，函式外的全域變數 total=0。

```
pyuser@deyu:~/zzz$ ./function7.py
Inside the function local total : 30
Outside the function global total : 0
```

3. 設定 global 變數 total 爲 sum 函式的回傳值。

```
pyuser@deyu:~/zzz$ cat function7a.py
#!/usr/bin/python

total = 0; # This is global variable.
# Function definition is here
def sum( arg1, arg2 ):
    # Add both the parameters and return them."
    total = arg1 + arg2; # Here total is local variable.
    print("Inside the function local total :", total)
    return total;

# Now you can call sum function
total = sum( 10, 20 );
print("Outside the function global total :", total)
```

4. 執行結果：函式內的區域變數 total=30，函式外的全域變數 total=30。

```
pyuser@deyu:~/zzz$ ./function7a.py
Inside the function local total : 30
Outside the function global total : 30
```

15.9 匿名函式 lambda

1. 匿名函式 (anonymous function) 是指沒有名稱的函式，匿名函式用 lambda 定義，arg1 是參數，可以有多個參數：

```
lambda arg1[, arg2,..., argn]: expression
```

2. 範例：square 匿名函式只有一個參數 x，計算 x 的平方；sum 匿名函式有 x, y 兩個參數，計算 x+y。

```
pyuser@deyu:~/zzz$ cat function8.py
#!/usr/bin/python

square = lambda x : x**2
sum = lambda x, y : x + y

print("x^2 = 3^2 =", square(3))
print("x + y = 5 + 9 =", sum(5,9))
```

3. 執行結果

```
pyuser@deyu:~/zzz$ ./function8.py
x^2 = 3^2 = 9
x + y = 5 + 9 = 14
```

15.10 匿名函式與 map 函式

1. 內建 map() 函式依序將 iterable 的元素傳入 function，然後將 function 執行的結果組成新的物件傳回。

```
map( function, iterable)
```

2. 範例：使用匿名函式對串列元素計算其平方值。

```
pyuser@deyu:~/zzz$ cat function9.py
#!/usr/bin/python

xlist = [1, 2, 3, 4, 5, 6, 7, 8, 9]
squarelist = map(lambda x : x**2, xlist)

print("square of xlist:", squarelist)
```

3. 執行結果：python map 回傳的是 map object at 記憶體位置。

```
pyuser@deyu:~/zzz$ ./function9.py
square of xlist: <map object at 0x7f8335e4d208>
```

4. 要輸出 map 的結果，必須將其轉換成 list。

```
pyuser@deyu:~/zzz$ cat function9a.py
#!/usr/bin/python

xlist = [1, 2, 3, 4, 5, 6, 7, 8, 9]
squarelist = list(map(lambda x : x**2, xlist))

print("square of xlist:", squarelist)
```

5. 執行結果：印出 map 的結果。

```
pyuser@deyu:~/zzz$ ./function9a.py
square of xlist: [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

15.11 實機練習題

15.11.1 average function

1. 前置作業

   (a) 在家目錄下建立 zzz 目錄。

   (b) 切換工作目錄到 zzz。

   (c) 請 下載 list.txt。

2. 在 zzz 目錄下寫一 python 程式 fun-avg1.py。
   (a) 執行程式時，提示「請輸入兩個數字，中間以空白隔開 :」
   (b) 使用 string 內建的 split 成員，將數字分開。
   (c) function 名稱 avg1，可傳入兩個數字參數 x1 及 x2。
   (d) 當傳入的兩個數字參數 x1 及 x2 都大於等於 0，回傳其平均值。
   (e) 參數 x1 或 x2 小於 0，回傳 100, x1, x2 的平均值。
   (f) 未傳入 x1 則 x1 預設 100，未傳入 x2 則 x2 預設 50。
   (g) 主程式呼叫 function varlen，使用`print('Average=%f' % xxx)` 印出「Average=xxx」，xxx 爲回傳的平均值。
   (h) 輸入的字串用字串內建的 split 函式，變成 list 串列。
3. 在 zzz 目錄下寫一 python 程式 fun-var-len1.py。
   (a) 使用 list4 串列。
   (b) 執行程式時，提示「請輸入一個數字 :」，假設是 nu。
   (c) function 名稱 varlen，第一個參數爲正整數 nu，其餘可變個數的參數傳入串列 list4。
   (d) function varlen 回傳串列 list4 的第 1 至 nu 個元素的平均值。
   (e) 主程式呼叫 function varlen，使用`print('Average=%f' % xxx)` 印出「Average=xxx」，xxx 爲回傳的平均值。

15.11.2 lambda function

1. 前置作業
   (a) 在家目錄下建立 zzz 目錄。
   (b) 切換工作目錄到 zzz。
2. 在 zzz 目錄下寫一 python 程式 fun-lambda1.py。
   (a) 執行程式時，提示「請輸入三個數字，中間以空白隔開 :」，假設是 x y z。
   (b) 提示：使用字串的 split 類別成員，拆成三個字串，並使用 map 函式將 x y z 字串都轉換成 int。
   (c) 使用名稱爲 xyz 的 lambda function 定義 x,y,z 三個變數的運算爲 x+y-2*z。
   (d) 主程式呼叫 xyz lambda function 印出「x+y-2*z=xxx」，xxx 爲 lambda function 回傳值。

Chapter 16

模組

16.1 簡介

1. 大程式一定是由許多函式或類別 (class) 組成，爲了讓程式可以依功能分割、增加程式可讀性，會將函式或類別儲存成模組 (module)。

2. 模組是包含 python 程式碼的一個檔案，可以定義函式、類別、變數。

16.2 自建模組

1. 自行建立一個加減乘除四則運算函式，存成 module1.py 做爲簡單的模組：

```
pyuser@deyu:~/zzz$ cat module1.py
#!/usr/bin/python

def add( x1, x2 ):
    return x1 + x2
def sub( x1, x2 ):
    return x1 - x2
def mul( x1, x2 ):
    return x1 * x2
def div( x1, x2 ):
    return x1 / x2
```

16.3 導入模組

1. Python 可以將任何 python 原始碼檔視爲一個模組，使用 import 將其導入。

```
import module1[, module2[,... moduleN]]
```

2. 範例：import 導入模組 module1，module1.functionname 引用 module1 中的函式。

```
pyuser@deyu:~/zzz$ cat module2.py
#!/usr/bin/python
import module1

print("8 + 4 =", module1.add(8,4))
print("8 - 4 =", module1.sub(8,4))
print("8 * 4 =", module1.mul(8,4))
print("8 / 4 =", module1.div(8,4))
```

3. 執行結果。

```
pyuser@deyu:~/zzz$ ./module2.py
8 + 4 = 12
8 - 4 = 4
8 * 4 = 32
8 / 4 = 2.0
```

16.4 導入部分函式

1. 從模組 modname 導入 name1, ... 等函式語法如下：

```
from modname import name1[, name2[, ... nameN]]
```

2. 範例：只從模組 module1 導入 add, sub 兩個函式，直接引用函式，不必另加 module1 模組名稱。

```
pyuser@deyu:~/zzz$ cat module3.py
#!/usr/bin/python
from module1 import add,sub

print("8 + 4 =", add(8,4))
print("8 - 4 =", sub(8,4))
```

3. 執行結果：輸出 add 及 sub 的結果。

```
pyuser@deyu:~/zzz$ ./module3.py
8 + 4 = 12
8 - 4 = 4
```

4. 範例：只從模組 module1 導入 add, sub 兩個函式，直接引用函式，不必另加 module1 模組名稱，增加呼叫 div 函式。

```
pyuser@deyu:~/zzz$ cat module3a.py
#!/usr/bin/python
from module1 import add,sub

print("8 + 4 =", add(8,4))
print("8 - 4 =", sub(8,4))
print("8 - 4 =", div(8,4))
```

5. 執行結果：add 及 sub 函式正常，但 div 函式沒有 import 進來，所以 div 沒定義。

```
pyuser@deyu:~/zzz$ ./module3a.py
8 + 4 = 12
8 - 4 = 4
Traceback (most recent call last):
  File "./module3a.py", line 7, in <module>
    print("8 - 4 =", div(8,4))
NameError: name 'div' is not defined
```

16.5 導入所有函式

1. 從模組 modname 導入所有函式語法如下：

```
from modname import *
```

2. 範例：從模組 module1 導入所有函式，此方式與直接 import 導入模組，差別在於引用函式時不必另加 module1 模組名稱。

```
pyuser@deyu:~/zzz$ cat module4.py
#!/usr/bin/python
from module1 import *
```

```
print("8 + 4 =", add(8,4))
print("8 - 4 =", sub(8,4))
print("8 * 4 =", mul(8,4))
print("8 / 4 =", div(8,4))
```

3. 執行結果。

```
pyuser@deyu:~/zzz$ ./module4.py
8 + 4 = 12
8 - 4 = 4
8 * 4 = 32
8 / 4 = 2.0
```

16.6 as 替代名稱

1. 若是導入的模組名稱太長可使用 as 指定替代名稱，語法如下：

```
import modname as newmodname
```

2. 範例：模組 module1 以 m1 替代其名稱。

```
pyuser@deyu:~/zzz$ cat module5.py
#!/usr/bin/python
import module1 as m1

print("8 + 4 =", m1.add(8,4))
print("8 - 4 =", m1.sub(8,4))
print("8 * 4 =", m1.mul(8,4))
print("8 / 4 =", m1.div(8,4))
```

3. 執行結果。

```
pyuser@deyu:~/zzz$ ./module5.py
8 + 4 = 12
8 - 4 = 4
8 * 4 = 32
8 / 4 = 2.0
```

4. 若是導入的模組中某函式名稱太長，或與主程式中的函式名稱相同，也可以使用 as 指定替代名稱，語法如下：

```
from modname import functionname as newfunctionname
```

5. 範例：導入模組 module1 的函式 add 時以 m1add 替代其名稱。

```
pyuser@deyu:~/zzz$ cat module6.py
#!/usr/bin/python
from module1 import add as m1add

print("8 + 4 =", m1add(8,4))
```

6. 執行結果。

```
pyuser@deyu:~/zzz$ ./module6.py
8 + 4 = 12
```

16.7 系統 sys 模組

1. Python 內建很多模組可以使用，其中 sys 模組可以控制 python shell 訊息，是常用的模組。以下範例列印 sys 四個函式：

   (a) `argv[0]`：執行的腳本名稱。

   (b) `argv[1]`：執行的腳本時，指定的第一個參數，可以指定很多參數，以 `argv[n]` 變數取得第 n 個參數。

   (c) `sys.version`：列印 python 的版本。

   (d) `sys.stdin.readline()[:-1]`：從鍵盤輸入，readline 函式讀取換行符號結束輸入，`[:-1]` 刪除讀取輸入的字串最後一個字元，也就是換行符號。

   (e) `sys.stdout.write("I like python.\n")`：輸出到螢幕，`\n` 是換行符號。

```
pyuser@deyu:~/zzz$ cat module7.py
#!/usr/bin/python
import sys

print("script name:", sys.argv[0])
print("The first argument:", sys.argv[1])
print("python version:", sys.version)
```

```
8 print("standard input:", sys.stdin.readline()[:-1])
  print("standard output:", sys.stdout.write("I like python.\n"))
```

2. 執行結果：沒給任何參數，sys.argv[1] 是 Null，所以回應 list index out of range。

```
1 pyuser@deyu:~/zzz$ ./module7.py
  script name: ./module7.py
3 Traceback (most recent call last):
    File "./module7.py", line 6, in <module>
5     print("The first argument:", sys.argv[1])
  IndexError: list index out of range
```

3. 執行結果：最後的 15 是 sys.stdout.write 的回傳值，也是其輸出的字元數。

```
  pyuser@deyu:~/zzz$ ./module7.py myfirstargv
2 script name: ./module7.py
  The first argument: myfirstarg
4 python version: 3.6.8 (default, Mar 19 2021, 05:13:41)
  [GCC 8.4.1 20200928 (Red Hat 8.4.1-1)]
6 sad
  standard input: sad
8 I like python.
  standard output: 15
```

16.8 實機練習題

16.8.1 自建模組

1. 前置作業

   (a) 在家目錄下建立 zzz 目錄。

   (b) 切換工作目錄到 zzz。

   (c) 在工作目錄 zzz 下，撰寫以下 python 程式：

2. moduleself1.py，其中有以下函式：

   (a) function 名稱 avg1，可傳入兩個數字參數 x1 及 x2。

      i. 當傳入的兩個數字參數 x1 及 x2 都大於等於 0，回傳其平均值。

ii. 參數 x1 或 x2 小於 0，回傳 100, x1, x2 的平均值。

iii. 未傳入 x1 則 x1 預設 100，未傳入 x2 則 x2 預設 50。

(b) function 名稱 sum1，可傳入兩個數字參數 x1 及 x2。

i. 當傳入的兩個數字參數 x1 及 x2 都大於等於 0，回傳其總合值。

ii. 參數 x1 或 x2 小於 0，回傳 100, x1, x2 的總合值。

iii. 未傳入 x1 則 x1 預設 50，未傳入 x2 則 x2 預設 150。

3. module-importself1.py

(a) 使用 import moduleself1 導入所有函式，不要使用 from ... import。

(b) 執行程式時，提示「請輸入一數字 :」，假設是 nu。

(c) 如果 nu 是偶數，呼叫 avg1 時傳入 x1 = nu, 呼叫 sum1 時傳入 x2 = nu。

(d) 如果 nu 是奇數，呼叫 avg1 時傳入 x2 = nu, 呼叫 sum1 時傳入 x1 = nu。

(e) 輸出「Input=nu Average=xxx Sum=yyy」，nu 是輸入的數字，xxx 爲 avg1 回傳值，yyy 爲 sum1 回傳值。

4. module-importself2.py

(a) 使用 import 從 moduleself1 導入函式 avg1，且使用 as 指定替代名稱爲 ms1avg。

(b) 執行程式時，提示「請輸入一數字 :」，假設是 nu。

(c) 呼叫 ms1avg 時傳入 x1 = nu。

(d) 輸出「Input=nu Average=xxx」，nu 是輸入的數字，xxx 爲 ms1avg 回傳值。

16.8.2 sys 模組

1. 前置作業

(a) 在家目錄下建立 zzz 目錄。

(b) 切換工作目錄到 zzz。

(c) 在工作目錄 zzz 下，撰寫以下 python 程式：

2. module-sys1.py

(a) 使用 python 內建 sys 模組，讓執行 python 程式時，可以指定參數。

(b) 計算所有數字參數的平均值並轉換成整數，假設爲 avg。

(c) 連結所有非數字參數 (不含執行命令本身) 成新字串，假設爲 newstr。

(d) 輸出「newstr=avg」。

(e) 範例：

```
pyuser@deyu:~/zzz$ ./module-sys1.py 1 ab 23 cd e 4
abcde=9
```

(f) 提示：使用 str.isnumeric() 判斷 str 是否可以轉換成數字。

Chapter 17

檔案讀取寫入

17.1 簡介

1. stdin 使用 input 從鍵盤輸入文字，stdout 使用 print 將文字輸出到螢幕。
2. Python 內建提供基本且必要的管理檔案函式。

17.2 open 開檔

1. open 函式產生 file 物件：

```
file object = open(file_name [, access_mode][, buffering])
```

   (a) file_name：要開啓的檔案名稱

   (b) access_mode：開啓檔案模式 (讀、寫、累加、覆蓋、二進位)

      i. r : reading only
      ii. rb : reading only in binary format
      iii. r+ : both reading and writing
      iv. rb+ : both reading and writing in binary format
      v. w : writing only, 檔案存在覆蓋，不存在則產生。
      vi. wb : writing only in binary format, 檔案存在覆蓋，不存在則產生。
      vii. w+ : both reading and writing, 檔案存在退出，不存在則產生。
      viii. wb+ : both reading and writing in binary format, 檔案存在退出，不存在則產生。
      ix. a : appending (累加), 不存在則產生。

x. ab：appending in binary format, 不存在則產生。

xi. a+：both appending and reading, 不存在則產生。

xii. ab+：both appending and reading in binary format, 不存在則產生。

(c) buffering：

i. 0：no buffering

ii. 1：line buffering (一行)

iii. n>1：n bytes buffering

iv. n<0：system 預設

v. 省略不指定: system 預設

17.3 file 物件屬性

1. 開檔產生 file 物件，相關屬性有：

   (a) file.closed：如果檔案關閉，回傳 true，反之 false。

   (b) file.mode：回傳檔案開啓的模式。

   (c) file.name：回傳檔案名稱。

2. 範例：開啓 fileio.txt 檔，印出 file 物件屬性。

```
pyuser@deyu:~/zzz$ cat fileio1.py
#!/usr/bin/python

# Open a file
fo = open("fileio.txt", "ab+")
print("Name of the file:", fo.name)
print("Closed or not :", fo.closed)
print("Opening mode :", fo.mode)
```

3. 執行結果：屬性如設定，且產生 fileio.txt 檔。

```
pyuser@deyu:~/zzz$ ./fileio1.py
Name of the file: fileio.txt
Closed or not : False
Opening mode : ab+
pyuser@deyu:~/zzz$ ll fileio.txt
-rw-rw-r--. 1 dywang dywang 0 Dec 23 14:13 fileio.txt
```

17.4 close method

1. file 物件的 close() method 更新還沒寫入的資料並關閉檔案，關閉不能再寫入資料。

```
fileObject.close()
```

2. 範例：關閉檔案後測試檔案是否開啓？

```
pyuser@deyu:~/zzz$ cat fileio2.py
#!/usr/bin/python

# Open a file
fo = open("fileio.txt", "ab+")
print("Name of the file:", fo.name)
print("1. Closed or not:", fo.closed)
# Close opend file
fo.close()
print("2. Closed or not:", fo.closed)
```

3. 執行結果：file 物件關閉後 fo.closed 回傳 True。

```
pyuser@deyu:~/zzz$ ./fileio2.py
Name of the file: fileio.txt
1. Closed or not: False
2. Closed or not: True
```

17.5 write method

1. write method 將字串寫入檔案，字串可以是文字或二進位資料。

```
fileObject.write(string)
```

2. 範例：將兩行文字寫入檔案 \n 是換行符號。

```
pyuser@deyu:~/zzz$ cat fileio3.py
#!/usr/bin/python

fo = open("fileio.txt", "ab+")
fo.write("Python 是很棒的程式語言。\n是的，確實很棒!!\n".encode())
```

```
fo.close()
```

3. 執行結果：執行程式後，查看檔案 fileio.txt 如程式寫入的資料，檔案大小是 59 bytes。

```
pyuser@deyu:~/zzz$ ./fileio3.py
pyuser@deyu:~/zzz$ cat fileio.txt
Python 是很棒的程式語言。
是的，確實很棒!!
pyuser@deyu:~/zzz$ ll fileio.txt
-rw-rw-r--. 1 dywang dywang 59 Dec 23 14:26 fileio.txt
```

17.6 read method

1. read method 從開啓的檔案物件中讀取字串，count 是讀取的字數，不給 count 則讀取整個檔案。

```
fileObject.read([count])
```

2. 範例：第一次讀 3 做字，接著讀 10 個字。

```
pyuser@deyu:~/zzz$ cat fileio4.py
#!/usr/bin/python

fo = open("fileio.txt", "r+")
print("Read 3 bytes is:", fo.read(3))
print("Read 10 bytes is:", fo.read(10))
fo.close()
```

3. 執行結果：

```
pyuser@deyu:~/zzz$ ./fileio4.py
Read 3 bytes is: Pyt
Read 10 bytes is: hon 是很棒的程式
```

17.7 File Positions

1. tell method 取得目前在檔案的讀寫位置。seek(offest[n, from]) method 改變目前的讀寫位置，n 是移動的 bytes 數，from 是開始的位置，from 設定值：
   (a) 0：檔案的開頭；
   (b) 1：目前位置；
   (c) 2：檔案結尾。
2. 範例：讀 10 bytes 後，使用 tell 查看目前的讀取位置。使用 seek 將讀取位置移到檔案開頭，再次讀取 10 bytes。

```
pyuser@deyu:~/zzz$ cat fileio5.py
#!/usr/bin/python

fo = open("fileio.txt", "r+")
print("Read 10 bytes is:", fo.read(10))
print("Current file position:", fo.tell())
fo.seek(0,0)
print("Current file position after seek:", fo.tell())
print("Again read 10 bytes is:", fo.read(10))
fo.close()
```

3. 執行結果：第一次讀取 10 個字後，讀取位置在 16，6 個字元加 1 個空白，每個中文字 3 個位元組，所以位置在 16，使用 seek 後回到 0，再次讀取還是從檔案開頭讀取。

```
pyuser@deyu:~/zzz$ ./fileio5.py
Read 10 bytes is: Python 是很棒
Current file position: 16
Current file position after seek: 0
Again read 10 bytes is: Python 是很棒
```

17.8 os 模組

1. Python 內建很多模組可以使用，其中 os 模組提供檔案管理員的 methods：
   (a) os.rename(current_file_name, new_file_name)：檔案重新命名。
   (b) os.remove(file_name)：刪除檔案。
   (c) os.mkdir("newdir")：建立目錄。

(d) os.chdir("newdir")：變更工作目錄。

(e) os.getcwd()：列印目前的工作目錄。

(f) os.rmdir('dirname')：刪除目錄。

2. 範例：開啓檔案 ostest.txt，寫入資料後關閉，使用 rename method 重新命名爲 newostest.txt，讀取檔案，關閉後刪除檔案。

```
pyuser@deyu:~/zzz$ cat fileio6.py
#!/usr/bin/python
import os

fo = open("ostest.txt", "w")
fo.write("file rename\n")
fo.close()
os.rename("ostest.txt", "newostest.txt")
fo = open("newostest.txt", "r")
print(fo.read()[:-1])
fo.close()
os.remove("newostest.txt")
```

3. 執行結果：列印 newostest.txt 結果無誤，且執行完 ostest.txt 及 newostest.txt 檔案都不存在。

```
pyuser@deyu:~/zzz$ ./fileio6.py
file rename
pyuser@deyu:~/zzz$ ll *ostest.txt
ls: cannot access *ostest.txt: No such file or directory
```

4. 範例：建立 ostest 目錄，改變工作目錄到 ostest，列印目前的工作目錄，回到上一層目錄，刪除 ostest 目錄。

```
pyuser@deyu:~/zzz$ cat fileio7.py
#!/usr/bin/python
import os

os.mkdir("ostest")
os.chdir("ostest")
print(os.getcwd()[:])
os.chdir("../")
os.rmdir("ostest")
```

5. 執行結果：列印工作目錄爲 /home/dywang/zzz/ostest，而且程式執行完 ostest 目錄也被刪除了。

```
pyuser@deyu:~/zzz$ ./fileio7.py
/home/dywang/zzz/ostest
pyuser@deyu:~/zzz$ ll ostest
ls: cannot access ostest: No such file or directory
```

17.9 實機練習題

1. 前置作業
   (a) 在家目錄下建立 zzz 目錄。
   (b) 切換工作目錄到 zzz，撰寫以下 python 程式：
2. file-write1.py
   (a) 使用 sys 模組，讓執行 file-write1.py 時，可以從命令列第一個參數指定寫入的檔案名稱。
   (b) 寫入與註冊的 sid 檔一樣的內容，例如：「電腦位置 IP 尾數學號姓名」。
   (c) 退出程式前必須關閉所有檔案。
3. file-read-write1.py
   (a) 使用 sys 模組，讓執行 file-read-write1.py 時，可以從命令列指定以下參數：
      i. 第一個參數：假設 fo，寫入的檔案名稱。
      ii. 第二個參數：假設 n1，檔案讀取的開始位置，必須是正整數。
      iii. 第三個參數：假設 n2，檔案讀取的 bytes，必須是正整數。
   (b) 以二進位寫入模式，開啓執行參數指定的檔案 fo。
   (c) 以二進位唯讀模式，開啓檔案 /usr/bin/yes。
   (d) 從第 n1 bytes 開始讀取 n2 bytes。
   (e) 將讀取的 n2 bytes 寫入指定的檔案 fo。
   (f) 退出程式前必須關閉所有檔案。
4. file-pos1.py
   (a) 從命令列第一個參數讀入檔名，以唯讀二進位模式開檔。
   (b) 讀取前 n 個字，n 爲命令列第二個參數。

(c) 列出目前的檔案讀寫位置。

(d) 改變檔案讀寫位置從目前位置位移 m 個 bytes，m 爲命令列第三個參數。

(e) 列出目前的檔案讀寫位置。

(f) 退出程式前必須關閉所有檔案。

5. file-os1.py

(a) 從命令列第一個參數讀入檔名，假設 aaa，開檔寫入與註冊的 sid 檔一樣的內容，例如：「電腦位置 IP 尾數學號姓名」。

(b) 使用 os 模組，變更檔名 aaa 爲 aaa1。

(c) 建立目錄 aaadir。

(d) 進入目錄 aaadir，列印目前工作目錄。

(e) 回到原工作目錄，變更目錄 aaadir 爲 aaadir1。

(f) 退出程式前必須關閉所有檔案。

Chapter 18

re 模組之正規表示式

18.1 前言

1. 什麼是正規表示法
   (a) 處理字串的方法，以行爲單位進行字串的處理行爲。
   (b) 透過一些特殊符號的輔助，讓使用者輕易的達到搜尋、取代，限定某特定字串的處理程序。
   (c) 延伸的正規表示法：可以作群組的字串『或 (or)』及『且 (and)』的處理。
   (d) 正規表示法與萬用字元不一樣，不可混淆。
2. 正規表示法用途
   (a) 系統管理員的可以透過『正規表示法』的功能，將登錄的資訊進行處理，僅取出『有問題』的資訊來進行分析。
   (b) 一般支援正規表示法的軟體可利用其進行字串的處理，例如：『郵件伺服器』以『字串』的比對來過濾郵件。
   (c) 軟體中輸入欄位的格式限制，例如：只能爲數字。
3. 正規表示式廣泛運用於 Linux，vi, grep, awk, sed 等工具皆支援正規表示法。
4. Python 內建 re 模組提供正規表示式支援。

18.2 re 模組函式

1. re 模組提供對正規表示式的支援，函式 match 「從字串 string 開頭找匹配」語法如下：

```
re.match(pattern, string[, flags])
```

2. search 函式從字串 string 任何位置找匹配

```
1 re.search(pattern, string[, flags])
```

3. split 函式依指定模式分割字串 string，存成 list

```
1 re.split(pattern, string[, maxsplit])
```

4. findall 函式從字串 string 找所有匹配，並回傳 list

```
1 re.findall(pattern, string[, flags])
```

5. finditer 函式從字串 string 找所有匹配，並回傳 iterator

```
1 re.finditer(pattern, string[, flags])
```

6. sub 函式從字串 string 找到匹配並取代成新字串 repl，最多取代 count 次，當 count=0 表示取代所有匹配。

```
1 re.sub(pattern, repl, string[, count])
```

7. subn 函式與 sub 函式相同，但回傳新字串及取代次數。

```
1 re.subn(pattern, repl, string[, count])
```

8. 以上函式必須傳入匹配模式 pattern，python 程式可透過 compile 函式產生匹配模式才能進行匹配，或將原始正規表示式控制模式，以單引號括起來，並以開頭 r 如 r'...'。

```
1 re.compile(pattern, flags=0)
```

18.3 匹配模式 pattern

1. 單一字元控制符號：

符號	描述	實例	匹配字串
^	行首	^word	word
$	行尾	word$	word
.	任意字元 (換行字元除外)	c.t	cat cbt cct
[...]	中括號中所列的字元	[abc]	a b c
[^...]	中括號中所列的字元外的所有字元	[^abc]	d e

2. 數量控制符號：必須要跟前一字元一起看，此表以 re 代表前一字元。

符號	描述	實例	匹配字串
re*	匹配 0 或多次前一個字元	xyz*	xy xyz xyzzz
re+	匹配 1 或多次前一個字元	xyz+	xyz xyzzz
re?	匹配 0 或 1 次前一個字	xyz?	xy xyz
re{n}	匹配剛好 n 次前一個字元	xy{3}z	xyyyz
re{n,}	匹配 n 次 (含) 以上前一個字元	xy{2,}z	xyyz xyyyyz
re{n,m}	匹配最少 n 次最多 m 次前一個字元	xy{2,3}z	xyyz xyyyz

3. 預定義控制符號：以 "\" 接一字元代表一群字元，可用於 [...] 字元集中。

符號	描述	實例	匹配字串
\d	數字 0 到 9，同 [0-9]	x\dz	x1z x5z
\D	非數字，同 [^\d]	x\Dz	xyz xAz
\s	空白字元，同 [\t\r\n\f\v]	x\sz	x z
\S	非空白字元，同 [^\s]	x\Sz	xBz
\w	數字與英文字母，同 [a-zA-Z0-9]	x\wz	x3z xBz
\W	非數字與英文字母，同 [^a-zA-Z0-9]	x\wz	x#z
\A	僅匹配字串開頭，不同於 ^ 是每行行首	\Axyz	xyz
\Z	僅匹配字串結尾，不同於 $ 是每行行尾	xyz\Z	xyz
\b	匹配 \w 與 \W 之間，尋找字串邊界	\bxyz\b xyz!	xyz

\B	[^\b，尋找非字串邊界	\Bxyz\B AxyzH	xyz

4. 邏輯、分組控制符號：

符號	描述	實例	匹配字串
abc|xyz	| 左右兩邊任意匹配一個	abc|xyz	abc xyz
(...)	小括號內爲一組，每遇到一個左括號編號 +1，可透過 re 模組的 group 函式取得匹配字串，或 \num 引用。	(xyz){2}	xyzxyz
\num	引用編號爲 num 的匹配字串	(\d)xyz\1	1xyz1 3xyz3
(?P<name>..)	同上的分組，但除了原編號外，再指定額外別名	(?P<id>xyz){2}	xyzxyz
(?P=name)	引用別名爲 name 的匹配字串	(?P<id>\d)xyz(?P=id)	1xyz1 3xyz3

5. 特殊控制符號：此時 (..) 不是做分組。

符號	描述	實例	匹配字串
(?:..)	(..) 不做分組編號	(?:xyz){2}	xyzxyz
(?ilmsux)	ilmsux 每個字元代表一個匹配模式，只能用在正規表示式開頭，可複選，於選項 flag 中介紹	(?i)xyz	xyz Xyz
(?#..)	# 後的內容爲註解	xy(?#abc)z?	xyz
(?=..)	接於字元後的匹配模式，但匹配時不印匹配模式。	xy(?=\d) xy3	xy
(?!..)	接於字元後的不匹配模式，但匹配時不印匹配模式。	xy(?!\d) xyz	xy
(?<=..)	接於字元前的匹配模式，但匹配時不印匹配模式。	(?<=\d)xy 3xy	xy
(?<!..)	接於字元前的不匹配模式，但匹配時不印匹配模式。	(?<!\d)xy zxy	xy

(?(id/name)yes-pattern|no-pattern)	如果編號爲 id 或別名爲 name 的分組匹配，使用 yes-pattern，否則使用 no-pattern	((\d)xyz(?(1)\d|\D)	1xyz3 xyzB

18.4 選項 flag

1. flag：指定搜尋選項，可以使用 bitwise OR (|) 指定不同的選項，re 可用的選項如下：
   (a) re.I(IGNORECASE): 忽略大小寫（括號內是完整拼字，下同）
   (b) re.M(MULTILINE): 多行模式，改變行首 '^' 和行尾 '$' 的行爲
   (c) re.S(DOTALL): 點任意匹配模式，改變 '.' 的行爲
   (d) re.L(LOCALE): 使預定字符類 \w \W \b \B \s \S 取決於當前語系設定
   (e) re.U(UNICODE): 使預定字符類 \w \W \b \B \s \S \d \D 取決於 unicode 定義的字符屬性
   (f) re.X(VERBOSE): 詳細模式。這個模式下正規表示式可以是多行，忽略空白字元，並可以加入註釋。

18.5 match 函式

1. 範例：match 函式「從字串開始位置」找正規表示式匹配的字串，函式回傳匹配的 object，需要使用 group(n) 方法取出匹配的字串，其中括號內的 n 爲匹配的第 n 分組，不指定則輸出全部分組。

```
pyuser@deyu:~/zzz$ cat re1.py
#!/usr/bin/python
import re, sys

str = sys.argv[1]
mo1 = re.match( r'^word', str, re.I)
mo2 = re.match( r'abc$', str, re.I)
mo3 = re.match( r'w.*', str, re.I)
mo4 = re.match( r'[wd]', str, re.I)
mo5 = re.match( r'[^wd]', str, re.I)
if mo1: print('1. ^word :', mo1.group())
if mo2: print('2. abc$ :', mo2.group())
if mo3: print('3. w.* :', mo3.group())
if mo4: print('4. [wd] :', mo4.group())
if mo5: print('5. [^wd] :', mo5.group())
```

2. 執行結果：

(a) 只匹配 1,3,4 三種，爲什麼 2 abc$ 不匹配，4 [wd] 也只匹配 w 沒有 d？
因爲 match 是從開頭找，也就是 abc$ 其實是 ^abc$

```
pyuser@deyu:~/zzz$ ./re1.py "word abc"
1. ^word : word
3. w.* : word abc
4. [wd] : w
```

(b) re.I 指定大小寫都可以，所以也匹配 1,3,4 三種。

```
pyuser@deyu:~/zzz$ ./re1.py "Word abc"
1. ^word : Word
3. w.* : Word abc
4. [wd] : W
```

(c) 因爲 match 函式「從字串開始位置」找正規表示式匹配的字串，也就是已經預設 ^ 開頭，此例中 pattern r'abc$' 等效 r'^abc$'。

```
pyuser@deyu:~/zzz$ ./re1.py "abc"
2. abc$ : abc
5. [^wd] : a
```

(d) 字串"abc word"，abc 不是結尾，所以不匹配，開頭是 a，匹配 [^wd] 開頭不是 w 或 d。

```
pyuser@deyu:~/zzz$ ./re1.py "abc word"
5. [^wd] : a
```

18.6 search 函式

1. 範例：search 函式「從字串任意位置」找正規表示式匹配的字串，函式回傳匹配的 object，需要使用 group(n) 方法取出匹配的字串，其中括號內的 n 爲匹配的第 n 分組，不指定則輸出全部分組。

```
pyuser@deyu:~/zzz$ cat re2.py
#!/usr/bin/python
import re, sys

str = sys.argv[1]
mo1 = re.search( r'^word', str, re.I)
mo2 = re.search( r'abc$', str, re.I)
mo3 = re.search( r'w.*', str, re.I)
mo4 = re.search( r'[wd]', str, re.I)
mo5 = re.search( r'[^wd]', str, re.I)
if mo1: print('1. ^word :', mo1.group())
if mo2: print('2. abc$ :', mo2.group())
if mo3: print('3. w.* :', mo3.group())
if mo4: print('4. [wd] :', mo4.group())
if mo5: print('5. [^wd] :', mo5.group())
```

2. 執行結果一：1,2,3,4,5 五種表示式都找到匹配，但 4,5 爲什麼只匹配 w 及 o？

```
pyuser@deyu:~/zzz$ ./re2.py "word abc"
1. ^word : word
2. abc$ : abc
3. w.* : word abc
4. [wd] : w
5. [^wd] : o
```

3. 執行結果二：3,4,5 三種表示式匹配。

```
pyuser@deyu:~/zzz$ python re2.py 'abc word'
3. w.* : word
4. [wd] : w
5. [^wd] : a
```

18.7 findall 函式

1. 範例：findall 函式找正規表示式匹配的所有字串並存成 list 回傳，將範例 re1.py 的 mo2 由 match 改成 findall，就可以找到字尾是 abc 的字串了。

```
pyuser@deyu:~/zzz$ cat re-findall.py
#!/usr/bin/python
import re, sys

str = sys.argv[1]
mo1 = re.match( r'^word', str, re.I)
```

```
mo2 = re.findall( r'abc$', str, re.I)
mo3 = re.match( r'w.*', str, re.I)
mo4 = re.match( r'[wd]', str, re.I)
mo5 = re.match( r'[^wd]', str, re.I)
if mo1: print('1. ^word :', mo1.group())
if mo2: print('2. abc$ :', mo2)
if mo3: print('3. w.* :', mo3.group())
if mo4: print('4. [wd] :', mo4.group())
if mo5: print('5. [^wd] :', mo5.group())
```

2. 執行結果：行尾 abc 的字串可以找到了。

```
pyuser@deyu:~/zzz$ ./re-findall.py "word abc"
1. ^word : word
2. abc$ : ['abc']
3. w.* : word abc
4. [wd] : w
```

18.8 sub 函式

1. 範例：sub 函式將字串中找到的正規表示字串取代指定的字串。

```
pyuser@deyu:~/zzz$ cat re3.py
#!/usr/bin/python
import re, sys

str = sys.argv[1]
mo1 = re.findall( r'c[aou]t', str, re.I)
mo2 = re.sub( r'(c)([aou])(t)', r'\3\2\1', str)
if mo1: print('1. findall :', mo1)
if mo2: print('2. sub :', mo2)
```

2. 執行結果：findall 找到三個匹配字串，sub 將找到的字串前後字元對調。

```
pyuser@deyu:~/zzz$ ./re3.py "cat cot cut cxt"
1. findall : ['cat', 'cot', 'cut']
2. sub : tac toc tuc cxt
```

18.9 匹配數量範例

1. 範例：使用 findall 函式找正規表示式指定的匹配次數的所有字串並存成 list 回傳。

```
pyuser@deyu:~/zzz$ cat re4.py
#!/usr/bin/python
import re, sys

str = sys.argv[1]
mo1 = re.findall( r'wo*d', str, re.I)
mo2 = re.findall( r'wo+d', str, re.I)
mo3 = re.findall( r'wo?d', str, re.I)
mo4 = re.findall( r'wo{3}d', str, re.I)
mo5 = re.findall( r'wo{2,}d', str, re.I)
mo6 = re.findall( r'wo{1,3}d', str, re.I)
if mo1: print('1. wo*d :', mo1)
if mo2: print('2. wo+d :', mo2)
if mo3: print('3. wo?d :', mo3)
if mo4: print('4. wo{3}d :', mo4)
if mo5: print('5. wo{2,}d :', mo5)
if mo6: print('6. wo{1,3}d :', mo6)
```

2. 執行結果：findall 找到指定 o 匹配次數的字串。

```
pyuser@deyu:~/zzz$ ./re4.py "wd wod wood woood xyz wooood woooood"
1. wo*d : ['wd', 'wod', 'wood', 'woood', 'wooood', 'woooood']
2. wo+d : ['wod', 'wood', 'woood', 'wooood', 'woooood']
3. wo?d : ['wd', 'wod']
4. wo{3}d : ['woood']
5. wo{2,}d : ['wood', 'woood', 'wooood', 'woooood']
6. wo{1,3}d : ['wod', 'wood', 'woood']
```

18.10 實機練習題

1. 前置作業

   (a) 在家目錄下建立 zzz 目錄。

   (b) 切換工作目錄到 zzz，撰寫以下 python 程式：

2. re-match1.py：

   (a) 使用 sys 模組，讓執行 re-match1.py 時，指定搜尋字串，假設 str1。

(b) 使用 re 模組，re.match 搜尋指定的字串。

(c) 不分大小寫，如果符合「h 開始，d 結尾」，則印出找到的字串。

(d) 否則印出「找不到字串 str1」，其中 str1 是輸入的字串。

3. re-search1.py：

(a) 使用 sys 模組，讓執行 re-search1.py 時，指定搜尋字串。

(b) 使用 re 模組，re.search 搜尋指定的字串。

(c) 不分大小寫，如果符合「行首爲 h，行尾爲 d」，則印出找到的字串。

(d) 否則印出「找不到字串 str1」，其中 str1 是輸入的字串。

4. re-sub1.py：

(a) 使用 sys 模組，讓執行 re-sub1.py 時，指定一串由 ()[]{} 所組成之字串，假設是 str2。

(b) 讀入 str2 字串，判斷字串是否合法？

(c) 一個 ()[]{} 必須對稱存在，且對應的括號間不能插入單獨的括號。
提示：while 迴圈中，三個獨立的對稱括號，取代成空字串，放回原字串變數，直到字串變數不變。

```
tmp=''
while ans!=tmp:
    tmp=ans
    ans=re.sub('\(\)','',ans)
    .....
```

(d) 範例輸出一：

```
({[()]()})
Ans=CORRECT
```

(e) 範例輸出二：

```
(){[()]()
Ans=ERROR
```

5. re-sub2.py：

(a) 使用 sys 模組，讓執行 re-sub2.py 時，指定搜尋的字串。

(b) 對於一個字串若起始字元爲 A，最後字元爲 L，且其間不含任何 A 或 L 字元則稱 AL 字串，AL 不分大小寫。

(c) 例：A_PENCIL 或 A_XXBBHCL （_ 表示空白字元）。

(d) 消去所有可能的 AL 字串，使得消去後之輸出不包含 AL 字串。

(e) 範例輸入：

```
This is a ball, not a bell.
```

(f) 範例輸出：

```
1 This is a ball, not a bell.
  This is , not l.
```

6. re-sub3.py：

(a) 使用 sys 模組，讓執行 re-sub3.py 時，指定搜尋的字串。

(b) 讀入一長串字串資料，字串資料是由字元（‘a’~’z’,’A’~’Z’）、數字（‘0’~’9’）以及空白字元所構成。

(c) 將字串中由字元或者空白隔開的數字部分進行加總，並且將原字串以及結果輸出。

提示：sub 取代做以下動作：

i. 所有非數字字元取代成空白

ii. 所有連續空白取代成”+” 號

iii. 刪除字串前後”+” 號後

iv. 刪除數字前的 0，例如：0234，改成 234，獨立的 0 不必刪。

(d) 例如輸入：

```
Cyut54is a356good 72school Ne0125ver 56give22 66up
```

(e) 則取代後的字串爲：

```
1 54+356+72+125+56+22+66
```

(f) 以 eval 函式計算” 字串運算式”，輸出範例如下：

```
Cyut54is a356good 72school Ne0125ver 56give22 66up
54+356+72+125+56+22+66=751
```

Chapter 19

除錯與異常處理

19.1 簡介

1. 程式異常 (exceptions) 是寫程式一定會碰到的問題。
2. 範例：定義一除法函式，呼叫時若除數爲 0，就可能出錯。

```
pyuser@deyu:~/zzz$ cat try1.py
#!/usr/bin/python

def div(x, y):
    return x/y

print("6/2=" , div(6,2))
print("3/0=" , div(3,0))
```

3. 執行結果：當呼叫時參數 6,2 回傳 3，但參數 3,0 時出現 ZeroDivisionError 的錯誤，程式無法繼續執行。

```
pyuser@deyu:~/zzz$ ./try1.py
6/2= 3.0
Traceback (most recent call last):
  File "/home/pyuser/10927001/try1.py", line 7, in <module>
    print("3/0=" , div(3,0))
                  ^^^^^^^^
  File "/home/pyuser/10927001/try1.py", line 4, in div
    return x/y
           ~^~
ZeroDivisionError: division by zero
```

4. Python 提供異常捕捉 (如 ZeroDivisionError) 並透過異常處理程序 (try - except)，讓程式可以繼續往下執行。

19.2 標準異常

異常名稱	描述
BaseException	所有異常的基類
SystemExit	翻譯器請求退出
KeyboardInterrupt	用户中斷執行 (通常是輸入^C)
Exception	常規錯誤的基類
StopIteration	疊代器沒有更多的值
GeneratorExit	生成器 (generator) 發生異常來通知退出
StandardError	所有的內建標準異常的基類
ArithmeticError	所有數值計算錯誤的基類
FloatingPointError	浮點計算錯誤
OverflowError	數值運算超出最大限制
ZeroDivisionError	除 (或取模) 零 (所有數據類型)
AssertionError	斷言語句失敗
AttributeError	對象沒有這個屬性
EOFError	沒有內建輸入, 到達 EOF 標記
EnvironmentError	操作系統錯誤的基類
IOError	輸入/輸出操作失敗
OSError	操作系統錯誤
WindowsError	系統調用失敗
ImportError	導入模塊/對象失敗
LookupError	無效數據查詢的基類
IndexError	序列中沒有此索引 (index)
KeyError	映射中沒有這個鍵
MemoryError	內存溢出錯誤 (對於 Python 解釋器不是致命的)
NameError	未聲明/初始化對象 (沒有屬性)
UnboundLocalError	訪問未初始化的本地變量
ReferenceError	弱引用 (Weak reference) 試圖訪問已經垃圾回收了的對象
RuntimeError	一般的運行時錯誤
NotImplementedError	尚未實現的方法
SyntaxError	Python 語法錯誤
IndentationError	縮進錯誤
TabError	Tab 和空格混用
SystemError	一般的解釋器系統錯誤
TypeError	對類型無效的操作
ValueError	傳入無效的參數
UnicodeError	Unicode 相關的錯誤
UnicodeDecodeError	Unicode 解碼時的錯誤

UnicodeEncodeError	Unicode 編碼時錯誤
UnicodeTranslateError	Unicode 轉換時錯誤
Warning	警告的基類
DeprecationWarning	關於被棄用的特征的警告
FutureWarning	關於構造將來語義會有改變的警告
OverflowWarning	舊的關於自動提升爲長整型 (long) 的警告
PendingDeprecationWarning	關於特性將會被廢棄的警告
RuntimeWarning	可疑的運行時行爲 (runtime behavior) 的警告
SyntaxWarning	可疑的語法的警告
UserWarning	用户代碼生成的警告

19.3 try-except

1. try - except 語法如下：

```
try:
    You do your operations here;
    ......................
except Exception:
    If there is Exception, then execute this block.
    ......................
```

2. 範例：使用 try - except 對 ZeroDivisionError 異常做處理。

```
pyuser@deyu:~/zzz$ cat try2.py
#!/usr/bin/python

def div(x, y):
    try:
        return x/y
    except ZeroDivisionError:
        print("除數不能爲 0")
        return

print("6/2=" , div(6,2))
print("3/0=" , div(3,0))
print("10/2=" , div(10,2))
```

3. 執行結果：當除數爲 0 時，輸出「除數不能爲 0」，其中 None 爲回傳值。

```
pyuser@deyu:~/zzz$ ./try2.py
6/2= 3.0
```

```
除數不能爲 0
3/0= None
10/2= 5.0
```

19.4 try-except-else

1. expect 語法：

   (a) except: 不接任何異常屬性 exceptions，表示所有 exceptions 都符合。

   (b) except Expection1: 接異常屬性 Exception1，表示符合 Exceptions1。

   (c) except(Exception1[, Exception2[,...ExceptionN]]]): 接多個異常屬性，表示符合列出的 Exceptions1, Exceptions2, ...。

2. try - except - else 語法如下，當 try 執行後，符合 Exception 屬性，執行 expcet 區塊程式，其餘執行 else 區塊程式：

```
try:
    You do your operations here;
    ......................
except Exception:
    If there is Exception, then execute this block.
else:
    If there is no exception then execute this block.
```

3. 範例：讀取命令指定的檔案，無法讀取時輸出Error: can't read file filename；成功讀取則列印檔案的字數。

```
pyuser@deyu:~/zzz$ cat try3.py
#!/usr/bin/python
import sys

try:
    fo = open(sys.argv[1], "r")
    data = fo.read()
except IOError:
    print("Error: can\'t read file", sys.argv[1])
else:
    wordlist = data.split()
    print("Word count for", sys.argv[1], "=", len(wordlist))
    fo.close()
```

4. 執行結果：參數 /etc/abc，因檔案不存在，回應無法讀取檔案。

```
pyuser@deyu:~/zzz$ ./try3.py /etc/abc
Error: can't read file /etc/abc
```

5. 執行結果：參數 /etc/issue，回應共有 6 個字。

```
pyuser@deyu:~/zzz$ ./try3.py /etc/issue
Word count for /etc/issue = 6
```

6. 查看檔案 /etc/issue。

```
pyuser@deyu:~/zzz$ cat /etc/issue
\S
Kernel \r on an \m
```

19.5 try-finally

1. finally 關鍵字放在 try 後，中間可以有 except 及 else，不論有沒有異常發生，一定會執行 finally 區塊的程式碼。

```
try:
    You do your operations here;
    ......................
except Exception:
    If there is Exception, then execute this block.
else:
    If there is no exception then execute this block.
finally:
    This would always be executed.
    ......................
```

2. 範例：無論有無異常發生，最後一定列印「程式結束」。

```
pyuser@deyu:~/zzz$ cat try4.py
#!/usr/bin/python
import sys

try:
    fo = open(sys.argv[1], "r")
    data = fo.read()
```

```
except IOError:
    print("Error: can\'t read file", sys.argv[1])
else:
    wordlist = data.split()
    print("Word count for", sys.argv[1], "=", len(wordlist))
    fo.close()
finally:
    print("程式結束")
```

3. 執行結果：參數爲不存在的檔案 /etc/cde，最後都印出「程式結束」。

```
pyuser@deyu:~/zzz$ ./try4.py /etc/cde
Error: can't read file /etc/cde
程式結束
```

4. 執行結果：參數爲可讀取的檔案 /etc/resolv.conf，最後也印出「程式結束」。

```
pyuser@deyu:~/zzz$ ./try4.py /etc/resolv.conf
Word count for /etc/resolv.conf = 6
程式結束
```

19.6 Exception 參數

1. 要取得異常發生之參數，可使用以下語法：

```
try:
    You do your operations here;
    ......................
# 舊語法
except ExceptionType, Argument:
# 新語法
except ExceptionType as Argument:
    You can print value of Argument here...
```

2. 範例：列印異常屬性 IOError 的編號，參數變數設爲 e。

```
pyuser@deyu:~/zzz$ cat try5.py
#!/usr/bin/python
import sys

try:
```

```
6     fo = open(sys.argv[1], "r")
      data = fo.read()
8 except IOError as e:
      print("Error: can\'t read file", sys.argv[1])
10    print("Error number =", e.errno)
  else:
12    wordlist = data.split()
      print("Word count for", sys.argv[1], "=", len(wordlist))
14    fo.close()
  finally:
16    print("程式結束")
```

3. 執行結果：IOError 的錯誤碼爲 2。

```
pyuser@deyu:~/zzz$ ./try5.py /etc/qwe
2 Error: can't read file /etc/qwe
  Error number = 2
4 程式結束
```

19.7 raise 觸發異常

1. 使用以下 raise 語法，自行觸發異常。

   (a) Exception：異常的類型，例如：NameError。

   (b) args：自行提供的異常參數。

   (c) trackback：如果存在，跟踪異常對象。

```
raise [Exception [, args [, traceback]]]
```

2. 範例：一個異常可以是一個字串、類別 (class)，python 內建的異常大多是類別。觸發異常後，後面的程式碼就不會再執行。

```
1 def functionName( level ):
      if level < 1:
3         raise Exception("Invalid level!", level)
```

3. 實例：定義一個異常函式 myerr，傳入的參數 level 小於 1，就會觸發異常 Exception，參數是”Invalid level!”。

```
pyuser@deyu:~/zzz$ vi try6.py
pyuser@deyu:~/zzz$ cat try6.py
1 Invalid level!
pyuser@ip123:~/10927001$ cat try6.py
#!/usr/bin/python
def myerr( level ):
    if level < 1:
        raise Exception("Invalid level!")

try:
    myerr(0)
except Exception as err:
    print(1,err)
else:
    print(2)
```

4. 執行結果：呼叫 myerr 時傳入 0，觸發異常，輸出自訂的異常"Invalid level!"。

```
pyuser@deyu:~/zzz$ ./try6.py
1 Invalid level!
```

19.8 自定義異常類型

1. 透過創建一個異常類別，用户可自行定義異常。異常應該是典型的繼承自 Exception 的類別，以下語法 SomeCustomException 爲自定義的異常名稱，Exception 可以指定使用 python 內建的任何「標準異常」。

```
class SomeCustomException(Exception):
    pass
```

2. 範例：自定義一個 NotPositiveError 異常類別，繼承標準異常 UserWaring。當判斷數字小於等於 0 時，觸發此自定義異常。

```
pyuser@deyu:~/zzz$ vi try7.py
pyuser@deyu:~/zzz$ cat try7.py
#!/usr/bin/python

class NotPositiveError(UserWarning):
    pass
```

```
8 string = input("輸入一個正整數： ")
try:
10     number = int(string)
    if number <= 0:
12         raise NotPositiveError
except ValueError:
14     print("這不是一個數字")
except NotPositiveError:
16     print("這不是一個大於 0 的整數")
```

3. 執行結果：輸入正整數 23，沒有觸發異常；輸入 0 ，觸發 NotPositiveError 異常；輸入 abc 觸發 ValueError 異常。

```
pyuser@deyu:~/zzz$ ./try7.py
2 輸入一個正整數： 23
pyuser@deyu:~/zzz$ ./try7.py
4 輸入一個正整數： 0
這不是一個大於 0 的整數
6 pyuser@deyu:~/zzz$ ./try7.py
輸入一個正整數： abc
8 這不是一個數字
```

19.9 實機練習題

1. 前置作業
   (a) 在家目錄下建立 zzz 目錄。
   (b) 切換工作目錄到 zzz，在 zzz 目錄下寫 python 程式：
2. try-except-else1.py：
   (a) 使用 sys 模組，讓執行 try-except-else1.py 時，可以使用參數指定字典的 key。
   (b) 建立字典 `{'a':1,'b':2,'c':3,'d':4}`
   (c) 使用 try-except-else 對讀 key 異常做處理。
   (d) 當 key 不存在時，印出「key 不存在」，key 爲指定的參數。
   (e) key 存在則從字典中刪除這個 key-value，並印出刪除後的字典。
3. try-except-else2.py：

(a) 使用 sys 模組，讓執行 try-except-else2.py 時，可以指定讀取的檔案，假設檔名 file。

(b) 使用 try-except-else 對讀檔異常做處理。

(c) 試著讀取 file

(d) 無法讀取檔案 file 時，開啓檔案 file，寫入「新建立的檔案 file」，file 後面要換行。

(e) else 印出「檔案 file 中共有 nu 個字」，nu 爲檔案中以空白隔開的字詞數。

(f) 關閉檔案。

4. try-except-finally1.py：

(a) 使用 sys 模組，讓執行 try-except-finally1.py 時，可以指定讀取年齡，假設是 age。

(b) 使用以下自定義異常類別 NotPositiveError，類別中定義 `__init__` member，提供錯誤內容"Not a positive number"。

```
class NotPositiveError(UserWarning):
    def __init__(self,err='Not a positive number'):
        Exception.__init__(self,err)
```

(c) 如果輸入的數字是負數或 0，則使用 raise 觸發此異常，印出「你輸入的年齡是負數或 0 xxxx」，其中 xxxx 是定義的錯誤內容。

(d) 非數字用標準異常 ValueError 排除異常，印出「你輸入的年齡不是數字 xxxx」，其中 xxxx 是定義的錯誤內容。

(e) 沒有輸入參數用標準異常 IndexError 排除異常，印出「你沒輸入年齡 xxxx」，其中 xxxx 是定義的錯誤內容。

(f) else 印出「你的年齡是 age」，age 是輸入的年齡。

(g) finally 印出「程式結束」。

Chapter 20

網路爬蟲

20.1 前言

1. 平常用瀏覽器與網站溝通，網路爬蟲 (Crawler) 則是寫程式利用 HTTP Request 抓取網路資料的技術。

2. 何時需要網路爬蟲？例如：

   (a) 自動取得網站最新消息。

   (b) 自動從不同網站搜集類似資訊。

   (c) 自動從網站複製貼上大筆資料。

20.2 簡單上網

1. webbrowser 模組，以 open 開啓網頁。執行 crawler.py 時會以預設瀏覽器開啓網頁。

```
[dywang@dywmac zzz]$ vim crawler.py
[dywang@dywmac zzz]$ cat crawler.py
#!/usr/bin/env python
# coding: utf-8

import webbrowser
webbrowser.open('http://dywang.csie.cyut.edu.tw/dywang/rhce7/index.html'
    )
[dywang@dywmac zzz]$ chmod +x crawler.py
[dywang@dywmac zzz]$ ./crawler.py
```

20.3 Google 地圖

1. 當在 google 地圖中輸入「朝陽科技大學」，進行查詢時網址如下，分成三部分

   (a) google 地圖網址 + 查詢地址

```
https://www.google.com.tw/maps/place/朝陽科技大學/
```

   (b) 經緯度

```
@24.069678,120.7126403,17z/
```

   (c) google 追縱記錄瀏覽者的資訊

```
data=!3m1!4b1!4m5!3m4!1s0x346924aa5ff8ce27:
0x7c8496caccaa905!8m2!3d24.069678!4d120.714829?hl=zh-TW
```

2. 修改 crawler.py，使用 raw_input 函式，在執行時輸入要查詢的地址，即可進行 google 地圖查詢。

```
[dywang@dywmac zzz]$ ./crawler.py
請輸入查詢地址： 朝陽科技大學
[dywang@dywmac zzz]$ cat crawler.py
#!/usr/bin/env python
# coding: utf-8

import webbrowser
address = raw_input("請輸入查詢地址： ")
url = 'https://www.google.com.tw/maps/place/'
#webbrowser.open('http://dywang.csie.cyut.edu.tw/dywang/rhce7/index.html
    ')
webbrowser.open(url + address)
```

20.4 requests 模組

1. resquests.get 取得網頁的原始檔案，型態是 Response 物件。

```
[dywang@dywmac zzz]$ vim crawler1.py
[dywang@dywmac zzz]$ cat crawler1.py
#!/usr/bin/env python
```

```
# coding: utf-8

import requests
url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce7/'
htmlfile = requests.get(url)
print(type(htmlfile))
[dywang@dywmac zzz]$ ./crawler1.py
<class 'requests.models.Response'>
```

2. Response 物件屬性 status_code 代表是否成功取得網頁，text 爲網頁內容。因爲範例網頁有中文，所以必須使用 sys.setdefaultencoding 設定預設編碼爲 utf-8，才能將網頁原始碼導向。

```
[dywang@dywmac zzz]$ vim crawler1.py
[dywang@dywmac zzz]$ cat crawler1.py
#!/usr/bin/env python
# coding: utf-8

import sys
reload(sys)
sys.setdefaultencoding( "utf-8" )

import requests
url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce7/'
htmlfile = requests.get(url)
if htmlfile.status_code == requests.codes.ok:
    print("Obtained web text successfully")
else:
    print("Failed to get web text")
print(htmlfile.text)
```

3. 執行程式，將網頁原始碼導向到 crawler1.html，使用 firefox 開啓 crawler1.html 就可看到網頁內容。

```
[dywang@dywmac zzz]$ ./crawler1.py > crawler1.html
```

20.5 網頁存取失敗

1. Response 物件 raise_for_status() 成員函式，可以對網址錯誤狀況產生異常處理，下例列出網頁存取失敗訊息代碼及內容。

```
[dywang@dywmac zzz]$ cat crawler2.py
```

```
#!/usr/bin/env python
# coding: utf-8

import sys
reload(sys)
sys.setdefaultencoding( "utf-8" )

import requests
url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce8/'
htmlfile = requests.get(url)
try:
    htmlfile.raise_for_status()
    print("Connected successfully")
except Exception as err:
    print("Failed: %s" % err)
```

2. 執行程式，錯誤代碼 404。

```
[dywang@dywmac zzz]$ ./crawler2.py
Failed: 404 Client Error: Not Found for url: http://dywang.csie.cyut.edu
    .tw/dywang/rhce8/
```

20.6 偽裝瀏覽器

1. 設定 headers 的 user-agent，使用 requests.get 時多加 headers 參數。

```
[dywang@dywmac zzz]$ cat crawler3.py
#!/usr/bin/env python
# coding: utf-8

import sys
reload(sys)
sys.setdefaultencoding( "utf-8" )

import requests

url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce7/'

headers = {
    'User-Agent': 'Mozilla/5.0',
    'From': 'dywang@csie.cyut.edu.tw'
}
htmlfile = requests.get(url, headers=headers)
try:
    htmlfile.raise_for_status()
    print("Connected successfully")
except Exception as err:
```

```
22      print("Failed: %s" % err)
```

2. 執行程式，成功連線。

```
[dywang@dywmac zzz]$ ./crawler3.py
Connected successfully
```

20.7 儲存網頁

1. 之前使用輸出導向，將網頁內容存檔，現在直接在程式中以 Response 物件的 iter_content() 存檔，每次儲存 5120 Bytes。

```
[dywang@dywmac zzz]$ vim crawler4.py
[dywang@dywmac zzz]$ cat crawler4.py
#!/usr/bin/env python
# coding: utf-8

import sys
reload(sys)
sys.setdefaultencoding( "utf-8" )

import requests

url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce7/'

headers = {
    'User-Agent': 'Mozilla/5.0',
    'From': 'dywang@csie.cyut.edu.tw'
}
htmlfile = requests.get(url, headers=headers)
try:
    htmlfile.raise_for_status()
    print("Connected successfully")
except Exception as err:
    print("Failed: %s" % err)

of = 'crawler4.txt'
with open(of, 'wb') as outfile:
    for ram in htmlfile.iter_content(5120):
        outfile.write(ram)
        print(len(ram))
    print("save to %s" % of)
```

2. 執行程式，儲存迴圈第一次儲存 5120 bytes，第二次儲存 4889 bytes。

```
[dywang@dywmac zzz]$ ./crawler4.py
Connected successfully
5120
4889
save to crawler4.txt
```

20.8 BeautifulSoup 安裝啓用

1. CentOS 6 的 python 2.6 無法直接以 pip install beautifulsoup4 安裝，先下載 beautifualsoup 套件。，解壓縮後將 bs 目錄複製到 python2.6 的套件目錄中。

```
[root@dywmac ~]# cd /usr/src/
[root@dywmac src]# wget https://www.crummy.com/software/BeautifulSoup/
    bs4/\
download/4.0/beautifulsoup4-4.1.0.tar.gz
[root@dywmac src]# tar zxvf beautifulsoup4-4.1.0.tar.gz
[root@dywmac ~]# cp -a beautifulsoup4-4.1.0/bs4 /usr/lib/python2.6/site-
    packages/
```

2. 導入 bs4 模組，使用 lxml 方式解析 HTML 文件。

```
[dywang@dywmac zzz]$ cat crawler5.py
#!/usr/bin/env python
# coding: utf-8
import requests, bs4

url = 'http://dywang.csie.cyut.edu.tw/dywang/rhce7/'
htmlfile = requests.get(url)
soup = bs4.BeautifulSoup(htmlfile.text, 'lxml')
```

3. 執行程式，輸出型態是 bs4.BeautifulSoup 物件，表示成功。

```
[dywang@dywmac zzz]$ ./crawler5.py
<class 'bs4.BeautifulSoup'>
```

4. 下載測試網頁 node2.html。

```
[dywang@dywmac zzz]$ wget https://dywang.csie.cyut.edu.tw/dywang/
    pythonProgram/node2.html
```

5. 直接讀取 node2.html 網頁原始碼檔，列印類別型態及 title。

```
[dywang@dywmac zzz]$ cat crawler6.py
#!/usr/bin/env python
# coding: utf-8
import bs4

htmlfile = open('node2.html')
soup = bs4.BeautifulSoup(htmlfile, 'lxml')
print("class tyep: ", type(soup))
print("title: ", soup.title)
```

6. 執行程式，輸出型態是 bs4.BeautifulSoup 物件，抬頭是「認識 Python」。

```
[dywang@dywmac zzz]$ ./crawler6.py
('class tyep: ', <class 'bs4.BeautifulSoup'>)
('title: ', <title>認識 Python</title>)
```

20.9 bs4 find

1. find() 找 h1 標籤。

```
[dywang@dywmac zzz]$ cat crawler7.py
#!/usr/bin/env python
# coding: utf-8
import bs4

htmlfile = open('node2.html')
soup = bs4.BeautifulSoup(htmlfile, 'lxml')
tag = soup.find('h1')
print("tyep: ", type(tag))
print("tag: ", tag)
print("tag text: %s" % tag.text)
```

2. 執行程式，找到標籤 h1，內容爲「認識 Python」。

```
[dywang@dywmac zzz]$ ./crawler7.py
('tyep: ', <class 'bs4.element.Tag'>)
('tag: ', <h1><a name="SECTION00200000000000000000">
認識 Python</a>
```

```
5 </h1>)
tag text:
7 認識 Python
```

3. find_all() 找所有標籤 li，並列印內容。

```
1 [dywang@dywmac zzz]$ cat crawler8.py
#!/usr/bin/env python
3 # coding: utf-8
import bs4
5
htmlfile = open('node2.html')
7 soup = bs4.BeautifulSoup(htmlfile, 'lxml')
tag = soup.find_all('li')
9 print("tyep: ", type(tag))
print("tag: ", tag)
11 print("tag text: ")
for data in tag:
13     print(data.text)
```

4. 執行程式，找到所有標籤 li。

```
1 [dywang@dywmac zzz]$ ./crawler8.py
('tyep: ', <type 'list'>)
3 ('tag: ', [<li><a href="node3.html" name="tex2html184">簡介</a>
</li>, <li><a href="node4.html" name="tex2html185">安裝與執行</a>
5 </li>, <li><a href="node5.html" name="tex2html186">內縮語法</a>
</li>, <li><a href="node6.html" name="tex2html187">括號、引號、換行、註
    解</a>
7 </li>])
tag text:
9 簡介

11 安裝與執行

13 內縮語法

15 括號、引號、換行、註解
```

20.10 bs4 getText

1. tag 回傳是 list 串列時，可用 getText() 取得內容。

```
1 [dywang@dywmac zzz]$ cat crawler9.py
```

```
#!/usr/bin/env python
# coding: utf-8
import bs4

htmlfile = open('node2.html')
soup = bs4.BeautifulSoup(htmlfile, 'lxml')
tag = soup.find_all('li')
print("tyep: ", type(tag))
print("tag: ", tag)
print("tag text: ")
for data in tag:
    print(data.text)
print("tag gettext: ")
for i in range(len(tag)):
    print(tag[i].getText())
```

2. 執行程式，使用 text 或 list 串列的 getText() 結果都一樣。

```
[dywang@dywmac zzz]$ ./crawler9.py
('tyep: ', <type 'list'>)
('tag: ', [<li><a href="node3.html" name="tex2html184">簡介</a>
</li>, <li><a href="node4.html" name="tex2html185">安裝與執行</a>
</li>, <li><a href="node5.html" name="tex2html186">內縮語法</a>
</li>, <li><a href="node6.html" name="tex2html187">括號、引號、換行、註
    解</a>
</li>])
tag text:
簡介

安裝與執行

內縮語法

括號、引號、換行、註解

tag gettext:
簡介

安裝與執行

內縮語法

括號、引號、換行、註解
```

20.11 bs4 select

1. select() 可以選擇 CSS 標籤，例如選擇 img 標籤。

```
[dywang@dywmac zzz]$ cat crawler10.py
#!/usr/bin/env python
# coding: utf-8
import bs4

htmlfile = open('node2.html')
soup = bs4.BeautifulSoup(htmlfile, 'lxml')
tag = soup.select('img')
print("tyep: ", type(tag))
print("tag: ", tag)
print("tag text: ")
for i in range(len(tag)):
    print(tag[i].getText())
```

2. 執行程式，使用 list 串列的 getText() 輸出結果都空白。

```
[dywang@dywmac zzz]$ ./crawler10.py
('tyep: ', <type 'list'>)
('tag: ', [<img align="BOTTOM" alt="next" border="0" height="24" src="
    next.png" width="37"/>, <img align="BOTTOM" alt="up" border="0"
    height="24" src="up.png" width="26"/>, <img align="BOTTOM" alt="
    previous" border="0" height="24" src="prev.png" width="63"/>, <img
    align="BOTTOM" alt="contents" border="0" height="24" src="contents.
    png" width="65"/>])
tag text:
('', 0)
('', 1)
('', 2)
('', 3)
```

3. 程式最後一行改成列出 list 串列的 index 才有輸出。

```
    print(str(tag[i]))

[dywang@dywmac zzz]$ ./crawler10.py
('tyep: ', <type 'list'>)
('tag: ', [<img align="BOTTOM" alt="next" border="0" height="24" src="
    next.png" width="37"/>, <img align="BOTTOM" alt="up" border="0"
    height="24" src="up.png" width="26"/>, <img align="BOTTOM" alt="
    previous" border="0" height="24" src="prev.png" width="63"/>, <img
    align="BOTTOM" alt="contents" border="0" height="24" src="contents.
    png" width="65"/>])
tag text:
('<img align="BOTTOM" alt="next" border="0" height="24" src="next.png"
    width="37"/>', 0)
('<img align="BOTTOM" alt="up" border="0" height="24" src="up.png" width
    ="26"/>', 1)
('<img align="BOTTOM" alt="previous" border="0" height="24" src="prev.
    png" width="63"/>', 2)
```

```
10 ('<img align="BOTTOM" alt="contents" border="0" height="24" src="
       contents.png" width="65"/>', 3)
```

4. 程式最後一行改成列出 list 串列的 get('src') 可以找到所有圖檔。

```
    print(tag[i].get('src'))
2
[dywang@dywmac zzz]$ ./crawler10.py('tyep: ', <type 'list'>)
4 ('tag: ', [<img align="BOTTOM" alt="next" border="0" height="24" src="
       next.png" width="37"/>, <img align="BOTTOM" alt="up" border="0"
       height="24" src="up.png" width="26"/>, <img align="BOTTOM" alt="
       previous" border="0" height="24" src="prev.png" width="63"/>, <img
       align="BOTTOM" alt="contents" border="0" height="24" src="contents.
       png" width="65"/>])
tag text:
6 next.png
up.png
8 prev.png
contents.png
```

Chapter 21

PyGTK

21.1 PyGTK 簡介與開始

1. PyGTK 簡介

   (a) PyGTK 是 GNOME 專案的一部分，作者是著名的 GNOME 開發者 James Henstridge。

   (b) PyGTK 是一套用 Python 封裝的，用於 GTK+ 的 GUI 庫。

   (c) PyGTK 是在 LGPL 授權下的自由軟體，可以自由修改、散佈及研究。

   (d) PyGTK 跨平台，未經修改下可在 Linux，Windows，MacOS X 和其他平台上使用。

   (e) PyGTK 所開發的應用程式更已被選應用於每童一電腦（One Laptop Per Child，OLPC）的系統之上。

   (f) PyGTK 非常容易使用，是最受歡迎的 GTK+ 函式庫之一。

   (g) PyGTK 提供所有圖形界面元件及有用的程式設施來產生桌面應用程式。

2. 第一支程式

```
[dywang@deyu python]$ cat mypy.py
#!/usr/bin/python
# coding: utf-8

import gtk

class myPy:
    def __init__(self):
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 爲其參數"
        window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title('視窗抬頭')
        window.set_keep_above(True)
```

```
        window.show()
15
myPy()
17 gtk.main()
```

3. 類別相關聯結

   (a) gtk.Widget class

   (b) gtk.Window class

   (c) gtk Functions

   (d) python: Lambda Functions

21.2 PyGTK 除錯

1. import 的 gtk modules 找不到，錯誤訊息如下：

```
1 [dywang@dywhd2 zzz]$ python mypy1.py
7
3 'import site' failed; use -v for traceback
Traceback (most recent call last):
5   File "mypy1.py", line 5, in <module>
      import gtk
7 ImportError: No module named gtk
```

2. 查詢安裝套件 pygtk2 已安裝。

```
1 [root@dywhd2 ~]# rpm -qa | grep pygtk
pygtk2-libglade-2.16.0-3.el6.x86_64
3 pygtksourceview-2.8.0-1.el6.x86_64
pygtk2-2.16.0-3.el6.x86_64
```

3. 查詢 pygtk module 在目錄 /usr/lib64/python2.6/site-packages/ 下。

```
[root@dywhd2 ~]# find /usr/lib64/python2.6 -name pygtk*
2 /usr/lib64/python2.6/site-packages/pygtk.pyc
/usr/lib64/python2.6/site-packages/pygtk.py
4 /usr/lib64/python2.6/site-packages/pygtk.pth
/usr/lib64/python2.6/site-packages/pygtk.pyo
```

4. 執行 python 後進入互動式模式，import pygtk 模組不成功，顯示 sys.path 發覺/usr/lib64/python2.6/site-packages/ 目錄並不在其中。

```
[dywang@dywhd2 zzz]$ python
7
'import site' failed; use -v for traceback
Python 2.6.5 (r265:79063, Nov 12 2010, 00:52:45)
[GCC 4.4.4 20100525 (Red Hat 4.4.4-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygtk; pygtk.require('2.0')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named pygtk
>>> import sys
>>> print sys.path
['', '', '/usr/local/google_appengine',
'/usr/local/google_appengine/lib/django_1.4',
'/usr/lib64/python26.zip',
'/usr/lib64/python2.6/',
'/usr/lib64/python2.6/plat-linux2',
'/usr/lib64/python2.6/lib-tk',
'/usr/lib64/python2.6/lib-old',
'/usr/lib64/python2.6/lib-dynload']
```

5. 解決方式：編輯 /etc/bashrc，將 PYTHONPATH 變數增加 /usr/lib64/python2.6/site-packages/ 目錄，讓任何用户登入後就生效。

```
[root@dywhd2 ~]# vim /etc/bashrc
[root@dywhd2 ~]# tail -n1 /etc/bashrc
export PYTHONPATH="/usr/lib64/python2.6/site-packages:${PYTHONPATH}"
```

6. 重新取得 bash shell 後，查詢 PYTHONPATH 變數已包含 /usr/lib64/python2.6/site-packages/ 目錄。

```
[dywang@dywhd2 ~]$ echo $PYTHONPATH
/usr/lib64/python2.6/site-packages::/usr/local/google_appengine:/usr/
    local/google_appengine/lib/django_1.4
```

7. 再執行 python 進入互動式模式，import pygtk 或 import gtk 都成功取得 module，不再出現錯誤訊息。

```
[dywang@dywhd2 ~]$ python
Python 2.6.5 (r265:79063, Nov 12 2010, 00:52:45)
[GCC 4.4.4 20100525 (Red Hat 4.4.4-5)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygtk
>>> import gtk
>>>
```

21.3 Buttons 與 Layout

1. Example

```
#!/usr/bin/python
# coding: utf-8

#import pygtk
import gtk

class myPy:
    def __init__(self):
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 爲其參數"
        window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title('視窗抬頭')
        window.set_keep_above(True)

        vbox1 = gtk.VBox(False, 2)
        "homogeneous=True 則內含物件平均分佈，spaceing=5 間隔5"
        hbox1 = gtk.HBox(homogeneous=True,spacing=5)
        hbox2 = gtk.HBox(homogeneous=True,spacing=5)
        hbox3 = gtk.HBox(homogeneous=True,spacing=5)
        yes = gtk.Button("yes")
        no = gtk.Button("no")
        close = gtk.Button(None,gtk.STOCK_CLOSE)
        hbox1.add(yes)
        hbox1.add(no)
        hbox2.add(close)
        vbox1.pack_start(hbox1, True, True, 0)
        vbox1.pack_start(hbox2, True, True, 0)
        vbox1.pack_start(hbox3, True, True, 0)
        window.add(vbox1)

        window.show_all()

myPy()
gtk.main()
```

2. 類別相關聯結

   (a) gtk.Vbox class

(b) gtk.Box class

(c) gtk.Hbox class

(d) gtk.Button class

(e) Stock Items

21.4 Signals, Events and Functions

1. Example1

```
#!/usr/bin/python
# coding: utf-8

#import pygtk
import gtk

class myPy:
    def callback(self, widget, data):
        if data == "yes":
            print "yes"
        elif data == "no":
            print "no"
        else:
            print "yes|no"

    def __init__(self):
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 爲其參數"
        window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title('視窗抬頭')
        window.set_keep_above(True)

        vbox1 = gtk.VBox(False, 2)
        hbox1 = gtk.HBox(homogeneous=True,spacing=5)
        hbox2 = gtk.HBox(homogeneous=True,spacing=5)
        hbox3 = gtk.HBox(homogeneous=True,spacing=5)
        yes = gtk.Button("yes")
        no = gtk.Button("no")
        close = gtk.Button(None,gtk.STOCK_CLOSE)
        close.connect("clicked", lambda x: gtk.main_quit())
        yes.connect("clicked", self.callback, 'yes')
        no.connect("clicked", self.callback, 'no')
        hbox1.add(yes)
        hbox1.add(no)
        hbox2.add(close)
        vbox1.pack_start(hbox1, True, True, 0)
        vbox1.pack_start(hbox2, True, True, 0)
        vbox1.pack_start(hbox3, True, True, 0)
        window.add(vbox1)

```

```
        window.show_all()
42
myPy()
44 gtk.main()
```

2. Example2

```
#!/usr/bin/python
2 # coding: utf-8

4 #import pygtk
import gtk
6 import subprocess

8 icondir="/usr/share/icons/gnome/16x16"
yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
10 no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

12 def img_label_box(parent, img_file, label):
    box = gtk.HBox(False, 0)
14  box.set_border_width(2)
    image = gtk.Image()
16  image.set_from_file(img_file)
    label = gtk.Label(label)
18  box.pack_start(image, False, False, 5)
    box.pack_start(label, False, False, 5)
20  image.show()
    label.show()
22  return box

24 class myPy:
    def callback(self, widget, data):
26      if data == "yes":
            subprocess.Popen("gnome-terminal",shell=True)
28      elif data == "no":
            print "no"
30      else:
            print "yes|no"
32
    def __init__(self):
34      window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 爲其參數"
36      window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title('視窗抬頭')
38      window.set_keep_above(True)

40      vbox1 = gtk.VBox(False, 2)
        hbox1 = gtk.HBox(homogeneous=True,spacing=5)
42      hbox2 = gtk.HBox(homogeneous=True,spacing=5)
        hbox3 = gtk.HBox(homogeneous=True,spacing=5)
44      yes = gtk.Button()
        no = gtk.Button()
```

```
46          close = gtk.Button(None,gtk.STOCK_CLOSE)
            close.set_use_stock(True)
48          close.connect("clicked", lambda x: gtk.main_quit())
            yes.connect("clicked", self.callback, 'yes')
50          no.connect("clicked", self.callback, 'no')
            box_yes = img_label_box(window, yes_icon, "YES")
52          box_no = img_label_box(window, no_icon, "NO")
            yes.add(box_yes)
54          no.add(box_no)
            hbox1.add(yes)
56          hbox1.add(no)
            hbox2.add(close)
58          vbox1.pack_start(hbox1, True, True, 0)
            vbox1.pack_start(hbox2, True, True, 0)
60          vbox1.pack_start(hbox3, True, True, 0)
            window.add(vbox1)
62
            window.show_all()
64
myPy()
66  gtk.main()
```

3. 類別相關聯結

   (a) gtk.Button class

21.5 Label and Entry

1. Example1: Label

```
    #!/usr/bin/python
2   # coding: utf-8

4   #import pygtk
    import gtk
6   import subprocess

8   icondir="/usr/share/icons/gnome/16x16"
    yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
10  no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

12  def img_label_box(parent, img_file, label):
        box = gtk.HBox(False, 0)
14      box.set_border_width(2)
        image = gtk.Image()
16      image.set_from_file(img_file)
        label = gtk.Label(label)
18      box.pack_start(image, False, False, 5)
        box.pack_start(label, False, False, 5)
```

```
20      image.show()
        label.show()
22      return box

24  class myPy:
        def callback(self, widget, data):
26          if data == "yes":
                subprocess.Popen("gnome-terminal",shell=True)
28          elif data == "no":
                self.label.set_text("no")
30          else:
                print "yes|no"
32
        def __init__(self):
34          window = gtk.Window(gtk.WINDOW_TOPLEVEL)
            "lambda 是匿名函式 x 爲其參數"
36          window.connect("destroy", lambda x: gtk.main_quit())
            window.set_title('視窗抬頭')
38          window.set_keep_above(True)

40          vbox1 = gtk.VBox(False, 2)
            hbox1 = gtk.HBox(homogeneous=True,spacing=5)
42          hbox2 = gtk.HBox(homogeneous=True,spacing=5)
            hbox3 = gtk.HBox(homogeneous=True,spacing=5)
44          yes = gtk.Button()
            no = gtk.Button()
46          close = gtk.Button(None,gtk.STOCK_CLOSE)
            close.set_use_stock(True)
48          close.connect("clicked", lambda x: gtk.main_quit())
            yes.connect("clicked", self.callback, 'yes')
50          no.connect("clicked", self.callback, 'no')
            box_yes = img_label_box(window, yes_icon, "YES")
52          box_no = img_label_box(window, no_icon, "NO")
            yes.add(box_yes)
54          no.add(box_no)
            self.label=gtk.Label("Yes|No")
56          hbox1.add(yes)
            hbox1.add(no)
58          hbox2.add(close)
            hbox3.add(self.label)
60          vbox1.pack_start(hbox1, True, True, 0)
            vbox1.pack_start(hbox2, True, True, 0)
62          vbox1.pack_start(hbox3, True, True, 0)
            window.add(vbox1)
64
            window.show_all()
66
    myPy()
68  gtk.main()
```

2. Example2: Entry

```
#!/usr/bin/python
# coding: utf-8

#import pygtk
import gtk
import subprocess

icondir="/usr/share/icons/gnome/16x16"
yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

def img_label_box(parent, img_file, label):
    box = gtk.HBox(False, 0)
    box.set_border_width(2)
    image = gtk.Image()
    image.set_from_file(img_file)
    label = gtk.Label(label)
    box.pack_start(image, False, False, 5)
    box.pack_start(label, False, False, 5)
    image.show()
    label.show()
    return box

class myPy:
    def callback(self, widget, data):
        if data == "yes":
            subprocess.Popen("gnome-terminal",shell=True)
        elif data == "no":
            self.label.set_text(data)
        else:
            self.label.set_text("Only yes|no")

    def entryChange(self,widget):
        if self.entry.get_text() == "yes":
            subprocess.Popen("gnome-terminal",shell=True)
            self.label.set_text("yes")
        elif self.entry.get_text() == "no":
            self.label.set_text("no")
        else:
            self.label.set_text("Only yes|no")

    def __init__(self):
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 爲其參數"
        window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title('視窗抬頭')
        window.set_keep_above(True)

        vbox1 = gtk.VBox(False, 2)
        hbox1 = gtk.HBox(homogeneous=True,spacing=5)
        hbox2 = gtk.HBox(homogeneous=True,spacing=5)
        hbox3 = gtk.HBox(homogeneous=True,spacing=5)
        yes = gtk.Button()
        no = gtk.Button()
```

```
        close = gtk.Button(None,gtk.STOCK_CLOSE)
56      close.set_use_stock(True)
        self.label=gtk.Label("Yes|No")
58      self.entry=gtk.Entry()
        close.connect("clicked", lambda x: gtk.main_quit())
60      yes.connect("clicked", self.callback, 'yes')
        no.connect("clicked", self.callback, 'no')
62      self.entry.connect("changed", self.entryChange)
        box_yes = img_label_box(window, yes_icon, "YES")
64      box_no = img_label_box(window, no_icon, "NO")
        yes.add(box_yes)
66      no.add(box_no)
        hbox1.add(yes)
68      hbox1.add(no)
        hbox2.add(close)
70      hbox3.add(self.label)
        hbox3.add(self.entry)
72      vbox1.pack_start(hbox1, True, True, 0)
        vbox1.pack_start(hbox2, True, True, 0)
74      vbox1.pack_start(hbox3, True, True, 0)
        window.add(vbox1)
76
        window.show_all()
78
if __name__ == "__main__":
80      print __name__
        myPy()
82      gtk.main()
else:
84      print __name__
```

3. 類別相關聯結

   (a) gtk.Label class

   (b) gtk.Entry class

21.6 實機練習題

21.6.1 練習一

1. 寫一 python 程式 mypygtk.py，上傳 moodle 作業，包含以下功能：

   (a) 視窗抬頭：學號及姓名 (中間空白隔開)

   (b) 水平盒子 Hbox

   (c) 垂直盒子 Vbox

(d) 按鈕 Button：至少一個按鈕執行某一程式 (自訂)，一個按鈕關閉視窗。

(e) Label

(f) Entry：輸入資料有三種以上變化，且 Label 隨著變化。

Chapter 22

Glade

22.1 簡介

1. 什麼是 glade ？

   (a) Glade 是 Rapid application development (RAD) 工具，提供使用者容易的 GTK+ 工具箱及 GNOME 桌面環境。

   (b) Glade 設計好的使用者界面存成 XML 檔，供程式使用。

   (c) Glade 的 XML 檔可供無數的程式語言使用，包括 C, C++, C#, Vala, Java, Perl, Python 等。

   (d) Glade 是 GNU GPL License 授權的自由軟體。

2. glade3 安裝

```
1 [root@dywHome2 ~]# yum install glade3
```

3. 執行 glade designer

```
1 圖形界面
Applications->Programming->Glade Interface Designer
3 文字界面
[dywang@deyu glade]$ glade-3
```

4. Glade Interface Designer Demo

5. 撰寫 python file example1.py

```
#!/usr/bin/python
# coding: utf-8
import gtk
import gtk.glade
import subprocess
import sys

class Example1:
    def __init__(self):
        self.gladefile = "example1.glade"
        self.wTree = gtk.glade.XML(self.gladefile)
        dic = {
            "on_window_example1_destroy" :self.quit,
            "on_button1_clicked" : self.ok,
            "on_button2_clicked" : self.quit,
            "on_entry1_changed" : self.entrychanged
            }
        self.wTree.signal_autoconnect(dic)
        self.window = self.wTree.get_widget("window1")
        self.window.set_keep_above(True)
        self.window.show()

    def entrychanged(self,widget):
        if self.wTree.get_widget("entry1").get_text() == "show":
            subprocess.Popen("gnome-terminal",shell=True)
            self.wTree.get_widget("label1").set_text("Show terminal")
        else:
            self.wTree.get_widget("label1").set_text("Only show")


    def ok(self,widget):
        subprocess.Popen("gnome-terminal",shell=True)
        self.wTree.get_widget("label1").set_text("Open terminal")

    def quit(self,widget):
        sys.exit(0)

if __name__ == "__main__":
    Example1()
    gtk.main()
```

6. 執行 python

```
[dywang@deyu glade]$ python example1.py
```

22.2 Calculator Example

1. 下載計算機範例檔
   calculator.glade
   calculatorglade.py

2. 下載計算機範例檔（增加開根號）
   calculator1.glade
   calculator1.py

3. 以 glade-3 開啓 calculator.glade

```
1. table1 Rows: 5 -> 6
2. insert ok button in table1 (0,5)
3. rename ok button Label and Name to Sqrt
4. set Sqrt button clicked Signals to on_Sqrt_clicked
5. vim calculatorglade.py

# coding: utf-8
import sys
import math # 匯入數學模組
try:
    import pygtk
    pygtk.require('2.0')
except:
    pass
.... omission ....

class Calculator:
.... omission ....
            "on_Add_clicked" : self.displayAdd,
            "on_Sqrt_clicked" : self.displaySqrt, # 定義sqrt按鈕接受函式
.... omission ....
        if operator == 'Sqrt':
            self.firstOperand = self.wTree.get_widget("displayText").
                get_text()
            result = math.sqrt(float(self.firstOperand))
            self.wTree.get_widget("displayText").set_text(str(result))
.... omission ....

    def displaySqrt(self,widget): #定義displaySqrt函式
        self.compute("Sqrt")
```

4. 執行

```
[dywang@deyu glade]$ python calculatorglade.py
```

5. 相關聯結

   (a) GTK+ and Glade3 GUI Programming Tutorial

   (b) 用 Libglade 快速開發 GTK+ 視窗程式

   (c) Calculator Program using Python and Glade

   (d) Python 教學

22.3 實機練習題

22.3.1 練習一

1. 下載計算機範例檔（增加開根號）
   calculator1.glade
   calculator1.py

2. 修改範例檔，至少包含以下要求：

   (a) 增加兩個 Label 分別顯示學號及姓名

   (b) 除了開根號外，至少增加一個工程運算函式。

   (c) 其他創意額外加分。