Week1:

Class & Object(工廠/小小兵)- 物件導向

左邊稱為class(類別/工廠),右邊稱為 特色Attributes/絕招Methods

整個左邊承襲下來的關係叫做 <Class Hierarchy >

Animal –Food

Mammals –Warm blood

Primates –5 fingers

Humans –brian

Humans生產出歐巴馬跟金正恩(小小兵/工人/Object/instance)

Super class

Inheritance

Sub class

Instance/Object/小小兵

Class/工廠/St Upper letter (工廠的話，第一個字一定是大寫!!)

attributes(instance variables)/特色

methods/絕招

GObject-這是一個class，所以第一個字大寫

GFillableObject可填滿的 GLabel文字 GLine

GOval圓形/橢圓 GRect 方形 GArc 弧形 GPolygon 多邊形

如果要import的話:

from campy.graphics.gobojects import GOval, GRect, GLabel, Gline

呼叫一個名為 constructor的部門做小小兵

face = GOval(大小,位置)

r\_eye = GOval(大小,位置)

l\_eye = GOval(大小,位置)

mouth = GRect(大小,位置)

圖片:Convas 🡪 graphics

文字:Console 🡪 text

圖片要用用GWindow叫出來

from campy.graphics.gwindow import GWindow

window = GWindow(大小,名稱)

window.add(face)

window.add(mouth)

Calling constructor

face = GOval (200,250,x=350, y=200)

Keyword arguments  
(可有可無-原本就有Default values)

Positional arguments  
(一定要有

face.filled = True 🡪Attribute (instance variables-ivar) 沒括弧-特色

window.add(face) 🡪 Method 有括弧-絕招

from campy.graphics.gobjects import GOval

campy是一個資料夾

graphics 是次資料夾

gobjects 的這個Python檔案(檔名一定是小寫)

GOval 是gobjects檔案裏面的Class(Class的第一個字一定是大寫)

from campy.graphics.gwindow import GWindow

campy是一個資料夾

graphics 是次資料夾

gobjects 的這個檔案(檔名一定是小寫)

GWindow 是gwindow檔案裏面的Class(Class的第一個字一定是大寫)

Animation (動畫)

While True:

update(Ex: Rect.move(vx,0) 前面是移動x, 後面是移動y)

check(條件: if rect x<=0,……)

pause (讓他慢慢走，單位是毫秒miliseconds)

from campy.gui.events.timer import pause

<GObject Class Hierarchy>

* GObject
  + GLine
  + GLabel
  + GFillableObject
    - GOval
    - GRect
    - GPloygon
    - GArc

GPolygon (多邊形)

triangle = Gpolygon()

triangle.add\_vertex((100,300))

triangle.add\_vertex((300,300))

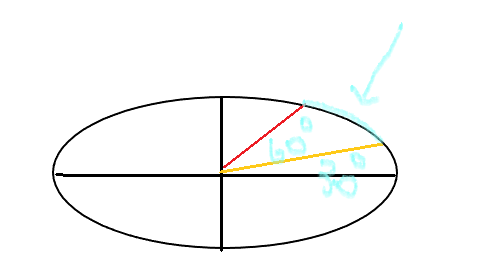
triangle.add\_vertex((150,200))

GArc(弧形)

arc= GArc(width, height, Start, Sweep)

寬/高/從幾度掃到幾度

先畫出一個Goval之後



Mouse Event

**Event-driven program**

from campy.gui.event.mouse import onmouseclicked, onmousemoved, onmousedragged

Asynchronous-突如其來的，mouseclick關不掉，你點了就會啟動

如果程式跑到一半，不小心又按到滑鼠，就又會先啟動mouseclick的function.

def main():

onmouseclicked(function) 在mouseclick當function要當數值的時候

不能寫function()

def function(mouse/event(隨便個定義名字)): 但是在這邊建立function的時候，一定要寫括弧滑鼠資訊。

mouse.x

mouse.y

Global Variable

1. lower case
2. above main()
3. bad style ☹ 除非想不到任何做法，不然非必要不要用

如何區分global variable / constant/ class

命名法則

ALL Upper 🡪 constant

第一個字母大寫 1st Upper 🡪 class

All lower 🡪 global variable

🡪local variable

GImage

from compy.graphics.gimage import GImage

def main():  
 image = GImage(‘檔名.jpg’) 要跟原本的地方same directory

window.add(image, 100,200)

如果get object不小心點到空白的地方  
則這個空白的地方叫做”None”

在物件導向的定義裡面，只能用is/is not，不能使用== 或 !=

Making Own Class

r1= Robot (183,60,color = “orange”)

r2= Robot (190,80,color = “red”)

r3= Robot (160,50)

=Robot( Weight =50, Height = 160)

Position arg🡪一定要 可以用Keyword 的方式

Methods

1. 因為 instance不同 而表現不同 🡪instance method

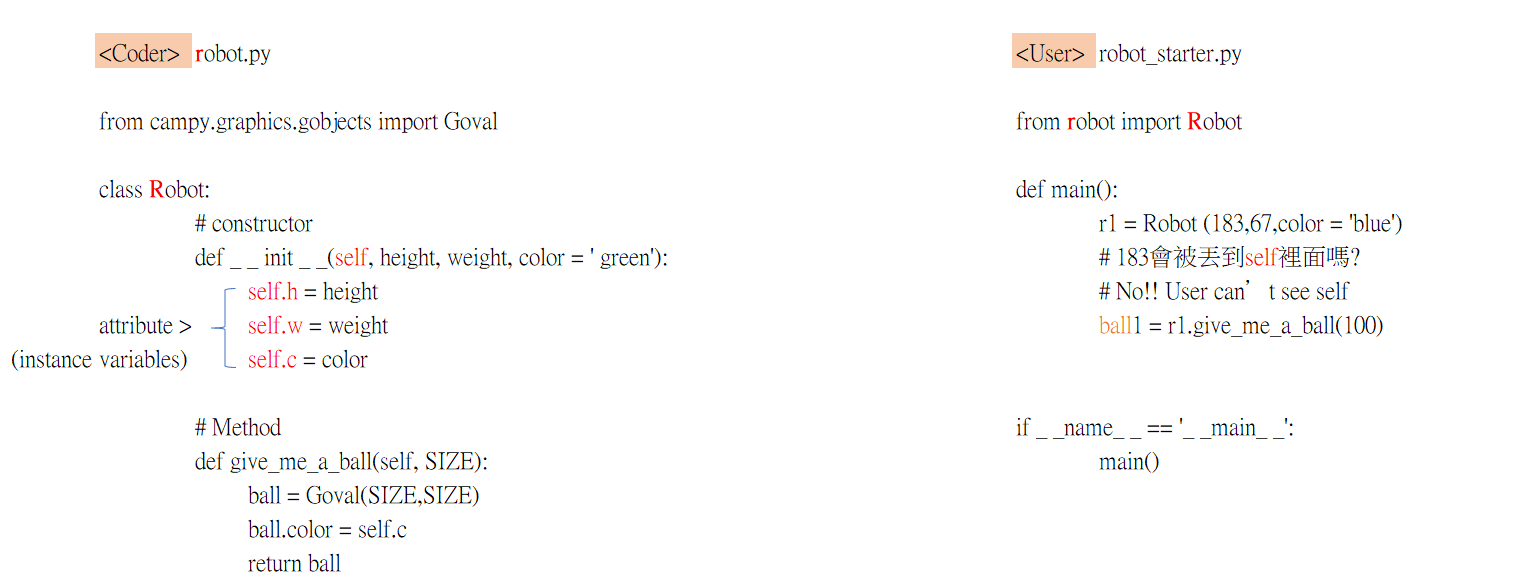
give me a ball(50)

1. 不會因為Instance不同而表現不同 🡪 static method

Say hi()

Week 2:

製造自己Robot的constructor跟Method



Python Prints n=4

1. Concatenation

print(“Score: “ +str(n))

#Score: 4

1. Comma Separated

print(“Score:“,n)

#Score: 4

逗號會自動加空格

1. Formatted String

(f-string)

print(f”Score: {n}”)

#Score: 4

Static Method

* Independent of instances (Not using self)
* Decorated with @staticmethod
* Can be use without an object/an instances

Robot.say\_hi()

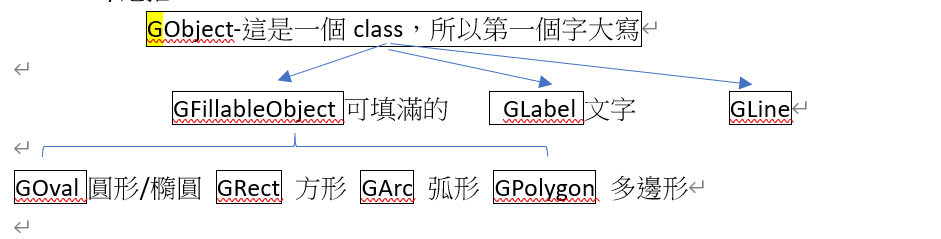
Ex:

img = SimpleImage(‘檔名.png’)

blank\_img = SimpleImage.blank(1000,500)

blank\_img = img.blank(1000,500)

Class Hierarchy class 承襲



Super class

Super class

Sub class

Sub class

import v.s from ..import

<import>

import random

roll = random,randrange(1,7)

import campy.graphics.gobjects

oval1= campy.graphics.gobjects.GOval(100,100)

oval2= campy.graphics.gobjects.GOval(200,200)

<from …import>

from random import randrange

roll = randrange(1,7)

from campy.graphics.gobjects import GOval

oval1= GOval(100,100)

oval2= GOval(200,200)

if \_ \_name\_ \_==”\_ \_main\_ \_”

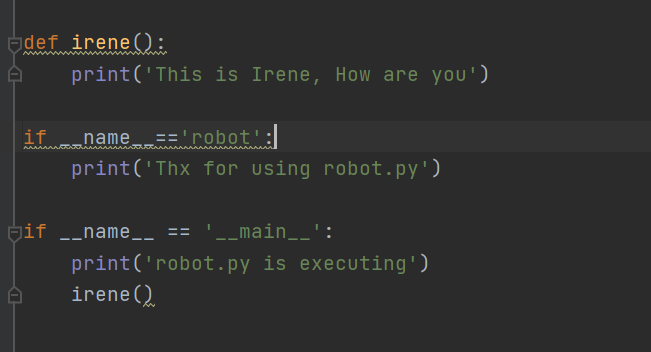
\_ \_name\_ \_ is a system variable, storing a str.

* If a file is executed =>\_ \_name\_ \_ stores “\_ \_main\_ \_”
* If a file is imported =>\_ \_name\_ \_ stores “filename”(robot)

如果執行有robot.py的話就print ‘Thx for using robot.py’

假設這個檔案要成為主要跑的程式檔(即按右鍵要modify要run這個.py的時候)

就print ‘ robot.py is executing’ 以及執行irene()底下的所有程式碼。



main() 不一定要叫main()也可以改名成irene()

Visibility of Instance Variables

<Public>

self.m

<Private>

self.\_ \_m

Always it private

Python underscores

<single Leading>\_var

class internal use (軟性private)-非強制性

but can be changed by users

<Double Leading>\_ \_var

strict internal use only (強制 不能使用)

<Single Tailing> var\_

when ‘var’ is used ( class\_=’SC101’)

<Double Leading & Tailing> \_ \_ var\_ \_

Python use only

Setter & Getter

When users need to change(setter)/obtain(getter) the information of the private instance variable.

好處:

1. User friendly.
2. Safe
3. Set only/Get only

If a:

Else:

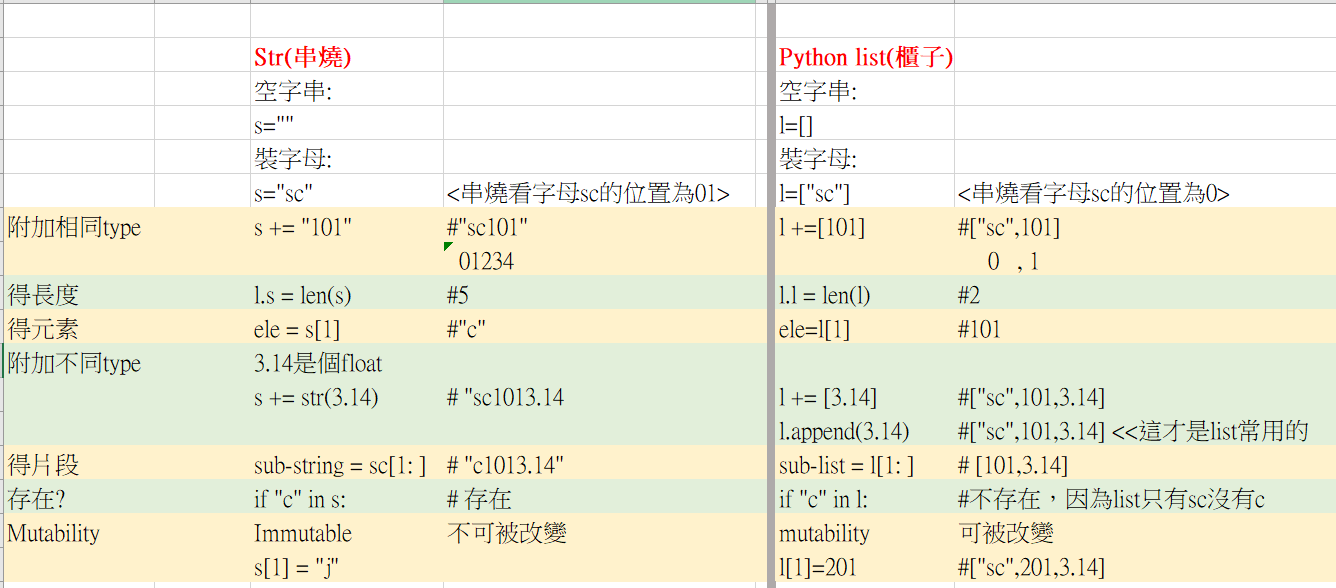
If not a:

else:

Python List

1. Ordered (Zero-indexed) 從0開始
2. Heterogeneous(store any data type)   
   能存多種形式

Homogeneous(store one data type)   
只能存一種形式



Stack & Queue

ADT🡪Abstract Data Type 一個資料型態的概念

<Stack>

1. Remove the element from the end- 從末端去拿元素
2. Last-In-First-Out (LIFO)
3. Add element 🡪 push(element), 在python list 用 “.append”
4. Get/ Remove element 🡪 pop(), 在python list 用 “.pop”

用python list 去做Stack:

l=[1,2,3]

l.append(4) #[1,2,3,4]

ele = l.pop() #4 (移除4), [1,2,3] (資料結構變成這個)

<Queue>

1. Remove the element from the front- 從前頭去拿元素
2. First-In-First-Out(FIFO)
3. Add element 🡪enqueue(element) , 在python list 用 “.append”
4. Get/Remove element 🡪 dequeue(), 在python list 用 “.pop(0)”

用python list 去做Queue:

queue=[1,2,3]

queue.append(4) #[1,2,3,4]

ele = queue.pop(0) #4 (移除1), [2,3,4] (資料結構變成這個)

Tokenization

s= ‘ I am Jerry Liao’

token就是用空格隔開的每個單字

l = s.split() #[“I”, “am”, “Jerry”, “Liao”] 這個是tsv檔

s= ‘ I,am,Jerry,Liao’

l = s.split(“,”) #[“I”, “am”, “Jerry”, “Liao”] 這個是csv檔

用什麼東西切開就是”delimiter”

如果要把l = [“I”, “am”, “Jerry”, “Liao”] 合起來:

“ “.join(l) 用空格切割

“,”.join(l) 用逗號切割

Import sys

sys.argv 儲存檔案被執行文字tokens

EX. py pick\_your\_pokemon.py bulbasaur

sys.argv 🡪 [‘pick\_your\_pokemon.py’,’bulbasaur’]

sys.argv[0] 永遠是檔名

Python Tuple



Return Mutiple?

def return\_3():

return 1,2,3

def main():

a,b,c = return\_3()

Swap

def swap(a,b):

c=a

a=b

b=c

但python可以這樣寫:  
b,a = a,b

Data Structures

1. list

* stack(LIFO)
* queue(FIFO)

1. tuple

* (1,3)/1,3
* a,b = b,a

Lucky Number

‘Jerry’🡪7

‘Cindy’🡪8

‘Becky’🡪20

‘Jewel’🡪3



移除’Jerry’可以做N次操作

加入’Jerry’加入只能一次做一次操作

Big Oh 計算複雜度

* Effiency
* Runtime
* Complexity

name\_lst.pop(0)

O(N)

name\_lst.append(‘Jay’)

O(1)



Total\_runtime = 2N^2 + 3N + 1

但電腦為了精簡，則big Oh = O(N^2)

EX: for i in range(N):

for j in range(i,N):

statement

big Oh?

A: O(N^2)

i == 0 🡪 j = N

i == 1 🡪 j = N-1

i == 2 🡪 j = N-2

.

.

.

i == N-1 🡪 j = 1

N+(N-1)+(N-2)….+3+2+1

(1+N)\*N/2

Python dictionary (dict)

Remove/get/add 都是O(1)?

速度很快!!

1. 也叫Map(C++,Java..)
2. 1對1
3. 單方向 (word 🡪 definition)
4. 左key右Value (Key-value pair)
5. Not ordered (沒有index)

dict

d = {}

1. 放key-value pair

d[key] = value

d[‘Jerry’] = 7

d[‘Cindy’] = 8

d[‘Becky’] = 20

d[‘Jewel’] = 3

1. 得value

value = d[key]

my\_lucky = d[‘Jerry’] #7

1. 改value

d[key] = new\_val

d[‘Jerry’] = 47

1. 幾筆data?

num\_data = len(d) #4

1. 移除value

d.pop(key)

d.pop(‘Jerry’)

6. 存在?

if key in d: (不能說if value in d，因為只能判別key)

if ‘Jerry’ in d:

#False

Loop over a dict

1. for key in d:

val = d[key]

1. for key, val in d.items():

跟上面1一樣都是取value

Python built-in

<max> <min> <sorted>

l = [9,5,10,1]

ele = max(l) #10

ele = min(l) #1

new\_l = sorted(l) #[1,5,9,10]

new\_l = sorted(l,reverse = True) #[10,9,5,1]

Tuple的比較的話

l2 = [(3,0), (1,1),(2,9)]

ele = max(l2) #(3,0) >>是比0的這個數字 即[(3,0), (1,1),(2,9)]

new\_l2 = sorted)l2) #[(1,1), (2,9), (3,0)]

Python single-line operations

(element-wise)



Python built-in with lambda

l1 =[‘c1’,’b9’,’a3’]

01, 01, 01

new\_l1 =sorted(l1) #[’a3’, ’b9’, ‘c1’]

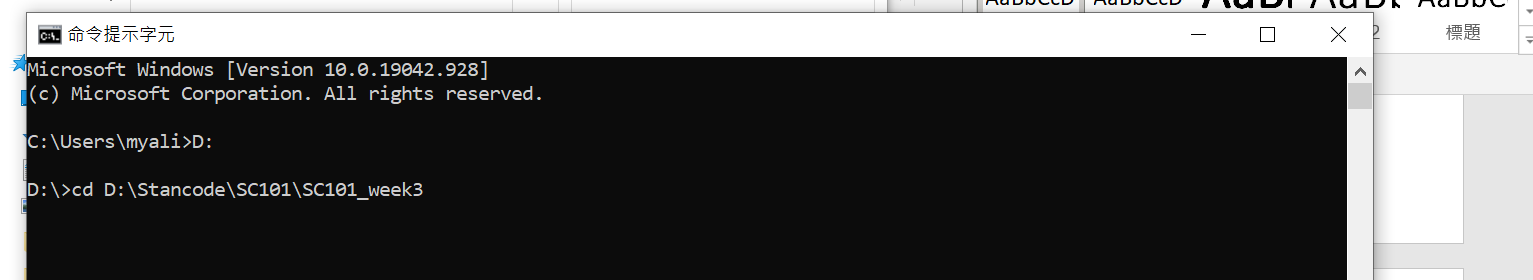
更改排列順序:

new\_l1 = sorted(l1,key=lambda s: s[1]) #[‘c1’,,’a3’,’b9’]

new\_l1 = max(l1,key=lambda s: s[1]) # ‘b9’

cmd如何使用拉進來的資料夾:

先按 cd 後在拉資料夾進來，在案enter進入資料夾。



Week 4:

Data Structure

* list

1. Stack🡪LIFO
2. Queue 🡪 FIFO

* tuple

1. (1,3)/1,3
2. a,b = b,a

* dict

1. d={}
2. d[key] = value
3. value = d[key]
4. if key in d: 🡪檢查是否存在
   * <loop over a dict>
5. for key in d:
6. for key, value in d.items()

#[(key1, value1),(key2, value2)]

dict是個沒有順序性的東西，但是python印出來的時候卻會有key的順序

Python的dict比其他的程式語言慢很多，因為他的dict會排順序

Computer Memory (RAM)

1. 0/1 🡪binary digit (bit)
2. 8 bit 🡪 1 byte (代表0-255的數字)
   * 存一個Int 電腦要幫忙留4 bytes (32台檯燈)
   * 存一個float 電腦要幫忙留4 bytes

存一個double 電腦要幫忙留8 bytes (double是指double precision，給你兩倍的準度) (對於float而言，double小數點會比較多)

* + 存一個long 電腦要幫忙留8 bytes (對於int而言，這可以存更多)

<Storage>

1KB = 1,000 bytes

電腦只能用2的倍數，所以只會給 210的bytes = 1,024 bytes

1MB = 1,000,000 bytes

220的bytes = 1,048,576 bytes

1GB = 1,000,000,000 bytes

230的bytes = 1,073,741,824 bytes

Numeral system 進位法則

<Binary> 1 0 0 1 要如何轉換成10進位?

23  22 21 20  
 1\*23+1\*20 == 9

<Octal> (8進位)

2 38

81 80

2\*81 + 3\*80 == 19

<Hexadecimal> (16進位) 0123456789ABCDEF (10開始用ABC表達)

3 C16

161 160

3\*161 + 12\*160 == 60

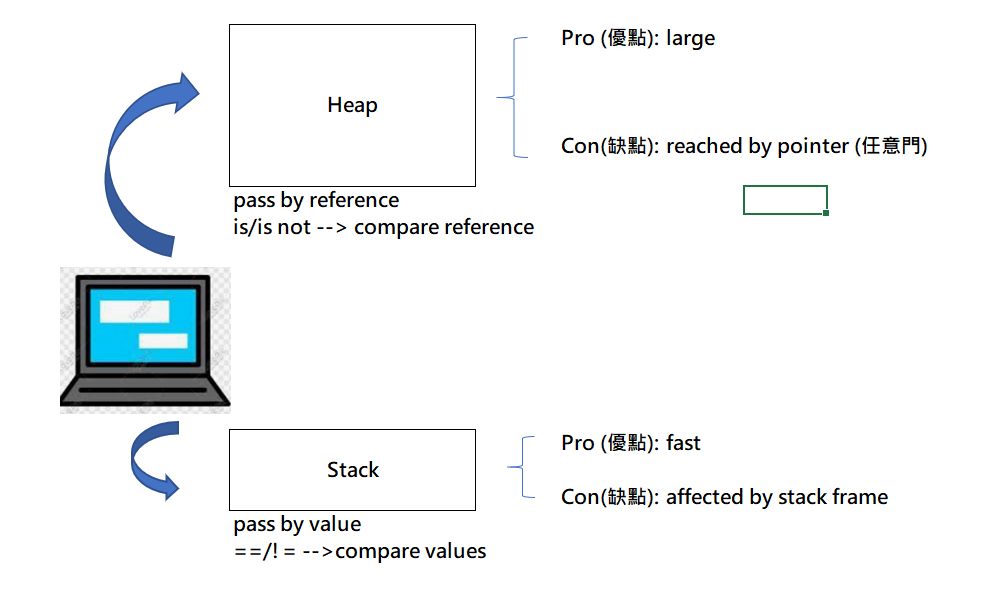
(記憶體位址)記憶體的家是用Hexadecimal 來表示

0X🡪代表16進位

0XFFFF🡪結尾 F越多代表記憶體越大  
 0X0001🡪開頭

A=10,B=11,C=12,D=13,E=14,F=15

<會不會受stack frame影響? List / dictionary in heap not do return.

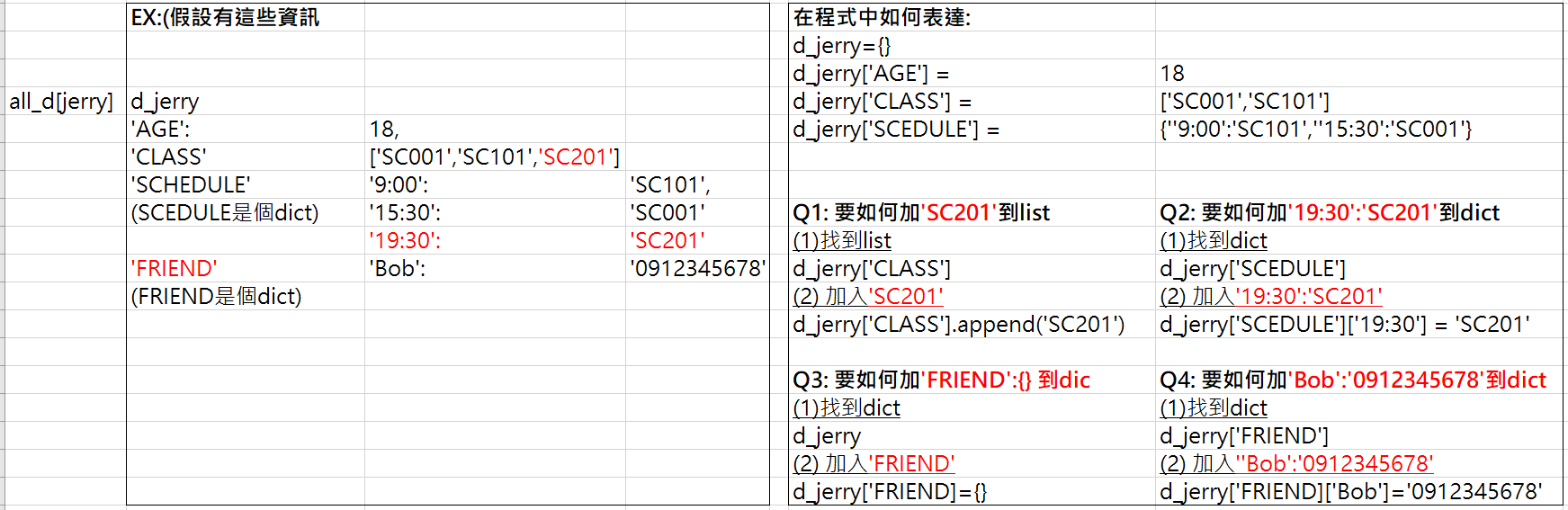


那如果我直接在window.add 之後換color也會有相同效果嗎?

如果我不進去stack frame 就不能換嗎?

[0][0]+=1

Nested data structure (DS)



bob

tkinter

import tkinter as tk

<make a canvas> (圖像化的東西)

w = make\_canvas(width, height)

<畫線draw a line>

w.create\_line(起點座標, 終點座標,線要多寬,顏色)

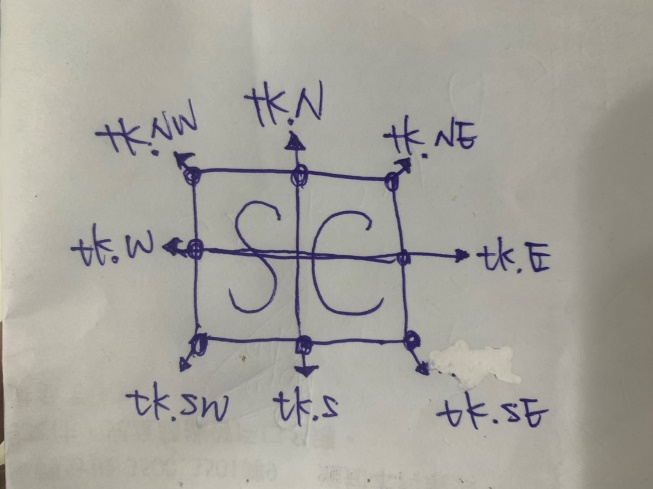
w.create\_line(x1,y1,x2,y2,width=5,fill=’red’)

<文字draw a label>

w.create\_text(文字的x,y,text=’SC’)

w.create\_ text (x,y,text = ‘SC’,anchor =tk.SW , font = ‘times 40’, fill = ‘blue’)

anchor 矛的意思(基準點的意思) 通常設在左下角



如果用campy的話就是:

from campy.graphics.gwindow import GWindow

w = GWindow(width=?, height=?)

Web Development



**HTTP GET request (url)**

**Internet**

Server

Client

Database

browser

**Response(Status)(content)…**

**Back End**

**Front End**

Backend Development

1. Server side
2. Help client communicate with database (log in account), (payment system)
3. 程式語言:Java/Python/PHP/….
4. 資料庫的語法: SQL, Oracal, …
5. Create API (Application Program Interface) to support Frontend

Frontend Development (client side)

<HTML> 非程式語言，只是語法

負責structure

1. Hyper Text Markup Language

超連結(links) tags

1. Handle
   * Web content
   * Web images
   * Web links

<CSS>非程式語言，只是語法

負責 appearance

1. Cascading Style Sheet
2. Fonts, Colors, Positions, …

<JavaScript>程式語言的一種

負責 behavior

1. Mouse Event
2. Run Web Apps

<HTML>:

如下:

<!DOCTYPE html> It’s a HTML5 document

<html> start

<head> 改變游標

</head>

<body> 網站的內容

<h1>Welcome to Irene & Hanz's Web</h1>

<h2>I created the Website for us</h2>

<p>

<a href="https://www.facebook.com/hsiaohan.chen.58/">Irene</a>  
 加網址

<img src="Irene.jpg" width="230" height="300">

加照片

</p>

</body>

</html> end

Week 5:

n! == n\*(n-1)!

5! == 5\*4\*3\*2\*1

4! == 4\*3\*2\*1

3! == 3\*2\*1

2! == 2\*1

1! == 1

0! == 1

Factorial 階層

def factorial(n):

return n\*factorial(n-1)

def factorial(n):

if n == 0:

return 1

else:

return n\*factorial(n-1)

Recursion 自己呼叫自己

**遞迴**（英語：Recursion），又譯為**遞歸**，在[數學](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%A6)與[電腦科學](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6)中，是指在[函數](https://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B0)的定義中使用函數自身的方法。

1. Calling itself
2. Self similarity
3. if base\_case:

return

else:

return recursion case

def main():

print(m(37,12)+m(14,10))

def m (a,b):

if a<b:

return a

else:

return m(a-b,b)

def main():

print(f(5))

def f(n):

#費式數列 Fibonacci Sequence

if n == 0:

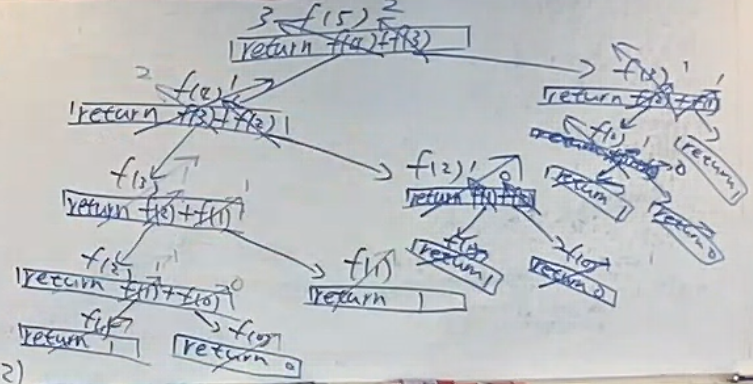
return 0 (base case)

elif n ==1:

return 1 (base case)

else:

return f(n-1)+f(n-2) return 完 代名詞往回退 note:f(4)+f(3) 要分開執行



回文

palindrome

在思考recursion的時候

1. base case?
   1. 一個字母 len(S)==1
   2. 空的 len(S)==0
2. Self similarity?

(該如何從recursive 逼近base case?)

（1）去頭去尾

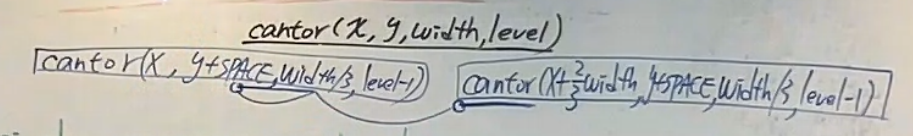
Fractals

定義:A self-similar recurring graphical pattern （用recursive 畫圖）

問題:

Self-similarity 在哪? (1) Draw a line (2) decrease width (3) left & right line (1/3) , (1/3) , 畫線左邊1/3, 右邊1/3.

Base case? 是什麼? Define our own base case! (自行定義)



Base case 可以兩個, recursive 也可以兩個

整數沒有長度 必須呼叫help function

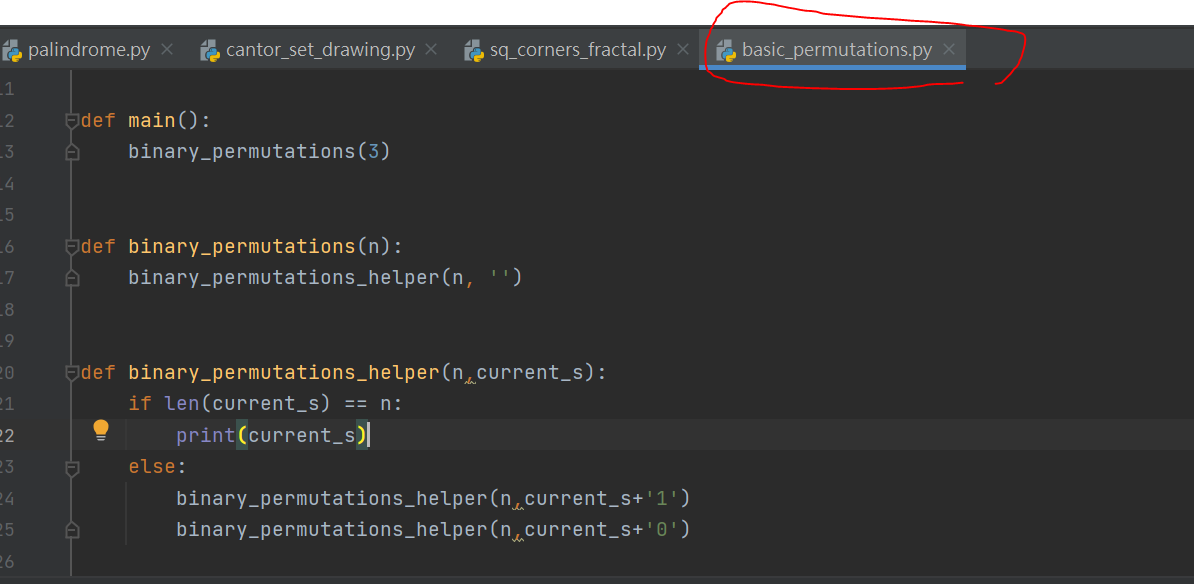
Helper Function

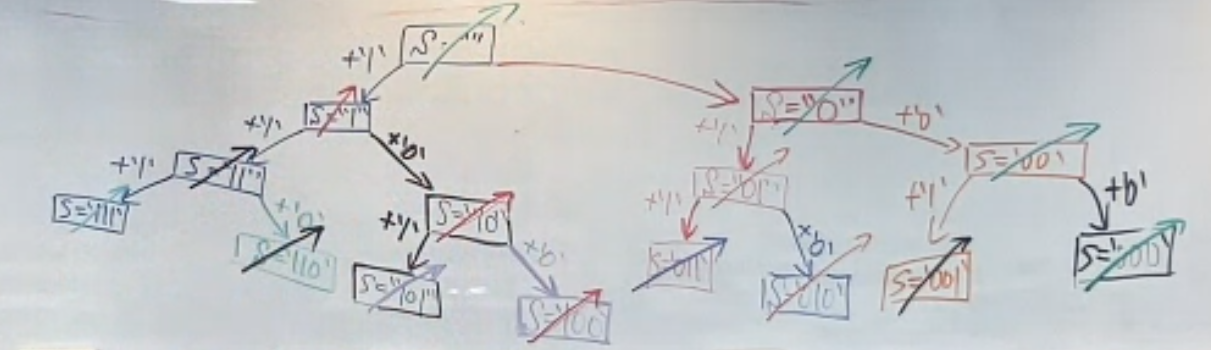
自己製造新的function , 且增加 argument/parameters

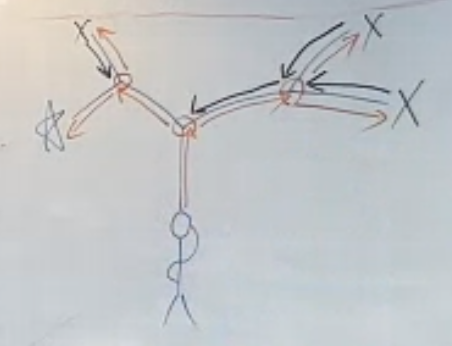
當題目要求太過分🡪自己加Variables

Helper Function

當題目要求太過分🡪自己加Variables





Backtracking (不會透過stack frame退掉，就要手動退)

Stanford的教學影片:<https://www.youtube.com/watch?v=-KbRfGR4Y-c>

定義: Trying all possible solutions, by abandoning ones that are not suitable.

1. “Brute Fore” Algorithm 暴力解法
2. Used in “exhaustive search” 每個都走走看
3. 3 step:
4. Choose 交叉路口.append
5. Explore 往前 lst
6. Un-choose 退後.pop

current\_lst = [ ]



Early Stopping

Pruning unnecessary recursive calls

Week 6:

Priority Queue

1. ADT sorted by priority

抽象資料型別（Abstract Data Type，ADT）是计算机科学中具有类似行为的特定类别的数据结构的数学模型；或者具有类似语义的一种或多种程序设计语言的数据类型。 抽象数据类型是间接定义的，通过其上的可执行的操作以及这些操作的效果的数学约束（与可能的代价）。

1. Used in advanced searching algorithm (Google Map)
2. Used in file compressor
3. 2 implementations
4. Python List Implementation

Bid O

* + - Prepend O(N)
    - Append O(1)
    - In Between
      * O(N) for loop
      * O(N) insert

🡪合起來還是 O(~~2~~N)

1. LinkedList Implementation

Bid O

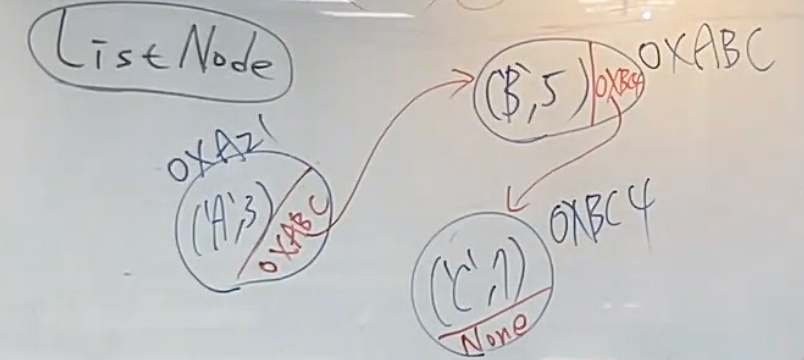
* + - Prepend O(1)
    - Append O(N)
    - In Between O(N)🡪比O(~~2~~N)小，不到N但被說成N

Linked List

1. Create this data structure by ourselves! (要寫一個class)
2. Organize non-contiguous (不連續) heap memory by pointers
3. Create ListNodes & Link them together!

ListNode

1. ListNode 包含Data + 下一筆data的記憶體位置(Pointer)
2. OXA21為此Data structure之開頭 (OXA21因為是開頭會被稱為Linkedlist)
3. 相連多個ListNode 🡪LinkedList



3 Common bugs in traversal

1. Infinite Loop (忘記往前)

While l\_l.next is not none:

Print(l\_I.value)

1. OBOB (加上最後一個)

While l\_l.nwxt is not none:

Print(l\_I.value)

l\_l = l\_l.next

1. Memory Leak，如果我一直讓Linked\_list動，我就無法找回開頭

(派出cur 去traverse)

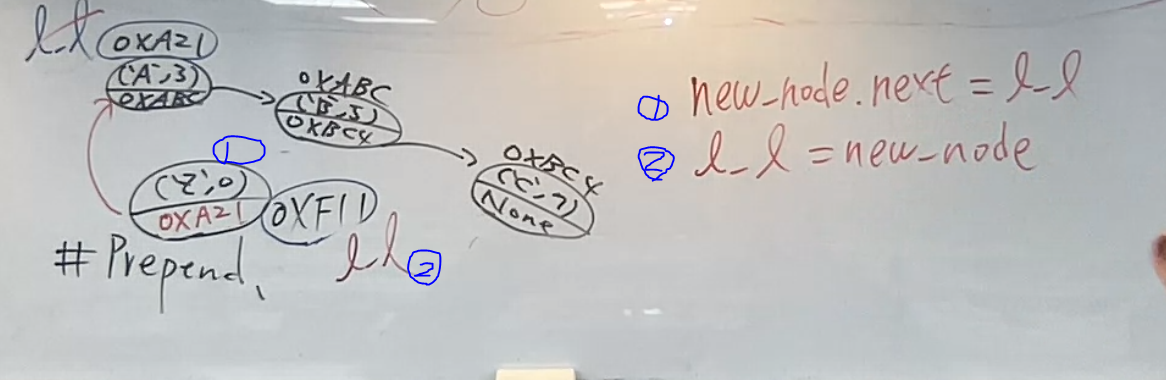
2 ways to traversal a l\_l (linked\_list)

1. While l\_l.next is not None:

🡪l\_l停在最後一個ListNode

1. While l\_l is not None:

🡪l\_l停在None

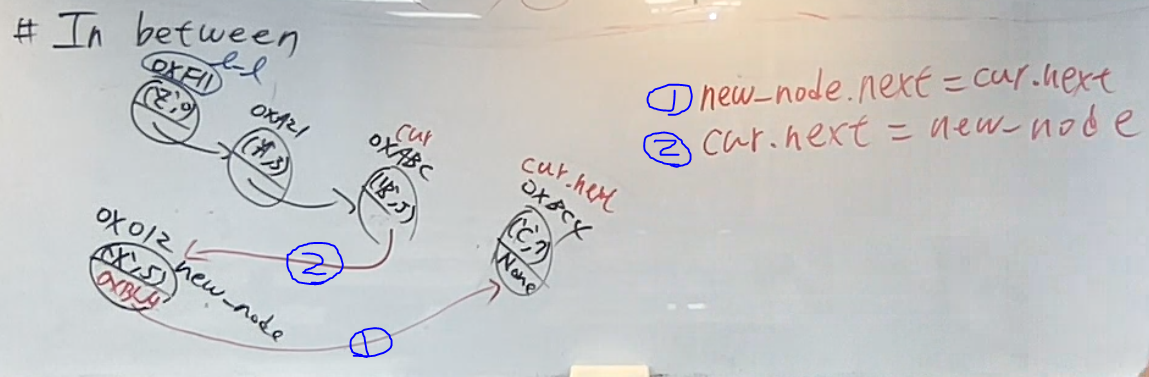


**Prepend:**

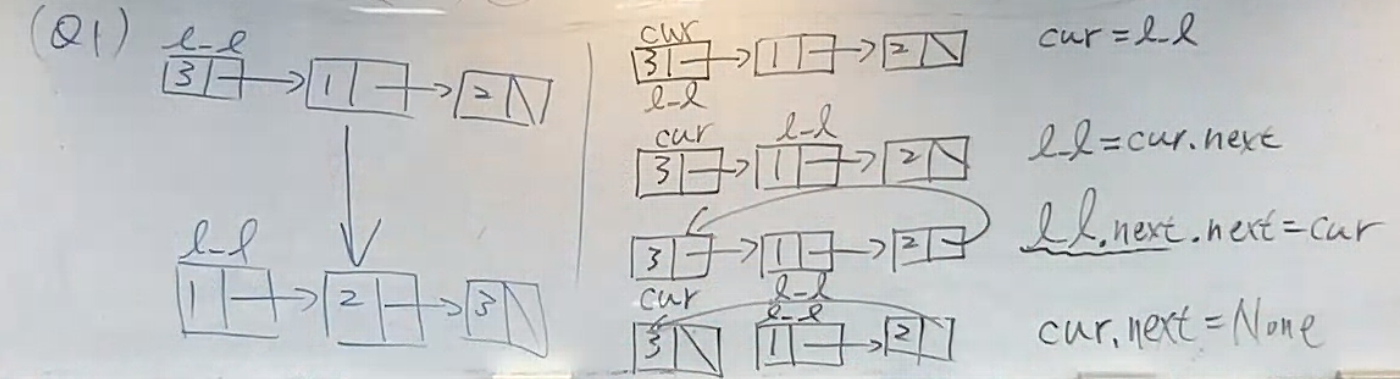
先連結再換開頭位置l\_l

**In Between:**

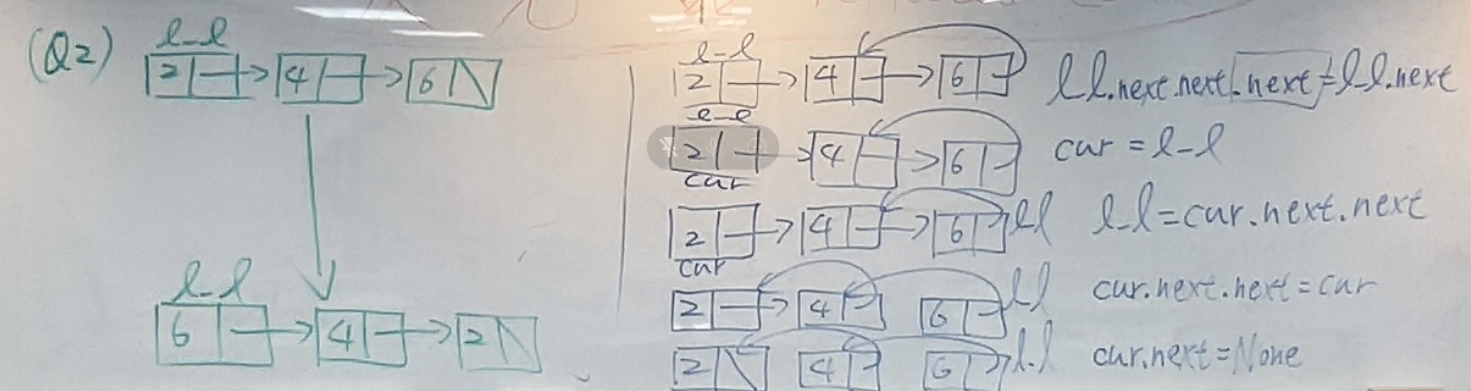
要先把new連到new的下一筆，在更新new的上一筆連向new



Q1



Q2



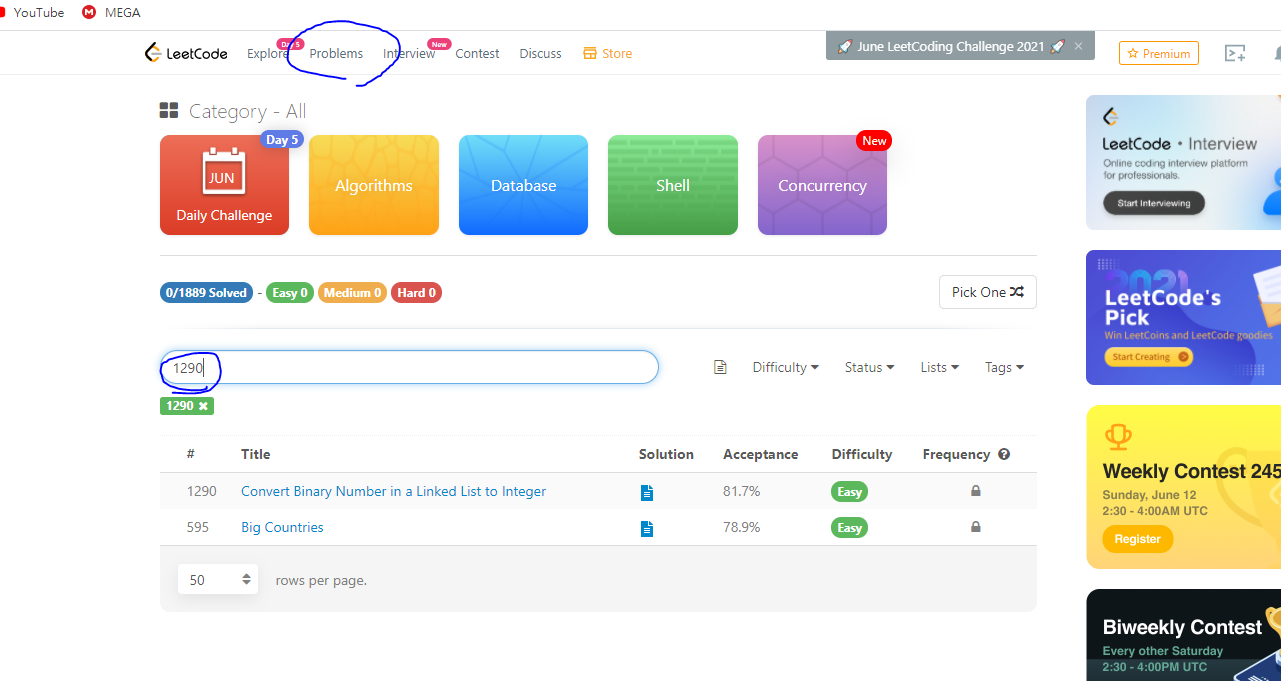
Cur = l\_l

l\_l = Cur.next.next

l\_l.next = cur.next

cur.next.next = cur

cur.next = None



def name(self, head):

“ “ “

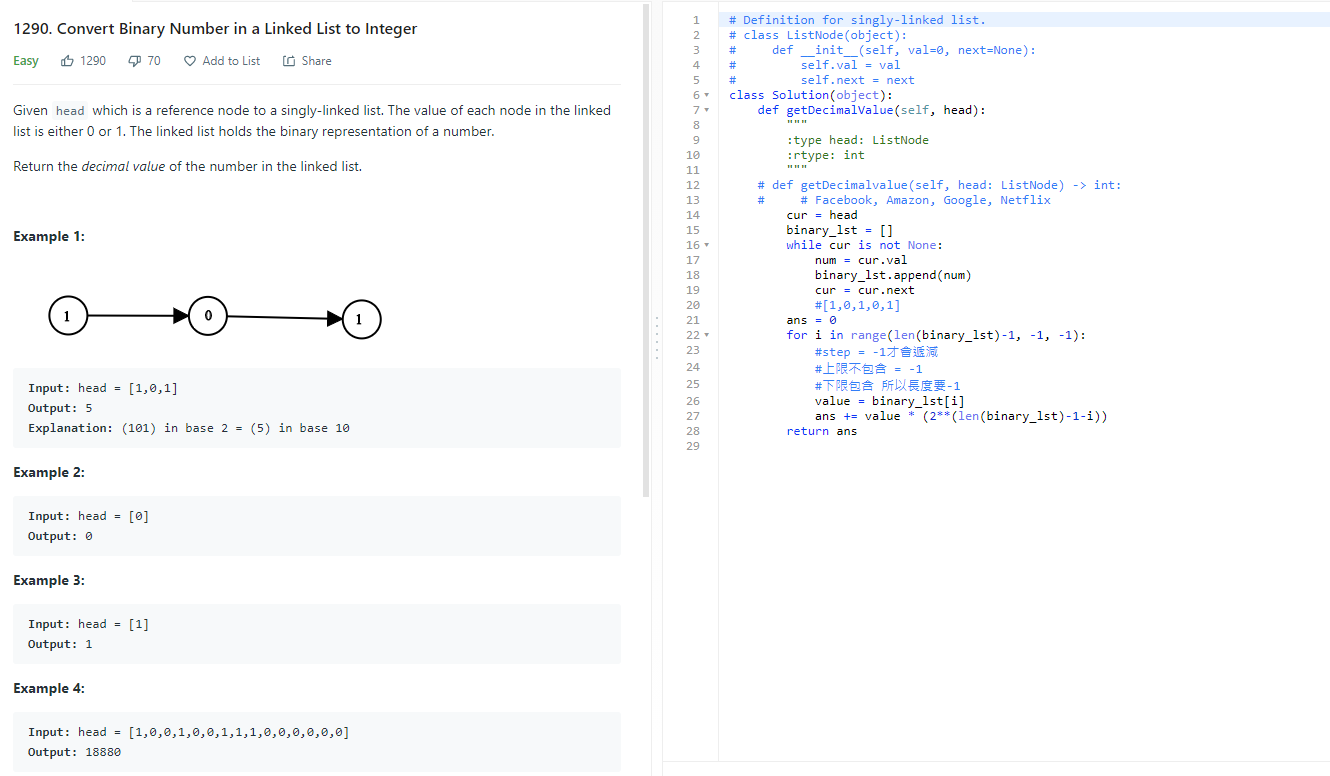
:param head: ListNode

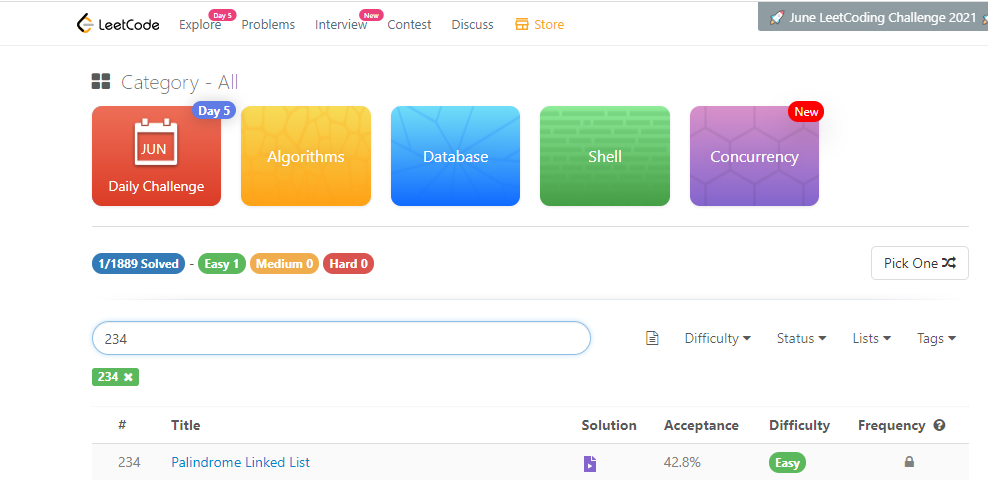
:return: int

“ “ “

OR

def name(self, head:ListNode)->int:







Follow up 的問題，如果你不用資料結構呢?

1>2>3>4>3>2>1

第一步 找到中點，在反接

1>2>3>4<3<2<1

有兩個東西，fast & slow

Fast 一次走兩步

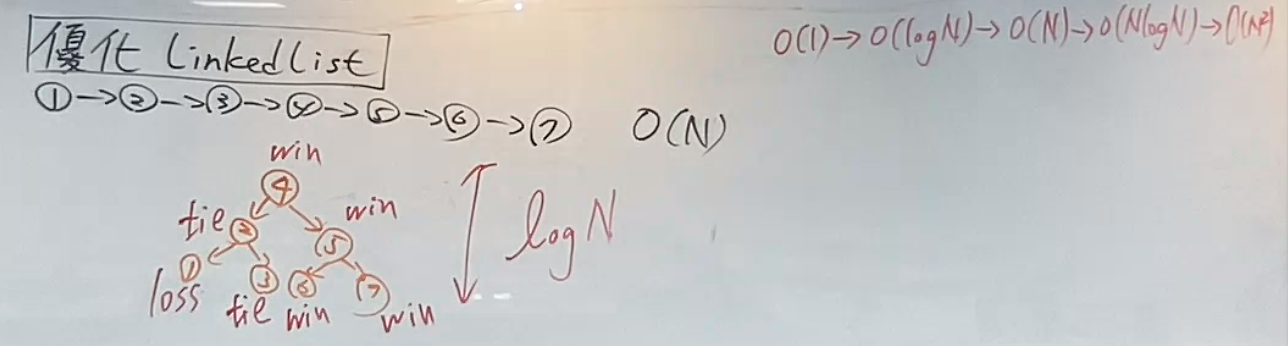
Slow一次走一步

Fast走完，slow就剛好在中點

優化Linked List

1>2>3>4>5>6>7

在電腦裡的log底都是2



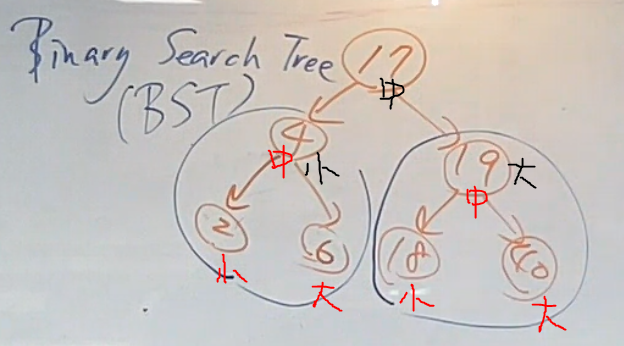
Tree

1. Directed (Parent -> Child)

2. Acyclic (1 path only)

3. Directed Acyclic Graph

(DAG)

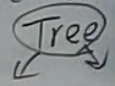


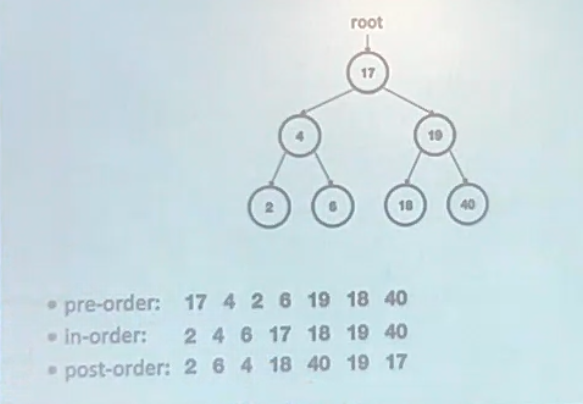
Recursive Definition

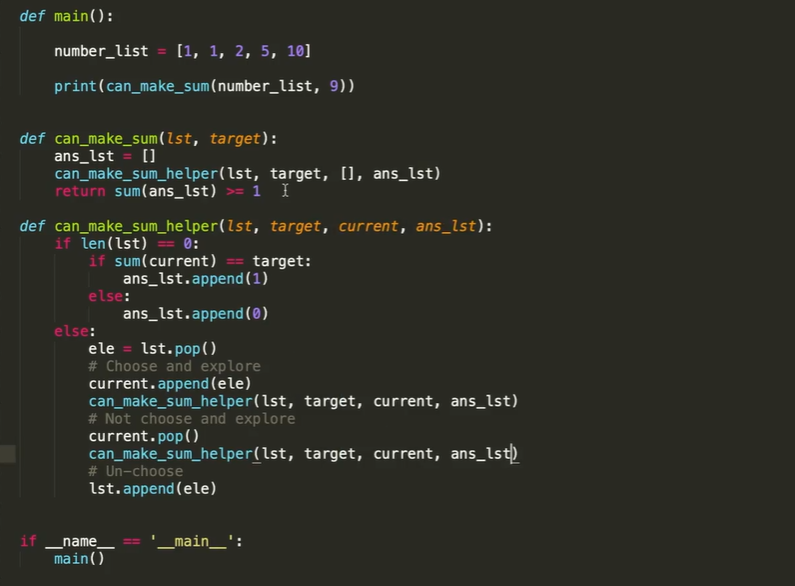
<Base case>

None

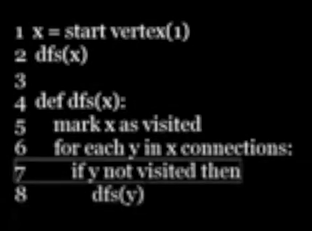
<Recursive>



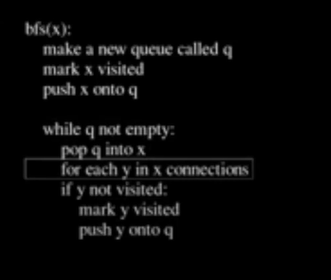




DFS: 以深度為優先的搜尋法則



BFS:以廣度為優先的搜尋法則



Dijkstra演算法:用 priorty queue 來搜尋

GOOGLE MAP的應用是BFS+ Dijkstra

Huffman演算法

Backtracking: 中文稱作「回溯法」，枚舉多維度數值的方法。運用遞迴依序窮舉各個維度的數值，製作所有可能的多維度數值，並且在遞迴途中避免枚舉不正確的多維度數值。

Linklist結構陣列是一種很常用的資料結構。

Linked-list是由一連串的節點（Node）所構成，每個節點指向下一個節點，而最後一個節點則指向Null（在python裡面是None）。

因此，每個節點本身應該要有兩種屬性（attribute），一個是本身帶有的值或者是資料，另一個則是指向下一個節點的指標（pointer）。

Priority Queue(優先權佇列)就像在處理「待辦事項」，能夠在眾多各自具有優先順序之資料堆中，取出「最重要/最不重要」的項目： Priority：對資料項目賦予「權重/優先權」，用以表示資料的重要程度。 Queue：隊伍、佇列、順序，意味著把資料整理成「某種順序」的資料結構。

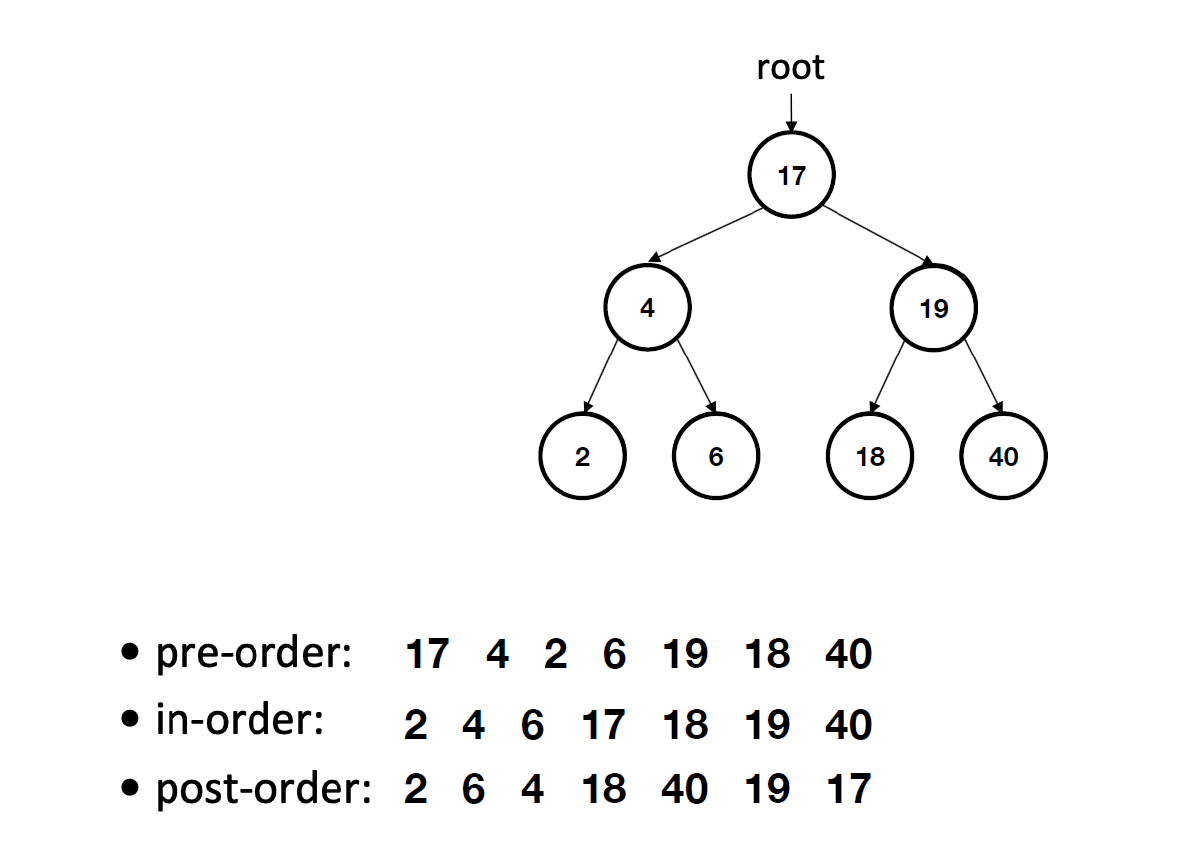
Lambda:

1. 使用Python写一些执行脚本时，使用Lambda可以省去定义函数的过程，让代码更加精简。 2、对于一些抽象的，不会别的地方再复用的函数，有时候给函数起个名字也是个难题，使用Lambda不需要考虑命名的问题。 3、使用Lambda在某些时候让代码更容易理解。

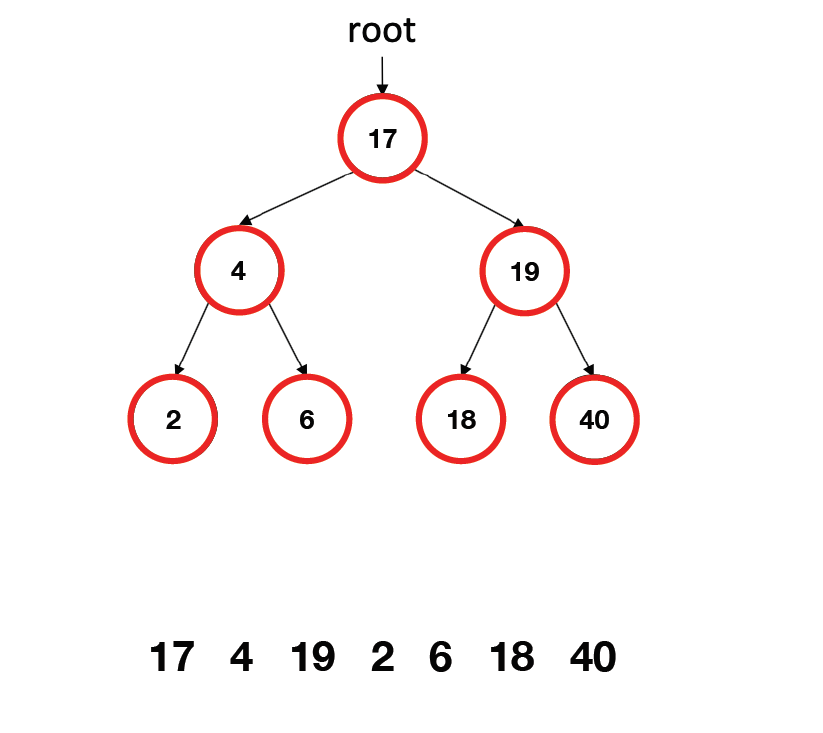
一個 lambda 函數是一個單行匿名函數，它可以接受任意數量的參數，但只能有一個表達式。

更具體地說，以下代碼是相同的：

我們需要學習 lambda 函數的原因是它可以修改現有的 Python 內置函數，例如 sorted、sum、max、min....，這使得 Python 在抑制代碼長度方面更加強大



BFS:



課程重點回顧:

[ SC101 - Week6 重點回顧 ]

-

各位同學晚安

課程轉眼來到最後了。最後一週同學的壓力一定非常大(boggle.py 之外還有期末考) 但請各位先不要擔心！我跟助教都會盡全力協助大家的。加油！我也會在 boggle.py 截止後私訊每一位同學詢問準備狀況，也請大家務必練習模擬考

-

Week6 課程我很喜歡的一點就是他扮演著 SC101 課程的集大成：運用物件導向、資料結構、演算法的每個觀念，幫大家把課程再複習，也讓同學們看到許多真實的程式應用實例（優先佇列 Priority Queue、自己管理記憶體走向的 LinkedList、搜尋 BFS、...）🥳

-

第六週同學要掌握的重點有以下6點：

(1.) ListNode 是一個我們自己創造的資料結構 (所有寫 Python 的都跟我們寫得一模一樣 🤩) 他會在 heap 隨便找一個區段塞入 self.val 以及 self.next（總共佔據大約 12 bytes 的記憶體空間）

(2.) 相較於 list，使用 linked list (串連後的 ListNode) 在插入 "In between" 的部分快很多！然而，Tree 又會比 linked list 更快。因此，這也是為什麼電腦大多數的資料都是儲存在 Tree 的結構

(3.) 在 linked list 搜尋時我們做 traversal 一定要小心三個常見 bugs - 無限迴圈、OBOB、Memory Leak。另外，我們提到 2 種 traversal 的方法有何差異？也麻煩同學務必了解～

(4.) 在樹狀結構，我們要 traversal 時會採取三種策略：

pre-order, in-order, post-order. 這三個分別為「中左右」、「左中右」、「左右中」請同學稍微記一下～

(5.) 樹狀結構在搜尋時的複雜度為何為 O(logN)🤔 另外 Tree 為什麼又稱為 DAG 🤔 這兩點再麻煩同學們複習一下了 😃 他們會是之後同學們若要繼續往上鑽研電腦科學的重點

(6.) 搜尋演算法分成 "Depth First Search" 以及 "Breadth First Search" 兩種主要脈絡。前者就是大家熟悉的 recursion 演算法！而後者一定要搭配一個 Queue 去使用其 FIFO 之特性來追蹤每一層跟層間的關係

然而，為了增加搜尋效率，我們會在搜尋前先「排列」。而「怎麼排列」就是 stanCode 團隊想要透過期末考教大家的重點之一了！我們會在期末檢討時幫同學整理所有常見的搜尋演算法！再麻煩大家務必出席～

[ SC101 - Week5 重點回顧 ]

-

各位同學辛苦了！

本週的課程已經進到非常 CS 的領域 🔥

真的很難！如果有任何問題請一定要跟我＆助教討論

教學團隊會盡全力幫助大家成功的 ✨

-

本週的重點回顧有以下 9 點：

1.) 從追蹤一個 Recursive Function 的過程，我們了解到程式碼前後的順序會影響 Recursion 執行之先後順序。這個概念結合了 Stack Frame！因此這邊可以幫各位補充一個新觀念：在 recursion 的世界，我們會稱他們的 Big Oh 為 Space Complexity（因為 recursion 一個 Stack Frame 的複雜度通常不高，但會有超級多的 Stack Frames 跑出來！因此我們要考慮：到底在整個演算法中跑出了幾個 Stack Frames？）

2.) 若一個 Recursive Function 的 base case 會 return，則 recursive case 也一定要 return! 不然答案會無法在每個 Stack Frame 間傳遞

(大家在寫作業時請務必小心）

3.) 我們在繪製 fractals 時發現：可以自己定義常數 level 來決定 base case。且 Recursive Function 可以被呼叫不只一次 (像是畫 cantor\_set 時為了畫左邊、畫右邊呼叫了兩次 ; 要畫 sq\_corners 時為了在四個角都有正方形呼叫了四次）

4.) 在繪製 fractals 時我們使用 pause(DELAY) 看到了一個有趣的現象：所有圖形都會先走到底（不管是左邊的底、還是右邊的底、還是由下而上）這個現象其實有一個專有名詞：Depth First Algorithm。我們在下週課程會更深度探討這個演算法的特色 🤓

5.) Recursion 應用在解題上時都會在不改變題目要求的情況下，再多呼叫一個 Helper Function 加入任何自己喜愛的變數（通常會加入一個 str / list）然而，我們發現：前者會受 Stack Frame 的影響 ; 後者不會！因此，為了手動歸還每一個 Stack Frame 當時的狀態，我們衍生了第六個重點的演算法：Backtracking

6.) Backtracking 要遵守三個步驟：Choose; Explore; Un-choose

這三個步驟可以讓存在 Heap 而不受 Stack Frame 影響的資料結構在退回每一個步驟時都回到「最初的狀態」。請同學務必手動追蹤一次、或是使用 debugger 跟著演算法跑一次 👀 才能更了解這個超爆爆難的演算法

7.) 很多時候我們需要限制 recursive function 呼叫的情況。舉例來說，在 dice\_rolls\_sum.py，電腦並沒有「超過 8 就該停止加數字」的概念。因此，我們要在 recursive function 一開始加入條件：不能讓超過 8 點的骰子組合進入 recursion

8.) Early Stopping 會是 Recursion 演算法加速的重要環節（因為少呼叫一次 Recursion 將可以少掉一大叢衍生出來的樹狀 Stack Frames）也因此，電腦科學家使用「修剪花木」的英文 - prune - 來描述這個概念 (很 Q 耶 🤩)

9.) 最後，我們用一題 Google 面試題看到了 subsets 的演算法：Choose

Explore without the ele

Explore with the ele

Un-choose

這個變形演算法將在日後同學們解「軟體工程師面試題」扮演關鍵的角色 ✨ 但作業不會用到這個變形～請大家不用擔心 💪

[ SC101 - Week4 重點回顧 ]

-

各位全端工程師們，期待看到大家的 babynames 🤩

-

本週的重點回顧有以下 8 點：

1.) Pass by value & Pass by reference 的差別就在於前者的資料存在記憶體名為「Stack」的區間、後者的資料存在記憶體名為「Heap」的區間

2.) 以後大家看到一個變數的 data type 要有另一層的觀念：他在告訴電腦要分配多少檯燈 💡 下方連結列出了常見的 data types 所佔的記憶體量：https://docs.oracle.com/.../chp-typeopexpr-2/index.html

3.) Stack Memory 儲存資料符合 Last-In-First-Out 的特色。因此，最後呼叫的 function call 通常是最先執行 👆 （幫大家補充一點：當多個 function calls 疊在一起，電腦會用一個叫做 Stack Pointer 的小幫手來告訴系統現在執行到哪一個 Stack Frame）在 Stack 上的資料通常都是 Primitive types (int, float, bool, ...) 他們會受到 Stack Frame 退出而隨之消失

4.) Heap Memory 儲存的資料為 objects (後來我們發現：資料結構其實也是一種 object) 。然而，由於程式執行所在位置為 Stack，因此若要使用在 Heap 上的資料，我們必須讓 Stack 上的變數儲存資料的 reference! 我們稱存著 reference 的變數為 pointer 👆👉👈👇

Pointer 會在 SC101 Week6 跟大家相見歡 🤩

5.) 網頁由三個部分組成：HTML, CSS, JavaScript. 這邊有一個滿有趣的影片跟大家介紹怎麼從頭使用上述的元件組成一個網站

https://www.youtube.com/watch?v=ZeDP-rzOnAA

6.) 爬蟲首先要請 Python 對我們的目標網站提出 HTTP requests!

HTTP（Hyper Text Transfer Protocol ）是一個管理由網路送出、接收的指令庫。「GET」是使用者向 server 要資料的指令。因此，我們才會在程式一開始使用 requests.get(url) 🤓

(URL 可以理解成網址，他的全名是 Uniform Resource Locator! 更多資訊可以點擊連結 👉 https://en.wikipedia.org/wiki/URL)

7.) 得到 HTML 之後，我們需要有人幫我們把這個龐大的文字巨獸整理成 object 的形式 🥺 因此，我們召喚 bs4 包裹裡面的「美麗的湯 🥣」。這個包裹可以先夠過 soup.find\_all 找出所有包含 class 變數的 <tag 欄位！之後我們透過「.」來進入欄位裡面的 <tag，或是透過 ['name'] 得到欄位裡的變數數值

8.) 最後，課程來到演算法核心 - Recursion - 的世界 🤯 請大家務必先從追蹤的過程了解 Recursion 的概念 ✍️ 或是透過 debugger 了解每一個 Stack Frame 的變化～～

[ SC101 - Week3 重點回顧 ]

各位同學午安！先恭喜 GObejcts 得獎者 🎉🎉

-

今天課程重點回顧有以下 9 點：

-

(1.) Python List 跟 Python str 非常像！如果有任何不了解的指令歡迎從 str 理解 👍 今天教的所有資料結構(以下簡稱 ds)都可以用相同的方法「取資料: element = ds[...]」、「查看是否存在: if element in ds:」、「得到每一筆資料: for element in ds」

-

(2.) ADT 是一種概念！我們今天教了 Python 不存在的兩個資料結構：Stack & Queue。他們會在最後兩週的演算法課程出現！再麻煩大家不要忘了他們 🥺

-

(3.) 我們今天終於學會怎麼用 Terminal/cmd 跑程式、用 Sublime 寫程式！大家快去佔咖啡廳門口的位置寫 code 🔥🔥🔥

-

(4.) 我們可以使用 sys.argv 取得使用者在 Console 輸入的文字！文字會以 Tokenize 之後的形式儲存，因此我們就可以使用 sys.argv[1] 去判斷使用者輸入的指令是要執行哪一個 function 了～

-

(5.) Tuple 是一個 Immutable 的資料結構！他會在 ai 領域瘋狂出現 🤩 而且 Python 會將「以逗號隔開的資料認定為 Tuple」，因此我們發現 Python Swap 演算法可以簡單地使用：a, b = b, a

(感謝 Hsiao Han Chen 示範 ❤️ )

-

(6.) 一個演算法/資料結構的效率我們會使用 Big Oh 來判斷！這個在往後大家面試會非常重要。今天稍微介紹，希望之後讓大家再次碰到的時候可以立馬上手 👍 以後上課或助教時間跟同學們溝通可以更專業了 🤓

-

(7.) 為了儲存相互對應的資料，電腦科學家發明了 Python dict。這個資料結構速度真的非常快！大家未來要加速程式歡迎使用它～ 然而，Python dict 沒有 index, 因此我們介紹了兩個得到 key-value 的方法：for key in d 以及 for key, value in d.items( )

-

(8.) 再來我們完成了我認為很適合當面試題的 word\_occurance.py 🎉 這份檔案包含了非常多重點概念：從讀檔到 tokenize、string manipulation、加入 dict、到最後如何以 value 排列輸出結果請大家務必複習～

-

(9.) 最後課程停在理解 Python Comprehension, Python lambda function 的部分。這些 element-wise 的指令在日後看其他工程師的 code 會一再出現～再麻煩大家記得看到要來這堂找找 stanCode 的筆記喔 😆

[ Python 物件導向 - 進階補充 ]

物件導向（OOP）是整理 data 的一種方式 - 我們用 object 來包裹特定種類的物品，來讓一個 data 擁有許多變數(attributes) 也擁有許多絕招(methods)。好像是這個 data 一誕生就是一個身懷絕藝的工人，可以直接做很多複雜的工作

在公司內部，專案一定要分工。這時候，每一個軟體工程師就是負責打造一個 class A，再把這個 class 丟給下一個 B工程師（B工程師可以用 class A 去做專案，或者 B 工程師也可以用 class A 當作 Super Class 來打造 class B）

因此 searchapparchitecture 這個網站是這樣形容 OOP 的：

"The organization of OOP makes the method beneficial to collaborative development, where projects are divided into groups.

Additional benefits of OOP include code reusability, scalability and efficiency. "

簡單來說，「物件導向」只是一種思維。我們的第二週上課範例全部都可以用非物件導向的方式寫！只是程式碼會在同一份檔案、重複的程式碼會比較多

Hope this helps!

最後，Jerry 又錄製了兩個影片來講解更艱深的 class 概念！如果想要再精進的同學歡迎找時間觀賞

「Python 在括弧裡放了一個 ＊ 到底是什麼意思?」

https://youtu.be/HsMXf752VxQ

「Python 括弧裡，變數前加上＊與＊＊是什麼意思？」

https://youtu.be/xEOfKR\_L3vU

Sincerely,

stanCode 團隊

[ SC101 - Week2 重點回顧 ]

-

各位同學午安！

希望今天物件導向的課程有打開大家的任督二脈 🔥

-

「Week2 - 物件導向」的重點有下列八點：

(1.) 製造小小兵的「constructor 部門」其實就是

def \_\_init\_\_

而它的功能是將所有 instance variables (attributes) 存到 self, 並讓所有透過它製造出來的小小兵 (instance/object) 擁有所有的 methods & attributes!

(2.) Methods 會根據「有沒有使用到 self」而分成 Instance Methods 以及 Static Methods. 前者的表現會因為小小兵不同而有些許不同，後者則表現完全一模一樣。因此，Static Methods 可以透過 class 來呼叫！

(3.) 想要承襲一個 class, 只要將該 class 放到定義後方的括弧、並在 def \_\_init\_\_ 裡面呼叫「super( ).\_\_init\_\_ 」啟動 Super Class 的所有 attributes＆methods

(4.) 我們透過承襲的上課範例想讓大家了解「attributes」與「parameters」之間的差別～ 前者是放在 self 的箱子名稱📦 後者則是接收使用者傳入 def \_\_init\_\_ 的箱子📦 舉例來說：

def \_\_init\_\_(self, weights):

self.w = weights

w 是我們這個 class 的 attribute, 而 weights 則是 parameter

(5.) 我們發現 \_\_name\_\_ 其實是每個 Python 檔案都有的系統變數！系統會根據一個檔案到底是「被執行」還是「被 import」去丟入 '\_\_main\_\_' 的文字或是 'filename' 的文字

(6.) 我們在 pypal.py 的檔案發現：把 keyword arguments 的 default value 數值存入檔案上方的常數可以讓程式碼維護/修改變得更加便利 🔥 也可以透過這個方法，讓使用者在呼叫您的 class 多了很多客製化的選項～（不給改的就不提供 😌 大家可以參考 zone.py 裡面我們 MAX\_SPEED 常數的例子）

(7.) 通常在 class 內部的 instance variables/attributes 都會以 private 的型式儲存！在 Python，我們會使用 \_var 以及 \_\_var 來表示 private 的概念：前者我會稱它為「非強制性的 private」後者我則會稱之為「強制性的 private」！請同學們在寫作業二的時候務必將 attributes 都寫成 private 🔥

(8.) 在 class 內部建造時若要使用內部 method, 要記得使用 self.method 來呼叫！就好像是使用端要使用 method 時需要透過 object.method 的概念 🤓

-

最後，期待看到大家打磚塊作品 🤩 見證 ++ 獎學金的誕生！

-

[ SC101 - Week1 重點回顧 ]

-

超級開心見到每一位同學！

很期待這七週跟大家一起燒腦 🔥 也期待課程帶給大家的蛻變 🤩

-

「Week1 - 物件導向」的重點有下列八點：

(1.) 一個 class 會去定義做出來的 object/instance 有什麼特色 (attributes) 與絕招 (methods)! 而負責做出「小小兵」的部門稱為 constructor

下一堂我們會依照自己的需求做出一個獨一無二的 class 🔥

(2.) 「被承襲的 class」我們稱之為 Super Class, 「承襲別人的 class」我們稱之為 Sub Class. 下一堂我們會實作承襲的概念 🤩

(3.) from campy.graphics.gobjects import GOval 這一行代表了進入 campy 資料夾裡的 graphics 資料夾裡的 gobjects.py 檔案。請注意：這邊我們用「點」進入資料夾 ; 而在讀檔時我們用「/」進入資料夾

(4.) Positional Arguments 代表著兩個概念：「以位置傳入」、「一定要有」; Keyword Arguments 代表著兩個概念：「以變數帶領傳入」、「可有可無」

(5.) Mouse Events 有 Asynchronous 的概念 ; 也就是說他是「關不掉的」、「隨機的」、「可隨時被執行的」

(6.) 請注意：onmouseclicked 這些 function 是要接收 function 的！傳入 function 不能加括弧 ; 而定義 function 時不僅要加括弧、還要提供一個變數給系統傳入「滑鼠資訊」🖱️ 再麻煩同學們參照上課講義 Week 1 的補充教材 📖

(7.) 在小畫家 mouse\_draw.py 的程式裡，我們使用 stroke 這個變數去追蹤「最後一個出現在視窗上的 object」! 而其他沒有被追蹤到的 objects 依舊存在電腦的記憶體，只是我們沒有追蹤而已。這個概念非常重要！再麻煩同學們務必花時間去理解 🤓

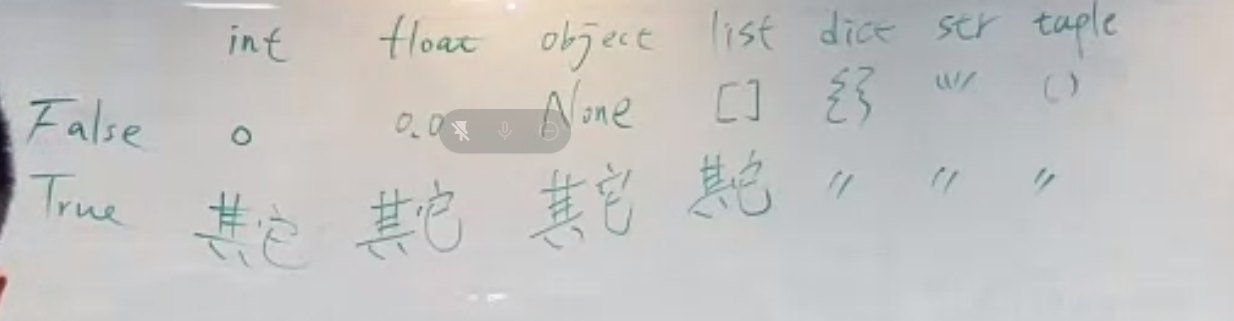
(8.) Global Variable 是一個可以被整個程式看到的變數（又稱為全域變數）然而，若您要改變 Global Variavle 存取的數值，請務必在更改之前，在寫上程式碼 global + 變數名稱! 讓 Python 知道我們是真的懂的 🤣

-

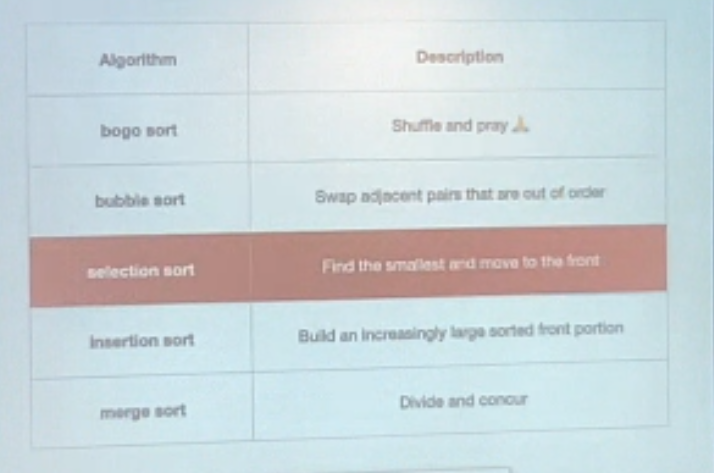
最後，期待看到大家 GObject 競賽作品 🤩

歡迎來到 SC101！

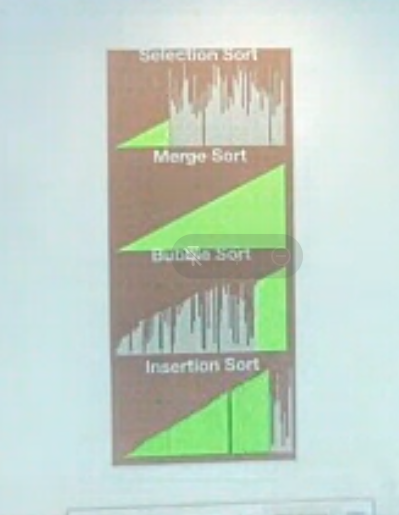
-



Sort種類:



速度: merge sort\_O(NlogN)>insertion sort\_O(N^2)>selection sort\_O(N^2)>bubble sort\_O(N^2)



期末考題第五題是LC2:FB, Google, Amazon

期末考最後一題:

