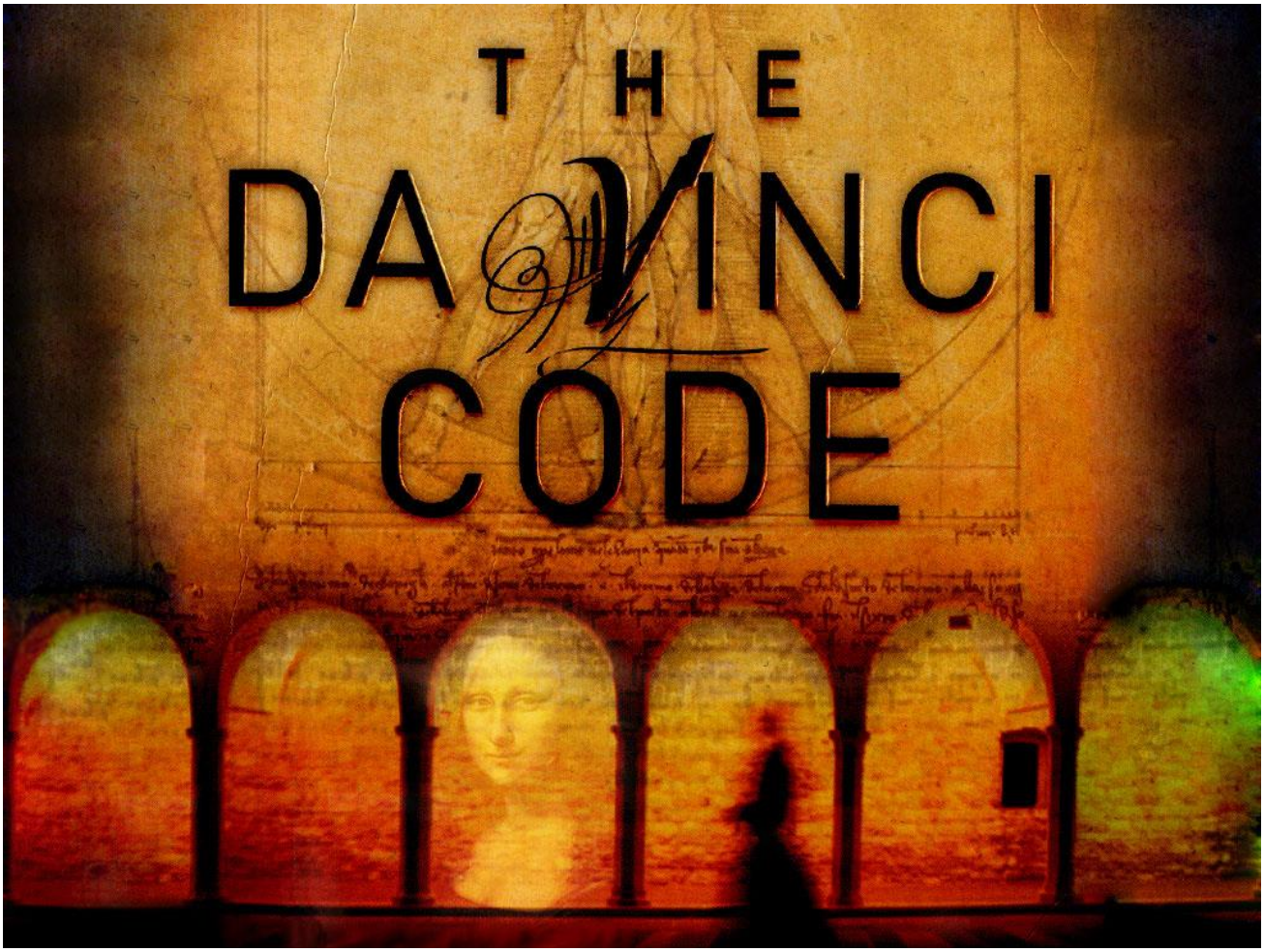
Computational Thinking & Programming

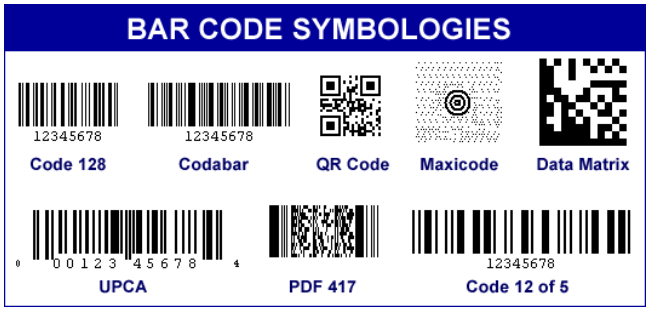
駭客任務 A -資料解析與文字處理

破解密碼 - 資料表示 Data presentation - 資料處理 Data processing -
資料編碼 Data encoding/decoding 資料加密 Data Encryption




```
01001001001000000110000101101101001000000111011101101000011000010111
01000010000001001001001000000110000101101101001011 0
#000000 #FFFFFF #00FF00 #FF00FF #A0CC00 #F0F0F0 #C0C0C0 #CCCCCC
```

--. --- --- --. .-.. .



A	·–	J	·–––	S	···		
B	–···	K	–·–	T	–		
C	–·–·	L	·–··	U	··–		
D	–··	M	––	V	···–		
E	·	N	–·	W	·––		
F	··–·	O	–––	X	–··–		
G	––·	P	·––·	Y	–·––		
H	····	Q	––·–	Z	––··		
I	··	R	·–·	1	·–––––		

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

- **編碼（Encoding）**在認知上是解釋傳入的刺激的一種基本知覺的過程。技術上來說，這是一個複雜的、多階段的轉換過程，從較為客觀的感覺輸入（例如光、聲）到主觀上有意義的體驗。
- **字符編碼（Character encoding）**是一套法則，使用該法則能夠對自然語言的字符的一個集合（如字母表或音節表），與其他東西的一個集合（如號碼或電脈衝）進行配對。
- **文字編碼（Text encoding）**使用一種標記語言來標記一篇文字的結構和其他特徵，以方便計算機進行處理。
- **神經編碼（Neural encoding）**是指信息在神經元中被如何描繪的方法。
- **記憶編碼（Memory encoding）**是把感覺轉換成記憶的過程。
- **加密（Encryption）**是為了保密而對信息進行轉換的過程。
- **轉碼（Transcoding）**是將已經編碼的信息從一種格式轉換到另一種格式的過程。
- **解碼（Decoding）**是編碼的相反，亦即把編碼過的信息恢復成原來樣式。

文字編碼

電腦中只能以 0、1 這種狀態來記錄，那麼我們輸入的文字又是如何被電腦所記錄的呢？其實這些文字在電腦中也是轉換成一連串 0、1 的狀態，這個文字對應成一連串 0、1 數字的方式我們稱之為編碼。最常見的編碼為美國資訊交換標準碼(American Standard Code for Information Interchange，ASCII)，這個編碼使用了 8 個位元，記錄了大小寫的字母 A-Z、0-9 的數字，以及一些英文的標點符號，例如：

位元	數值	字元	位元	數值	字元	位元	數值	字元
00110000	48	0	01000001	65	A	01100001	97	a
00110001	49	1	01000010	66	B	01100010	98	b
00110010	50	2	01000011	67	C	01100011	99	c
00110011	51	3	01000100	68	D	01100100	100	d
00110100	52	4	01000101	69	E	01100101	101	e

至於中文的部分，在台灣是使用大五碼(Big-5)，而中國大陸是使用國標碼(GB)，系統中有一個「內碼輸入法」就是直接以 Big-5 碼輸入一個字。這些編碼都是使用 16 bits 也就是兩個位元組來記錄一個文字，為了和 ASCII 碼相容，所以第一個位元組必須大於 128 (ASCII 只使用到 0~127)，才會將這個位元組以及後面位元組算成同一個中文字。不過，同一個字在 Big-5 碼及 GB 碼所對應的數字是不同的，而同一個編碼在這兩個編碼所對應的文字也是不同的，這樣的問題也出現在日文、韓文等不是使用字母的語系，為了解決這個問題，於是出現了萬用碼(Unicode)，它的目標是把全世界所有國家的文字有一個統一的編碼，目前 Win2000/WinXP 系統即是使用 Unicode

a009/ACM 458: 解碼器

在密碼學裡面有一種很簡單的加密方式，就是把明碼的每個字元加上某一個整數 K 而得到密碼的字元（明碼及密碼字元一定都在 ASCII 碼中可列印的範圍內）。例如若 K=2，那麼 apple 經過加密後就變成 crrng 了。解密則是反過來做。這個問題是給你一個密碼字串，請你依照上述的解密方式輸出明碼。

至於在本任務中 K 到底是多少，請自行參照 Sample Input 及 Sample Output 推出來吧！相當簡單的。

輸入說明 ：每筆測試資料一列。每列有 1 個字串，就是需要解密的明碼。

輸出說明 ：對每一測試資料，請輸出解密後的密碼。

範例輸入 ：

1JKJ'pz'{ol'{yhklthyr'vm'{ol'Jvu{yvs'Kh{h'Jvywvyh{pvu5

1PIT"pz'h'{yhklthyr'vm'{ol'Pu{lyuh{pvuhs'I|zpulzz'Thjopul'Jvywvyh{pvu5

範例輸出:

*CDC is the trademark of the Control Data Corporation.

*IBM is a trademark of the International Business Machine Corporation.

提示：

ord(x)： x 字元的 ascii 數值

chr(t)：數值 t 對應的 ascii 字元

b428: 凱薩加密

在西元前就存在的一種加密－**凱薩加密**為目前最早發現的替換加密 (substitution cipher)。其原理很簡單，將一段明文往替換成往後數的第 k 個英文字母。

若用數學式表示凱薩加密和解密，如下：

加密 C=EK(P)=(P+k)mod26

解密 P=DK(P)=(C−k)mod26

例如 k=3 時，發生的情況如下：

明文字母表：ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文字母表：DEFGHIJKLMNOPQRSTUVWXYZABC

從數學的觀點來看，每一個字母就是一個數字。

A = 0, B = 1, C = 2, ...,X = 23, Y = 24, Z = 25

輸入說明 ：

多組測資，每一組測資兩行，分別是明文和經過凱薩加密的密文，保證明文和密文長度相同，長度小於等於 1000 大於 0 且只由大寫英文字母構成。

輸出說明 ：

對於每組測資輸出一行，凱薩加密使用的 k 值為何，其 0≤k<26

範例輸入 ：

ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
DLQXABXEEQMEUQYLZPEK
YGLSVWSZZLHZPLTGUKZF
Z
Z

範例輸出:

3
21
0

c109: 00306 – Cipher

Bob 和 Alice 開始使用一種全新的編碼策略。令人訝異的他們並未採用公開金鑰密碼系統(Public Key Cryptosystem），而是採用秘密鍵（secret keys）的方式來編碼及解碼。在 1996 年 2 月 16 日在費城他們上次見面的時候，他們選定了他們的秘密鍵。這些秘密鍵是由 1 到 n 的整數所構成，但是排列 的順序卻是他們任意挑選的。在編碼的時候採用以下的原則：

把要編碼的訊息（明文）寫在秘密鍵下面，每個字元與秘密鍵的一數字對齊。位於位置 i 的字元經編碼後其位置為 a_i，a_i 為秘密鍵中第 i 個位置的值。明文中的每個字元編碼後就得到密文了。這密文還可以用同樣的策略再加密，經過了 k 次加密後他們交換他們的密文。

明文的長度一定小於等於 n。如果明文的長度小於 n，就在其後方加上空白字元使得其長度剛好為 n。你的任務是幫助 Bob 和 Alice 寫一個程式讀入秘密鍵，以及 k，以及一連串要加密的明文，然後輸出密文。

輸入說明 ：

輸入含有多組測試資料，每組測試資料的第一列有一個整數 n（0 < n <= 200），代表秘密鍵的長度。接下來的一列為秘密鍵，含有 n 個整數，內容為 1 到 n 的某種排列。再接下來的每列為 k 與明文，其中以一空白字元分隔。請注意：每列以換列符號(End of Line)當作結束，但是換列符號不列入明文。當遇到 k=0 的一列時代表此組測試資料結束。另外，k 可能相當大（但是不會超出 C 語言中 int 的範圍），所以請注意你程式的演算法，否則可能導致 Time Limit Exceeded.

當遇到 n=0 時代表整個輸入結束。請參考 Sample Input。

輸出說明 ：

對每一組測試資料中的每列明文，輸出加密後的密文（長度一定為 n）。每組測試資料後請空一列。請參考 Sample Output。

範例輸入 ：

10
4 5 3 7 2 8 1 6 10 9
1 Hello Bob
2 Hello Bob
1995 CERC
0
5
1 2 3 5 4
1 Hello
0
0

範例輸出:

BolHeol b
lelBo Hob
C RCE

Helol

c044: 10008 - What's Cryptanalysis

內容 ：

密碼翻譯（cryptanalysis）是指把某個人寫的密文（cryptographic writing）加以分解。這個程序通常會對密文訊息做統計分析。你的任務就是寫一個程式來對密文作簡單的分析。

輸入說明 ：

輸入的第 1 列有一個正整數 n，代表以下有多少列需要作分析的密文。 接下來的 n 列，每列含有 0 或多個字元（可能包含空白字元）

輸出說明 ：

每列包含一個大寫字元（A~Z）和一個正整數。這個正整數代表該字元在輸入中出現的次數。輸入中大小寫（例如：A 及 a）視為相同的字元。輸出時請按照字元出現的次數由大到小排列，如果有 2 個以上的字元出現次數相同的話，則按照字元的大小（例如：A 在 H 之前）由小到大排列。 請注意：如果某一字元未出現在輸入中，那他也不應出現在輸出中。

範例輸入 ：

3
This is a test.
Count me 1 2 3 4 5.
Wow!!!! Is this question easy?

範例輸出:

S 7
T 6
I 5
E 4
O 3
A 2
H 2
N 2
U 2
W 2
C 1
M 1
Q 1
Y 1

Python 3.1 快速導覽 - 內建字串型態 (string)

字串 (string) 屬於不可變 (immutable) 的序列 (sequence) 型態. 可進行以下序列通用的計算

計算	描述
x in s	判斷 x 是否在 s 中
x not in s	判斷 x 是否不在 s 中
s + t	連接 s 及 t
s * n, n * s	將 s 重複 n 次連接 s 本身
s[i]	取得索引值 i 的元素
s[i:j]	取得索引值 i 到 j 的子序列
s[i:j:k]	取得索引值 i 到 j ，間隔 k 的子序列
len(s)	回傳 s 的元素個數
min(s)	回傳 s 中的最小值
max(s)	回傳 s 中的最大值
s.index(i)	取得 s 中第一次出現 i 的索引值
s.count(i)	累計 s 中 i 出現的個數

字串型態有以下的方法 (method)

方法	描述
str.capitalize()	回傳將 str 改成首字母大寫，其餘字母小寫的字串
str.center(width[, fillchar])	回傳一個將 str 設置字串中央，長度 width 的新字串，fillchar 為填充字元，預設為空格
str.count(sub[, start[, end]])	計算 sub 出現的次數， start 為起始計算索引值， end 為結束索引值
str.encode(encoding="utf-8", errors="strict")	回傳 encoding 版本的 bytes 物件
str.endswith(suffix[, start[, end]])	判斷 str 是否以 suffix 結尾
str.expandtabs([tabsize])	將 tab 符號以 tabsize 的空格數替換
str.find(sub[, start[, end]])	回傳 sub 第一次出現的索引值
str.format(*args, **kwargs)	進行格式化字串運算
str.index(sub[, start[, end]])	回傳 sub 第一次出現的索引值
str.isalnum()	判斷字串中的字元是否都是字母或數字
str.isalpha()	判斷字串中的字元是否都是字母
str.isdecimal()	判斷字串中所有字元是否是十進位數字
str.isdigit()	判斷字串中所有字元是否是數字

方法	描述
str.isidentifier()	判斷字串是否可作為合法的識別字
str.islower()	判斷字串中所有字母字元是否都是小寫字母
str.isnumeric()	判斷字串中所有字元是否是數字
str.isprintable()	判斷字串中所有字元是否都屬於可見字元
str.isspace()	判斷字串是否為空格字元
str.istitle()	判斷字串是否適合當作標題
str.isupper()	判斷字串中所有字母字元是否都是大寫字母
str.join(iterable)	回傳將 str 連結 iterable 各元素的字串
str.ljust(width[, fillchar])	回傳將 str 在寬度 width 向左對齊的字串， fillchar 為填充字元，預設為空格
str.lower()	將 str 的英文字母都改成小寫
str.lstrip([chars])	回傳將 str 左邊具有 chars 字元去除的拷貝版本， chars 預設為空格符號
static str.maketrans(x[, y[, z]])	回傳 x 與 y 配對的 Unicode 編碼字典，若有提供 z ，z 中的字元會跟 None 配對
str.partition(sep)	以 sep 分割 str 為三個部份，結果回傳具有三個子字串的序對
str.replace(old, new[, count])	將 str 中的 old 子字串以 new 代換
str.rfind(sub[, start[, end]])	尋找最右邊的 sub ，也就是索引值最大的 sub
str.rindex(sub[, start[, end]])	尋找最右邊的 sub ，也就是索引值最大的 sub
str.rjust(width[, fillchar])	回傳將 str 在寬度 width 向右對齊的字串， fillchar 為填充字元，預設為空格
str.rpartition(sep)	以 sep 從最右端分割 str 為三個部份，結果回傳具有三個子字串的序對
str.rsplit([sep[, maxsplit]])	將 str 從最右端以 sep 分割成子字串，回傳儲存子字串的串列， maxsplit 為子字串最多的數量
str.rstrip([chars])	從 str 的最右端中移除 chars 字元，預設為空白字元
str.split([sep[, maxsplit]])	將 str 以 sep 分割成子字串，回傳儲存子字串的串列，maxsplit 為子字串最多的數量
str.splitlines([keepends])	將 str 以新行符號分割成子字串，回傳儲存子字串的串列
str.startswith(prefix[, start[, end]])	判斷 str 是否以 prefix 開頭
str.strip([chars])	從 str 中移除 chars 字元，預設為空白字元
str.swapcase()	將 str 中的英文字母進行大小寫轉換
str.title()	將 str 轉換成作為標題的字串
str.translate(map)	將 str 中的字元以 map 中配對的字元轉換
str.upper()	將 str 的英文字母都改成大寫
str.zfill(width)	回傳以 0 塞滿 width 的新字串

Python 3.1 快速導覽 - 內建串列型態 (list)

串列 (list) 屬於可變 (mutable) 的序列 (sequence) 型態，可進行以下序列通用的計算

計算	描述
x in s	判斷 x 是否在 s 中
x not in s	判斷 x 是否不在 s 中
s + t	連接 s 及 t
s * n, n * s	將 s 重複 n 次連接 s 本身
s[i]	取得索引值 i 的元素
s[i:j]	取得索引值 i 到 j 的子序列
s[i:j:k]	取得索引值 i 到 j ，間隔 k 的子序列
len(s)	回傳 s 的元素個數
min(s)	回傳 s 中的最小值
max(s)	回傳 s 中的最大值
s.index(i)	取得 s 中第一次出現 i 的索引值
s.count(i)	累計 s 中 i 出現的個數

由於串列是可變的複合資料型態 (compound data type) ，也是 Python 中大量運用的工作型態種類，因此有額外以下的計算

計算	描述
s[i] = x	將索引值 i 的元素指派為 x
s[i:j] = t	將索引值 i 到 j 的元素指派為 t ， t 為迭代器
del s[i:j]	刪除索引值 i 到 j 的元素
s[i:j:k] = t	將索引值 i 到 j ，間隔 k 的元素指派為 t ， t 為迭代器
del s[i:j:k]	刪除索引值 i 到 j ，間隔 k 的元素

串列型態有以下的方法 (method)

方法	描述
list.append(x)	將 x 附加到 list 的最後
list.extend(x)	將 x 中的元素附加到 list 的最後
list.count(x)	計算 list 中 x 出現的次數
list.index(x[, i[, j]])	回傳 x 在 list 最小的索引值
list.insert(i, x)	將 x 插入 list 索引值 i 的地方
list.pop([i])	取出 list 中索引值為 i 的元素，預設是最後一個
list.remove(x)	移除 list 中第一個 x 元素
list.reverse()	倒轉 list 中元素的順序
list.sort([key[, reverse]])	排序 list 中的元素