《程式語言》

一、Java與Python是目前被廣泛使用的程式語言，試就下列特性比較它們之間的不同，包括Typing、Coding（撰寫程式之難易）、執行環境、程式執行效能、開發App的難易度。（15分）

試題評析	本題比較Python與Java語言，屬105年地特相似題，當年比較的是Python與C語言。相較於以往的考題，本題必須依照題意分項進行兩種語言比較，因此雖然比較有方向，但因限制了範圍可能讓考生更難下筆。本題應依照題意繪製表格分項比較，不僅要寫出簡答，例如較容易、效能較高，還必須要解釋背後之理由，才能獲得高分。
考點命中	《高點・高上程式語言講義》第三回，金乃傑編撰，頁74。 《高點・高上程式語言講義》第五回，金乃傑編撰，頁61-64。

答：

依題意比較Java與Python語言之差異如下表：

	Java	Python
型態	使用動態型態繫結（Dynamic type binding）、弱型態語言（Weak Typing），變數與型態在執行程式時才繫結，且執行中繫結可以再改變，亦即一變數可以在執行不同時期擁有不同型態。	使用靜態型態繫結（Static type binding），變數與型態在執行前繫結，並在執行中宣告的型態不會改變。但值得一提的是，若使用多型（Polymorphism），可以使物件實際型態與宣告不同，但僅限於使用宣告時的子類別。
可寫性	1.撰寫容易。 2.由於變數不必事先宣告，在需要時直接使用即可，並能在使用中不斷改變型態，符合初學者直覺。 3.提供豐富函式庫，並有許多簡化語法，讓程式結構精簡並能達到完整功能。不過由於相同的功能可以用許多不同語法達成，因此可以說是入門容易但要寫得好並不容易。	1.難度較高。 2.語法嚴謹，並承襲C/C++的語法結構，在使用變數前均需要宣告，並在特定敘述時還需要加上例外處理語法。 3.簡化的語法較少，程式結構較制式。 4.提供強大函式庫，在功能上非常完整，因此入門難度高，但熟悉後幾乎可處理所有任務需求。
執行環境	透過直譯器執行，幾乎可以執行在任何系統上，且通常需要的系統資源較少。	使用混和式編譯，先將Java語法編譯成位元碼（Byte Code），在透過系統上的虛擬機器（JVM）執行，通常需要之資源較多。
效能	使用直譯器效能受到影響，但由於許多重要函數及外掛使用C/C++開發，因此在執行效能上比其他直譯語言來得出色。	由於仍有直譯階段，效能受JVM限制，通常並不優異。
開發App	通常用於開發資料分析相關程式，如網路爬蟲、斷詞統計等，提供多種功能，可開發網頁應用程式或單機程式，但通常不會以Python作為主要開發手機應用程式之語言。	能夠以Java語法開發Android手機應用程式，並擁有大量範例程式可參考，因此要完成一般需求的App門檻不高。

二、諾姆・荷姆斯基（Noam Chomosky）定義了type-0、type-1、type-2、type-3四種語法類型，請說明各類型可接受之語言（Language Accepted）是什麼？相對應之自動機（Automaton）是什麼？（20分）

試題評析	本題屬於程式語言文法相關的背景知識，是BNF文法的基礎，屬於記憶型題目。由於本題有標準答案，因此考生僅需依照題意撰寫表格依序填入作答即可。
考點命中	《高點・高上程式語言講義》第六回，金乃傑編撰，頁1-2。

答：

根據Chomosky所定義，各type是層層疊加的，type-0是機器能接受中限制最鬆散的語言，type-3是最嚴格的語言，因此type-3是type-2的子集合、type-2是type-1的子集合而type-1是type-0的子集合。各可接受語言及對應之自動機說明如下表：

文法	可接受之語言	對應之自動機
type-0	遞迴可計數（recursive enumerable）	圖靈機
type-1	上下文有關（context-sensitive）語言	線性界線自動機
type-2	上下文無關（context-free）語言	下推自動機
type-3	正規（regular）語言	有限狀態自動機

三、(一)請解釋程式語言中之變動態範圍（Dynamic Scope）與語句參考環境（Referencing Environment）。（6分）

(二)以下為一動態範圍之程式，請分別列出a,b,c三點之參考環境。（9分）

```
void      subprogram1(){
        int  X,Y;
        ...      <-------------a
        }    /* end of subprogram1
void      subprogram2(){
        int  Y,Z;
        ...      <-------------b
        }    /* end of subprogram2
void    main(){
        int  W,Z;
        ...      <-------------c
        }    /* end of main
```

試題評析	本題考點為領域的概念，相似於101年關務四等的程式語言題目。前面6分為名詞解釋，由於配分不高，不需要花過多篇幅說明，僅需要提出重點即可；後面9分則需要看程式碼作答。 值得注意的是，本題之虛擬碼就一般程式語言觀點，因為只有宣告而沒有呼叫，僅能判斷會執行main()函數，而subprogram1()跟subprogram2()都不會執行，故就無所謂參考環境。 因此為符合題意，應在作答前先以合理假設其呼叫順序，再根據假設作答。由於本題須使用「動態範圍」方式，故變數之取值若無區域變數定義，會根據動態鏈往呼叫者去尋找定義。
考點命中	《高點・高上程式語言講義》第三回，金乃傑編撰，頁93-94。

答：

(一) 動態範圍(Dynamic Scope)：又稱流動繫結法（fluid binding），是一種定義變數在程式本文中可以用變數的名稱存取記憶體位址範圍的方法，對於區段內沒有宣告的變數，往呼叫方向反向尋找定義。因此變數名稱所代表的記憶體位址會因為執行順序改變，新潮的程式語言多用此方法，如LISP、Perl、Logo。

(二) 語句參考環境(Referencing Environment)：當副程式被呼叫時所建立的環境，用以提供變數的定義讓程式得以執行。一般來說一程式會先將收到的實際參數、區域變數加入參考環境中，在執行運算式時就會根據參考環境中變數的定義提取其值，若運算式中使用到的變數沒有在參數或區域變數中，會根據所使用的語言定義，往結構上層的全域變數或呼叫者的動態鏈方向尋找定義。

(三) 假設此程式main()呼叫subprogram2()；subprogram2()呼叫subprogram1()。以下說明a、b與c三點之參考環境：

	W	X	Y	Z
a	main()	subprogram1()	subprogram1()	subprogram2()
b	main()	不可見	subprogram2()	subprogram2()

c	main()	不可見	不可見	main()

四、在一concurrent環境下，有一共享變數X其初始值為1，程序（process）A必須加2到X，程序B必須將X乘以4。A與B都要執行三個動作1.讀取X；2.進行算數運算；3.將算好的值寫回X。試列出最後可能得到的X值。（20分）

試題評析	本題為同步問題，相較於以往以撰寫程式或看程式寫輸出，本題僅是要求考生依照同步問題會有的遺失更新列出所有可能的結果。因此僅需在列出結果後，以圖示方式說明其計算過程，應不難拿分。
考點命中	《高點・高上程式語言講義》第三回，金乃傑編撰，頁131-132。

答：

最後可能得到的X值有4種可能：12、6、4或3，將原因說明如下：

(一) A執行完換B執行：

	時間→					
X值	1	1	3	3	3	12
A程序	讀取1	運算 1 + 2 = 3	寫回3			
B程序				讀取3	運算 3 * 4 = 12	寫回12

(二) B執行完換A執行：

	時間→					
X值	1	1	4	4	4	6
A程序				讀取4	運算 4 + 2 = 6	寫回6
B程序	讀取1	運算 1 * 4 = 4	寫回4			

(三) B在A寫入後覆蓋寫入結果：

	時間→					
X值	1	1	1	3	3	4
A程序	讀取1		運算 1 + 2 = 3		寫回3	
B程序		讀取1		運算 1 * 4 = 4		寫回4

(四) A在B寫入後覆蓋寫入結果：

	時間→					
X值	1	1	1	3	4	3
A程序	讀取1		運算 1 + 2 = 3			寫回3
B程序		讀取1		運算 1 * 4 = 4	寫回4	

五、假設有一C++中的namespace，稱之為MyStack，該namespace中有一變數topPtr，試列出參考該變數的三種方式。（15分）

試題評析	本題考點為C++中的基本觀念namespace，為解決多檔案專案中程式變數相衝突的問題，但在大多數物件導向的教材中因可利用封裝解決此問題而較少被討論。本題需依照題意列出三種考察的方式，並以片段程式碼說明其實作方法。
考點命中	《高點・高上程式語言講義》第四回，金乃傑編撰，頁116-117。

答：

依照題意，假設MyStack宣告在MyStack.h檔案中，列出參考topPtr的三種方式：

1. 使用using namespace 名稱空間：

```
#include "MyStack.h"
using namespace MyStack;

int main(){
  int i;
  topPtr = &i;
}
```

2. 使用using 名稱空間::成員：

```
#include "MyStack.h"
using MyStack::topPtr;

int main(){
  int i;
  topPtr = &i;
}
```

3. 直接在需要參考變數時使用範圍解析運算子::：

```
#include "MyStack.h"

int main(){
  int i;
  MyStack::topPtr = &i;
}
```

六、試說明SQL與NoSQL，包括資料庫結構、資料庫可擴展性。並列舉它們的優點各三項。（15分）

試題評析	本題為資料庫相關題型，測驗考生對SQL與NoSQL的了解。考生需要依照題意繪製表格，並根據題目中的比較項目進行說明，並列出各三項優點，才能滿足題目要求。
考點命中	《高點・高上程式語言講義》第六回，金乃傑編撰，頁66。

答：

將SQL與NoSQL資料庫比較如下表：

	SQL資料庫	NoSQL資料庫
內容	是建立在關聯模型基礎上的資料庫，藉助於集合代數等數學概念和方法來處理資料庫中的資料。現如今雖然對此模型有一些批評意見，但它還是資料儲存的傳統標準。標準資料查詢語言SQL就是一種基於關聯式資料庫的語言，這種語言執行對關聯式資料庫中資料的檢索和操作。主流的資料庫如Microsoft SQL Server、Oracle SQL及MySQL都屬於關聯	是一種新型態的資料庫技術，對於此名詞有許多不同的解釋，但大多數學者認同NoSQL是Not Only SQL的簡稱。NoSQL有別於傳統的關聯式資料庫在資料表中有主鍵、外鍵對應對應，NoSQL不使用關聯，根據型態可分為四種：Key-Value資料庫，欄導向資料庫（Column-oriented Database）、文件導向資料庫（Document-oriented Database）以及圖學資

	式資料庫。	料庫（Graph Database）。
特色	1.原子性（Atomicity）：一個交易中的所有操作，只有全部完成或全部沒完成，如果失敗會被回滾（Rollback）而不會卡在執行一半。 2.一致性（Consistency）：在交易開始之前和交易結束以後，資料庫的完整性沒有被破壞。 3.隔離性（Isolation）：可以防止多個交易並發生執行時由於交叉執行而導致資料的不一致。。 4.持久性（Durability）：交易處理結束後，對資料的修改就是永久的，即使系統故障也不會丟失。	1.核心可用性（Basically Available）：指分散式系統在出現故障或負載過重時，只保證核心可用性，允許損失部分進階功能。 2.允許同步延遲（Soft-state）：允許系統存在中間狀態，而該中間狀態不會影響系統整體可用性。 3.最終一致性（Eventual Consistency）：所有資料副本經過一定時間後，最終能夠達到一致的狀態。
優點	1.保持資料的一致性：在同時執行多個交易後都能確保資料是一致的，不會有遺失寫入等情形。 2.資料更新成本低：由於資料庫分欄的設計，讓更新欄位的數值時只要做部分的寫入，因此更新速度快。 3.可進行Join等複雜的查詢：資料庫結構提供主鍵、外鍵，可將不同資料表合而為一進行查詢或更新。	1.資料欄位可輕易擴展：不須預先定義好資料欄位，資料中的每筆記錄都可能有不同的屬性和格式。 2.彈性可擴展：可以在系統運行時動態增加或刪除結點；資料不需要集中儲存於中央伺服器，可將資料劃分後儲存在local端，加速處理。 3.非同步複製：資料可以儘快地寫入一個節點，而不會被網路傳輸引起遲延。