- Ex2-1Add1: Python 與 JavaScript 有那些相似處？有那些相異處，至少各舉出三點以上。
- Ex2-1Add2: JavaScript 的主要規範是什麼？簡介之。
- Ex2-1Add3: 有關 Python 增強規範(或稱為 Python 改進建議書)是什麼？簡介之。
- Ex2-1Add4: JavaScript 在 ES6 加入新符號 「=>」作用是什麼？ Python 在 3.0 加入「->」作用與 JS 不同是什麼？簡單說明之。
- Ex2-1Add5: 2015 的年底Python3.5 增加兩個關鍵字 async,await。有趣的情況，在 2016 的ES7, JavaScript 也有加入新關鍵字 async 和 await。 請簡單說明其作用。

Ex2-1Add1: Python 與 JavaScript 有那些相似處？有那些相異處，至少各舉出三點以上。
Ans:
相似處：
都不需經過編譯，就可執行。（so,它們是直釋式語言(interpreter))
都有互動式直釋器，所以在逐行輸入程式碼行時，可立即查看執行結果。
兩者都有垃圾回收(garbage collection)機制。
兩種語言都沒有標頭檔，無謂的重複樣板檔等等。
兩者都只要文字編輯器(text editor)就可進行開發工作，而非一定要有 IDE 才能開發。
這兩種語言，「函數都是一等公民(first-class citizens)」，所以函數可當作參數，在函數之間傳遞。
相異處：
差異最大的地方是 JavaScript 是使用單一執行緒執行環境、非阻斷式函數呼叫(single-threaded and non-blocking)，非同步式 I/O (asynchronous I/O) 。意味著，就像是簡單的動作，如檔案存取，都會有回呼函式(callback function)。
JS 原本是只出現在Web 開發的，直到近幾年(有了node.js 之後)才逐漸脫離瀏覽器的牢籠。而Python 則在各個領域皆可見到其身影。
JS 是網路瀏覽器上的唯一語言，Python 則是就算有瀏覽器翻譯器，也還會有很多問題。
Python 具有完善的標準庫，而 JS 則僅有限(極為受限)的輔助物件程式（例如 JSON ， Math)。
Python 具有相當典型的物件導象的類別(object-oriented classes)，而 JS 則沒有類別(classless)，是以原型為基礎(Prototype-based programming)去定義物件的鏈結關係。
JS 相比於 python，只有較少的通用數據處理庫(general-purpose data-processing)。

Ex2-1Add2: JavaScript 的主要規範是什麼？簡介之。
Ans:
在 1996 年，一個名為 ECMA（歐洲計算機製造商協會）國際標準組織製定了一個名為 ECMAScript（ES）的標準規範，所有瀏覽器供應商都可以實施。
關於這個手稿語言規範：
在 2009 年 12 月：發表 ES5. 所以是長時期大量被使用的規範。
在 2015 年 6 月：發表 ES6 (ES2015, ECMAScript 2015)，此版本具有向下相容, 而且開發很多新的寫法。例如：開始添加了類和模組的語法，但是其中的 class 其實是語法糖，本質仍相同，是 prototype-based.
2016~2020 每年都公佈新加概念和語言特性的規範 ES7, ES8, ES9,Es10,ES11

Ex2-1Add3: 有關 Python 增強規範(或稱為 Python 改進建議書)是什麼？簡介之。
Ans:
第一個 PEP 誕生於 2000 年，由 python 官網，可查詢：到 2018 年為止，已擁有 478 個“兄弟姐妹”，到 2020 年(編號到 PEP8101)已擁有 528 個兄弟姐妹。
其中的 PEP8 -- Style Guide for Python Code (Python 程式碼風格指引) 是編碼必讀的規範，以下只列舉較重要的幾點：
關於程式碼編排:
縮進規範：PEP 8 規範告訴我們，請選擇四個空格的縮進，不要使用 Tab，更不要 Tab 和空格混用。第二個要注意的是，每行最大長度請限制在 79 個字符。

Ex2-1Add4: JavaScript 在 ES6 加入新符號 「=>」作用是什麼？ Python 在 3.0 加入「->」作用與 JS 不同是什麼？簡單說明之。
Ans:
※有趣的新指令「箭頭」=> -> ??
JS 的箭頭函式運算式(=>)（arrow function expression）擁有比函式運算式還簡短的語法。
本函式運算式適用於非方法的函式，但是要小心，因為 this 會固定第一次呼叫，所以不能被用作建構式（constructor）。
例如：

```
var mm = [
   'a', 'bbb', 'ccccc', 'dd'
];
console.log(mm.map(mm => mm.length));
//out: Array [1, 3, 5, 2]
```

※ 2006 年的 PEP 3107 -- Function Annotations
設計了 -> (單線箭頭) 來代表返回值的型態！
說明：Python 不像 JavaScript 一樣使用箭頭符號=> ，而是單線節頭，是一個返回值註釋，它是函數註釋的一部分，它是 Python3.0 開始就在這裡！
Return Values :
def sum() -> expression: ...
例如：

```
def fff(x) -> int: # 此處的單線箭頭，是說明 fff 返迴值的型態！ 只是讓設計師參考用！
      x=x*2
      return x
print(fff(100))
print(type(fff(100)))
def fff(x) -> float: # ※故意寫錯！ 因為只讓設計師參考用，所以並沒作太多的正確性檢查！
      x=x*2
      return x
print(fff(100))
print(type(fff(100)))
####out
200 <class 'int'>
200 <class 'int'>
```

Ex2-1Add5: 2015 的年底 Python3.5 增加兩個關鍵字 async,await。有趣的情況，在 2016 的 ES7, JavaScript 也有加入新關鍵字 async 和 await。 請簡單說明其作用。

Ans:

基本上，asnyc 關鍵字是定義一個非同步函數，在這個函數裡面才可寫 await 的關鍵字，表示 await 後的函數，要等待到完成，才會執行下一行程式。

兩種語言的關鍵字，都有類似的設計。實際應用例子都較複雜，請參考其他資料。

補充：==========

```
var delay = (s) => {
   return new Promise(resolve =>
      {setTimeout(resolve,s);
   });
};
delay().then(() =>
   { console.log(1);      // 顯示 1
   return delay(1000); // 延遲丨秒
}).then(() => {
   console.log(2);        // 顯示 2
   return delay(2000); // 延遲二秒
}).then(() => {
   console.log(3);        // 顯示 3
});
//////////////
//用 async 定義函數，然後在其中，使用 await 強迫阻塞，直到完成，才會執行下一行。
var go=async function () {                  // ~ 開頭
   var delay = (s) => {
      return new Promise(function(resolve){   // 回傳一個 promise
         setTimeout(resolve,s);                    // 等待多少秒之後 resolve()
      });
   };
   //////下方的程式碼，會變的比較清楚！！
   console.log(1);          // 顯示 1
   await delay(1000);     // 延遲丨秒
   console.log(2);          // 顯示 2
   await delay(2000);     // 延遲二秒
   console.log(3);          // 顯示 3
};
///undefined
go();
///1
///Promise {<pending>}__proto__: Promise[[PromiseState]]: "fulfilled"[[PromiseResult]]: undefined
///2
///3
```