Lab 4- Các kiểu dữ liệu Python (Addition)

1. Mục đích
   - Làm quen với các bài lập trình liên quan đến một số cấu trúc dữ liệu hàm.
2. Khi hoàn thành Lab04, sinh viên có thể:
   - Có thể phân tích và xây dựng được hàm. Lập trình các bài toán có liên quan đến cấu trúc dữ liệu và hàm, …
3. Một số bài lập trình

   3.1. Viết hàm nhận tham số là hai danh sách **a** và **b** và trả về danh sách các tập hợp chứa: (**a** giao với **b**, **a** hợp với **b**, **a - b**, **b - a**)

   Hint:

   - Để giao hai danh sách, sử dụng lệnh for: **`a_giao_b = [value for value in list1 if value in list2]`**
   - Cách 2: để giao hai danh sách, ta biến đổi list thành set, rồi thực hiện phép giao (intersection hoặc &) của set.
   - Cách 3: Để giao hai danh sách, sử dụng lệnh sau: **`a_giao_b = [list(filter(lambda x: x in lst1, sublist)) for sublist in lst2]`**
   - Sử dụng các hàm **`symmetric(), symmetric_difference()`** cho các chức năng khác.

   3.2. Viết hàm nhận một chuỗi làm tham số và trả về một từ điển trong đó các khóa là các ký tự trong chuỗi ký tự và các giá trị là số lần xuất hiện của ký tự đó trong văn bản đã cho.

   Ví dụ: Nhập chuỗi S=**`"Ana has apples."`** là tham số, hàm sẽ trả về từ điển: **`{'a': 3, 's': 2, '.': 1, 'e': 1, 'h': 1, 'l': 1, 'p': 2, ' ': 2, 'A': 1, 'n': 1}`**

   Hint:

   - Cho biến ch "scan" tập hợp các ký tự trong chuỗi S, **`dictionary[ch]=S.count(ch)`**. Nên nhớ rằng, muốn chuyển chuỗi ký tự thành tập hợp, sử dụng hàm **`set()`**.
   - Sử dụng vòng lặp ***`for`*** cho tập hợp các ký tự trong chuỗi S: **`{ch:S.count(ch) for ch in set(S)}`**
   - Nếu sử dụng nhiều hơn một phương pháp để trả về kết quả, sử dụng phép so sánh "==" để xác định 2 từ điển có giống nhau hay không?

   3.3. Viết hàm nhận tham số là một danh sách các số nguyên và trả về một bộ dữ liệu (***a, b***), với ***a*** biểu thị số phần tử duy nhất trong danh sách và ***b*** biểu thị số phần tử trùng lặp trong danh sách.

   Ví dụ: `intList=[1, 2, 3, 4, 5, 6, 2, 3, 2, 4, 4, 5, 5, 7, 8]` → `([1, 6, 7, 8], [2, 3, 4, 5])`.

   Hint:

   - Sử dụng hàm tập hợp set() để các định các phần tử khác nhau trong list. Ví dụ: **`set(intList)`** → **`[1, 2, 3, 4, 5, 6, 7, 8]`**
   - Sử dụng vòng lặp *for* để duyệt qua các phần tử với điều kiện phần tử đó duy nhất. List các phần tử duy nhất: **`[i for i in set(intList) if intList.count(i)==1]`**

- Sau khi đã xác định được các phần tử duy nhất, các phần từ không duy nhất sẽ bằng các phần tử khác nhau trong list loại bỏ đi các phần tử duy nhất (sử dụng các hàm trong set: **`symmetric()`**, **`symmetric_difference()`**, …).
- Bonus: Xác định phần tử xuất hiện nhiều nhất trong list? Ví dụ: `intList = [1, 2, 3, 4, 5, 6, 2, 3, 2, 4, 4, 5, 5, 7, 8]` → `[2, 4, 5]`

3.4. Viết hàm *bin2dec* biến đổi chuỗi nhị phân thành số thập phân (không sử dụng hàm thư viện). Ví dụ *bin2dec*('1010') → 10.

Hint:

- Sử dụng hàm pow(x, y) → x^y.
- Sử dụng vòng lặp *for* để duyệt qua các ký tự chử số trong chuỗi với điều kiện ký tự đó bằng '1' thì sử dụng hàm pow(x, y). Ví dụ: bin2dec('1010') → sum([2^3, 2^1])