Lab 3 – Các kiểu dữ liệu Python

1. Mục đích
   - Làm quen với các bài lập trình liên quan đến các cấu trúc dữ liệu string, list, set,…
2. Khi hoàn thành Lab02, sinh viên có thể:
   - Phân tích và lập trình các bài toán có liên quan đến các cấu trúc dữ liệu Python.
3. Một số lệnh cơ bản Python: đọc hiểu và thực hiện một phần câu lệnh nếu chưa rõ.
   https://www.freecodecamp.org/news/python-code-examples-sample-script-coding-tutorial-for-beginners/

   Lưu ý: Giáo viên có thể hỏi bất kỳ câu hỏi nào trong phần này.
4. Một số bài lập trình liên quan đến ác kiểu dữ liệu: *string, set, list, dictionary, …*

   4.1. Viết chương trình đảo ngược các từ trong một chuỗi. Ví dụ s = "geeks quiz practice code" → "code practice quiz geeks"
   Phân tích:
   - Sử dụng hàm `split()` để phân tách từng từ trong một chuỗi. Ví dụ: tách chuỗi và đảo ngược → `s = string.split()[::-1]`
   - Đảo ngược danh sách: sử dụng stack hoặc hàm đảo ngược.
   - Nối mỗi từ trong một list string cách bằng phương thức " ".join() trong python.

```
reverse_sentence = ''.join(reversed_words)
reverse_sentence = ''.join(words[i] for i in range(len(words)-1, -1, -1))
```

   4.2. Viết chương trình đảo ngược các từ trong một list các chuỗi ký tự. Ví dụ:
   `fruits = ['banana', 'orange', 'mango', 'lemon']` → `lemon mango orange banana`
   Hint: Tìm hiểu hàm `range(len(fruits)-1, -1, -1)`

   4.3. Viết chương trình Python để đếm số nguyên âm trong một chuỗi cho trước.
   Ví dụ: `s = "GeeksforGeeks"` → tổng số các ký tự nguyên âm là 5.
   Hint:
   - `vowels = "aeiouAEIOU"` → các ký tự nguyên âm (trong tiến Anh)
   - Sử dụng phương thức *count()*

```
count = sum(s.count(vowel) for vowel in vowels)
```

   - Sử dụng danh sách: `count=len([char for char in str if char in vowels])`
   - Sử dụng biểu thức ReGex:

```
vowels = r'[aeiouAEIOU]'
count = len(re.findall(vowels, string))
```

   4.4. Viết chương trình Python để tìm các từ có độ dài lớn hơn một số k cho trước.
   Hint:
   - Sử dụng list để collect các từ có độ dài lớn hơn k

```
print([word for word in sentence.split() if len(word) > length])
```

   - Sử dụng hàm lambda: `list(filter(lambda x: (len(x) > K), s))`
   - Sử dụng hàm enumerate: `print([a for i, a in enumerate(s) if len(a) > length])`

4.5. In tất cả các kết hợp có thể có từ ba chữ số. Input: [1, 2, 3], Output: 1 2 3, 1 3 2, 2 1 3, 2 3 1, 3 1 2, 3 2 1.

Hint:

- Sử dụng Brute force (3 vòng lặp) hoặc sử dụng hàm *permutations()*
- Load hàm permutation: `from itertools import permutations`
- Hàm `permutations([1, 2, 3], 3)` trả về một tuple gồm các giá trị hoán đổi vị trí của ba phần tử trong danh sách [1, 2, 3].

4.6. Viết chương trình kiểm tra tính hợp lệ của mật khẩu người dùng nhập vào. Sau đây là các tiêu chí để kiểm tra mật khẩu:

1. Ít nhất 1 chữ cái giữa [a-z]
2. Ít nhất 1 số trong khoảng [0-9]
1. Ít nhất 1 chữ cái giữa [A-Z]
3. Ít nhất 1 ký tự từ [$#@]
4. Độ dài mật khẩu giao dịch tối thiểu: 6
5. Độ dài tối đa của mật khẩu giao dịch: 12

Chương trình của bạn phải chấp nhận một chuỗi mật khẩu được phân tách bằng dấu phẩy và sẽ kiểm tra chúng theo các tiêu chí trên. Mật khẩu phù hợp với tiêu chí sẽ được in, mỗi mật khẩu được phân tách bằng dấu phẩy.

Ví dụ: Nếu các mật khẩu sau được cung cấp làm đầu vào cho chương trình: ABd1234@1,a F1#,2w3E*,2We3345

Khi đó, đầu ra của chương trình sẽ là: ABd1234@1

Hint:

- Để tách các từ trong một chuỗi S cách nhau bởi dấu phẩy, sử dụng hàm `split(',')`
- Để biết được một trong các ký tự từ a-z có trong chuỗi S nào đó, ta dùng hàm `re.search("[a-z]",S)`. Để sử dụng được hàm search, ta phải `import re`
- Hàm `re.search("[a-z]","aBBB")` đối tượng `re.Match` nếu tìm thấy một trong những ký tự từ a-z, ngược lại sẽ trả về `None`, ví dụ: `re.search("[a-z]","ABBB")`. Đối tượng `re.Match` tương đương True trong lệnh điều kiện ***if***. None tương đương False trong lệnh ***if***.
- Hàm `len` để xác định độ dài một chuỗi ký tự.