Lab 5 – Functions, Module, Class, Regular Expression

1. Mục đích
   - Làm quen với các bài lập trình liên quan đến Hàm, Module, Class, và Regular Expression.
2. Khi hoàn thành Lab05, sinh viên có thể:
   - Có thể phân tích và xây dựng được lớp và các phương thức trong lớp. Lập trình các bài toán có liên quan đến cấu trúc hàm, module, và regular expression …
3. Một số bài lập trình

   3.1. **Xây dựng module tính toán số học mở rộng**:

   Hãy tạo một module *advanced_math* với các hàm toán học như sau:

   - factorial(n): Tính giai thừa của n (sử dụng đệ quy). Ví dụ: factorial(4) = 24
   - fibonacci(n): Trả về dãy Fibonacci lên tới n phần tử. Ví dụ: fibonacci(8)=21
   - prime_factors(n): Tìm tất cả các số nguyên tố của n. Ví dụ: 20 = 2 × 2 × 5
   - is_prime(n): Kiểm tra xem n có phải là số nguyên tố không. Ví dụ: is_prime(5): true

   Yêu cầu:

   - Không sử dụng các hàm thư viện của Python.
   - Sử dụng *"if __name __ == "__main__"* trong module *advanced_math.py*.
   - Có khả năng thực thi các hàm toán học trong *advanced_math.py*.
   - Có khả năng thực thi các hàm toán học trong module advanced_math ở một file tự mở *check_math.py*.

   3.2. **Xây dựng một module phân tích văn bản với biểu thức chính quy (regular expression): tạo một module *regex_checking* có các hàm sau**:

   - *count_words(text)*: Đếm số lượng từ trong văn bản. Ví dụ: "Nha Trang University abc@gmail.com" có 6 ký tự.
   - *find_emails(text)*: Tìm tất cả các địa chỉ email trong văn bản. Ví dụ: "Nha Trang University abc@gmail.com" có một email abc@gmail.com
   - *find_dates(text)*: Tìm tất cả các ngày có định dạng DD/MM/YYYY hoặc MM-DD-YYYY trong văn bản. Ví dụ: Cho văn bản sau:

     text = """
     Hôm đó là ngày 01/11/2023. Liên hệ với tôi qua email john.doe@example.com. Một ngày đẹp trời khác là 10-12-2022. Một số lời nói xấu như idiot, badword cần được kiểm duyệt.
     Nha Trang University, today is 4/11/2024.
     """

     Kết quả của việc so khớp ngày trong văn bản trên là: `['01/11/2023', '10-12-2022']`

- *censor_bad_words(text, bad_words)*: Censor (thay thế) tất cả các từ xấu (*bad_words*) trong văn bản bằng dấu *. Ví dụ: kết quả thay thế từ xấu trong văn bản trên là:
  """
  Hôm đó là ngày 01/11/2023. Liên hệ với tôi qua email john.doe@example.com.
  Một ngày đẹp trời khác là 10-12-2022. Một số lời nói xấu như *****, ******* cần được kiểm duyệt.
  Nha Trang University, today is 4/11/2024.
  """

Hint:

- Sử dụng biểu thức chính quy ***r'\b\w+\b'*** để xác định các từ (*words*) có trong một văn bản với hàm ***findall*** của thư viện ***re***. Từ đó có thể đếm các từ trong một văn bản.
- Để xác định các email có trong một văn bản, ta có thể sử dụng biểu thức chính quy sau **r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'**.
- Để xác định ngày trong văn bản, ta sử dụng biểu thức chính quy **r'\b\d{2}/\d{2}/\d{4}\b|\b\d{2}-\d{2}-\d{4}\b'**.
- Sau đây là hàm kiểm tra từ xấu:

```
def censor_bad_words(text, bad_words):
    for word in bad_words:
        text = re.sub(r'\b' + re.escape(word) + r'\b', '*' * len(word), text, flags=re.IGNORECASE)
    return text
```

Yêu cầu:

- Không sử dụng bất kỳ thư viện nào để thực hiện nội dung trên (ngoại trừ *re* (tức là: *import re*))

3.3. **Quản lý ngân hàng**: Xây dựng một hệ thống quản lý ngân hàng bao gồm các module/class quản lý tài khoản (*account*), giao dịch (*transaction*), và ngân hàng (*bank*). Các lớp (class) hệ thống quản lý:

- Account (tài khoản, *account.py*): lớp Account có các thuộc tính *account_id*, *owner*, *balance.* Các phương thức *deposit(amount)*, *withdraw(amount)*. Ngoài ra, có thêm 1 phương thức *__str__()* để hiển thị thông tin tài khoản.
- Transaction (giao dịch, *transaction.py*): lớp Transaction để lưu các giao dịch. Lớp Transaction có các thuộc tính *transaction_id*, *account* (là một đối tượng Account), *type* (loại giao dịch: gửi/rút), *amount* (số tiền giao dịch). Các phương thức *execute()* để thực hiện giao dịch (trừ hoặc cộng vào tài khoản); phương thức *__str__()* để hiển thị thông tin giao dịch.
- Bank (ngân hàng, *bank.py*): lớp Bank để quản lý tài khoản và giao dịch. Lớp *Bank* có các thuộc tính: Thuộc tính *accounts* (danh sách các tài khoản), *transactions* (danh sách các giao dịch). Lớp *Bank* có các phương thức: phương thức *create_account(owner)* để tạo tài khoản mới. Phương thức *get_account*(*account_id*) để tìm tài khoản theo *account_id*. Phương thức *process_transaction(transaction)* để xử lý giao dịch.

Hint:

Lớp Account

```
class Account:
    def __init__(self, account_id, owner):
        self.account_id = account_id
        self.owner = owner
        self.balance = 0

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            return True
        return False

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            return True
        return False

    def __str__(self):
        return f"Account[ID: {self.account_id}, Owner: {self.owner}, Balance:
{self.balance}]"
```

Lớp Transaction

```
class Transaction:
    def __init__(self, transaction_id, account, type, amount):
        self.transaction_id = transaction_id
        self.account = account
        self.type = type
        self.amount = amount

    def execute(self):
        if self.type == "deposit":
            return self.account.deposit(self.amount)
        elif self.type == "withdraw":
            return self.account.withdraw(self.amount)
        return False

    def __str__(self):
        return f"Transaction[ID: {self.transaction_id}, Type: {self.type},
Amount: {self.amount}]"
```

Lớp Bank

```
#from account import Account
#from transaction import Transaction

class Bank:
    def __init__(self):
        self.accounts = []
        self.transactions = []

    def create_account(self, owner):
        account_id = len(self.accounts) + 1
        account = Account(account_id, owner)
        self.accounts.append(account)
        return account

    def get_account(self, account_id):
        for account in self.accounts:
            if account.account_id == account_id:
                return account
        return None

    def process_transaction(self, transaction):
```

```
        if transaction.execute():
            self.transactions.append(transaction)
            return True
        return False

    def list_accounts(self):
        for account in self.accounts:
            print(account)

    def list_transactions(self):
        for transaction in self.transactions:
            print(transaction)
```

Chương trình chính

```
#from bank import Bank
#from transaction import Transaction

# Tạo ngân hàng và tài khoản
bank = Bank()
account1 = bank.create_account("Alice")
account2 = bank.create_account("Bob")

# Thực hiện các giao dịch
transaction1 = Transaction(1, account1, "deposit", 1000)
transaction2 = Transaction(2, account1, "withdraw", 200)
transaction3 = Transaction(3, account2, "deposit", 500)

# Xử lý các giao dịch
bank.process_transaction(transaction1)
bank.process_transaction(transaction2)
bank.process_transaction(transaction3)

# In thông tin tài khoản và giao dịch
bank.list_accounts()
bank.list_transactions()
```

3.4. Hệ thống quản lý đơn hàng: Xây dựng một hệ thống quản lý đơn hàng chia thành các module/class riêng biệt như sản phẩm (*Product*), đơn hàng (*Order*), và khách hàng (*Customer*).

- Product (sản phẩm, *product.py*): Định nghĩa lớp *Product* có các thuộc tính *product_id*, *product_name*, *price*, *stock*.
    - ✓ Phương thức *update_stock*(quantity) để cập nhật số lượng hàng tồn kho.
    - ✓ Phương thức *__str__()* để hiển thị thông tin sản phẩm.
- Customer (khách hàng, *customer.py*): Định nghĩa lớp *Customer* có các thuộc tính *customer_id*, *name*, *email*.
    - ✓ Phương thức *__str__()* để hiển thị thông tin khách hàng.
- Order (đơn hàng, *order.py*): Định nghĩa lớp *Order* có các thuộc tính *order_id*, *customer* (là một đối tượng *customer*), *products* (danh sách các sản phẩm), và *total_price*.
    - ✓ Phương thức *add_product*(*product, quantity*) để thêm sản phẩm vào đơn hàng và cập nhật tổng giá tiền.
    - ✓ Phương thức *calculate_total()* để tính tổng giá tiền của đơn hàng.
    - ✓ Phương thức *__str__()* để hiển thị thông tin đơn hàng.