BÀI 4 LẬP TRÌNH THỰC HIỆN CÁC PHÉP TOÁN

Trong bài học này, chúng ta sẽ lập trình để máy tính có thể thực hiện các phép toán cộng và trừ. Phần thực hành lập trình thực hiện phép nhân và phép chia là bài tập thực hành cho các bạn. Dưới đây là những lệnh được bổ sung hoặc chỉnh sửa so với các bài học trước.

```
def addition():
    global num1
    num1 = int(e.get())
    e.delete(0, END)
    global operator
    operator = 1

def subtraction():
    global num1
    num1 = int(e.get())
    e.delete(0, END)
    global operator
    operator = 2

def calculation():
    global num1
    global operator
    num2 = int(e.get())
    e.delete(0, END)
    if operator == 1:
        e.insert(0, num1 + num2)
    if operator == 2:
        e.insert(0, num1 - num2)

bt_sub = Button(text="-", width=10, height=3,
                                    command=subtraction)
bt_eq = Button(text="=", width=10, height=3,
                                    command=calculation)
bt_add = Button(text="+", width=10, height=3,
                                    command=addition)
```

4.1. Lập trình chức năng thực hiện phép toán cộng

Khi thực hiện phép toán cộng, chúng ta cần trải qua các bước:

Bước 1: Nhập số hạng thứ nhất.

Bước 2: Nhấn nút "+", lúc này các chữ số trên hộp văn bản sẽ được lưu thành số hạng thứ nhất, sau đó hộp văn bản được xóa trống.

Bước 3: Nhập số hạng thứ hai.

Bước 4: Nhấn nút "=", lúc này chương trình sẽ thực hiện phép tính tổng số hạng thứ nhất và số hạng thứ 2, sau đó hiển thị kết quả lên hộp văn bản.

Chúng ta tạo một hàm thực hiện việc tính tổng (ví dụ: **addition()**), hàm sẽ được thực hiện khi nút "+" được nhấn. Trong thân hàm, các bạn cần tạo biến để lưu số hạng thứ nhất (ví dụ: **num1**) với từ khóa **global**.

Phạm vi (Scope) trong Python

Biến nhớ được tạo ở khu vực nào sẽ chỉ được sử dụng bên trong khu vực đó, khu vực ở đây có thể là định nghĩa lớp (class), định nghĩa hàm (function) hoặc thân chương trình chính. Điều này có nghĩa nếu biến được tạo bên trong một hàm thì sẽ không sử dụng được ở bên ngoài hàm cũng như không sử dụng được ở hàm khác, biến này được gọi là biến cục bộ (local variable).

```
def func1():
    a = "Hello VIETSTEM"

print(a)     # lỗi
```

Chương trình có lỗi, biến a chưa được định nghĩa ở thân chương trình chính

Biến được tạo ở bên thân chương trình chính được gọi là biến toàn cục (global variable). Các hàm trong chương trình đều nằm trong thân chương trình chính, vì vậy biến toàn cục có thể sử dụng được ở tất cả các hàm trong chương trình.

```
a = "Hello VIETSTEM"
def func1():
    print(a)

func1()     # kết quả in ra "Hello VIETSTEM"
```

Chương trình chạy đúng

Nếu các bạn cố tình khởi tạo một biến cục bộ có tên giống một biến toàn cục khác, hàm sẽ sử dụng biến cục bộ đó và không tác động đến biến toàn cục. Việc này không xảy ra lỗi nhưng rất dễ gây nhầm lẫn nên các bạn tránh việc đặt tên trùng như vậy. Nếu các bạn muốn tạo biến toàn cục hoặc muốn truy cập biến toàn cục ở trong hàm, các bạn sử dụng từ khóa **global** để khai báo biến.

```
a = 1
def func1():
    a = 2
    print(a)


func1()   # kết quả in ra 2
print(a)  # kết quả in ra 1
```

Biến toàn cục a không bị thay đổi giá trị

```
a = 1
def func1():
    global a
    a = 2
    print(a)

func1()   # kết quả in ra 2
print(a)  # kết quả in ra 2
```

Biến toàn cục a đã bị thay đổi giá trị

Lệnh **global a** bên trong hàm có ý nghĩa khai báo biến **a** là biến toàn cục. Nếu biến toàn cục **a** đã tồn tại từ trước thì sẽ sử dụng biến đó, còn nếu biến **a** chưa tồn tại thì cần khởi tạo biến **a** (biến **a** lúc này được coi là biến toàn cục). Mặc dù đã khai báo **global** nhưng nếu biến chưa tồn tại mà không được khởi tạo (gán giá trị) thì sẽ không sử dụng được.

```
def func1():
    global a
    a = 2

func1()
# biến a được tạo mới khi
# chạy hàm func1()
print(a) # kết quả in ra 2
```

*Biến **a** được tạo mới khi chạy hàm **func1()**, nếu không chạy hàm **func1()** thì biến **a** không được tạo, lệnh **print(a)** sẽ báo lỗi*

```
def func1():
    global a


func1()
# chương trình báo lỗi
# biến a chưa được khởi tạo
print(a)
```

*Biến **a** được khai báo nhưng chưa được khởi tạo, chương trình sẽ báo lỗi*

Trong hàm **addition()**, ta tạo biến **num1** với từ khóa **global**, sau đó lấy giá trị đoạn văn bản hiện tại của hộp văn bản bằng cách sử dụng phương thức **get()**, chuyển giá trị này thành kiểu dữ liệu **int** bằng hàm **int()** và gán vào biến **num1**. Cuối cùng, các bạn xóa hộp văn bản. Trong thân chương trình chính, hàm này có thể được đặt sau hàm **bt_click()**. Các bạn cũng lập trình cho nút "+" thực hiện hàm này khi được nhấn.

```
6  def addition():
7    global num1
8    num1 = int(e.get())
9    e.delete(0, END)
…  ...
26 bt_add = Button(text="+", width=10, height=3,
                                    command=addition)
```

Sau khi nhấn nút "+", hộp văn bản được xóa đi, người dùng tiếp tục nhấn các nút chữ số để nhập số hạng thứ hai. Cuối cùng, khi người dùng nhấn nút "=", kết quả sẽ được hiển thị trên hộp văn bản. Các bạn tạo hàm thực hiện việc tính kết quả khi nhấn dấu "=" (ví dụ: **calculation()**).

```
10 def calculation():
11   global num1
12   num2 = int(e.get())
13   e.delete(0, END)
14   e.insert(0, num1 + num2)
…    ...
30 bt_eq = Button(text="=", width=10, height=3,
                                  command=calculation)
```

Các bạn tiến hành chạy thử sẽ thấy chúng ta đã thực hiện được phép toán cộng khi bấm đúng trình tự theo các bước. Nếu chúng ta nhấn dấu "+" khi chưa nhập số thì hàm **e.get()** có giá trị rỗng và hàm chuyển đổi **int()** sẽ báo lỗi. Nếu các bạn nhấn nút "=" ngay sau khi nhấn dấu "+" mà không nhập số hạng thứ 2 thì chương trình cũng báo lỗi với lý do tương tự như vậy. Trong trường hợp chạy lần đầu tiên, chúng ta nhập số hạng thứ nhất và nhấn dấu "=", chương trình xảy ra lỗi với lý do chưa khởi tạo biến **num1** (biến **num1** được khởi tạo khi nhấn nút "+"). Việc sửa những lỗi này sẽ được đưa vào phần bài tập để các bạn thực hành.

4.2. Lập trình thực hiện thêm phép toán trừ

Chúng ta tạo thêm các hàm thực hiện công việc khi nhấn các nút phép tính trừ, nhân, chia tương tự như trên, tuy nhiên khi nhấn nút "=", máy tính cần biết phép toán sẽ thực hiện là phép tính nào trong 4 phép tính cộng, trừ, nhân, chia. Để làm được điều này, các bạn tạo một biến để lưu loại phép tính (ví dụ: **operator**). Biến **operator** sẽ nhận 4 giá trị khác nhau do ta quy ước (tương ứng với 4 loại phép tính), ví dụ:

- Giá trị 1 là phép cộng;
- Giá trị 2 là phép trừ;
- Giá trị 3 là phép nhân;
- Giá trị 4 là phép chia.

Mỗi khi nhấn các nút cộng, trừ, nhân, chia, biến này cần được đặt giá trị tương ứng. Ví dụ chúng ta lập trình gán giá trị 1 cho biến **operator** khi nút cộng được nhấn, gán giá trị 2 khi nút trừ được nhấn. Các bạn thực hiện tương tự để gán giá trị 3 và 4 cho biến **operator** khi nút nhân và nút chia được nhấn.

```
    # phép cộng
6   def addition():
7     global num1
8     num1 = int(e.get())
9     e.delete(0, END)
10    global operator
11    operator = 1

    # phép trừ
12  def subtraction():
13    global num1
14    num1 = int(e.get())
15    e.delete(0, END)
16    global operator
17    operator = 2
…     ...
35  bt_sub = Button(text="-", width=10, height=3,
                                    command=subtraction)
```

Khi nút "=" được nhấn, chúng ta cần kiểm tra biến **operator** có giá trị bằng bao nhiêu để thực hiện phép toán tương ứng.

```
18 def calculation():
19   global num1
20   global operator
21   num2 = int(e.get())
22   e.delete(0, END)
     # kiểm tra trường hợp phép tính cộng
23   if operator == 1:
24     e.insert(0, num1 + num2)
     # kiểm tra trường hợp phép tính trừ
25   if operator == 2:
26     e.insert(0, num1 - num2)
```

Khi bổ sung phép toán nhân và chia, các bạn cũng thực hiện các bước tương tự, gồm tạo hàm thực hiện công việc khi nút nhân và chia được nhấn, sau đó thêm trường hợp tính toán cho phép nhân và phép chia khi nút "=" được nhấn.

Tóm tắt lý thuyết và bài tập thực hành

Bài học này đã hướng dẫn các bạn tạo hàm bằng từ khóa **def** và truyền giá trị cho hàm. Chúng ta đã thực hành tạo các hàm thực hiện việc tính phép cộng và phép trừ, sau đó lưu thông tin phép tính vào các biến toàn cục sử dụng từ khóa **global**, cuối cùng hiển thị kết quả trên màn hình.

Bài tập 1. Lập trình chức năng thực hiện phép tính nhân

Bài tập 2. Lập trình chức năng thực hiện phép tính chia

Bài tập 3. Chương trình báo lỗi khi chúng ta nhấn nút phép tính hoặc nút "=" mà để trống hộp văn bản, các bạn hãy sửa lỗi này sao cho chương trình không báo lỗi nữa mà thay vào đó không thực hiện việc gì cả. Gợi ý: Trước khi đặt số hạng là các chữ số trên hộp văn bản, chúng ta cần kiểm tra xem hộp văn bản có rỗng hay không bằng lệnh kiểm tra độ dài một xâu bất kỳ: **len(<xâu>)**. Ví dụ **len("abc")** có giá trị bằng 3.

Bài tập 4. Khi chạy chương trình lần đầu, nếu điền số vào hộp văn bản và nhấn dấu "=" trước khi nhấn các phím phép toán, chương trình sẽ báo lỗi biến **operator** chưa được khởi tạo. Hãy lập trình để nếu chưa nhấn phép toán thì việc nhấn nút "=" không thực hiện việc gì cả.

Ý tưởng phát triển 1: Khi khởi tạo và khi bị xóa, hộp văn bản luôn hiển thị số 0, như vậy thì sẽ không có trường hợp hộp văn bản bị rỗng.

Ý tưởng phát triển 2*: Thêm nút thực hiện phép tính căn bậc 2 và bình phương.