TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BÀI TẬP KẾT THÚC MÔN HỌC

NGÀNH : KỸ THUẬT MÁY TÍNH

HỆ : ĐẠI HỌC CHÍNH QUY

MÔN HỌC: LẬP TRÌNH PYTHON

THÁI NGUYÊN - 2025

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ thông tin

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC

LẬP TRÌNH PYTHON

Sinh viên: Nguyễn Đức Anh Tú

Lớp: K58.KTP

Giáo viên GIẢNG DẠY: TS. Nguyễn Văn Huy

Link GitHub:

Thái Nguyên – 2025

TRƯỜNG ĐHKTCN

KHOA ĐIỆN TỬ

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC: LẬP TRÌNH PYTHON

BỘ MÔN CÔNG NGHỆ THÔNG TIN

Họ tên sinh viên : Nguyễn Đức Anh Tú MSV : K225480106070

Lớp : K58.KTP Ngành : Kỹ thuật máy tính

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy

Ngày giao đề tài: 31/05/2025 Ngày hoàn thành: 09/06/2025

Tên đề tài : Viết chương trình máy tính có giao diện GUI cho phép người dùng nhập hai số và chọn phép toán +, –, ×, ÷ để tính toán.

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày….tháng…..năm…..

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

Mục lục

LỜI NÓI ĐẦU

Ứng dụng Máy Tính Đơn Giản là một dự án lập trình nhằm mục đích thực hành các kỹ năng phát triển giao diện người dùng đồ họa (GUI) bằng Python và thư viện Tkinter. Bài toán này được thực hiện trong khuôn khổ bài tập học phần lập trình Python, với mục tiêu xây dựng một ứng dụng cho phép người dùng thực hiện các phép toán cơ bản (cộng, trừ, nhân, chia) trên hai số thực, đồng thời xử lý các lỗi nhập liệu và cung cấp giao diện thân thiện.

Mục đích của bài tập này là trình bày quá trình thiết kế, xây dựng, và kiểm thử ứng dụng, từ cơ sở lý thuyết đến thực nghiệm thực tế. Bài tập kết thúc môn học sẽ làm rõ các tính năng, thách thức, và kiến thức được áp dụng trong dự án, đồng thời cung cấp các sơ đồ khối, cấu trúc dữ liệu, và kết quả thực nghiệm để minh họa cách hoạt động của chương trình. Qua bài tập này, em không chỉ củng cố kiến thức về lập trình Python mà còn học được cách tổ chức mã nguồn, xử lý lỗi, và tối ưu hóa giao diện người dùng.

Bài tập kết thúc môn học được chia thành bốn chương chính: Giới thiệu đầu bài, Cơ sở lý thuyết, Thiết kế và xây dựng chương trình, Thực nghiệm và kết luận. Hy vọng bài tập này sẽ cung cấp cái nhìn toàn diện về quá trình phát triển ứng dụng và những bài học em đã rút ra.

LỜI CẢM ƠN

Em xin chân thành cảm ơn thầy Nguyễn Văn Huy đã tận tình hướng dẫn, cung cấp đề bài và tài liệu quý báu giúp em hoàn thành bài tập lớn này. Sự hỗ trợ và góp ý từ thầy đã giúp em hiểu rõ hơn về lập trình giao diện, xử lý lỗi và tổ chức mã nguồn theo hướng đối tượng. Dự án là cơ hội quý giá giúp em củng cố kiến thức, phát triển kỹ năng lập trình và tư duy giải quyết vấn đề. Em xin trân trọng cảm ơn!

DANH MỤC HÌNH ẢNH

CHƯƠNG I : TỔNG QUAN VỀ ĐỀ BÀI

1.1. Giới thiệu tổng quan

Chương trình cần xây dựng là một ứng dụng "Máy tính đơn giản" với giao diện người dùng đồ họa (GUI). Mục tiêu chính là cung cấp một công cụ tính toán cơ bản cho phép người dùng thực hiện các phép toán số học: cộng (+), trừ (−), nhân (×), và chia (÷).
Các yêu cầu cụ thể của chương trình bao gồm:

- Giao diện người dùng (GUI): Chương trình phải được phát triển với một giao diện đồ họa trực quan, thân thiện, cho phép người dùng tương tác dễ dàng.
- Đầu vào:
  + Người dùng có thể nhập hai số thực thông qua các ô nhập liệu chuyên biệt (Entry).
  + Lựa chọn phép toán mong muốn được thực hiện thông qua các nút chọn (Radiobutton) hoặc một menu tùy chọn.
- Đầu ra: Kết quả của phép tính phải được hiển thị rõ ràng trên giao diện người dùng, cụ thể là trên một nhãn văn bản (Label).
- Tính năng tính toán: Chương trình phải thực hiện phép toán đã chọn ngay lập tức khi người dùng nhấn nút "Tính".
- Xử lý lỗi: Chương trình cần có khả năng kiểm tra và xử lý các trường hợp lỗi phổ biến, bao gồm:
  + Lỗi nhập liệu: Khi người dùng nhập các ký tự không phải là số hợp lệ.
  + Lỗi chia cho 0: Khi người dùng cố gắng thực hiện phép chia với số chia bằng 0.
  + Trong trường hợp phát hiện lỗi, một hộp thoại thông báo (messagebox) cần được hiển thị để cảnh báo người dùng.
- Tính năng Reset: Chương trình phải cung cấp một nút "Reset" cho phép người dùng xóa nội dung của cả hai ô nhập liệu và đặt lại trạng thái hiển thị kết quả.
- Thiết kế bố cục: Giao diện cần được tổ chức một cách hợp lý và khoa học sử dụng trình quản lý bố cục Grid của thư viện GUI.

1.2. Các tính năng chính của Chương trình

Dựa trên các yêu cầu đã nêu, chương trình "Máy tính đơn giản" sẽ có các tính năng cốt lõi sau:

- Giao diện nhập liệu trực quan: Hai ô nhập liệu cho phép người dùng dễ dàng điền vào các số cần tính toán.
- Lựa chọn phép toán rõ ràng: Các nút chọn (Radiobutton) giúp người dùng dễ dàng lựa chọn phép cộng, trừ, nhân, hoặc chia.
- Hiển thị kết quả tức thì: Kết quả tính toán sẽ được hiển thị ngay lập tức trên một nhãn chuyên dụng sau khi nhấn nút "Tính".
- Thông báo lỗi thân thiện: Hộp thoại thông báo sẽ xuất hiện khi có lỗi nhập liệu hoặc lỗi chia cho 0, cung cấp phản hồi rõ ràng cho người dùng.
- Đặt lại trạng thái tiện lợi: Nút "Reset" giúp người dùng nhanh chóng xóa dữ liệu và bắt đầu một phép tính mới.

1.3. Thách thức và Giải pháp đề xuất

Việc xây dựng một máy tính GUI đơn giản, dù không quá phức tạp, vẫn đặt ra một số thách thức cần được giải quyết một cách hiệu quả:

- Thách thức 1: Xử lý đầu vào không hợp lệ.

  - Mô tả: Người dùng có thể nhập các ký tự không phải là số (ví dụ: chữ cái, ký hiệu đặc biệt) vào ô nhập liệu, hoặc để trống các ô. Nếu không được xử lý, điều này có thể gây ra lỗi chương trình khi cố gắng chuyển đổi sang kiểu số học.
  - Giải pháp: Vận dụng cơ chế bắt ngoại lệ (try-except block) của Python. Cụ thể, sẽ sử dụng try-except ValueError để phát hiện và xử lý lỗi khi chuyển đổi chuỗi sang số thực (float), và kiểm tra rỗng trước khi chuyển đổi. Một messagebox sẽ được hiển thị để thông báo lỗi cho người dùng.

- Thách thức 2: Xử lý phép chia cho số 0.

  - Mô tả: Phép chia cho 0 là một trường hợp không xác định trong toán học và sẽ gây ra lỗi ZeroDivisionError trong Python.
  - Giải pháp: Trước khi thực hiện phép chia, chương trình sẽ kiểm tra giá trị của số chia. Nếu số chia bằng 0, một ZeroDivisionError tùy chỉnh sẽ được tạo (raise) và bắt bởi khối try-except, sau đó một messagebox sẽ cảnh báo người dùng.

- Thách thức 3: Tổ chức bố cục GUI một cách trực quan và dễ mở rộng.

  - Mô tả: Với nhiều thành phần khác nhau (ô nhập liệu, nút chọn, nút chức năng, nhãn hiển thị), việc sắp xếp chúng sao cho hợp lý và dễ nhìn trên giao diện là quan trọng.
  - Giải pháp: Sử dụng trình quản lý bố cục Grid của tkinter. Grid cho phép định vị các widget theo hàng và cột, cung cấp khả năng kiểm soát chính xác vị trí, kích thước tương đối và khả năng giãn nở của từng thành phần, tạo ra một layout có cấu trúc và dễ bảo trì.

1.4. Vận dụng kiến thức

Để hoàn thành chương trình này, các kiến thức và công nghệ sau sẽ được vận dụng:

- Thư viện tkinter: Thư viện chuẩn của Python dùng để phát triển các ứng dụng GUI. Cụ thể, sẽ sử dụng các widget như Tk, Label, Entry, Radiobutton, Button.
- Trình quản lý bố cục Grid: Một thành phần của tkinter được sử dụng để sắp xếp và định vị các widget theo cấu trúc lưới (hàng và cột).
- Xử lý ngoại lệ (Exception Handling): Sử dụng cấu trúc try-except để quản lý và phản hồi các lỗi phát sinh trong quá trình chạy chương trình (ví dụ: ValueError, ZeroDivisionError).
- Hộp thoại thông báo (tkinter.messagebox): Để hiển thị các cảnh báo và thông báo lỗi một cách rõ ràng cho người dùng.
- Kiến thức toán học cơ bản: Các phép toán số học cơ bản (cộng, trừ, nhân, chia).

Tóm tắt chương.

Trong chương này, em đã giới thiệu tổng quan về chương trình "Máy tính đơn giản" với giao diện người dùng đồ họa (GUI). Em đã nêu rõ mục tiêu, các yêu cầu tính năng chi tiết, và những đầu vào/đầu ra mà chương trình cần xử lý. Các tính năng chính như nhập số, thực hiện phép toán cơ bản, hiển thị kết quả, xử lý lỗi nhập liệu và chia cho 0, cùng với chức năng reset, đều đã được trình bày.

CHƯƠNG II : CƠ SỞ LÝ THUYẾT

2.1. Ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình thông dịch, cấp cao, đa năng, được tạo ra bởi Guido van Rossum và phát hành lần đầu vào năm 1991. Python được biết đến với cú pháp rõ ràng, dễ đọc và cấu trúc mã nguồn đơn giản, làm cho nó trở thành lựa chọn lý tưởng cho cả người mới bắt đầu và các nhà phát triển chuyên nghiệp.

Các đặc điểm nổi bật của Python:

- Dễ học và dễ sử dụng: Cú pháp đơn giản, giống tiếng Anh, giúp giảm thời gian học và tăng tốc độ phát triển.
- Đa nền tảng: Có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS, Linux.
- Mã nguồn mở: Python là mã nguồn mở, cho phép cộng đồng đóng góp và phát triển rộng rãi.
- Thư viện phong phú: Python sở hữu một kho tàng thư viện khổng lồ, hỗ trợ hầu hết các lĩnh vực từ phát triển web, khoa học dữ liệu, trí tuệ nhân tạo đến phát triển ứng dụng desktop.
- Hỗ trợ đa mô hình lập trình: Hỗ trợ lập trình hướng đối tượng (OOP), lập trình chức năng (functional programming) và lập trình thủ tục (procedural programming).

Python đóng vai trò là ngôn ngữ nền tảng để viết toàn bộ logic của ứng dụng máy tính, từ xử lý đầu vào, thực hiện phép tính đến quản lý giao diện người dùng.

2.2. Thư viện chương trình

2.2.1 Tkinter

Tkinter là bộ công cụ GUI chuẩn được tích hợp sẵn trong Python, cung cấp một cách nhanh chóng và dễ dàng để tạo ra các ứng dụng giao diện người dùng đồ họa. Tkinter là một wrapper (lớp bọc) của bộ công cụ Tk GUI toolkit, vốn được phát triển cho Tcl/Tk.

Các thành phần chính của Tkinter:

Widgets**:** Là các phần tử cơ bản cấu thành giao diện người dùng. Mỗi widget có một chức năng cụ thể và các thuộc tính riêng biệt để tùy chỉnh hiển thị và hành vi. Trong ứng dụng máy tính, chúng ta sẽ sử dụng các widget sau:

- Tk: Đại diện cho cửa sổ chính của ứng dụng.
- Label: Dùng để hiển thị văn bản hoặc hình ảnh tĩnh. Trong chương trình này, Label sẽ được dùng để hiển thị các tiêu đề ("Số thứ nhất", "Số thứ hai", "Kết quả") và đặc biệt là hiển thị kết quả của phép tính.
- Entry: Cung cấp một ô nhập liệu đơn dòng cho phép người dùng nhập dữ liệu văn bản (trong trường hợp này là các số).
- Radiobutton: Cho phép người dùng chọn một trong một nhóm các tùy chọn loại trừ lẫn nhau (chỉ một tùy chọn có thể được chọn tại một thời điểm). Trong chương trình, Radiobutton sẽ được sử dụng để chọn phép toán (+, –, ×, ÷).
- Button: Tạo một nút có thể nhấn được. Khi nút được nhấn, một hàm (callback function) được chỉ định sẽ được thực thi.

Biến điều khiển (Control Variables): Tkinter cung cấp các lớp biến đặc biệt (StringVar, IntVar, DoubleVar, BooleanVar) cho phép các widget liên kết dữ liệu trực tiếp với các biến Python. Khi giá trị của biến thay đổi, widget sẽ tự động cập nhật, và ngược lại, khi người dùng tương tác với widget, giá trị của

biến cũng sẽ được cập nhật. Trong dự án này, StringVar sẽ được sử dụng để liên kết với nhóm Radiobutton để theo dõi phép toán đã chọn.

Vòng lặp sự kiện chính (mainloop()): Đây là một phương thức quan trọng của đối tượng Tk. Sau khi tất cả các widget đã được tạo và sắp xếp, mainloop() sẽ khởi động vòng lặp sự kiện, lắng nghe các tương tác của người dùng (nhấn nút, nhập liệu, v.v.) và kích hoạt các hàm xử lý sự kiện tương ứng. Chương trình GUI sẽ tiếp tục chạy cho đến khi cửa sổ được đóng.

Việc sử dụng Tkinter giúp đơn giản hóa quá trình tạo GUI, cho phép tập trung vào logic tính toán mà không cần đi sâu vào chi tiết của các API đồ họa cấp thấp.

2.2.2 Trình quản lý bố cục Grid (Grid Layout Manager)

Trong Tkinter, các widget không tự động xuất hiện trên màn hình sau khi được tạo. Chúng cần được đặt vào một vị trí cụ thể bằng cách sử dụng một trong các trình quản lý bố cục. Grid là một trong những trình quản lý bố cục mạnh mẽ và linh hoạt nhất của Tkinter, cho phép sắp xếp các widget theo một cấu trúc dạng bảng (grid), tức là theo hàng và cột.

Nguyên lý hoạt động của Grid:

- Hàng và Cột: Mọi widget được đặt vào một ô xác định bởi chỉ số hàng (row) và chỉ số cột (column). Chỉ số hàng và cột bắt đầu từ 0.
- grid() method: Mỗi widget trong Tkinter có một phương thức grid() được sử dụng để đặt nó vào layout: widget.grid(row=idx_hang, column=idx_cot, ...)
- Tham số quan trọng của grid():
    - row: Chỉ định chỉ số hàng nơi widget sẽ được đặt.
    - column: Chỉ định chỉ số cột nơi widget sẽ được đặt.

- rowspan: Cho phép widget kéo dài qua nhiều hàng.
- columnspan: Cho phép widget kéo dài qua nhiều cột. Rất hữu ích cho các thành phần như màn hình hiển thị hoặc nút "Tính" kéo dài toàn bộ chiều rộng.
- padx, pady: Thêm khoảng đệm (padding) theo chiều ngang và dọc bên ngoài ranh giới của widget.
- ipadx, ipady: Thêm khoảng đệm (internal padding) theo chiều ngang và dọc bên trong ranh giới của widget.
- sticky: Chỉ định cách widget sẽ "dính" vào các cạnh của ô lưới nếu ô đó lớn hơn kích thước của widget. Giá trị có thể là sự kết hợp của N (North/trên), S (South/dưới), E (East/phải), W (West/trái). NSEW (hoặc ew, ns, nsew) làm cho widget giãn nở để lấp đầy toàn bộ ô.

- *grid_rowconfigure()* và grid_*columnconfigure*(): Các phương thức này được gọi trên đối tượng container (thường là cửa sổ Tk hoặc một Frame) để kiểm soát hành vi giãn nở của các hàng và cột trong lưới.
  - container.grid_rowconfigure(index, weight=value): weight xác định mức độ giãn nở của hàng. Hàng có weight cao hơn sẽ giãn nở nhiều hơn khi không gian khả dụng tăng lên.
  - container.grid_columnconfigure(index, weight=value): Tương tự cho cột.

Việc sử dụng Grid trong ứng dụng máy tính giúp chúng ta tổ chức các ô nhập liệu, nút chọn phép toán, nút chức năng và nhãn kết quả một cách rõ ràng, tạo ra một bố cục cân đối và dễ dàng điều chỉnh.

2.2.3. Xử lý ngoại lệ (Exception Handling)

Xử lý ngoại lệ là một cơ chế quan trọng trong lập trình để quản lý các lỗi hoặc sự kiện bất thường có thể xảy ra trong quá trình thực thi chương trình mà

không làm dừng chương trình đột ngột. Python sử dụng khối try-except để thực hiện việc này.

- try block: Chứa đoạn mã có khả năng gây ra lỗi. Nếu một lỗi xảy ra trong khối try, Python sẽ dừng việc thực thi khối try đó và chuyển quyền điều khiển sang khối except tương ứng.
- except block: Chứa mã để xử lý một loại lỗi cụ thể. Nếu loại lỗi xảy ra trong try khớp với loại lỗi được chỉ định trong except, mã trong except sẽ được thực thi.
- raise statement: Cho phép lập trình viên tự tạo ra một ngoại lệ tùy chỉnh. Điều này hữu ích khi một điều kiện lỗi cụ thể được phát hiện mà không phải là lỗi được Python tự động ném ra.

Trong chương trình máy tính, xử lý ngoại lệ được áp dụng để:

- ValueError: Xảy ra khi một hàm nhận được một đối số có kiểu dữ liệu đúng nhưng giá trị không phù hợp (ví dụ: float("abc")). Trong trường hợp này, try-except ValueError sẽ bắt lỗi khi người dùng nhập văn bản không phải số vào ô Entry.
- ZeroDivisionError: Xảy ra khi thực hiện phép chia với số chia bằng 0. Chương trình sẽ chủ động kiểm tra trường hợp này và raise ZeroDivisionError một cách tường minh trước khi thực hiện phép chia, sau đó except ZeroDivisionError sẽ bắt và xử lý.
- Exception: Là lớp cơ sở cho hầu hết các ngoại lệ khác. Bắt Exception có thể được sử dụng để xử lý bất kỳ lỗi không lường trước nào khác, đảm bảo chương trình không bị sập.

Việc sử dụng xử lý ngoại lệ đảm bảo rằng ứng dụng máy tính vẫn hoạt động ổn định và cung cấp thông báo rõ ràng cho người dùng ngay cả khi có lỗi xảy ra.

CHƯƠNG III : THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống

❖ **Chương trình được tổ chức bao gồm các module chính sau:**

➢ **Giao diện người dùng (GUI - User Interface):**

- Chức năng: Chịu trách nhiệm hiển thị tất cả các thành phần trực quan của máy tính như ô nhập liệu, nút bấm, nhãn hiển thị kết quả. Nó cũng tiếp nhận các tương tác từ người dùng (nhấn nút, nhập số).
- Thành phần: Tk (cửa sổ chính), Label (hiển thị văn bản và kết quả), Entry (ô nhập liệu), Radiobutton (chọn phép toán), Button (nút "Tính", "Reset").
- Sử dụng: Thư viện tkinter, trình quản lý bố cục Grid.

➢ **Xử lý Phép toán (Input & Operation Handler):**

- Chức năng: Đọc dữ liệu từ các ô nhập liệu của GUI, xác định phép toán đã chọn. Sau đó, nó thực hiện các phép tính số học cơ bản (cộng, trừ, nhân, chia) dựa trên đầu vào và lựa chọn của người dùng.
- Thành phần chính: Hàm tinh(), các biến toàn cục lưu trữ trạng thái biểu thức.
- Sử dụng: Các phép toán số học cơ bản, chuyển đổi kiểu dữ liệu (float()).

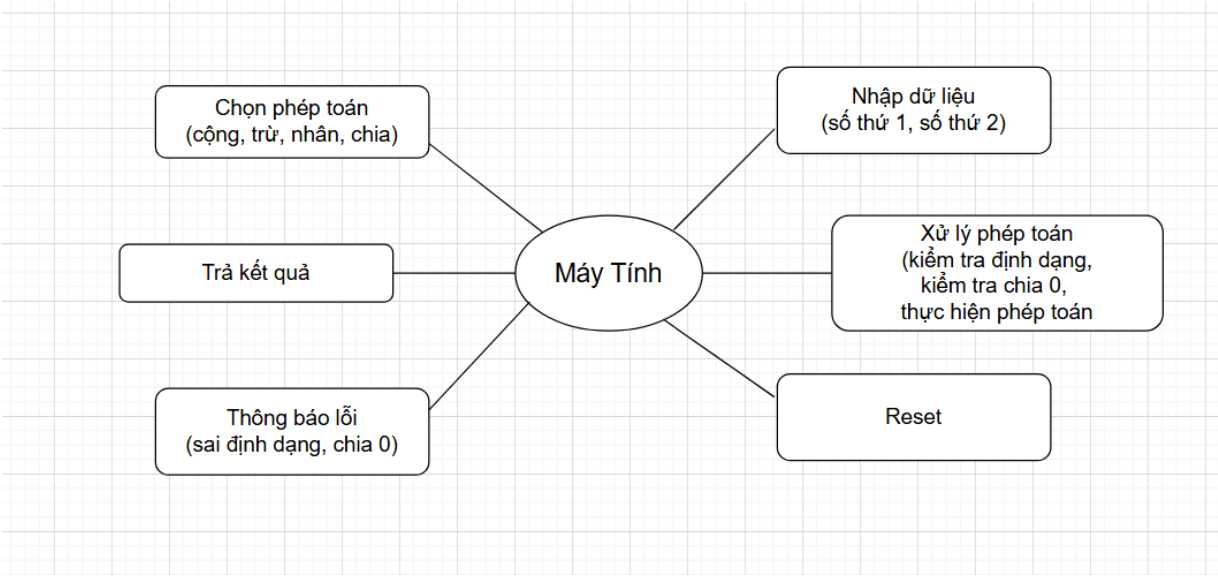
➢ **Xử lý lỗi (Error Handler):**

- Chức năng: Phát hiện và quản lý các tình huống lỗi có thể xảy ra trong quá trình thực thi chương trình, đặc biệt là lỗi nhập liệu không hợp lệ và lỗi chia cho 0.
- Thành phần chính: Khối try-except, hàm messagebox.showerror().

➢ **Reset:**

- Chức năng: Xóa dữ liệu và đặt lại trạng thái ban đầu
- Thành phần chính: hàm dat_lai()

❖ **Biểu đồ phân cấp chức năng:**



Hình 1 Biểu đồ phân cấp chức năng

Mô tả biểu đồ: Biểu đồ là một cây phân cấp với nút gốc "Máy Tính". Mỗi nhánh con là một chức năng, chia nhỏ thành các tác vụ cụ thể.

Các chức năng chính được liệt kê là:

1. Nhập dữ liệu: Bao gồm việc nhập số thứ nhất và nhập số thứ hai. Đáng chú ý là nguồn cũng liệt kê các phép toán Cộng, Trừ, Nhân, Chia dưới nhánh "Nhập dữ liệu" này.15

2. Xử lý tính toán: Nhánh này chi tiết hóa các bước xử lý, bao gồm kiểm tra định dạng số, kiểm tra chia cho 0, và thực hiện phép toán.

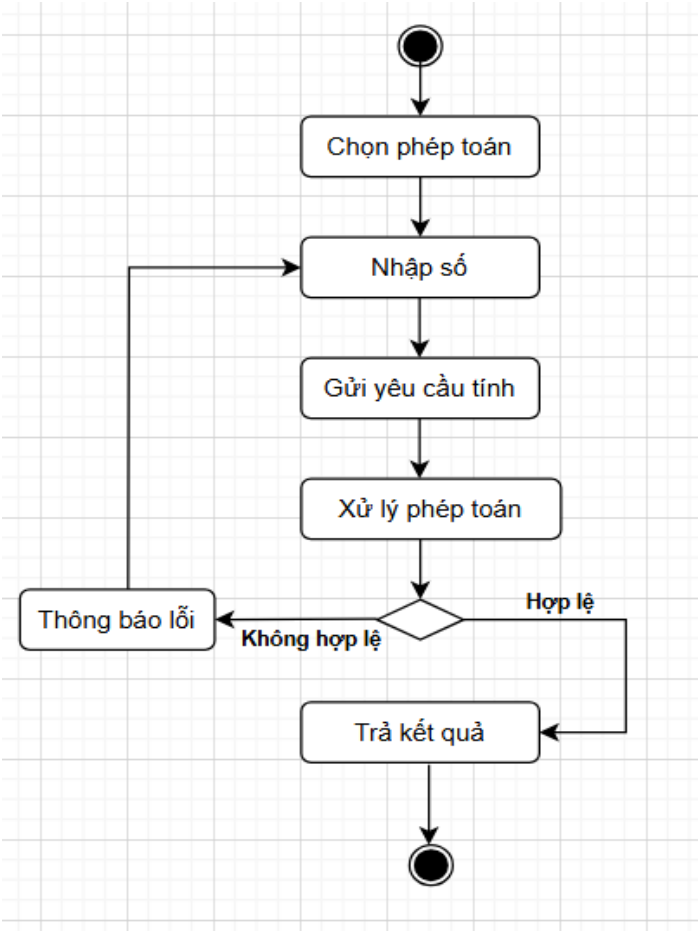
3. Hiển thị kết quả

4. Reset: Chức năng này bao gồm các hoạt động như xóa ô nhập1 và đặt lại phép toán

5. Thông báo lỗi: Nhánh này xử lý các trường hợp lỗi, cụ thể là sai định dạng hoặc chia cho 0

3.2. Sơ đồ khối các thuật toán chính

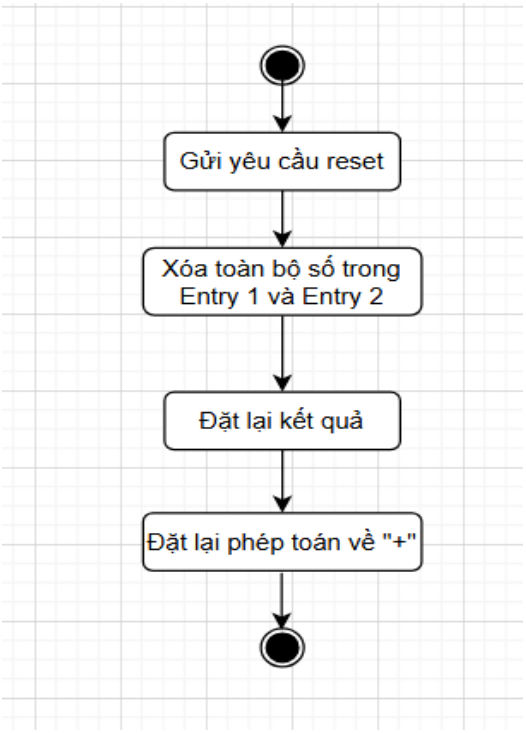
3.2.1. Thuật toán tính toán (tinh())



Hình 3.2.1 Biểu đồ hoạt động tính toán

- Chức năng: Đọc giá trị từ các ô nhập liệu, thực hiện phép toán đã chọn và hiển thị kết quả. Bắt các lỗi nhập liệu và chia cho 0.
- Đầu vào: Chuỗi từ so1, chuỗi từ so2, và lựa chọn phép toán từ phep_toan.
- Xử lý:
    + Kiểm tra nếu phép toán là chia và số thứ hai bằng 0 → phát sinh ZeroDivisionError.
    + Thực hiện phép toán theo đúng ký tự đã chọn.
- Đầu ra: Cập nhật ket_qua với kết quả hoặc thông báo lỗi; hiển thị hộp thoại messagebox nếu có lỗi.

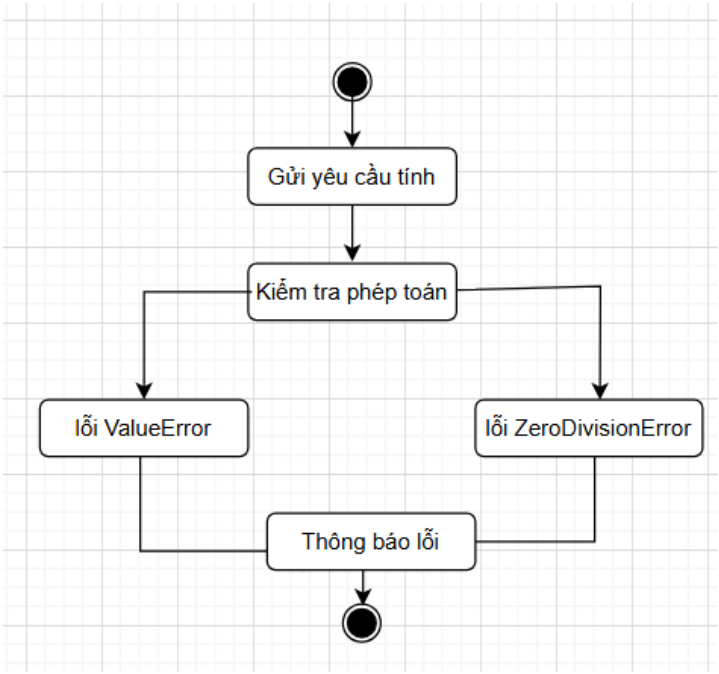
3.2.2. Thuật toán Reset (dat_lai())



Hình 3.2.2 Biểu đồ hoạt động reset

- Chức năng: Khôi phục trạng thái mặc định của các thành phần nhập liệu và hiển thị kết quả trên GUI.
- Đầu vào: Sự kiện nhấn nút “Reset” hoặc sau khi xảy ra lỗi.
- Xử lý:
    + Gọi Entry.delete(0, END) để xóa dữ liệu ở hai ô nhập.
    + Đặt lại biến phép toán (StringVar) về dấu “+”.
    + Cập nhật Label kết quả về giá trị mặc định.
- Đầu ra: Các ô nhập liệu (Entry) trống rỗng, nhãn kết quả (ket_qua) trở về văn bản mặc định, và lựa chọn phép toán (phep_toan) được đặt lại về dấu cộng.

3.2.3. Thuật toán bắt ngoại lệ và hiển thị hộp thoại



Hình 3.2.3. Biểu đồ hoạt động bắt ngoại lệ và hiển thị hộp thoại

- Chức năng: Quản lý lỗi để chương trình không bị sập và thông báo cho người dùng.
- Đầu vào: Một ngoại lệ (ValueError, ZeroDivisionError) được phát hiện trong khối try-except
- Xử lý: Dựa vào loại lỗi, chuẩn bị thông báo lỗi phù hợp
  + ValueError: dùng messagebox.showerror() để hiển thị thông báo "Vui lòng nhập số hợp lệ".
  + ZeroDivisionError: hiển thị "Không thể chia cho 0!".
- Đầu ra: Hộp thoại lỗi hiển thị; Label kết quả hiển thị lỗi; chương trình tiếp tục hoạt động.

3.3. Chương trình

Thuộc tính lớp:

- goc: Cửa sổ chính (tk.Tk).
- so1, so2: Ô nhập liệu (ttk.Entry). Kiểu dữ liệu Chuỗi → float
- phep_toan: Biến tk.StringVar lưu phép toán (+, -, ×, ÷).

- ket_qua: Nhãn hiển thị kết quả

Hàm trong lớp MayTinh:

- __init__: Hàm khởi tạo của lớp thiết lập cửa sổ ứng dụng (master) và (widgets)
- tinh: Đọc giá trị từ các ô nhập liệu, xác định phép toán, thực hiện tính toán.
- dat_lai: Xóa dữ liệu và đặt lại trạng thái.

Hàm ngoài lớp MayTinh:

cua_so_goc: Tạo cửa sổ chính và chạy vòng lặp mainloop

❖ Code :

```
import tkinter as tk
from tkinter import messagebox

class MayTinh:
    # Lớp MayTinh quản lý toàn bộ giao diện và logic của ứng dụng máy tính
    def __init__(self, master):

        # Khởi tạo cửa sổ chính và các thành phần GUI.
        # master: đối tượng cửa sổ Tkinter gốc (root).
        self.goc = master # gốc Lưu trữ đối tượng cửa sổ chính
        self.goc.title("May Tinh")
        self.goc.geometry("300x330")
        self.goc.resizable(False, False) # Không cho phép thay đổi kích thước
cửa sổ
        self.goc.configure(bg="#30e8db") # Đặt màu nền cho cửa sổ chính

        # Cấu hình các cột để chúng giãn nở đều khi cửa sổ được thay đổi kích
thước
        self.goc.grid_columnconfigure(0, weight=1)
        self.goc.grid_columnconfigure(1, weight=1)

        # --- Tạo và sắp xếp các thành phần GUI bằng Grid ---

        # Hàng 0: Label "Số thứ nhất" và Entry nhập số thứ nhất
        tk.Label(self.goc, text="So thu nhat:").grid(row=0, column=0, padx=5,
pady=5, sticky="w") #
        self.so1 = tk.Entry(self.goc, width=20)
        self.so1.grid(row=0, column=1, padx=5, pady=5, sticky="ew")

        # Hàng 1: Label "Số thứ hai" và Entry nhập số thứ hai
```

```
        tk.Label(self.goc, text="So thu hai:").grid(row=1, column=0, padx=5,
pady=5, sticky="w") #
        self.so2 = tk.Entry(self.goc, width=20)
        self.so2.grid(row=1, column=1, padx=5, pady=5, sticky="ew")

         # Hàng 2 & 3: Các Radiobutton để chọn phép toán
        self.phep_toan = tk.StringVar(value="+")

        tk.Radiobutton(self.goc, text="Cong (+)", variable=self.phep_toan,
value="+").grid(row=2, column=0, padx=5, pady=2, sticky="w")
        tk.Radiobutton(self.goc, text="Tru (-)", variable=self.phep_toan,
value="-").grid(row=2, column=1, padx=5, pady=2, sticky="w")
        tk.Radiobutton(self.goc, text="Nhan (x)", variable=self.phep_toan,
value="*").grid(row=3, column=0, padx=5, pady=2, sticky="w")
        tk.Radiobutton(self.goc, text="Chia (:)", variable=self.phep_toan,
value="/").grid(row=3, column=1, padx=5, pady=2, sticky="w")

        # Hàng 4: Nút "Tính"
        tk.Button(self.goc, text="Tinh", command=self.tinh, width=15,
height=2).grid(row=4, column=0, columnspan=2, padx=5, pady=10, sticky="ew")

        # Hàng 5: Nút "Reset"
        tk.Button(self.goc, text="Reset", command=self.dat_lai, width=15,
height=2).grid(row=5, column=0, columnspan=2, padx=5, pady=5, sticky="ew")

        # Hàng 6: Label hiển thị kết quả
        self.ket_qua = tk.Label(self.goc, text="Ket qua", font=("Helvetica",
12, "bold"), fg="blue", bg="lightgray", padx=10, pady=10)
        self.ket_qua.grid(row=6, column=0, columnspan=2, padx=5, pady=10,
sticky="ew") #

    def tinh(self):
        # Đọc giá trị từ các ô nhập liệu, thực hiện phép toán đã chọn
        # và hiển thị kết quả trên Label. Bắt lỗi ValueError và
ZeroDivisionError
        try:
            so1_str = self.so1.get()
            so2_str = self.so2.get()

            if not so1_str or not so2_str:
                messagebox.showerror("Loi", "Vui long nhap day du ca hai
so.") #
                return
            num1 = float(so1_str) #
```

```
            num2 = float(so2_str) #
            phep = self.phep_toan.get() # Phép toán (operation_var)

            kq = 0 # kết quả
            if phep == "+":
                kq = num1 + num2
            elif phep == "-":
                kq = num1 - num2
            elif phep == "*":
                kq = num1 * num2
            elif phep == "/":
                if num2 == 0:
                    raise ZeroDivisionError("Khong the chia cho so 0!")
                kq = num1 / num2
            else:
                messagebox.showerror("Loi", "Vui long chon mot phep toan.")
                return

            self.ket_qua.config(text=f"Ket qua = {kq}")

        except ValueError:
            messagebox.showerror("Loi", "Vui long nhap so hop le.")
            self.ket_qua.config(text="Ket qua: Loi nhap lieu!")
        except ZeroDivisionError as e:
            messagebox.showerror("Loi", str(e))
            self.ket_qua.config(text="Ket qua: Loi chia 0!")

    def dat_lai(self):
        # Xóa nội dung của cả hai ô nhập liệu và đặt lại nhãn kết quả.
        self.so1.delete(0, tk.END)
        self.so2.delete(0, tk.END)
        self.ket_qua.config(text="Ket qua")
        self.phep_toan.set("+")

# Khởi tạo cửa sổ Tkinter gốc
cua_so_goc = tk.Tk()

# Tạo một thể hiện của lớp MayTinh, truyền cửa sổ gốc vào
may_tinh_cua_toi = MayTinh(cua_so_goc)

# Chạy vòng lặp sự kiện chính của Tkinter
cua_so_goc.mainloop() #
```

Tóm tắt Chương 3 – Thiết kế và xây dựng chương trình: Chương trình được thiết kế theo hướng đối tượng với lớp MayTinh là trung tâm điều khiển toàn bộ giao diện và chức năng. Giao diện gồm các thành phần chính: ô nhập liệu, nhóm nút chọn phép toán, nút "Tính" và "Reset", cùng nhãn hiển thị kết quả. Mỗi chức năng được chia thành các thuật toán riêng như: nhập số và kiểm tra lỗi, xử lý phép toán, hiển thị kết quả, bắt ngoại lệ và reset.

CHƯƠNG IV : THỰC HIỆN CHƯƠNG TRÌNH

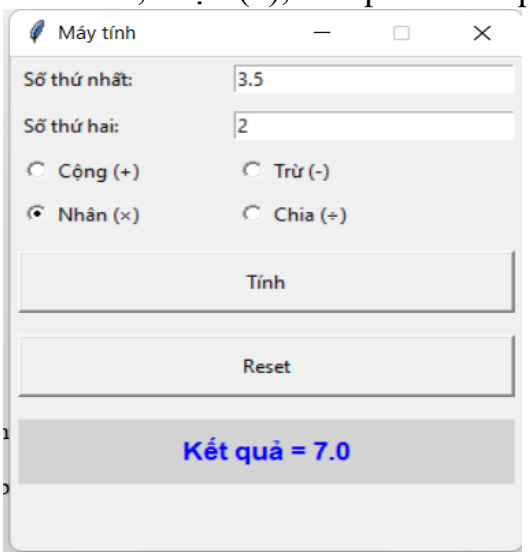
4.1. Kiểm thử chương trình

❖ **Quy trình xử lý diễn ra như sau:**

- Khởi chạy file mã py sẽ được thực thi bởi trình thông dịch của Python.
- Cửa sổ chính của ứng dụng (cua_so_goc = tk.Tk()) được tạo. Đây là đối tượng đại diện cho cửa sổ mà người dùng sẽ tương tác.
- Giao diện người dùng được xây dựng
- Khởi động một vòng lặp sự kiện chính, ví dụ sự kiện Tính được xảy ra sẽ kích hoạt hàm tương ứng (ví dụ tinh())
- Xử lý sự kiện khi hàm xử lý sự kiện được gọi. Hàm đó thực hiện các thao tác logic đã được định nghĩa (ví dụ: đọc giá trị từ Entry, thực hiện phép tính, kiểm tra lỗi).
    + Nếu có lỗi, một hộp thoại messagebox được hiển thị và ket_qua được cập nhật.
    + Nếu thành công, ket_qua được cập nhật với kết quả.
- Kết thúc chương trình: Vòng lặp sự kiện cua_so_goc.mainloop() sẽ tiếp tục chạy cho đến khi người dùng đóng cửa sổ ứng dụng

❖ **Chạy thử chương trình**

1. Phép nhân: Nhập 3.5 và 2, chọn (x), kết quả: "Kết quả = 7.0"

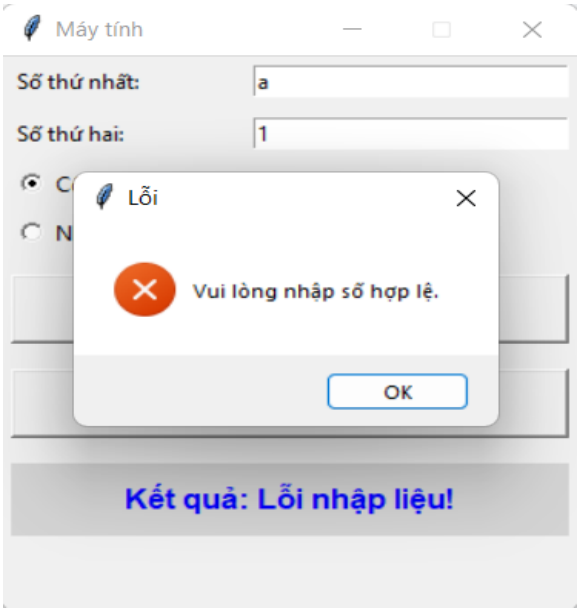


Hình 4.2.1 Kết quả phép nhân

2. Phép trừ: Nhập 25 và 10, chọn (-), kết quả: “Kết quả = 15.0”

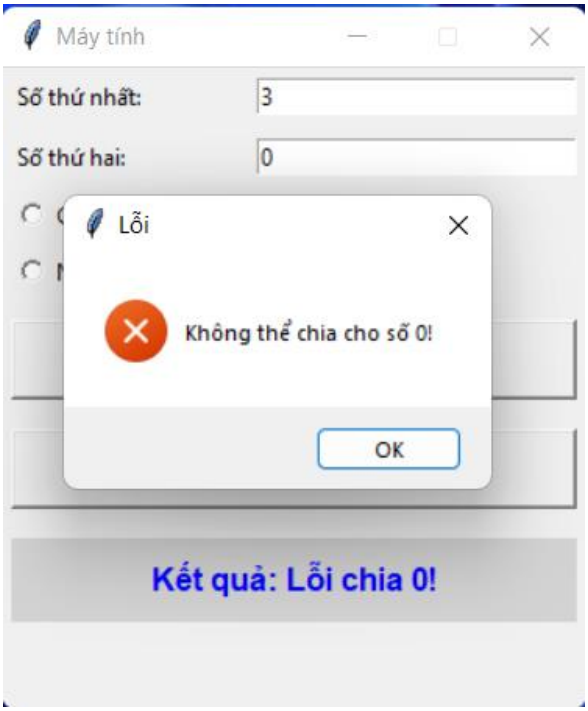
Hình 4.2.2 Báo lỗi khi phép trừ

3. Lỗi nhập liệu: Nhập "a" và 1, chọn "+", hiển thị thông báo: "Vui lòng nhập số hợp lệ."



Hình 4.2.3 Báo lỗi khi phép tính không hợp lệ

4. Phép chia: Nhập 3 và 0, chọn (÷), kết quả: “ Kết quả: Lỗi chia 0”



Hình 4.2.4 Báo lỗi khi không chia hết cho 0

4.2. Kết luận

Ứng dụng "Máy tính" được phát triển đã hoàn thành tốt các yêu cầu đề ra và đạt được các mục tiêu sau:

- Xây dựng thành công giao diện GUI: Chương trình đã tạo ra một giao diện người dùng đồ họa trực quan và thân thiện bằng cách sử dụng thư viện tkinter của Python. Các thành phần như ô nhập liệu (Entry), nút chọn phép toán (Radiobutton), nút chức năng (Button) và nhãn hiển thị kết quả (Label) được sắp xếp rõ ràng.
- Triển khai bố cục linh hoạt với Grid: Toàn bộ giao diện được tổ chức một cách hiệu quả và có cấu trúc bằng cách sử dụng trình quản lý bố cục Grid, đảm bảo các thành phần được căn chỉnh và sắp xếp hợp lý.
- Thực hiện các phép tính số học cơ bản: Chương trình có khả năng thực hiện chính xác bốn phép toán cơ bản: cộng, trừ, nhân, chia giữa hai số thực.

- Xử lý lỗi mạnh mẽ: Chương trình đã tích hợp cơ chế xử lý ngoại lệ (try-except) để bắt và thông báo cho người dùng về các lỗi phổ biến như:
    - Nhập liệu không phải là số hợp lệ (ValueError).
    - Phép chia cho số 0 (ZeroDivisionError).
    - Các hộp thoại messagebox.showerror được sử dụng để cung cấp phản hồi lỗi rõ ràng và thân thiện.
- Chức năng Reset tiện lợi: Nút "Reset" cho phép người dùng nhanh chóng xóa dữ liệu đã nhập và đặt lại trạng thái máy tính về ban đầu.
- Hoạt động ổn định: Chương trình chạy ổn định theo mô hình hướng sự kiện, phản hồi nhanh chóng các tương tác của người dùng.

Trong quá trình thực hiện dự án này, em đã học hỏi và củng cố được nhiều kiến thức, kỹ năng quan trọng trong lập trình Python và phát triển ứng dụng GUI:

- Nắm vững kiến thức về tkinter: Hiểu sâu hơn về các widget cơ bản (Label, Entry, Button, Radiobutton) và cách sử dụng chúng để xây dựng giao diện người dùng.
- Kỹ năng xử lý ngoại lệ (Exception Handling): Áp dụng hiệu quả try-except để viết mã mạnh mẽ, có khả năng chống lỗi, giúp chương trình không bị sập khi gặp đầu vào không mong muốn.
- Kiến thức về lập trình hướng sự kiện: Hiểu rõ cách cua_so_goc.mainloop() hoạt động và cách các hàm xử lý sự kiện được kích hoạt bởi các tương tác của người dùng.
- Kỹ năng gỡ lỗi (Debugging): Phát triển khả năng tìm kiếm và sửa lỗi trong mã nguồn dựa trên các thông báo lỗi và hành vi không mong muốn của chương trình.
- Quản lý luồng chương trình: Hiểu rõ hơn về cách các hàm và module tương tác với nhau trong một ứng dụng thực tế.

Mặc dù chương trình đã hoàn thành các yêu cầu cơ bản, vẫn còn nhiều tiềm năng để cải tiến và mở rộng trong tương lai:

- Phép tính phức tạp hơn: Mở rộng các phép toán để bao gồm lũy thừa, căn bậc hai, phần trăm, và các hàm lượng giác.
- Xử lý chuỗi phép tính: Hiện tại, máy tính chỉ xử lý một phép toán tại một thời điểm. Cải tiến để cho phép người dùng nhập chuỗi phép tính dài hơn (ví dụ: 2 + 3 * 4) và tuân thủ thứ tự ưu tiên toán tử.
- Lịch sử tính toán: Thêm một tính năng để lưu trữ và hiển thị lịch sử các phép tính đã thực hiện.
- Kiểm tra đầu vào theo thời gian thực: Cải tiến để cung cấp phản hồi ngay lập tức cho người dùng khi nhập liệu không hợp lệ (ví dụ: đổi màu ô nhập liệu, hiển thị cảnh báo nhỏ) thay vì chỉ khi nhấn nút "Tính".

TÓM TẮT CHƯƠNG

Trong chương này em giới thiệu tổng quan về Python – ngôn ngữ chính được sử dụng để xây dựng chương trình. Em đã mô tả cơ chế kiểm thử bao gồm việc xác định các trường hợp kiểm thử (test cases) cụ thể với đầu vào, các bước thực hiện, kết quả mong đợi và kết quả thực tế. Các trường hợp kiểm thử đã được thiết kế để bao quát đầy đủ các chức năng, từ phép tính cơ bản đến xử lý các tình huống lỗi như chia cho 0 và nhập liệu không hợp lệ, cũng như kiểm tra chức năng reset.

TÀI LIỆU THAM KHẢO

1. Giáo trình môn Lập trình Python : Python Programming for the Absolute Beginner- 3rd Edition
2. W3Schools. Python Try Except. https://www.w3schools.com/python/python_try_except.asp