TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN BÀI 1 MÔN HỌC HK241

LẬP TRÌNH PYTHON NÂNG CAO

XÂY DỰNG HỆ THỐNG RA QUYẾT ĐỊNH TRONG VIỆC DỰ BÁO THỜI TIẾT

Sinh viên thực hiện : Nguyễn Phúc Nguyên

MSSV: 2274802010586

GVHD: Huỳnh Thái Học

TP. Hồ Chí Minh – năm 2024

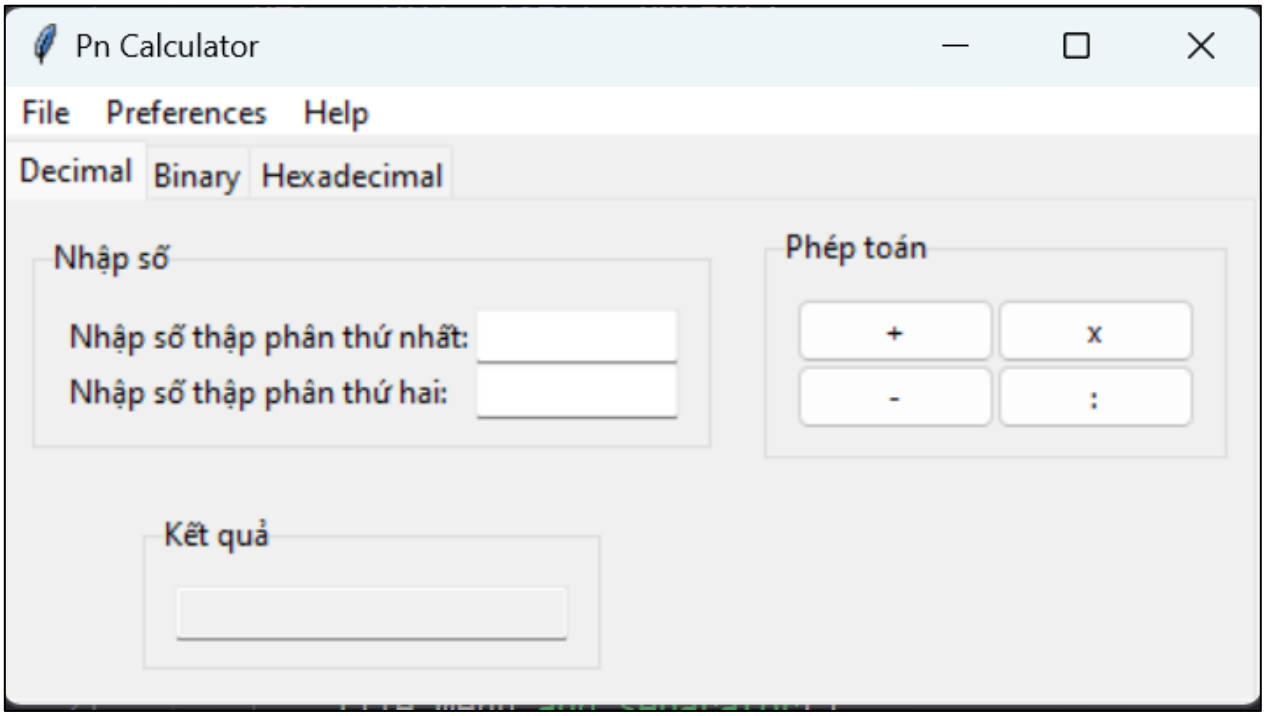
Nguyễn Phúc Nguyên
2274802010586

Mục lục

CHƯƠNG 1. Giao diện người dùng (GUI)

Ứng dụng sử dụng thư viện Tkinter để tạo giao diện đồ họa. Giao diện chính bao gồm một số thành phần cơ bản sau:

Giao diện chính:



Hình 1: Giao diện chính của Máy tính.

1. Thanh Menu: có ba mục chính:

- File: Cho phép mở tệp, lưu tệp và thoát chương trình.
- Preferences: Cho phép thay đổi giao diện và ngôn ngữ (chưa được triển khai đầy đủ).
- Help: Hiển thị hướng dẫn sử dụng và thông tin về ứng dụng.

2. Tab Control:

Chương trình có một bảng điều khiển dưới dạng các tab với 3 loại máy tính: Decimal, Binary, và Hexadecimal. Mỗi tab sẽ xử lý phép tính tương ứng với hệ số của nó.

3. Khung nhập liệu và phép toán:

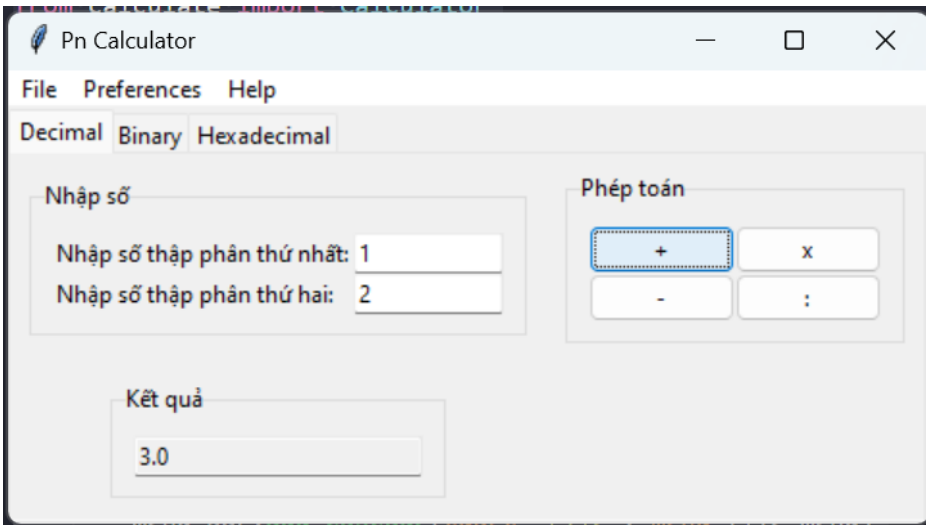
- Ở mỗi tab, người dùng có thể nhập hai số để thực hiện các phép tính.

- Các nút bấm tương ứng với các phép tính được hiển thị trong các tab khác nhau.
- Kết quả của phép tính sẽ được hiển thị trong một ô nhập liệu có trạng thái chỉ đọc.

CHƯƠNG 2: Chức năng cơ bản

1. Decimal (Số thập phân):

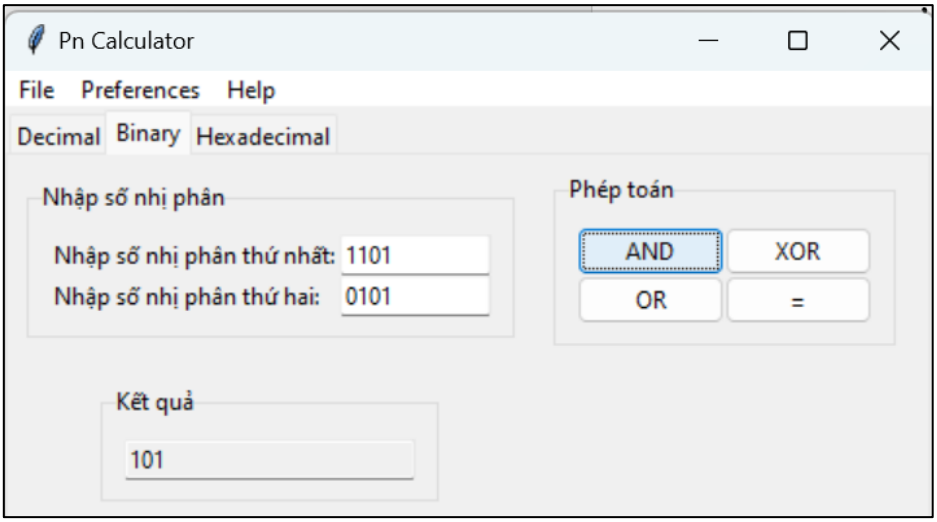
Các phép toán cơ bản: +, - , x, :



Hình 2: GUI của Decimal.

2. Binary (Số nhị phân)

Các phép tính cơ bản: and, xor, or, =

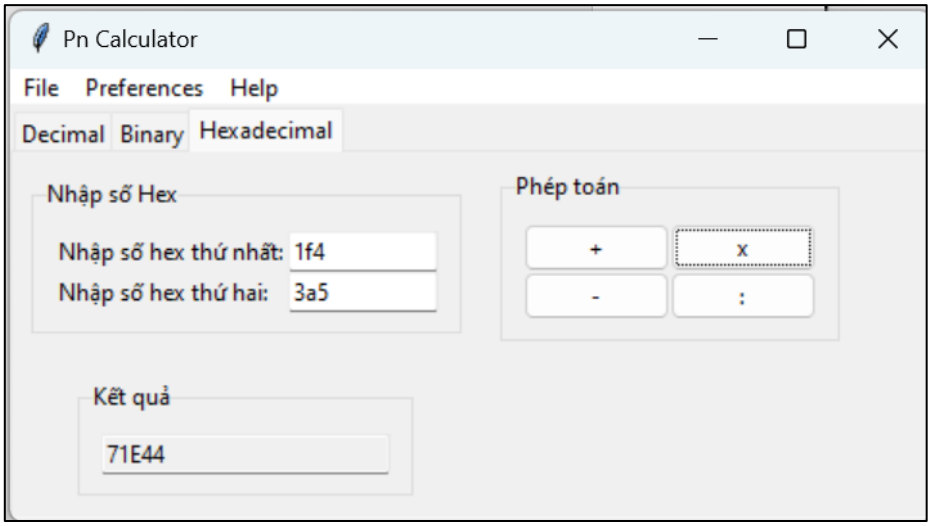


Hình 3: GUI của Binary.

Nguyễn Phúc Nguyên
2274802010586

1. Hexadecimal (Số thập lục phân)

Các phép tính cơ bản: +, - , x, :



Hình 4: GUI của Hexadecimal.

CHƯƠNG 3: Mã nguồn

1. File calculate.py

```python
class Calculator:
    def __init__(self, num1, num2, method):
        self.num1 = num1
        self.num2 = num2
        self.method = method

    def TinhDec(self):
        if self.method == "+":
            return self.num1 + self.num2
        elif self.method == "-":
            return self.num1 - self.num2
        elif self.method == "x":
            return self.num1 * self.num2
        elif self.method == ":":
            if self.num2 != 0:
                return self.num1 / self.num2
            else:
                raise ZeroDivisionError("Cannot divide by zero.")
        else:
            raise ValueError(f"Unknown method {self.method}")
```

Hình 5: Class Calculator và hàm tính Dec.

Gồm class Calculator: đây là class chứa các chức năng chính của chương trình, xử lí tính toán dựa trên 2 số num1, num2 và method (toán tử) được truyền vào. Ở đây ta xét cho số thập phân.

Khi ta thực hiện với tính Bin, ta cần đảm bảo rằng dữ liệu sử dụng cần đưa về đúng dạng bin (gồm 0 và 1). Ta cần sử dụng try, except để bắt lỗi:

```
22      def TinhBin(self):
23          try:
24              num1_int = int(str(self.num1), 2)
25              num2_int = int(str(self.num2), 2)
26          except ValueError:
27              raise ValueError("Invalid binary input")
28
29          if self.method == "AND":
30              return bin(num1_int & num2_int)[2:]
31          elif self.method == "XOR":
32              return bin(num1_int ^ num2_int)[2:]
33          elif self.method == "OR":
34              return bin(num1_int | num2_int)[2:]
35          elif self.method == "=":
36              return num1_int == num2_int
37          else:
38              raise ValueError("Unknown binary operation")
39
```

Hình 6: Hàm tính Bin.

Tương tự với tính Bin, tính Hex cũng cần xử lí đầu vào.

```
def TinhHex(self):
    try:
        num1_hex = int(str(self.num1), 16)
        num2_hex = int(str(self.num2), 16)
```

Hình 7: Xử lí đầu vào của số Hex.

Bên cạnh đó khi thực hiện tính toán cần xử lí lại kết quả trả về khi có thể chữ cái trong hệ thập lục phân

```
if self.method == "+":
    return hex(num1_hex + num2_hex)[2:].upper()
elif self.method == "-":
    return hex(num1_hex - num2_hex)[2:].upper()
elif self.method == "x":
    return hex(num1_hex * num2_hex)[2:].upper()
elif self.method == ":":
    if num2_hex == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return hex(num1_hex // num2_hex)[2:].upper()
else:
    raise ValueError(f"Unknown method {self.method}")
```

Hình 8: Tinh toán và xử lí kết quả trả về.

2. File GUI

```
import tkinter as tk
from tkinter import ttk
from tkinter import Menu, messagebox as msg, filedialog
from calculate import Calculator

class BaseCalculatorApp:
    def __init__(self, parent):
        self.parent = parent
        self.create_menu()

    def create_menu(self):
        # Tạo menu cho cửa sổ chính
        menu_bar = Menu(self.parent.winfo_toplevel())
        self.parent.winfo_toplevel().config(menu=menu_bar)

        # Menu File
        file_menu = Menu(menu_bar, tearoff=0)
        menu_bar.add_cascade(label="File", menu=file_menu)
        file_menu.add_command(label="Open", command=self.open_file)
        file_menu.add_command(label="Save", command=self.save_file)
        file_menu.add_separator()
        file_menu.add_command(label="Exit", command=self.parent.quit)
```

Hình 9: File GUI và các hàm bên trong

1.1. Class BaseCalculatorApp.

Dùng để tạo giao diện chính cho Máy tính, các phần cơ bản trong Menu:

- File gồm Open file đã lưu, Save dung để lưu đáp án vào file .txt và Exit dung để đóng máy tính.
- Freferences gồm Change Theme, và Language sẽ được cập nhật sau này.
- Help gồm Manual hiện thì hướng dẫn sử dụng, About hiện thị thông tin máy tính.

1.2 . Class DecCalculatorApp.

Kế thùa BaseCalculatorApp để sử dụng giao diện lúc nãy vừa mới tạo.

Khởi tạo __init__ với các biến là 2 số num1 và num2 được sử dụng để thực hiện tính toán và lưu kết quả vào biến result.

```
class DecCalculatorApp(BaseCalculatorApp):
    def __init__(self, parent):
        super().__init__(parent)
        self.num1 = tk.StringVar()
        self.num2 = tk.StringVar()
        self.result = tk.StringVar()
        self.create_dec_tab(parent)
```

Hình 10: Hàm __init__ của DecCalculator.

Hàm create_dec_tab dung để tạo giao diện cho phần tab tính Dec.

- LabelFrame để tạo các khung cho cho GUI như Nhập số, Phép toán và Kết quả.

- Label để tạo nhãn, thể hiện nội dung.

- Entry để nhập 2 số vào từ bàn phím và lưu trữ kết quả.

- Button để tạo nút nhấn cho các phép toán.

```
def create_dec_tab(self, parent):
    dec_frame = ttk.Frame(parent)  # Sử dụng Frame cho tab Decimal
    parent.add(dec_frame, text="Decimal")  # Thêm frame vào tab_control

    input_frame = ttk.LabelFrame(dec_frame, text="Nhập số", padding=(10, 10))
    input_frame.grid(column=0, row=0, padx=10, pady=10)

    ttk.Label(input_frame, text="Nhập số thập phân thứ nhất:").grid(column=0, row=0, sticky='W')
    ttk.Entry(input_frame, width=12, textvariable=self.num1).grid(column=1, row=0, sticky='W')

    ttk.Label(input_frame, text="Nhập số thập phân thứ hai:").grid(column=0, row=1, sticky='W')
    ttk.Entry(input_frame, width=12, textvariable=self.num2).grid(column=1, row=1, sticky='W')

    button_frame = ttk.LabelFrame(dec_frame, text="Phép toán", padding=(10, 10))
    button_frame.grid(column=1, row=0, padx=10, pady=10)

    ttk.Button(button_frame, text="+", command=self.add).grid(column=0, row=0)
    ttk.Button(button_frame, text="-", command=self.subtract).grid(column=0, row=1)
    ttk.Button(button_frame, text="x", command=self.multiply).grid(column=1, row=0)
    ttk.Button(button_frame, text=":", command=self.divide).grid(column=1, row=1)

    result_frame = ttk.LabelFrame(dec_frame, text="Kết quả", padding=(10, 10))
    result_frame.grid(column=0, row=1, padx=10, pady=10)

    ttk.Entry(result_frame, width=24, textvariable=self.result, state='readonly').grid(column=0, row=0)
```

Hình 11: Hàm create_dec_tab.

Hàm perform_calculation dung để lấy giá trị của 2 số đã nhập vào từ GUI. Sau đó sử dụng module calculate.py và gọi đến Class Calculator để xử lí tính toán.

```
def perform_calculation(self, method):
    try:
        num1 = float(self.num1.get())
        num2 = float(self.num2.get())
        calc = Calculator(num1, num2, method)
        return calc.TinhDec()
    except ValueError:
        msg.showerror("Lỗi", "Vui lòng nhập vào các số hợp lệ.")
        return ""
```

Hình 12: Hàm perform_calculation

Sau đó tạo lần lượt các hàm add, subtract, multiply và divide để set kết quả dựa trên phép toán đã chọn.

```python
def add(self):
    self.result.set(self.perform_calculation("+"))

def subtract(self):
    self.result.set(self.perform_calculation("-"))

def multiply(self):
    self.result.set(self.perform_calculation("x"))

def divide(self):
    if float(self.num2.get()) == 0:
        msg.showerror("Lỗi", "Không thể chia cho 0")
    else:
        self.result.set(self.perform_calculation(":"))
```

Hình 13: Các hàm thực hiện tính toán và ghi kết quả.

1.3. Class BinCalculatorApp.

Được xây dựng GUI tương tự như class DecCalculatorApp. Nhưng các phép toán được xây đựng phù hợp với mã nhị phân như: and, xor, or, =

```python
def perform_calculation(self, method):
    try:
        calc = Calculator(self.num1.get(), self.num2.get(), method)
        return calc.TinhBin()
    except ValueError as e:
        msg.showerror("Lỗi", str(e))
        return ""

def and_op(self):
    self.result.set(self.perform_calculation("AND"))

def or_op(self):
    self.result.set(self.perform_calculation("OR"))

def xor_op(self):
    self.result.set(self.perform_calculation("XOR"))

def equal_op(self):
    result = self.perform_calculation("=")
    self.result.set("True" if result else "False")
```

Hình 14: Các hàm trong Bin.

1.4. Class HexCalculatorApp.

Tương tự như DecCalculator, ta xây dựng GUI để thực hiện nhập xuất cho tab Hex

```
class HexCalculatorApp(BaseCalculatorApp):
    def create_hex_tab(self, parent):
        button_frame = ttk.LabelFrame(hex_frame, text="Phep toan", padding=(10, 10))
        button_frame.grid(column=1, row=0, padx=10, pady=10)

        ttk.Button(button_frame, text="+", command=self.add).grid(column=0, row=0)
        ttk.Button(button_frame, text="-", command=self.subtract).grid(column=0, row=1)
        ttk.Button(button_frame, text="x", command=self.multiply).grid(column=1, row=0)
        ttk.Button(button_frame, text=":", command=self.divide).grid(column=1, row=1)

        result_frame = ttk.LabelFrame(hex_frame, text="Kết quả", padding=(10, 10))
        result_frame.grid(column=0, row=1, padx=10, pady=10)

        ttk.Entry(result_frame, width=24, textvariable=self.result, state='readonly').grid(column=0, row=0)

    def perform_calculation(self, method):
        try:
            num1 = self.num1.get()
            num2 = self.num2.get()
            calc = Calculator(num1, num2, method)
            return calc.TinhHex()
        except ValueError:
            msg.showerror("Lỗi", "Vui lòng nhập vào các số hợp lệ.")
            return ""

    def add(self):
        self.result.set(self.perform_calculation("+"))

    def subtract(self):
        self.result.set(self.perform_calculation("-"))

    def multiply(self):
        self.result.set(self.perform_calculation("x"))
```

Hình 15: Class HexCalculator.

1.4. Hàm __main__.

Sau khi đã hoàn thành các class và hàm ta tiến hành thực hiện chương trình.

Tạo GUI bằng tkinner tên là root

Dùng Notebook của tkinter để tạo ra GUI chứa được các tab như Dec, Bin và Hex.

Truyền Notebook vào các Class DecCalculatorApp, BinCalculatorApp, HexCalculatorApp để khởi tạo.

Dung hàm mainloop() để thực hiện GUI.

CHƯƠNG 4. GitHub

Link github của dự án: https://github.com/PhucNguyenne/Python_NC_Bai1