**1. وارد کردن کتابخانه‌ها**

import tkinter as tk  
from tkinter import ttk

در این بخش، ما دو کتابخانه اصلی tkinter و ttk را وارد می‌کنیم. tkinter برای ساخت رابط گرافیکی و ttk برای استفاده از ویجت‌های گرافیکی پیشرفته‌تر مانند دکمه‌ها و فریم‌ها به کار می‌رود.

**2. تعریف کلاس CalculatorApp**

class CalculatorApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Calculator")  
 self.root.geometry("300x400")  
 self.root.resizable(True, True)  
  
 self.current\_input = ""  
 self.result\_var = tk.StringVar()  
  
 self.create\_widgets()  
 self.bind\_keys()

self.bind\_keys()در این بخش، کلاس CalculatorApp تعریف می‌شود که مسئول مدیریت رابط کاربری ماشین حساب است. در متد \_\_init\_\_:

* root (پنجره اصلی برنامه) به عنوان ورودی دریافت می‌شود.
* عنوان پنجره و ابعاد آن تعیین می‌شود.
* current\_input متغیری است که ورودی فعلی ماشین حساب را ذخیره می‌کند.
* result\_var یک متغیر از نوع StringVar است که برای نمایش نتیجه محاسبات استفاده می‌شود.
* متد create\_widgets() برای ساخت ویجت‌های گرافیکی (مثل دکمه‌ها) فراخوانی می‌شود.
* متد bind\_keys() برای پشتیبانی از ورودی‌های صفحه‌کلید فراخوانی می‌شود.

**3. ساخت ویجت‌ها (create\_widgets)**

def create\_widgets(self):  
 # Entry widget for display  
 display = ttk.Entry(self.root, textvariable=self.result\_var, font=("Arial", 24), justify="right", state="readonly")  
 display.pack(fill="x", padx=10, pady=10)  
  
 # Button layout  
 buttons = [  
 ['%', 'CE', 'C', '⌫'],  
 ['1/x', 'x²', '√x', '÷'],  
 ['7', '8', '9', '×'],  
 ['4', '5', '6', '-'],  
 ['1', '2', '3', '+'],  
 ['±', '0', '.', '=']  
 ]  
  
 for row in buttons:  
 frame = ttk.Frame(self.root)  
 frame.pack(fill="x", padx=10)  
 for btn\_text in row:  
 button = ttk.Button(frame, text=btn\_text, command=lambda b=btn\_text: self.on\_button\_click(b))  
 button.pack(side="left", expand=True, fill="both", padx=2, pady=2)

در این متد:

* ابتدا یک Entry ویجت برای نمایش ورودی و نتیجه محاسبات ایجاد می‌شود.
* سپس یک لیست buttons حاوی دکمه‌ها تعریف می‌شود. این دکمه‌ها شامل اعداد، عملگرها و عملکردهای خاص (مانند C برای پاک کردن ورودی) هستند.
* برای هر ردیف از دکمه‌ها یک Frame ایجاد می‌شود و دکمه‌ها در داخل آن فریم‌ها قرار می‌گیرند. هر دکمه با متد on\_button\_click() مرتبط است که در صورت کلیک روی دکمه فراخوانی می‌شود.

**4. پشتیبانی از ورودی‌های صفحه‌کلید (bind\_keys)**

def bind\_keys(self):  
 self.root.bind("<Key>", self.on\_key\_press)

این متد برای پشتیبانی از ورودی‌های صفحه‌کلید استفاده می‌شود. با فشردن هر کلید، متد on\_key\_press() فراخوانی می‌شود که ورودی صفحه‌کلید را پردازش می‌کند.

**5. واکنش به کلیک دکمه‌ها (on\_button\_click)**

def on\_button\_click(self, button):  
 if button in "0123456789":  
 self.current\_input += button  
 elif button in "+-×÷":  
 self.current\_input += {"×": "\*", "÷": "/"}[button]  
 elif button == "C":  
 self.current\_input = ""  
 elif button == "⌫":  
 self.current\_input = self.current\_input[:-1]  
 elif button == "=":  
 self.calculate\_result()  
 elif button == "CE":  
 self.current\_input = ""  
 elif button == "x²":  
 self.current\_input += "\*\*2"  
 elif button == "√x":  
 self.current\_input += "\*\*0.5"  
 elif button == "1/x":  
 self.current\_input = f"1/({self.current\_input})"  
 elif button == "±":  
 self.current\_input = f"-({self.current\_input})"  
 elif button == ".":  
 self.current\_input += "."  
 self.update\_display()

این متد زمانی که یک دکمه کلیک می‌شود فراخوانی می‌شود. بسته به دکمه‌ای که فشار داده شده، ورودی به‌روز می‌شود:

* اگر دکمه یک عدد یا علامت ریاضی باشد، آن را به current\_input اضافه می‌کند.
* اگر دکمه خاصی مانند C (پاک کردن ورودی) یا (حذف آخرین ورودی) باشد، عمل متناسب انجام می‌شود.
* اگر دکمه = باشد، متد calculate\_result() برای محاسبه نتیجه فراخوانی می‌شود.

**6. پشتیبانی از ورودی‌های صفحه‌کلید (on\_key\_press)**

def on\_key\_press(self, event):  
 char = event.char  
 if char in "0123456789.+-\*/":  
 self.current\_input += char  
 elif event.keysym == "BackSpace":  
 self.current\_input = self.current\_input[:-1]  
 elif event.keysym in ("Return", "KP\_Enter"):  
 self.calculate\_result()  
 self.update\_display()

این متد به ورودی‌های صفحه‌کلید پاسخ می‌دهد:

* اگر کلیدی از اعداد، علامت‌های ریاضی یا نقطه فشرده شود، آن را به ورودی اضافه می‌کند.
* اگر کلید BackSpace فشرده شود، آخرین کاراکتر از ورودی حذف می‌شود.
* اگر کلید Enter (یا Return) فشرده شود، نتیجه محاسبه می‌شود.

**7. محاسبه نتیجه (calculate\_result)**

def calculate\_result(self):  
 try:  
 result = eval(self.current\_input)  
 self.current\_input = str(result)  
 except Exception:  
 self.current\_input = "Error"

در این متد، با استفاده از تابع eval(), ورودی به‌عنوان یک عبارت ریاضی محاسبه می‌شود. اگر محاسبه موفقیت‌آمیز باشد، نتیجه در current\_input ذخیره می‌شود، در غیر این صورت "Error" نمایش داده می‌شود.

**8. به‌روزرسانی نمایشگر (update\_display)**

def update\_display(self):  
 self.result\_var.set(self.current\_input)

این متد برای به‌روزرسانی محتوای نمایشگر (ویجت Entry) استفاده می‌شود. current\_input به عنوان محتوای نمایشگر تنظیم می‌شود.

**9. اجرای برنامه**

if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = CalculatorApp(root)  
 root.mainloop()

در این بخش، برنامه اجرا می‌شود. یک نمونه از CalculatorApp ایجاد شده و پنجره اصلی Tk نمایش داده می‌شود.

**نتیجه‌گیری:**

این برنامه یک ماشین حساب ساده با رابط کاربری گرافیکی ایجاد می‌کند که از عملیات پایه‌ای ریاضی مثل جمع، تفریق، ضرب و تقسیم پشتیبانی می‌کند. همچنین از قابلیت‌های پیشرفته‌تری مانند محاسبه معکوس (1/x)، ریشه (√x)، توان (x²)، و پشتیبانی از ورودی صفحه‌کلید برخوردار است.