perplexity

Simple Python Interest Calculator App (Tkinter)

Below is a simple Python application using the `tkinter` library. This app calculates:

- **Total Interest** based on the principal amount, annual interest rate, and savings duration (in months)
- **Monthly Payment** (total amount divided by the number of months)

Features

- User-friendly graphical interface
- Input validation with error messages for invalid entries
- Instant calculation of results

Complete Code Example

```
import tkinter as tk
from tkinter import messagebox

def calculate_interest():
    try:
        amount = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())
        # Calculate total interest
        total_interest = amount * (annual_rate / 100) * (months / 12)
        # Calculate monthly payment
        monthly_payment = (amount + total_interest) / months
        label_result_total.config(text=f"Total Interest: {total_interest:.2f}")
        label_result_monthly.config(text=f"Monthly Amount: {monthly_payment:.2f}")
    except ValueError:
        messagebox.showerror("Input error", "Please enter valid numeric values.")

# Create main window
root = tk.Tk()
root.title("Interest Calculator")

# Widgets
tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
entry_amount = tk.Entry(root)
entry_amount.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=5)
entry_rate = tk.Entry(root)
entry_rate.grid(row=1, column=1, padx=10, pady=5)
```

```
tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
entry_months = tk.Entry(root)
entry_months.grid(row=2, column=1, padx=10, pady=5)

tk.Button(root, text="Calculate", command=calculate_interest).grid(row=3, column=0, colum

label_result_total = tk.Label(root, text="Total Interest: ")
label_result_total.grid(row=4, column=0, columnspan=2)

label_result_monthly = tk.Label(root, text="Monthly Amount: ")
label_result_monthly.grid(row=5, column=0, columnspan=2)

root.mainloop()
```

How It Works

1. **Enter** the principal amount, annual interest rate, and duration in months.
2. **Click** "Calculate".
3. **View** the total interest and monthly payment displayed below.

Notes

- This app assumes simple interest (not compound).
- The monthly payment is calculated as:

$$\text{Monthly Payment} = \frac{\text{Principal} + \text{Total Interest}}{\text{Months}}$$

- Error handling is included for non-numeric input.

You can copy and run this code in any Python environment with `tkinter` installed.

⁂

monthly amount is wrong show only interest also show the total amount after end of period

Got it! I'll adjust the app to:

- Show **only the total interest** as the "Total Interest"
- Add a label to show the **total amount after the period**, which is principal + interest
- Remove the "monthly amount" display since you want to see interest totals only

Here's the updated Python tkinter app code:

```
import tkinter as tk
from tkinter import messagebox

def calculate_interest():
    try:
```

```
        amount = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())

        # Calculate total interest
        total_interest = amount * (annual_rate / 100) * (months / 12)
        # Calculate total amount after the period
        total_amount = amount + total_interest

        # Display results
        label_result_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_result_total.config(text=f"Total Amount after {months} months: {total_amour
    except ValueError:
        messagebox.showerror("Input error", "Please enter valid numeric values.")

# Create main window
root = tk.Tk()
root.title("Interest Calculator")

# Widgets
tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
entry_amount = tk.Entry(root)
entry_amount.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=5)
entry_rate = tk.Entry(root)
entry_rate.grid(row=1, column=1, padx=10, pady=5)

tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
entry_months = tk.Entry(root)
entry_months.grid(row=2, column=1, padx=10, pady=5)

tk.Button(root, text="Calculate", command=calculate_interest).grid(row=3, column=0, colum

label_result_interest = tk.Label(root, text="Total Interest: ")
label_result_interest.grid(row=4, column=0, columnspan=2)

label_result_total = tk.Label(root, text="Total Amount after period: ")
label_result_total.grid(row=5, column=0, columnspan=2)

root.mainloop()
```

This now clearly shows total interest earned and the total amount at the end of the saving period without the previous monthly installment calculation. Let me know if you'd like any other changes!

shoq monthly interest amount

Sure! To show the **monthly interest amount**, we can add a calculation for the interest earned **each month** and display it as well.

Since this is simple interest, the monthly interest is just the total interest divided by the number of months.

I'll update the app to show:

- Total interest
- Total amount after the period
- Monthly interest amount

Here is the updated code:

```
import tkinter as tk
from tkinter import messagebox

def calculate_interest():
    try:
        amount = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())

        # Calculate total interest (simple interest)
        total_interest = amount * (annual_rate / 100) * (months / 12)
        # Calculate total amount after the period
        total_amount = amount + total_interest
        # Calculate monthly interest amount
        monthly_interest = total_interest / months if months != 0 else 0

        # Display results
        label_result_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_result_total.config(text=f"Total Amount after {months} months: {total_amoun
        label_result_monthly_interest.config(text=f"Monthly Interest Amount: {monthly_int
    except ValueError:
        messagebox.showerror("Input error", "Please enter valid numeric values.")

# Create main window
root = tk.Tk()
root.title("Interest Calculator")

# Widgets
tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
entry_amount = tk.Entry(root)
entry_amount.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=5)
entry_rate = tk.Entry(root)
entry_rate.grid(row=1, column=1, padx=10, pady=5)

tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
entry_months = tk.Entry(root)
entry_months.grid(row=2, column=1, padx=10, pady=5)

tk.Button(root, text="Calculate", command=calculate_interest).grid(row=3, column=0, colum

label_result_interest = tk.Label(root, text="Total Interest: ")
label_result_interest.grid(row=4, column=0, columnspan=2)

label_result_total = tk.Label(root, text="Total Amount after period: ")
label_result_total.grid(row=5, column=0, columnspan=2)
```

```
label_result_monthly_interest = tk.Label(root, text="Monthly Interest Amount: ")
label_result_monthly_interest.grid(row=6, column=0, columnspan=2)

root.mainloop()
```

This version adds a **Monthly Interest Amount** display showing the interest earned per month for the given duration.

Let me know if you'd like it further customized!

Can you explain how you're calculating the monthly amount and interest? What formula are you using?

Certainly! Here's the explanation of the formulas used to calculate the interest and amounts in the app:

1. Total Interest (Simple Interest)

The app uses the **simple interest formula**, which assumes interest is calculated only on the original principal amount (not compounded).

$$\text{Total Interest} = \text{Principal} \times \frac{\text{Annual Interest Rate}}{100} \times \frac{\text{Months}}{12}$$

- **Principal** = initial amount saved or invested
- **Annual Interest Rate** = interest rate per year (in %)
- **Months** = duration the money is saved for, in months
- We divide months by 12 to convert the duration to years, because the interest rate is annual.

2. Total Amount After the Period

$$\text{Total Amount} = \text{Principal} + \text{Total Interest}$$

This is the total money you'll have at the end of the saving period, including your original principal plus the interest earned.

3. Monthly Interest Amount

For simplicity, the app assumes the total interest is evenly distributed across all months (since this is simple interest, not compounded monthly):

$$\text{Monthly Interest} = \frac{\text{Total Interest}}{\text{Months}}$$

This represents how much interest you earn on average each month over the saving duration.

Summary:

- The formula is based on simple (not compound) interest.
- Interest is calculated proportionally based on the time period in years.
- Monthly interest is the average monthly share of the total interest.

This method is straightforward and widely used for estimating interest when compounding is not considered.

Let me know if you'd like the formulas for compound interest or anything else!

Can you refactor the code to improve readability and maintainability?

Certainly! Here is the refactored code with improved readability and maintainability by:

- Encapsulating the calculation logic in a separate function
- Grouping UI creation in a dedicated function
- Adding comments for clarity
- Using consistent naming conventions

```
import tkinter as tk
from tkinter import messagebox

def calculate_simple_interest(principal, annual_rate, months):
    """
    Calculate simple interest details:
    - total interest earned
    - total amount after interest
    - monthly interest amount
    """
    total_interest = principal * (annual_rate / 100) * (months / 12)
    total_amount = principal + total_interest
    monthly_interest = total_interest / months if months else 0
    return total_interest, total_amount, monthly_interest

def on_calculate():
    """
    Handler for calculate button.
    Fetch inputs, validate, compute values, and update UI with results.
    """
    try:
        principal = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())

        total_interest, total_amount, monthly_interest = calculate_simple_interest(
            principal, annual_rate, months)

        label_total_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_total_amount.config(text=f"Total Amount after {months} months: {total_amour
```

```
        label_monthly_interest.config(text=f"Monthly Interest Amount: {monthly_interest:.
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values.")

def create_ui():
    """
    Set up the tkinter UI elements.
    """
    root = tk.Tk()
    root.title("Interest Calculator")

    tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
    global entry_amount
    entry_amount = tk.Entry(root)
    entry_amount.grid(row=0, column=1, padx=10, pady=5)

    tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=
    global entry_rate
    entry_rate = tk.Entry(root)
    entry_rate.grid(row=1, column=1, padx=10, pady=5)

    tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
    global entry_months
    entry_months = tk.Entry(root)
    entry_months.grid(row=2, column=1, padx=10, pady=5)

    btn_calculate = tk.Button(root, text="Calculate", command=on_calculate)
    btn_calculate.grid(row=3, column=0, columnspan=2, pady=10)

    global label_total_interest, label_total_amount, label_monthly_interest
    label_total_interest = tk.Label(root, text="Total Interest: ")
    label_total_interest.grid(row=4, column=0, columnspan=2)

    label_total_amount = tk.Label(root, text="Total Amount after period: ")
    label_total_amount.grid(row=5, column=0, columnspan=2)

    label_monthly_interest = tk.Label(root, text="Monthly Interest Amount: ")
    label_monthly_interest.grid(row=6, column=0, columnspan=2)

    return root

if __name__ == "__main__":
    window = create_ui()
    window.mainloop()
```

Improvements:

- Calculation logic separated in `calculate_simple_interest()` for easy testing or reuse.
- UI building in `create_ui()` keeps main script clean.
- Event handler `on_calculate()` clearly manages input/output.
- Comments explain purpose of functions and key logic.

This structure improves clarity and makes future extension easier. Let me know if you'd like it further modularized or additional features!Certainly! Here's the refactored code with clear separation of concerns, well-named functions, concise comments, and organized UI setup for better readability and maintainability:

```
import tkinter as tk
from tkinter import messagebox

def calculate_simple_interest(principal, annual_rate, months):
    """
    Calculate simple interest, total amount, and monthly interest.
    """
    total_interest = principal * (annual_rate / 100) * (months / 12)
    total_amount = principal + total_interest
    monthly_interest = total_interest / months if months else 0
    return total_interest, total_amount, monthly_interest

def on_calculate():
    """
    Retrieve inputs, validate, compute interest, and update output labels.
    """
    try:
        principal = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())

        total_interest, total_amount, monthly_interest = calculate_simple_interest(
            principal, annual_rate, months)

        label_total_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_total_amount.config(text=f"Total Amount after {months} months: {total_amour
        label_monthly_interest.config(text=f"Monthly Interest Amount: {monthly_interest:.
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values.")

def create_ui():
    """
    Setup the Tkinter UI elements.
    """
    root = tk.Tk()
    root.title("Interest Calculator")

    tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
    global entry_amount
    entry_amount = tk.Entry(root)
    entry_amount.grid(row=0, column=1, padx=10, pady=5)

    tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=
    global entry_rate
    entry_rate = tk.Entry(root)
    entry_rate.grid(row=1, column=1, padx=10, pady=5)

    tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
    global entry_months
    entry_months = tk.Entry(root)
```

```
    entry_months.grid(row=2, column=1, padx=10, pady=5)

    tk.Button(root, text="Calculate", command=on_calculate).grid(row=3, column=0, columns

    global label_total_interest, label_total_amount, label_monthly_interest
    label_total_interest = tk.Label(root, text="Total Interest: ")
    label_total_interest.grid(row=4, column=0, columnspan=2)

    label_total_amount = tk.Label(root, text="Total Amount after period: ")
    label_total_amount.grid(row=5, column=0, columnspan=2)

    label_monthly_interest = tk.Label(root, text="Monthly Interest Amount: ")
    label_monthly_interest.grid(row=6, column=0, columnspan=2)

    return root

if __name__ == "__main__":
    app = create_ui()
    app.mainloop()
```

This version keeps the interest calculation logic separated and clearly handles the UI events and element creation. This makes future updates or testing easier. Let me know if you want additional features or enhancements!

Can you add input validation to ensure users don't enter negative numbers or non-numeric values?

Certainly! I've enhanced the input validation in the `on_calculate` function to ensure users cannot enter negative numbers or non-numeric values. If invalid inputs are detected, an error message is shown.

Here is the updated code with strict validation:

```
import tkinter as tk
from tkinter import messagebox

def calculate_simple_interest(principal, annual_rate, months):
    """
    Calculate simple interest, total amount, and monthly interest.
    """
    total_interest = principal * (annual_rate / 100) * (months / 12)
    total_amount = principal + total_interest
    monthly_interest = total_interest / months if months else 0
    return total_interest, total_amount, monthly_interest

def validate_inputs(principal, annual_rate, months):
    """
    Validate inputs to be positive numbers.
    """
    if principal < 0:
        raise ValueError("Amount must be non-negative.")
    if annual_rate < 0:
```

```
        raise ValueError("Annual Interest Rate must be non-negative.")
    if months <= 0:
        raise ValueError("Duration (months) must be greater than zero.")

def on_calculate():
    """
    Retrieve inputs, validate, compute interest, and update output labels.
    """
    try:
        principal = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())

        # Validate inputs
        validate_inputs(principal, annual_rate, months)

        total_interest, total_amount, monthly_interest = calculate_simple_interest(
            principal, annual_rate, months)

        label_total_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_total_amount.config(text=f"Total Amount after {months} months: {total_amoun
        label_monthly_interest.config(text=f"Monthly Interest Amount: {monthly_interest:.
    except ValueError as e:
        messagebox.showerror("Input Error", str(e))

def create_ui():
    """
    Setup the Tkinter UI elements.
    """
    root = tk.Tk()
    root.title("Interest Calculator")

    tk.Label(root, text="Amount:").grid(row=0, column=0, padx=10, pady=5)
    global entry_amount
    entry_amount = tk.Entry(root)
    entry_amount.grid(row=0, column=1, padx=10, pady=5)

    tk.Label(root, text="Annual Interest Rate (%):").grid(row=1, column=0, padx=10, pady=
    global entry_rate
    entry_rate = tk.Entry(root)
    entry_rate.grid(row=1, column=1, padx=10, pady=5)

    tk.Label(root, text="Duration (months):").grid(row=2, column=0, padx=10, pady=5)
    global entry_months
    entry_months = tk.Entry(root)
    entry_months.grid(row=2, column=1, padx=10, pady=5)

    tk.Button(root, text="Calculate", command=on_calculate).grid(row=3, column=0, columns

    global label_total_interest, label_total_amount, label_monthly_interest
    label_total_interest = tk.Label(root, text="Total Interest: ")
    label_total_interest.grid(row=4, column=0, columnspan=2)

    label_total_amount = tk.Label(root, text="Total Amount after period: ")
    label_total_amount.grid(row=5, column=0, columnspan=2)
```

```
    label_monthly_interest = tk.Label(root, text="Monthly Interest Amount: ")
    label_monthly_interest.grid(row=6, column=0, columnspan=2)

    return root

if __name__ == "__main__":
    app = create_ui()
    app.mainloop()
```

What was added:

- A new function `validate_inputs()` that checks:
  - Amount and interest rate are not negative
  - Duration is greater than zero (no zero or negative months)
- The `on_calculate()` calls `validate_inputs()` and shows an error popup if validation fails.

This prevents invalid or nonsensical inputs ensuring more robust app behavior. Let me know if you'd like me to add any other validation rules!

Can you improve the app's user interface to make it more user-friendly?

Certainly! Here are some improvements to make the app's user interface more user-friendly:

- Use padding and consistent spacing for better layout
- Add field labels aligned to the left
- Group inputs and results visually separated
- Use larger font for headings or results for emphasis
- Add placeholder text to input fields (using default text cleared on focus)
- Add basic input focus behavior to clear placeholders
- Use a fixed window size to prevent resizing issues

Here is the improved UI code:

```
import tkinter as tk
from tkinter import messagebox, font

def calculate_simple_interest(principal, annual_rate, months):
    total_interest = principal * (annual_rate / 100) * (months / 12)
    total_amount = principal + total_interest
    monthly_interest = total_interest / months if months else 0
    return total_interest, total_amount, monthly_interest

def validate_inputs(principal, annual_rate, months):
    if principal < 0:
        raise ValueError("Amount must be non-negative.")
    if annual_rate < 0:
```

```
        raise ValueError("Annual Interest Rate must be non-negative.")
    if months <= 0:
        raise ValueError("Duration (months) must be greater than zero.")

def on_calculate():
    try:
        principal = float(entry_amount.get())
        annual_rate = float(entry_rate.get())
        months = int(entry_months.get())
        validate_inputs(principal, annual_rate, months)

        total_interest, total_amount, monthly_interest = calculate_simple_interest(
            principal, annual_rate, months)

        label_total_interest.config(text=f"Total Interest: {total_interest:.2f}")
        label_total_amount.config(text=f"Total Amount after {months} months: {total_amoun
        label_monthly_interest.config(text=f"Monthly Interest Amount: {monthly_interest:.
    except ValueError as e:
        messagebox.showerror("Input Error", str(e))

def clear_placeholder(event, entry, placeholder):
    if entry.get() == placeholder:
        entry.delete(0, tk.END)
        entry.config(fg='black')

def add_placeholder(entry, placeholder):
    entry.insert(0, placeholder)
    entry.config(fg='grey')
    entry.bind("<FocusIn>", lambda event: clear_placeholder(event, entry, placeholder))
    entry.bind("<FocusOut>", lambda event: add_placeholder_on_focus_out(event, entry, pla

def add_placeholder_on_focus_out(event, entry, placeholder):
    if entry.get() == '':
        entry.insert(0, placeholder)
        entry.config(fg='grey')

def create_ui():
    root = tk.Tk()
    root.title("Interest Calculator")
    root.geometry("350x350")
    root.resizable(False, False)

    # Fonts
    heading_font = font.Font(root=root, family="Helvetica", size=14, weight="bold")
    result_font = font.Font(root=root, family="Helvetica", size=12)

    # Heading
    tk.Label(root, text="Simple Interest Calculator", font=heading_font).pack(pady=10)

    # Input frame for cleaner group
    input_frame = tk.Frame(root)
    input_frame.pack(pady=10, padx=20, fill='x')

    tk.Label(input_frame, text="Amount:").grid(row=0, column=0, sticky='w', pady=5)
    global entry_amount
    entry_amount = tk.Entry(input_frame)
```

```
    entry_amount.grid(row=0, column=1, pady=5)
    add_placeholder(entry_amount, "e.g. 1000")

    tk.Label(input_frame, text="Annual Interest Rate (%):").grid(row=1, column=0, sticky=
    global entry_rate
    entry_rate = tk.Entry(input_frame)
    entry_rate.grid(row=1, column=1, pady=5)
    add_placeholder(entry_rate, "e.g. 5")

    tk.Label(input_frame, text="Duration (months):").grid(row=2, column=0, sticky='w', pa
    global entry_months
    entry_months = tk.Entry(input_frame)
    entry_months.grid(row=2, column=1, pady=5)
    add_placeholder(entry_months, "e.g. 12")

    tk.Button(root, text="Calculate", command=on_calculate, bg='blue', fg='white').pack(p

    # Results frame
    result_frame = tk.Frame(root)
    result_frame.pack(pady=5, padx=20, fill='x')

    global label_total_interest, label_total_amount, label_monthly_interest
    label_total_interest = tk.Label(result_frame, text="Total Interest: ", font=result_fo
    label_total_interest.pack(anchor='w', pady=3)

    label_total_amount = tk.Label(result_frame, text="Total Amount after period: ", font=
    label_total_amount.pack(anchor='w', pady=3)

    label_monthly_interest = tk.Label(result_frame, text="Monthly Interest Amount: ", for
    label_monthly_interest.pack(anchor='w', pady=3)

    return root

if __name__ == "__main__":
    app = create_ui()
    app.mainloop()
```

UI Enhancements include:

- Clear heading with large bold text
- Input fields grouped with labels aligned left
- Placeholder text in input fields for guidance
- Blue colored calculate button with padding for better clickability
- Results displayed with larger font and padding
- Fixed window size and padding for neat layout
- Placeholder text disappears when entry is focused, reappears if left blank

This should provide a much cleaner, user-friendly experience. Let me know if you'd like me to add any other UI improvements or features!