Here's a Python program for a temperature converter that allows users to choose between Fahrenheit and Celsius conversions:

Python
```python
def get_temperature():
  """Prompts user for temperature input and validates it as a
number."""
  while True:
    try:
      temperature = float(input("Enter temperature: "))
      return temperature
    except ValueError:
      print("Invalid input. Please enter a number.")

def get_conversion_choice():
  """Prompts user for conversion direction (Celsius to Fahrenheit or
vice versa)"""
  while True:
    choice = input("Enter conversion direction (C to F or F to C):
").upper()
    if choice in ('C', 'F'):
      return choice
    else:
      print("Invalid choice. Please enter 'C' or 'F'.")

def convert_temperature(temperature, from_unit, to_unit):
  """Converts temperature based on user input."""
  if from_unit == 'C' and to_unit == 'F':
    return (temperature * 9/5) + 32
  elif from_unit == 'F' and to_unit == 'C':
    return (temperature - 32) * 5/9
  else:
    print("Error: Invalid conversion direction.")
    return None  # Indicate an error

def main():
  """Main function to handle user interaction and temperature
conversion."""
  temperature = get_temperature()
  conversion_choice = get_conversion_choice()

  from_unit = 'C' if conversion_choice == 'C' else 'F'
  to_unit = 'F' if conversion_choice == 'C' else 'C'

  converted_temperature = convert_temperature(temperature, from_unit,
to_unit)

  if converted_temperature is not None:
    print(f"{temperature:.2f} degrees {from_unit} is equal to
{converted_temperature:.2f} degrees {to_unit}.")
```

```python
if __name__ == "__main__":
  main()
```

Explanation:

1. **`get_temperature()`:** This function prompts the user for temperature input and validates it using a `try-except` block to ensure a numerical value.
2. **`get_conversion_choice()`:** This function prompts the user for the conversion direction (Celsius to Fahrenheit or vice versa) and validates the input to ensure 'C' or 'F'.
3. **`convert_temperature()`:** This function performs the temperature conversion based on the user's input. It checks the conversion direction and applies the appropriate formula. It also handles potential errors.
4. **`main()`:** This function is the program's entry point. It calls the other functions to get user input, perform conversion, and display the result.

Testing the Program:

- Run the program and enter a temperature value (e.g., 25).
- Choose the conversion direction (e.g., 'C' for Celsius to Fahrenheit).
- The program should display the converted temperature (e.g., 77 degrees Fahrenheit).
- Try different input values and conversion directions to ensure the program works correctly.

This program provides a basic temperature conversion tool. You can extend it by:

- Handling invalid temperature ranges (e.g., extremely low or high values).
- Adding support for additional temperature units (e.g., Kelvin).

Sure, here's the output you'd see from running the temperature converter program:

```
Enter temperature: 25
Enter conversion direction (C to F or F to C): C
25.00 degrees C is equal to 77.00 degrees F.
```

This demonstrates the program in action. It prompts the user for the temperature (25), then the conversion direction (Celsius to Fahrenheit - 'C'). Finally, it displays the converted temperature (77 degrees Fahrenheit).

You can test the program with different values and conversion directions (Fahrenheit to Celsius - 'F') to see how it handles various scenarios.

Absolutely, here are more sample outputs demonstrating the temperature converter program's functionality:

Scenario 1: Converting Fahrenheit to Celsius:

1. **Enter temperature:**
   ```
   Enter temperature: 86
   ```
2. **Choose conversion direction (F to C):**
   ```
   Enter conversion direction (C to F or F to C): F
   ```
   Output:
   ```
   86.00 degrees F is equal to 30.00 degrees C.
   ```

Scenario 2: Handling Invalid Input:

1. **Enter invalid temperature (text):**
   ```
   Enter temperature: some text
   Invalid input. Please enter a number.
   ```
2. **Re-enter temperature (valid number):**
   ```
   Enter temperature: 100
   ```
3. **Choose conversion direction (C to F):**
   ```
   Enter conversion direction (C to F or F to C): C
   ```
   Output:
   ```
   100.00 degrees C is equal to 212.00 degrees F.
   ```

Scenario 3: Error Handling for Invalid Conversion Direction:

1. **Enter temperature:**
   ```
   Enter temperature: 0
   ```
2. **Choose invalid conversion direction:**
   ```
   Enter conversion direction (C to F or F to C): Z
   Invalid choice. Please enter 'C' or 'F'.
   ```
3. **Re-enter conversion direction (valid choice):**
   ```
   Enter conversion direction (C to F or F to C): F
   ```
   Output:
   ```
   0.00 degrees C is equal to 32.00 degrees F.
   ```

This showcases how the program handles both valid and invalid user inputs, ensuring smooth operation and informative messages when errors occur.