**Topic 77: A More Practical Example of Exception Handling**

**What**  
This topic extends the previous chapter’s error handling by introducing a loop that allows the user to keep trying until they enter a valid filename. It uses a while loop along with try and except to repeatedly ask the user for input until a valid file is opened successfully.

**Why**

* **Repeated User Input**: If the user makes a mistake (like entering a non-existent file name), we don’t want the program to crash. Instead, the user should have another chance to input the correct filename.
* **Better User Interaction**: By providing another chance, the program can continue running smoothly, enhancing the user experience.
* **Avoid Program Termination**: Without a loop, after an error occurs, the program would stop, leading to a poor experience. Using a loop ensures the program continues asking for the correct input.

**How**

1. **Using a while True loop**:  
   The while True loop keeps running indefinitely until a valid file is opened. If an error occurs, the loop continues to ask for input:

python

Copy code

while True:

1. **try block to handle errors**:  
   Inside the loop, you place the code that may cause an error (in this case, asking for a filename and opening it):

python

Copy code

try:

filename = input("What text file to open? ")

with open(filename) as f:

print(f.read())

1. **Using except to handle FileNotFoundError**:  
   If the user enters a file that doesn’t exist, the except block catches the error, displays a message, and the loop repeats, asking for the filename again:

python

Copy code

except FileNotFoundError:

print("Sorry, " + filename + " not found.")

1. **break to exit the loop**:  
   If the file is found and opened successfully, the program displays its contents and then exits the loop using break:

python

Copy code

break

1. **Indents for readability**:  
   As usual in Python, the blocks of code inside the while, try, and except must be properly indented to show their scope:

python

Copy code

while True:

try:

filename = input("What text file to open? ")

with open(filename) as f:

print(f.read())

break

except FileNotFoundError:

print("Sorry, " + filename + " not found.")

**Things to Remember**

* **while True**: This loop keeps running until explicitly stopped with a break statement. It's ideal for repeatedly asking the user for input.
* **Exception Handling**: Use try and except to catch errors (e.g., FileNotFoundError) and provide meaningful error messages.
* **Breaking the Loop**: Once the file is successfully opened and read, use break to exit the loop.
* **Indentation**: All code within the try, except, and loop blocks must be properly indented.
* **Repeated User Input**: This approach ensures that users can try again until they enter valid input, improving the program's robustness and user experience.