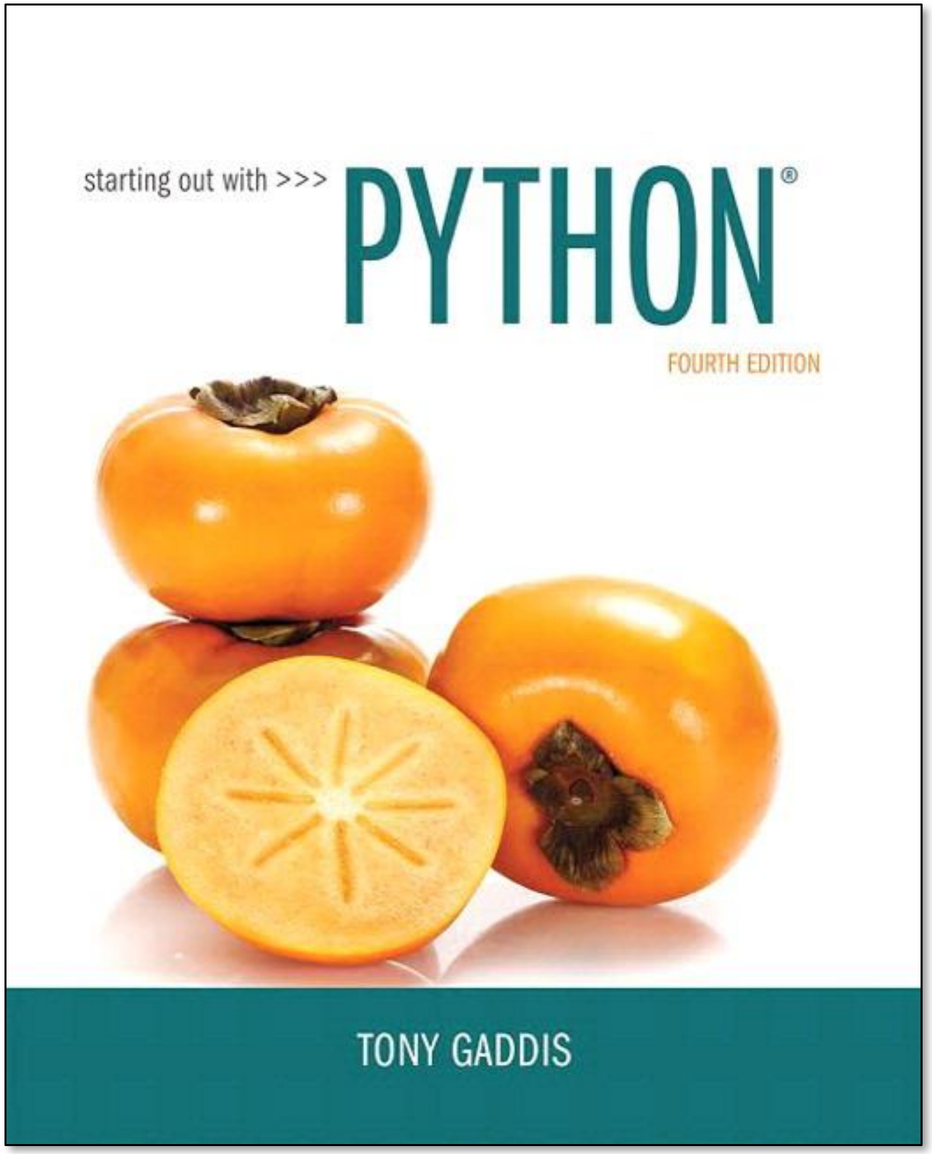
CHAPTER 6

Files and Exceptions



Dr. Bita Shadgar

Learning Outcomes

- At the end of this week the students must be able to:
  - Define an exception
  - Handle an exception
  - Handle multiple exceptions
  - Create and use an exception object

Exceptions

- **Exception**: error that occurs while a program is running
  - Usually causes program to sudden halt
- **Traceback**: error message that gives information regarding line numbers that caused the exception
  - Indicates the type of exception and brief description of the error that caused exception to be raised
    - Practice with division.py, enter 0 as second input.

Preventing Exceptions

- Many exceptions can be prevented by careful coding
  - Example: input validation
  - Usually involve a simple decision construct (devision2.py)
- Some exceptions cannot be avoided by careful coding
  - Examples
    - Trying to convert non-numeric string to an integer
    - Trying to open for reading a file that doesn’t exist

Exception Handling

- **Exception handler**: code that responds when exceptions are raised and prevents program from crashing
- Python syntax: `try/except` statement

```
try:
    statements
except exceptionName:
    statements
```

  - **Try block**: statements that can potentially raise an exception
  - **Handler**: statements contained in `except` block

- Practice with display_file.py and display_file2.py, enter a filename that doesn't exist.

How an Exception works?

- If statement in try block raises exception:
  - Exception specified in except clause:
    - Handler immediately following except clause executes
    - Continue program after try/except statement
  - Other exceptions:
    - Program halts with traceback error message
- If no exception is raised, handlers are skipped

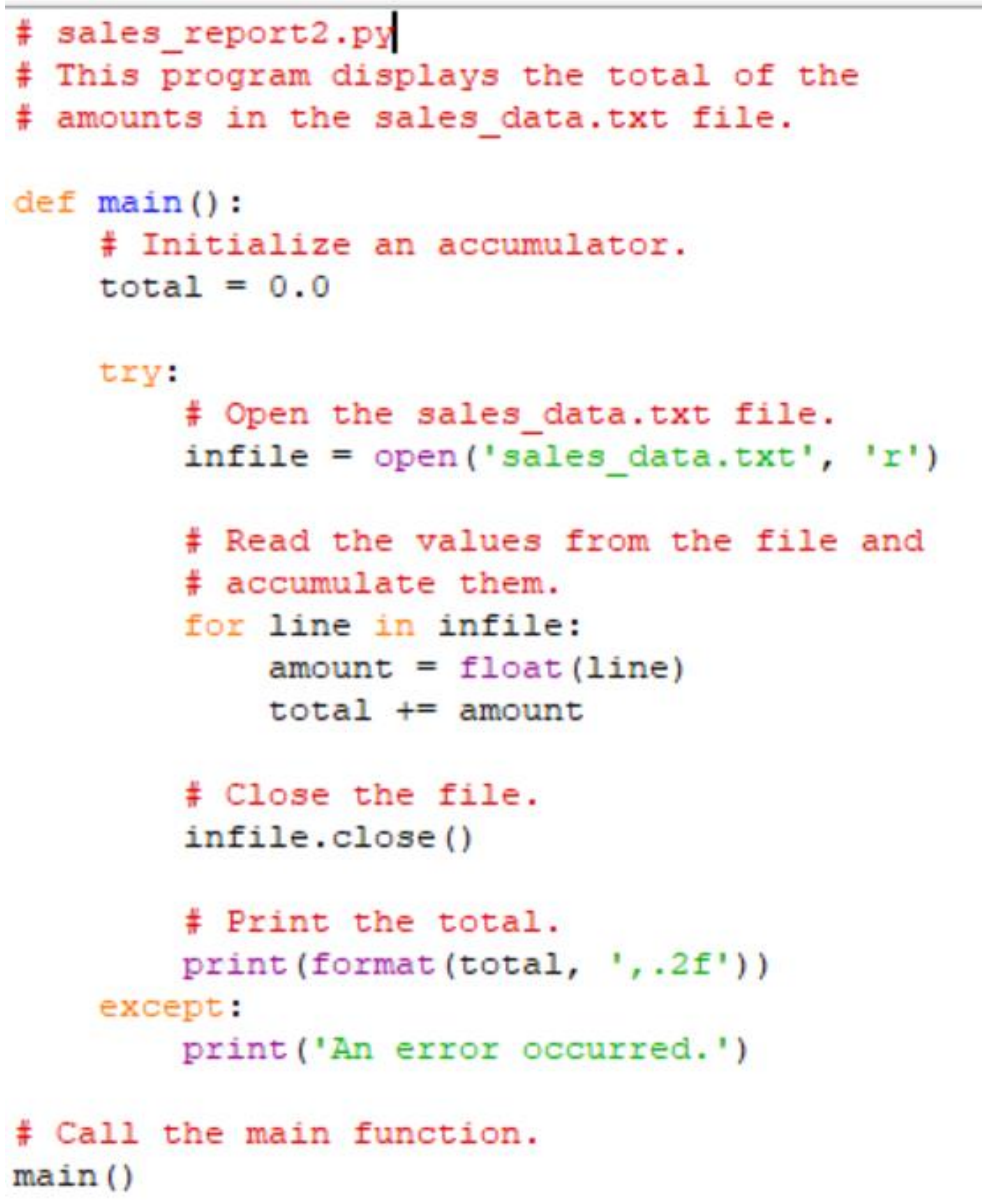
Handling Multiple Exceptions

- Often code in try block can throw more than one type of exception
  - Need to write `except` clause for each type of exception that needs to be handled
- An `except` clause that does not list a specific exception will handle any exception that is raised in the try block
  - Should always be last in a series of `except` clause

Multiple `except` Clause

```python
# This program displays the total of the
# amounts in the sales_data.txt file.

def main():
    # Initialize an accumulator.
    total = 0.0

    try:
        # Open the sales_data.txt file.
        infile = open('sales_data.txt', 'r')

        # Read the values from the file and
        # accumulate them.
        for line in infile:
            amount = float(line)
            total += amount

        # Close the file.
        infile.close()

        # Print the total.
        print(format(total, ',.2f'))

    except IOError:
        print('An error occured trying to read the file.')

    except ValueError:
        print('Non-numeric data found in the file.')

    except:
        print('An error occured.')

# Call the main function.
main()
```

except Clause

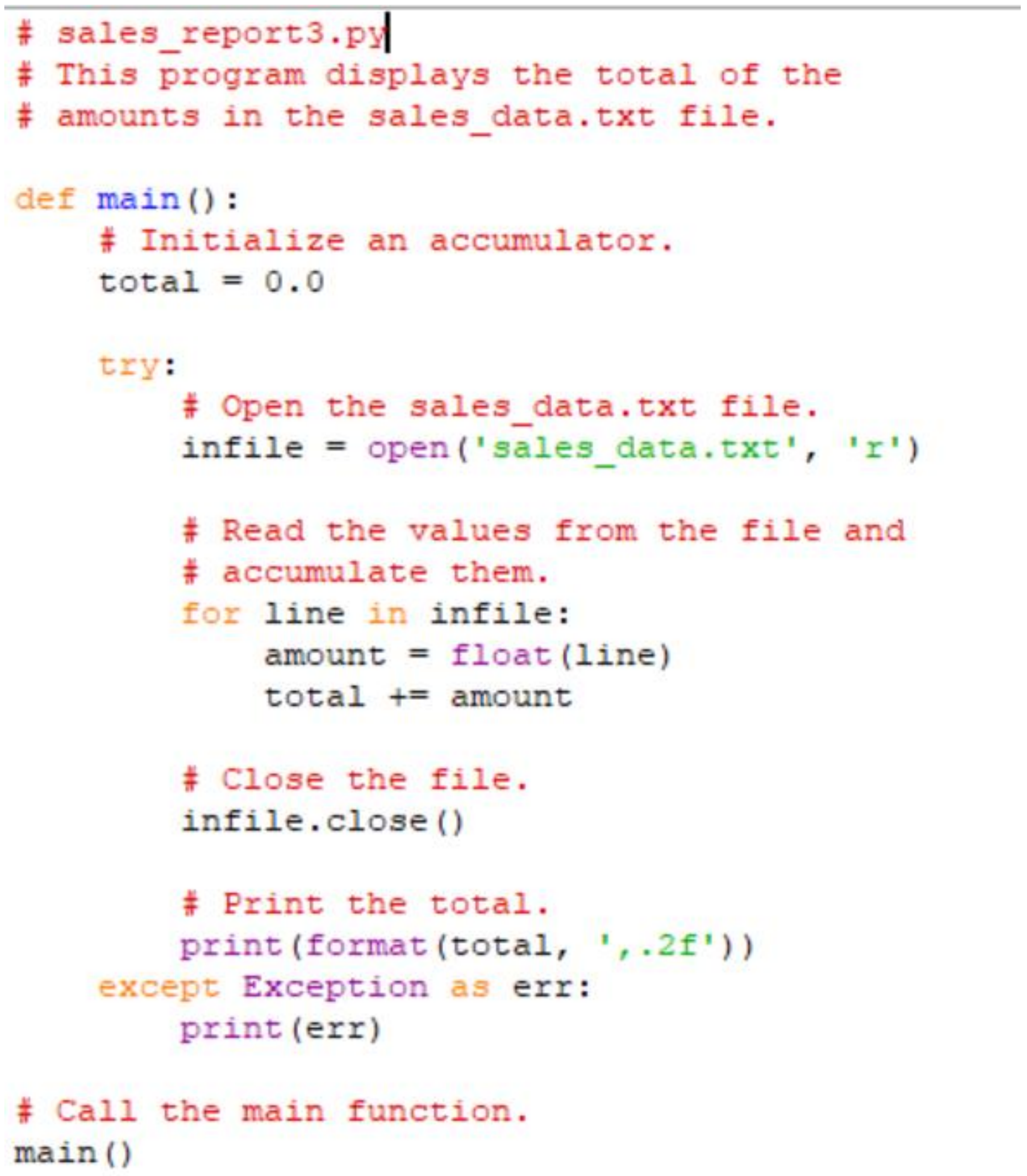


```
# sales_report2.py
# This program displays the total of the
# amounts in the sales_data.txt file.

def main():
    # Initialize an accumulator.
    total = 0.0

    try:
        # Open the sales_data.txt file.
        infile = open('sales_data.txt', 'r')

        # Read the values from the file and
        # accumulate them.
        for line in infile:
            amount = float(line)
            total += amount

        # Close the file.
        infile.close()

        # Print the total.
        print(format(total, ',.2f'))
    except:
        print('An error occurred.')

# Call the main function.
main()
```

Exception's Default Error Message

- **Exception object**: object created in memory when an exception is thrown
  - Usually contains default error message pertaining to the exception
  - Can assign the exception object to a variable in an `except` clause
    - Example: `except ValueError as err:`
  - Can pass exception object variable to `print` function to display the default error message

Default Error Message



```
# sales_report3.py
# This program displays the total of the
# amounts in the sales_data.txt file.

def main():
    # Initialize an accumulator.
    total = 0.0

    try:
        # Open the sales_data.txt file.
        infile = open('sales_data.txt', 'r')

        # Read the values from the file and
        # accumulate them.
        for line in infile:
            amount = float(line)
            total += amount

        # Close the file.
        infile.close()

        # Print the total.
        print(format(total, ',.2f'))
    except Exception as err:
        print(err)

# Call the main function.
main()
```

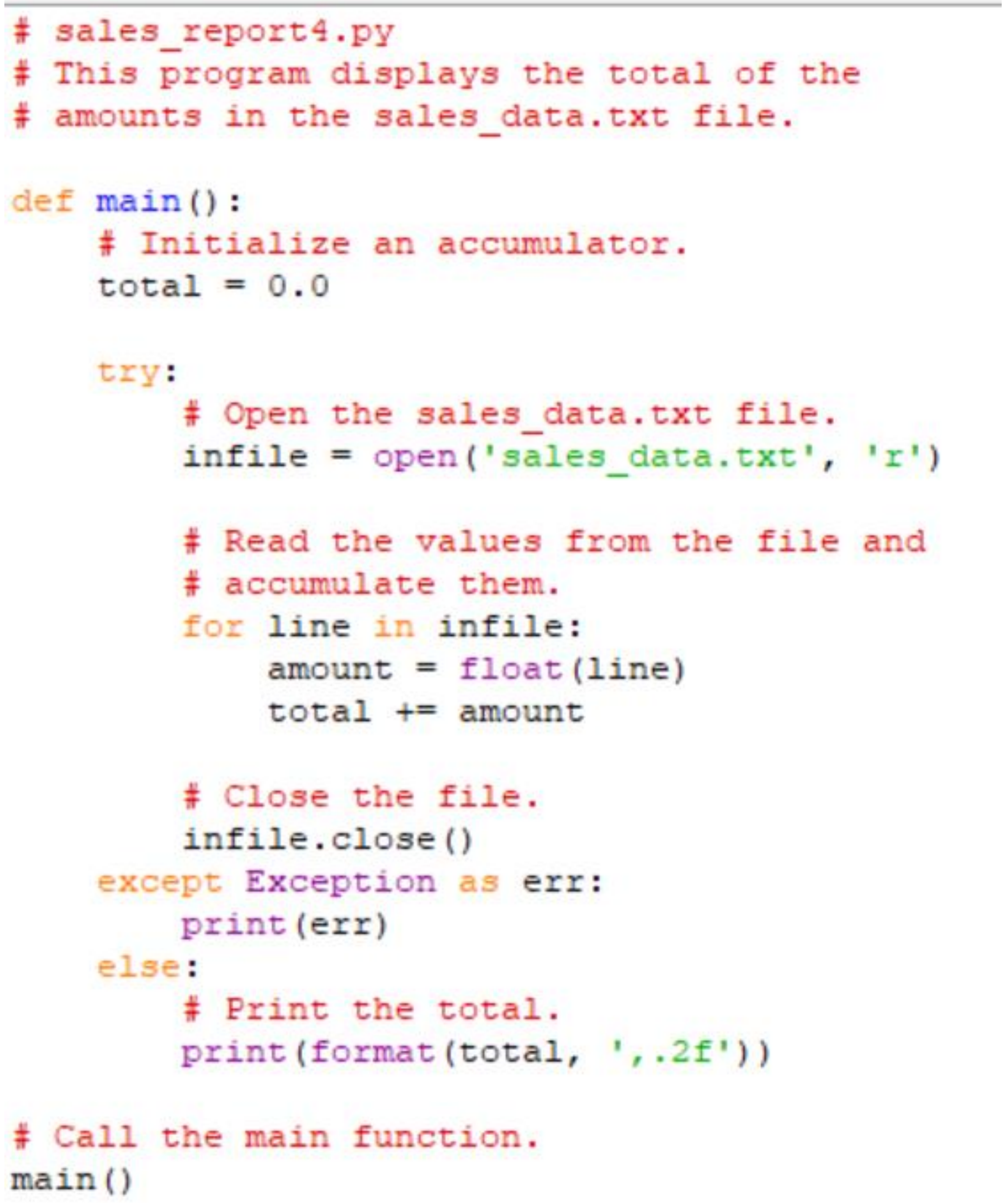
More Practice

- Practice with gross_pay1.py , gross_pay2.py and gross_pay3.py

The `else` Clause

- `try/except` statement may include an optional `else` clause, which appears after all the `except` clauses
  - Aligned with `try` and `except` clauses
  - Syntax similar to `else` clause in decision structure
  - Else block: block of statements executed after statements in try block, only if no exceptions were raised
    - If exception was raised, the else suite is skipped

else Clause

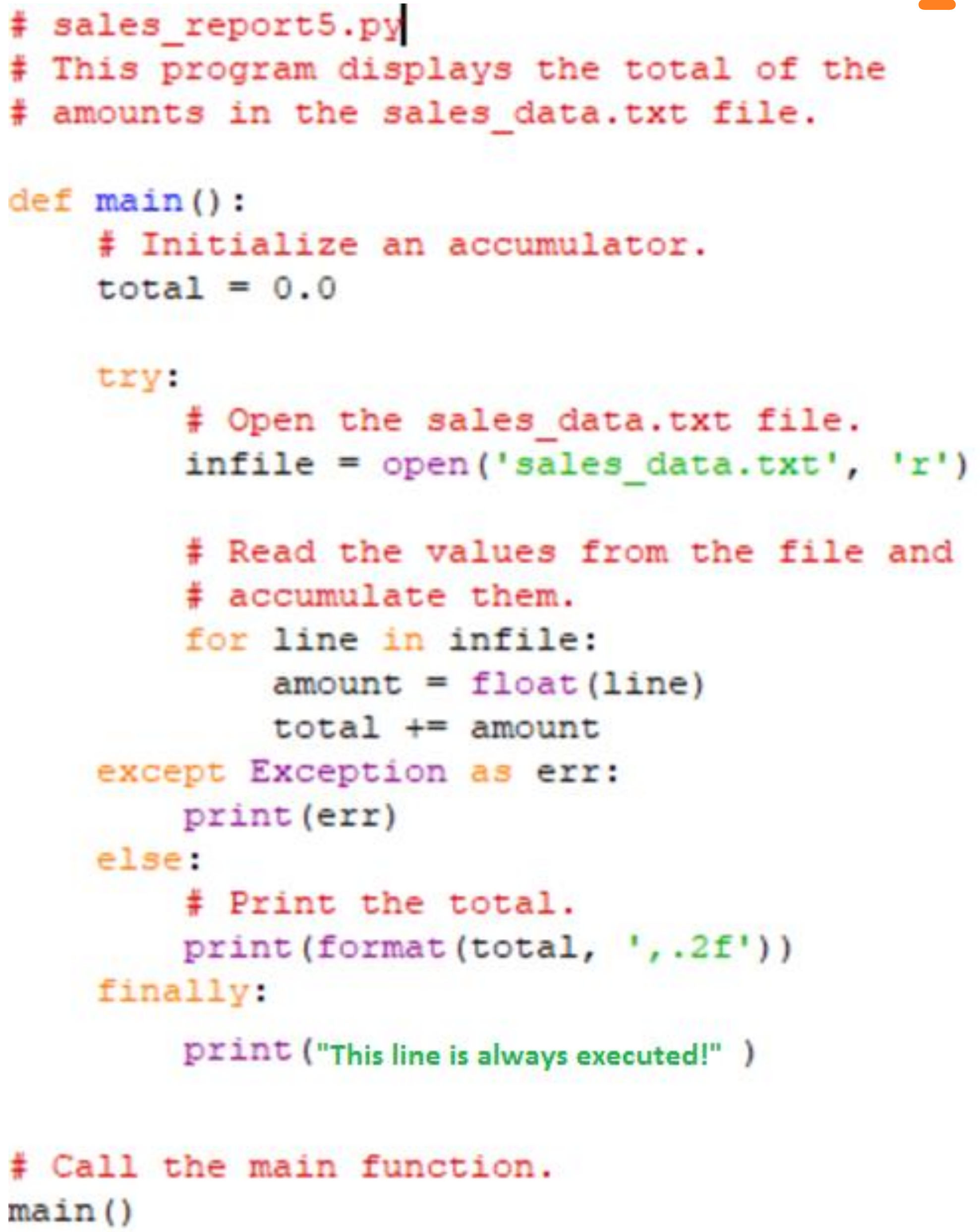


```
# sales_report4.py
# This program displays the total of the
# amounts in the sales_data.txt file.

def main():
    # Initialize an accumulator.
    total = 0.0

    try:
        # Open the sales_data.txt file.
        infile = open('sales_data.txt', 'r')

        # Read the values from the file and
        # accumulate them.
        for line in infile:
            amount = float(line)
            total += amount

        # Close the file.
        infile.close()
    except Exception as err:
        print(err)
    else:
        # Print the total.
        print(format(total, ',.2f'))

# Call the main function.
main()
```

The `finally` Clause

- `try/except` statement may include an optional `finally` clause, which appears after all the `except` clauses
  - Aligned with `try` and `except` clauses
  - General format: `finally:`
    ```
    finally:
        statements
    ```
  - Finally block: block of statements after the `finally` clause
    - Execute whether an exception occurs or not
    - Purpose is to perform cleanup before exiting

finally Clause



```python
# sales_report5.py
# This program displays the total of the
# amounts in the sales_data.txt file.

def main():
    # Initialize an accumulator.
    total = 0.0

    try:
        # Open the sales_data.txt file.
        infile = open('sales_data.txt', 'r')

        # Read the values from the file and
        # accumulate them.
        for line in infile:
            amount = float(line)
            total += amount
    except Exception as err:
        print(err)
    else:
        # Print the total.
        print(format(total, ',.2f'))
    finally:

        print ("This line is always executed!"  )


# Call the main function.
main()
```

What If an Exception Is Not Handled?

- Two ways for exception to go unhandled:
  - No except clause specifying exception of the right type
  - Exception raised outside a try block
- In both cases, exception will cause the program to halt

Summary

- Exceptions, including:
  - Traceback messages
  - Handling exceptions

More Practice

- Check out review questions in chapter 6 of the textbook including :
  - Multiple Choices,
  - True or False
  - Short Answer
  - Algorithm WorkBench
  - Programming Exercises (9-12)