Exceptions

Exceptions

- Exception: error that occurs while a program is running
  - Usually causes program to abruptly 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

Exceptions (cont'd.)

- Many exceptions can be prevented by careful coding
  - Example: input validation
  - Usually involve a simple decision construct
- 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

Exceptions (cont'd.)

- Exception handler: code that responds when exceptions are raised and prevents program from crashing
  - In Python, written as `try/except` statement
    - General format:

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

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

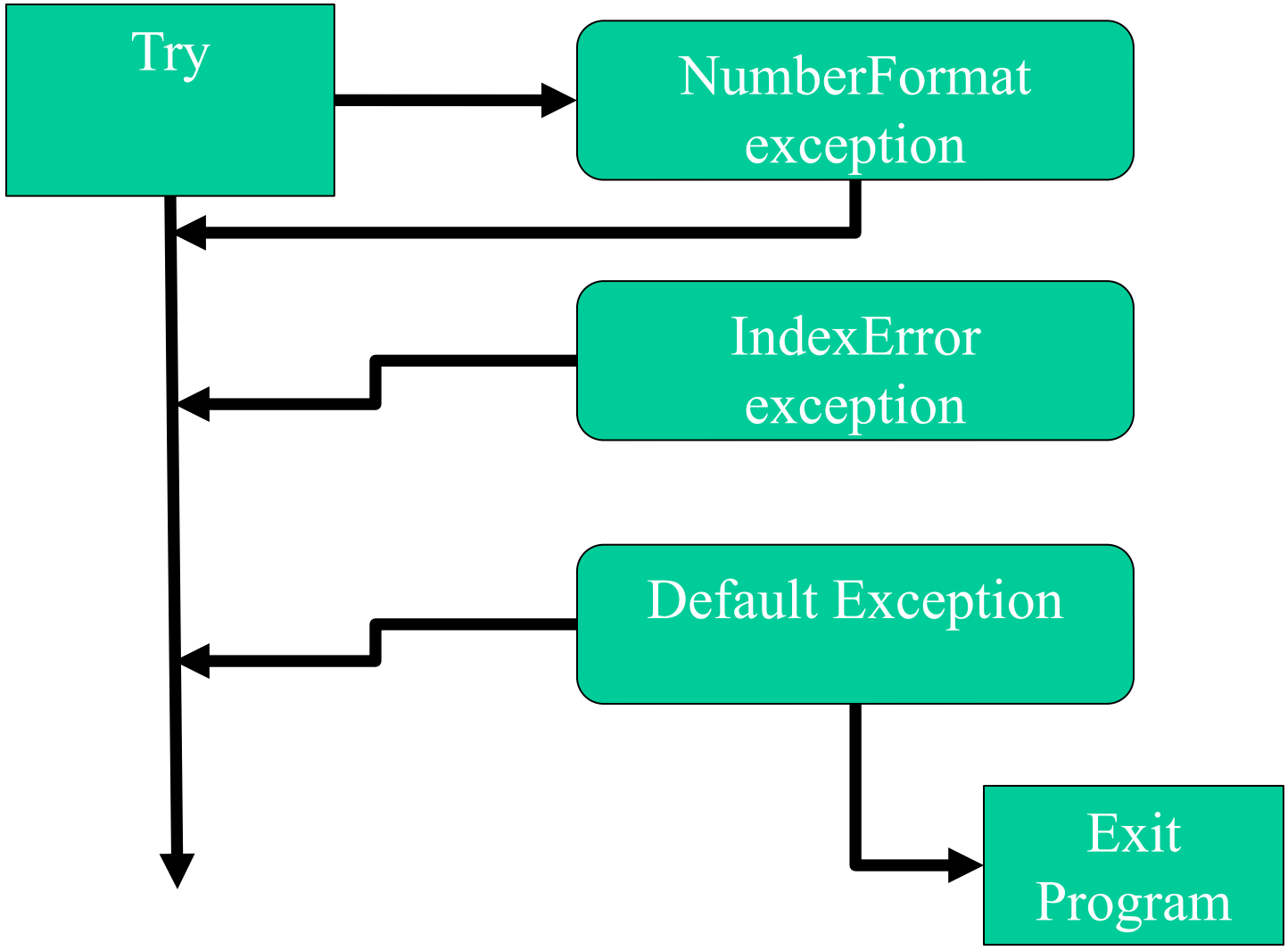
Exceptions (cont'd.)

- If statement in try suite 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 suite 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 suite
  - Should always be last in a series of `except` clauses

Exception Flow Chart



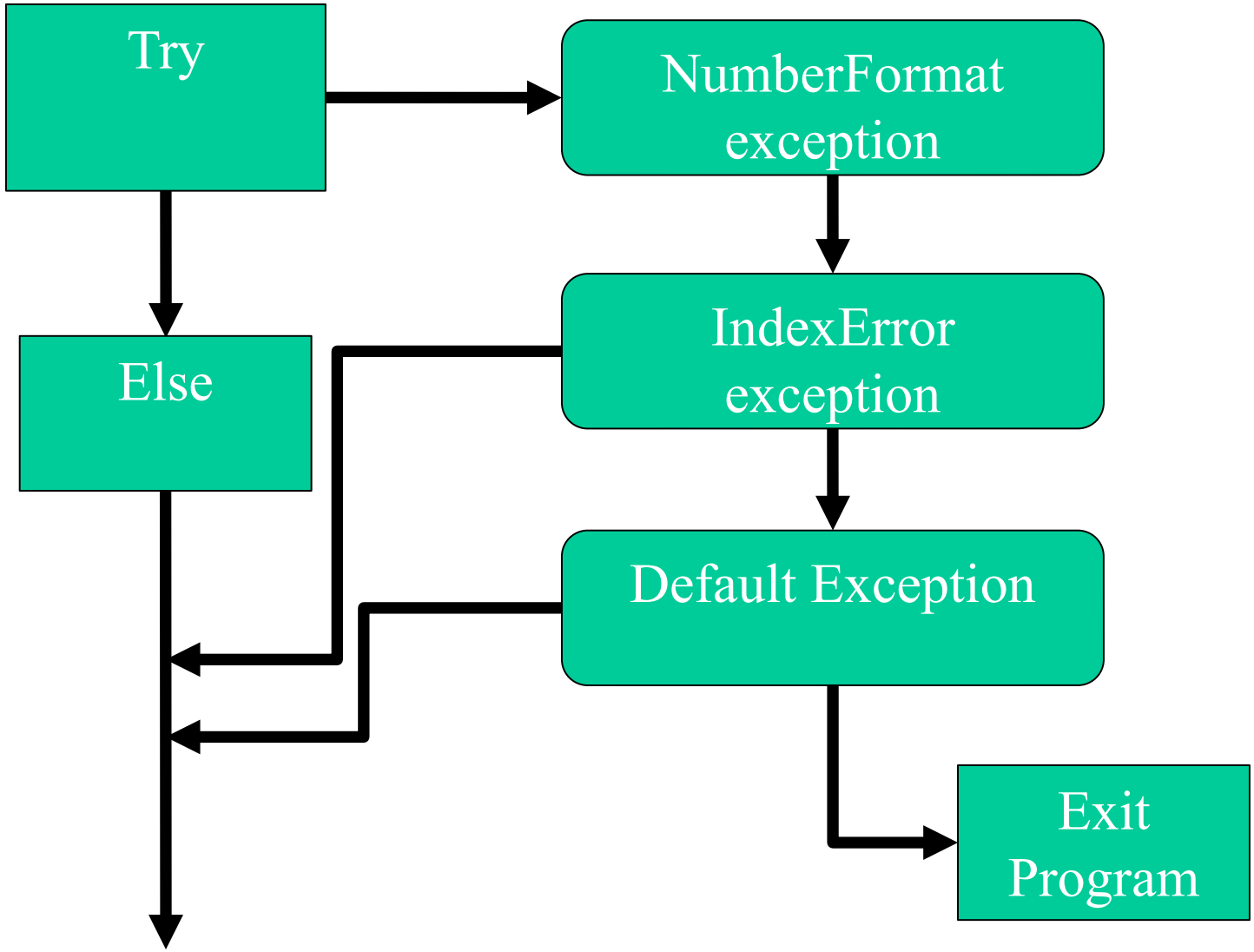
Displaying an Exception's Default Error Message

- 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

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 suite: block of statements executed after statements in try suite, only if no exceptions were raised
    - If exception was raised, the else suite is skipped

Exception Flow Chart



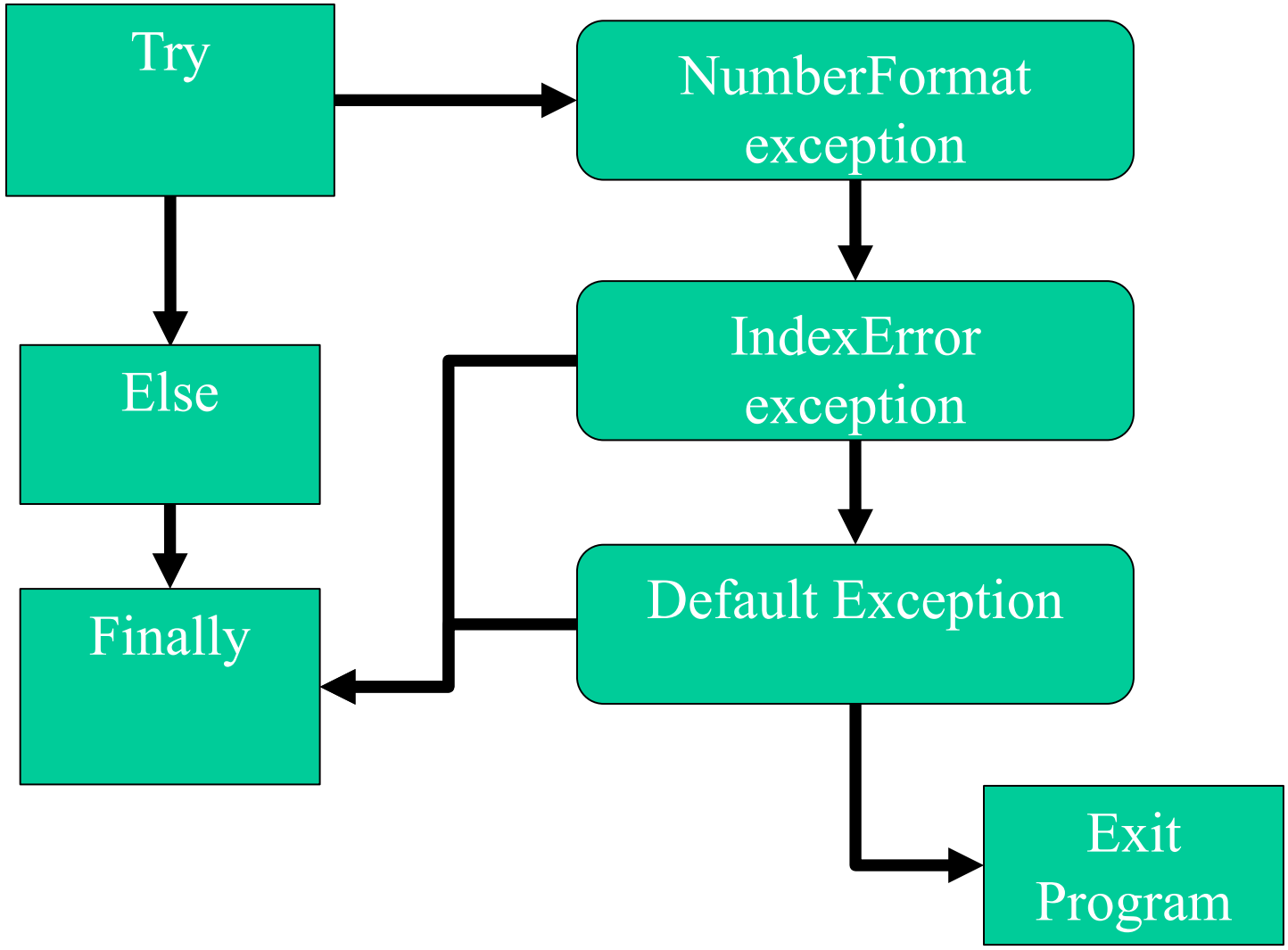
The `finally` Clause

- `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:
        statements
```

- Finally suite: block of statements after the `finally` clause
  - Execute whether an exception occurs or not
  - Purpose is to perform cleanup before exiting

Exception Flow Chart



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 suite
- In both cases, exception will cause the program to halt
  - Python documentation provides information about exceptions that can be raised by different functions

All Python Exceptions

- AssertionError
- AttributeError
- EOFError
- FloatingPointError
- GeneratorExit
- ImportError
- IndexError
- KeyError
- KeyboardInterrupt
- MemoryError
- NameError
- NotImplementedError
- OSError
- OverflowError
- ReferenceError
- RuntimeError
- StopIteration
- SyntaxError
- IndentationError
- TabError
- SystemError
- SystemExit
- TypeError
- UnboundLocalError
- UnicodeError
- UnicodeEncodeError
- UnicodeDecodeError
- UnicodeTranslateError
- ValueError
- ZeroDivisionError

Custom Exceptions

- If that isn't enough, you can create your own custom exceptions
- Just define a class that inherits from `Exception`

```
class YourCustomError(Exception):
    pass
```

And later:

```
raise YourCustomError
```

Input Validation

- Yes…we *finally* got here….

- The same old story

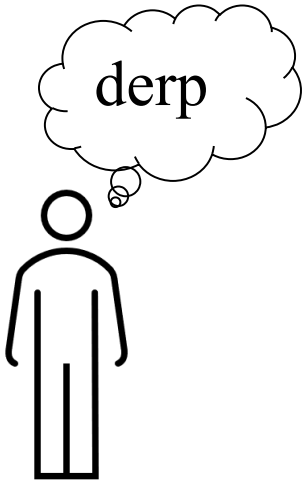
```
Enter a number:
```

Input Validation

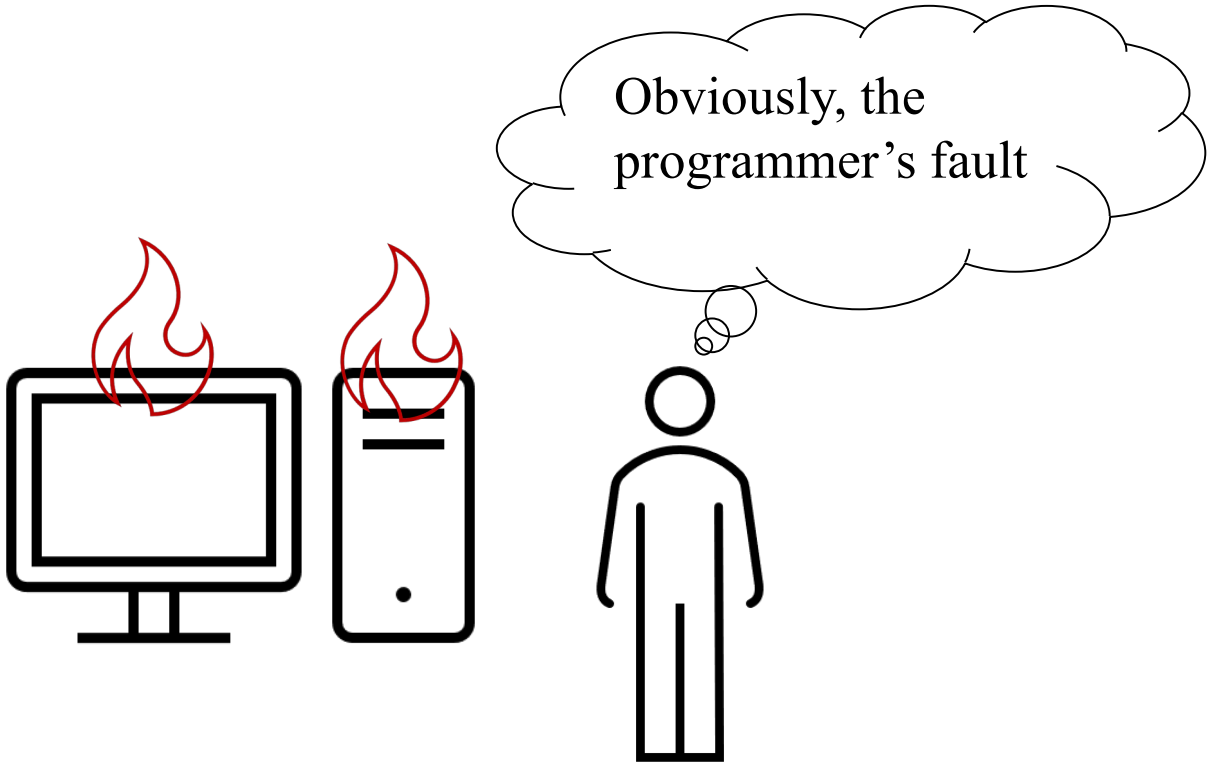
- Yes…we *finally* got here….

- The same old story

```
Enter a number: iquwienbe
```



And the Result



The Solution?

- Basically do what we've already been doing.

- Write a program that accepts only numbers as valid input. The user is prompted to try again if an invalid input is given

Python Has Helpers

- capitalize(): Converts the first character to upper case
- casefold(): Converts string into lower case
- center(): Returns a centered string
- count(): Returns the number of times a specified value occurs in a string
- encode(): Returns an encoded version of the string
- endswith(): Returns true if the string ends with the specified value
- expandtabs(): Sets the tab size of the string
- find(): Searches the string for a specified value and returns the position of where it was found
- format(): Formats specified values in a string
- format_map(): Formats specified values in a string
- index(): Searches the string for a specified value and returns the position of where it was found
- isalnum(): Returns True if all characters in the string are alphanumeric
- isalpha(): Returns True if all characters in the string are in the alphabet
- isascii(): Returns True if all characters in the string are ascii characters
- isdecimal(): Returns True if all characters in the string are decimals
- isdigit(): Returns True if all characters in the string are digits
- isidentifier(): Returns True if the string is an identifier
- islower(): Returns True if all characters in the string are lower case
- isnumeric(): Returns True if all characters in the string are numeric
- isprintable(): Returns True if all characters in the string are printable
- isspace(): Returns True if all characters in the string are whitespaces
- istitle(): Returns True if the string follows the rules of a title
- isupper(): Returns True if all characters in the string are upper case
- join(): Converts the elements of an iterable into a string
- ljust(): Returns a left justified version of the string
- lower(): Converts a string into lower case
- lstrip(): Returns a left trim version of the string
- maketrans(): Returns a translation table to be used in translations
- partition(): Returns a tuple where the string is parted into three parts
- replace(): Returns a string where a specified value is replaced with a specified value
- rfind(): Searches the string for a specified value and returns the last position of where it was found
- rindex(): Searches the string for a specified value and returns the last position of where it was found
- rjust(): Returns a right justified version of the string
- rpartition(): Returns a tuple where the string is parted into three parts
- rsplit(): Splits the string at the specified separator, and returns a list
- rstrip(): Returns a right trim version of the string
- split(): Splits the string at the specified separator, and returns a list
- splitlines(): Splits the string at line breaks and returns a list
- startswith(): Returns true if the string starts with the specified value
- strip(): Returns a trimmed version of the string
- swapcase(): Swaps cases, lower case becomes upper case and vice versa
- title(): Converts the first character of each word to upper case
- translate(): Returns a translated string
- upper(): Converts a string into upper case
- zfill(): Fills the string with a specified number of 0 values at the beginning

So: Two Options

- Option 1:
  - Use the build in helpers
    - isdecimal(): Returns True if all characters in the string are decimals
    - isdigit(): Returns True if all characters in the string are digits

- Option 2
  - Write our own helpers
    - Good for languages that might not have them already

Exercises 9A

- Write a method that accepts a string and returns true only if the string can be converted to a float.

- Write a method that accepts a string and returns true only if the string can be converted to an int

Exercises 9B

- You're going to write an interactive calculator! User input is assumed to be a formula that consist of a number, an operator (+, -, *, /), and another number, separated by white space
  - e.g. 1 + 1

- If the input does not consist of 3 elements, raise a FormulaError, which is a custom Exception.
- Try to convert the first and third input to a float. Report any invalid inputs and what they were.
- If the second input is not +, -, *, /, again raise a FormulaError
- If the input is valid, perform the calculation and print out the result.