10 Exceptions

Key terms: `try except finally raise`

Reading: Aaron Maxwell's "Exceptional Logging of Exceptions in Python"[13]

Exercise: Write a program that raises a `ValueError` if the first command line argument cannot be represented as a floating point number.

Syntax errors make a program invalid and unrunnable, while runtime errors occur during execution. Only the latter can be caught, or excepted. The errors themselves are objects descended from class `Exception`. You can expect to run into the `IndexError` (no such value exists in a collection), the `TypeError` (an object is of the wrong class), and the `ValueError` (a value is unacceptable). These are defined in the "Built-in Exceptions" chapter of the official documentation.[14].

To except an error is to branch execution upon some kind of failure, which may only be as severe as an invalid input or the end of data. This is analogous to conditional branching but triggered by the exception mechanism. In this model, we `try` what we want to do, and `except` what to do otherwise. The blocks are set off with colons like functions or loops. The exception block should be written as tightly as possible: not doing much more than taking the questionable step, and catching only a specific error.

```
# Demonstrate that ints and strings cannot be added:

print(22+33)
print("Two"+"Words")

try:
 print(4+"@@@")
except TypeError:
 print ("Addition does not support these types.")
```

The `raise` keyword causes, or throws, an exception. If those already defined don't match your use case well, a new class can be easily defined and raised.

```
# Validate that all arguments passed were integers.
import sys

found = list()
for argument in sys.argv[1:]:
 try:
  found.append(int(argument))
 except ValueError:
  pass

# Propagate an error if there were no integers at all.
if found:
 pass
else:
 raise TypeError('None of the arguments were integers.')
```

Catching `Exception` itself is a terrible idea, as discussed in Maxwell's post, because we won't know what happened. We should catch only the specific exception we know how to handle.

Next, we will address file input and output.

[13] `https://www.loggly.com/blog/exceptional-logging-of-exceptions-in-python/`
[14] `https://docs.python.org/3/library/exceptions.html`