CS61A Lecture 32
Amir Kamil
UC Berkeley
April 5, 2013

Announcements

- Hog revisions due Monday
- HW10 due Wednesday
- Make sure to fill out survey on Piazza
  - We need to schedule alternate final exam times for those who have a conflict, so if you do, let us know on the survey when you are available

The Begin Special Form

Begin expressions allow sequencing

\[
\text{(begin } \exp_1 \exp_2 \ldots \exp_n)\]

\[
\text{(define \{repeat } \exp k \text{ fn)}
\]

\[
\text{if } (> k 0)
\]

\[
\text{(begin fn) (repeat } (- k 1) \text{ fn)}
\]

\[
\text{‘done)\}
\]

\[
\text{(define \{tri } \exp \text{ fn)}
\]

\[
\text{repeat 3 (lambda } () \exp (\lt 120))))
\]

\[
\text{(define \{sier } \exp \text{ d k)}
\]

\[
\text{tri (lambda } () \exp (k 1) \exp d \exp (leg d k))))
\]

\[
\text{(define \{leg } \exp \text{ d k)}
\]

\[
\text{sier (/ d 2) (- k 1)) (penup) \exp d \exp (pendown)\}
\]

Handling Errors (Back to Python)

Sometimes, computers don't do exactly what we expect
- A function receives unexpected argument types
- Some resource (such as a file) is not available
- A network connection is lost

Exceptions

A built-in mechanism in a programming language to declare and respond to exceptional conditions

Python raises an exception whenever an error occurs

Exceptions can be handled by the program, preventing a crash

Unhandled exceptions will cause Python to halt execution

Mastering exceptions:

Exceptions are objects! They have classes with constructors

They enable non-local continuations of control:

If \( x \) calls \( g \) and \( g \) calls \( h \), exceptions can shift control from \( h \) to \( x \)

without waiting for \( g \) to return

However, exception handling tends to be slow

Assert Statements

Assert statements raise an exception of type \texttt{AssertionError}

\[
\text{assert } \exp, \text{ <string>}
\]

Assertions are designed to be used liberally and then disabled in production systems

\[
\text{python3 -O}
\]

"O" stands for optimized. Among other things, it disables assertions

Whether assertions are enabled is governed by the built-in bool \texttt{__debug__}
Raise Statements

Exceptions are raised with a `raise statement`

```
raise <expression>
```

`<expression>` must evaluate to an exception instance or class.

Exceptions are constructed like any other object; they are just instances of classes that inherit from `BaseException`

- `TypeError` -- A function was passed the wrong number/type of argument
- `NameError` -- A name wasn't found
- `KeyError` -- A key wasn't found in a dictionary
- `RuntimeError` -- Catch-all for troubles during interpretation

Try Statements

Try statements handle exceptions

```
try:
    <try suite>
except <exception class> as <name>:
    <except suite>
    ...
```

Execution rule:
- The `<try suite>` is executed first;
- If, during the course of executing the `<try suite>`, an exception is raised that is not handled otherwise, and
- If the class of the exception inherits from `<exception class>`, then
- The `<except suite>` is executed, with `<name>` bound to the exception

Handling Exceptions

Exception handling can prevent a program from terminating

```python
>>> try:
    x = 1/0
    except ZeroDivisionError as e:
        print('handling a', type(e))
    x = 0
    handling a <class 'ZeroDivisionError'>
>>> x
0
```

**Multiple try statements:** Control jumps to the except suite of the most recent try statement that handles that type of exception.

WWPD: What Would Python Do?

How will the Python interpreter respond?

```python
def invert(x):
    result = 1/x  # Raises a ZeroDivisionError if x is 0
    print('Never printed if x is 0')
    return result

def invert_safe(x):
    try:
        return invert(x)
    except ZeroDivisionError as e:
        return str(e)

>>> invert_safe(1/0)
>>> try:
    invert_safe(0)
    except BaseException:
        print('Handled!')
>>> invert_safe(1/0)
```

```python
>>> invert_safe(1/0)
>>> try:
    invert_safe(0)
    except BaseException:
        print('Handled!')
>>> invert_safe(1/0)
```