Exception handling

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 1

When things go wrong

- Our code could generate many types of errors
  - `y = x/z`, but `z` has value `0`
  - `y = int(s)`, but string `s` does not represent a valid integer
  - `y = 5*x`, but `x` does not have a value
  - `y = l[i]`, but `i` is not a valid index for list `l`
  - Try to read from a file, but the file does not exist
  - Try to write to a file, but the disk is full

When things go wrong

- Our code could generate many types of errors
  - `y = x/z`, but `z` has value `0`
  - `y = int(s)`, but string `s` does not represent a valid integer
  - `y = 5*x`, but `x` does not have a value
  - `y = l[i]`, but `i` is not a valid index for list `l`
  - Try to read from a file, but the file does not exist
  - Try to write to a file, but the disk is full
- Recovering gracefully
  - Try to anticipate errors
  - Provide a contingency plan
  - Exception handling

Types of errors

- Python flags the type of each error

Types of errors

- Python flags the type of each error
- Most common error is a syntax error
  - `SyntaxError: invalid syntax`
  - Not much you can do!

Types of errors

- Python flags the type of each error
- Most common error is a syntax error
  - `SyntaxError: invalid syntax`
  - Not much you can do!
- We are interested in errors when the code is running
  - Name used before value is defined
    `NameError: name 'x' is not defined`
  - Division by zero in arithmetic expression
    `ZeroDivisionError: division by zero`
  - Invalid list index
    `IndexError: list assignment index out of range`

Terminology

- Raise an exception
  - Run time error $\rightarrow$ signal error type, with diagnostic information
    `NameError: name 'x' is not defined`
- Handle an exception
  - Anticipate and take corrective action based on error type
- Unhandled exception aborts execution

Terminology

- Raise an exception
  - Run time error → signal error type, with diagnostic information `NameError: name 'x' is not defined`
- Handle an exception
  - Anticipate and take corrective action based on error type
- Unhandled exception aborts execution

Handling exceptions

```
try:
    ...  ← Code where error may occur
    ...
except IndexError:
    ...  ← Handle IndexError
except (NameError,KeyError):
    ...  ← Handle multiple exception types
except:
    ...  ← Handle all other exceptions
else:
    ...  ← Execute if try runs without errors
```

Using exceptions "positively"

- Collect scores in dictionary

```
scores = {"Shefali":[3,22],
          "Harmanpreet":[200,3]}
```

- Update the dictionary
- Batter `b` already exists, append to list

```
scores[b].append(s)
```

- New batter, create a fresh entry

```
scores[b] = [s]
```

Using exceptions "positively"

- Collect scores in dictionary

```
scores = {"Shefali":[3,22],
          "Harmanpreet":[200,3]}
```

- Update the dictionary
- Batter `b` already exists, append to list

```
scores[b].append(s)
```

- New batter, create a fresh entry

```
scores[b] = [s]
```

Traditional approach

```
if b in scores.keys():
  scores[b].append(s)
else:
  scores[b] = [s]
```

Using exceptions "positively"

- Collect scores in dictionary

```
scores = {"Shefali":[3,22],
          "Harmanpreet":[200,3]}
```

- Update the dictionary
- Batter `b` already exists, append to list

```
scores[b].append(s)
```

- New batter, create a fresh entry

```
scores[b] = [s]
```

Traditional approach

```
if b in scores.keys():
  scores[b].append(s)
else:
  scores[b] = [s]
```

Using exceptions

```
try:
  scores[b].append(s)
except KeyError:
  scores[b] = [s]
```

Flow of control

```
...
x = f(y,z)
```

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

```
def g(m):
    ...
    h(m)
```

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

```
def g(m):
    ...
    h(m)
```

```
def h(s):
    ...
    h(s)
```

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

```
def g(m):
    ...
    h(m)
```

```
def h(s):
    ...
    h(s)
```

`IndexError` not handled in `h()`

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

```
def g(m):
    ...
    h(m)
```

`IndexError` inherited from `h()`

```
def h(s):
    ...
    h(s)
```

`IndexError` not handled in `h()`

Flow of control

```
...
x = f(y,z)
```

```
def f(a,b):
    ...
    g(a)
```

`IndexError`
inherited from `g()`

```
def g(m):
    ...
    h(m)
```

`IndexError`
inherited from `h()`
Not handled?

```
def h(s):
    ...
    h(s)
```

`IndexError` not handled in `h()`

Flow of control

```
...
x = f(y,z)
```

`IndexError`
inherited from `f()`

```
def f(a,b):
  ...
  g(a)
```

`IndexError`
inherited from `g()`
Not handled?

```
def g(m):
  ...
  h(m)
```

`IndexError`
inherited from `h()`
Not handled?

```
def h(s):
  ...
  h(s)
```

`IndexError` not handled in `h()`

Flow of control

```
...
x = f(y,z)
```

`IndexError`
inherited from `f()`
Not handled?
Abort!

```
def f(a,b):
  ...
  g(a)
```

`IndexError`
inherited from `g()`
Not handled?

```
def g(m):
  ...
  h(m)
```

`IndexError`
inherited from `h()`
Not handled?

```
def h(s):
  ...
  h(s)
```

`IndexError` not handled in `h()`