#### NPTEL MOOC

# PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 5, Lecture 1

Madhavan Mukund, Chennai Mathematical Institute http://www.cmi.ac.in/~madhavan

# When things go wrong

- \* y = x/z, but z has value 0
- \* y = int(s), but string s is not 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 ...

- \* Some errors can be anticipated
- \* Others are unexpected
- \* Predictable error exception
  - \* Normal situation vs exceptional situation
- Contingency plan exception handling

### Exception handling

- \* If something goes wrong, provide "corrective action"
  - \* File not found display a message and ask user to retype filename
  - List index out of bounds provide diagnostic information to help debug error
- \* Need mechanism to internally trap exceptions
- \* An untapped exception will abort the program

### Types of errors

- \* Python notifies you of different types of errors
- \* Most common error, invalid Python code

SyntaxError: invalid syntax

- \* Not much you can do with this!
- \* We are interested in errors that occur when code is being executed

# Types of errors

Some errors while code is executing (run-time errors)

- \* 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 → 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:
  . . . ← What to do if IndexError occurs
except (NameError, KeyError):
 · · · Common code to handle multiple errors
except:
 . . . ← Catch all other exceptions
else:
```

### "Positive" use of exceptions

\* Add a new entry to this dictionary

```
scores = {'Dhawan':[3,22],'Kohli':[200,3]}
```

\* Batsman b already exists, append to list

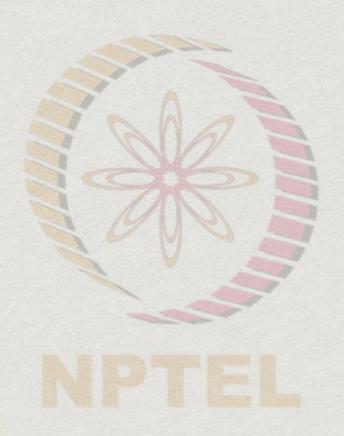
```
scores[b].append(s)
```

\* New batsman, create fresh entry

$$scores[b] = [s]$$

# "Positive" use of exceptions

$$x = f(y,z)$$



x = f(y,z) def f(a,b): g(a) NPTEL

```
x = f(y,z)
def f(a,b):
g(a)
def g(m):
h(m)
```

```
x = f(y,z)
            def f(a,b):
              g(a)
                         def g(m):
                           h(m)
                                   def h(s):
```

```
x = f(y,z)
             def f(a,b):
               g(a)
                          def g(m):
                            h(m)
                                    def h(s):
       IndexError, not handled in h()→
```

```
x = f(y,z)
               def f(a,b):
                  g(a)
                            def g(m):
IndexError inherited from h()→ h(m)
                                      def h(s):
         IndexError, not handled in h()→
```

```
x = f(y,z)
               def f(a,b):
                 g(a) ←IndexError inherited from g()
                            def g(m):
IndexError inherited from h()→ h(m)
                                      def h(s):
Not handled?
```

IndexError, not handled in h()→

```
x = f(y,z)
                def f(a,b):
IndexError
inherited
                   g(a) ←IndexError inherited from g()
from f()
                                            Not handled?
                              def g(m):
IndexError inherited from h() \rightarrow h(m)
                                         def h(s):
Not handled?
          IndexError, not handled in h()→
```

```
x = f(y,z)
                def f(a,b):
IndexError
inherited
                   g(a) ←IndexError inherited from g()
from f()
                                            Not handled?
Not handled?
                              def g(m):
Abort!
IndexError inherited from h() \rightarrow h(m)
                                          def h(s):
Not handled?
```

IndexError, not handled in h()→

# Summary

- \* Exception handling allows us to gracefully deal with run time errors
- \* Can check type of error and take appropriate action based on type
- \* Can change coding style to exploit exception handling
- \* When dealing with files and input/output, exception handling becomes very important