withrestart

structured error recovery
using named restart points

Ryan Kelly
ryan@rfk.id.au

Common Lisp

Common Lisp

exceptions

Common Lisp

~~exceptions~~

conditions + handlers + restarts

```
def parse_file(fname):
    return open(fname).read()
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
>>>
```

```
>>> os.listdir("test")
['3.txt', '1.txt', '2.txt']
>>>
```

```
>>> os.listdir("test")
['3.txt', '1.txt', '2.txt']
>>>
>>>
>>> open("test/1.txt").read()
'one'
>>>
```

```
>>> os.listdir("test")
['3.txt', '1.txt', '2.txt']
>>>
>>>
>>> open("test/1.txt").read()
'one'
>>>
>>>
>>> summarise_dir("test")
'one two three'
>>>
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    # what if a file gets deleted after calling listdir()?
    return summarise_files(fs)
```

```
>>> summarise_dir("test")
>>> summarise_dir("test")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "examples.py", line 26, in summarise_dir
    return summarise_files(sorted(fs))
  File "examples.py", line 13, in summarise_files
    results.append(parse_file(fname))
  File "examples.py", line 6, in parse_file
    return open(fname).read()
IOError: [Errno 2] No such file or directory: 'test/2.txt'
>>>
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    # Nothing sensible we can do here
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        try:
            results.append(parse_file(fname))
        except IOError:
            pass
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        try:
            results.append(parse_file(fname))
        except IOError:
            results.append(“MISSING”)
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        try:
            results.append(parse_file(fname))
        except IOError:
            # Can fix it, but what to do?
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    try:
        return summarise_files(fs)
    except IOError:
        return summarise_dir(dname)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    try:
        return summarise_files(fs)
    except IOError, e:
        with open(e.filename,”w”) as f:
            f.write(“MISSING”)
        return summarise_dir(dname)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    try:
        return summarise_files(fs)
    except IOError, e:
        # Can fix it, but it throws away all that work!
```

```
summarise_files: best place for error recovery
                 mechanics

summarise_dirs:  best place for error recovery
                 policy
```

summarise_files: best place for error recovery
mechanics

summarise_dirs: best place for error recovery
policy

Need ***cooperation*** between these functions

“Beyond Exception Handling”

“Beyond Exception Handling”

Throwing an exception is always ***fatal***

“Beyond Exception Handling”

Throwing an exception is always ***fatal***

Handling an exception is always ***solitary***

“Beyond Exception Handling”

Throwing an exception is always ***fatal***

Handling an exception is always ***solitary***

We should be able to fix the error
then continue with what we were doing

```
“Restart”:   a checkpoint for resuming execution
             after the occurrence of an error

“Handler”:   like an except clause, but executed
             before unwinding the stack
```

“Restart”: a checkpoint for resuming execution after the occurrence of an error

“Handler”: like an except clause, but executed *before* unwinding the stack

Handlers can invoke a restart if they are able to correct the error

```
from withrestart import *
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        results.append(parse_file(fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts() as invoke:
            results.append(invoke(parse_file,fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip) as invoke:
            results.append(invoke(parse_file,fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value) as invoke:
            results.append(invoke(parse_file,fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value,retry) as invoke:
            results.append(invoke(parse_file,fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
>>> summarise_dir("test")
>>> summarise_dir("test")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "examples.py", line 25, in summarise_dir
    return summarise_files(sorted(fs))
  File "examples.py", line 12, in summarise_files
    results.append(invoke(parse_file,fname))
  File "/.../withrestart/__init__.py", line 433, in __call__
    raise exc_type, exc_value, traceback
IOError: [Errno 2] No such file or directory: 'test/2.txt'
>>>
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value,retry) as invoke:
            results.append(invoke(parse_file,fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    return summarise_files(fs)
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value,retry) as invoke:
            results.append(invoke(parse_file,fname))
    return " ".join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    with Handler(IOError,"skip"):
        return summarise_files(fs)
```

```
>>> summarise_dir("test")
>>> summarise_dir("test")
'one three'
>>>
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value,retry) as invoke:
            results.append(invoke(parse_file,fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    with Handler(IOError,"use_value",”MISSING”):
        return summarise_files(fs)
```

```
>>> summarise_dir("test")
>>> summarise_dir("test")
'one MISSING three'
>>>
```

```
def parse_file(fname):
    return open(fname).read()


def summarise_files(fnames):
    results = []
    for fname in sorted(fnames):
        with restarts(skip,use_value,retry) as invoke:
            results.append(invoke(parse_file,fname))
    return “ “.join(results)


def summarise_dir(dname):
    fs = [os.path.join(dname,f) for f in os.listdir(dname)]
    with handlers() as h:
        @h.add_handler
        def handle_IOError(err):
            with open(err.filename,"w") as f:
                f.write("RECREATED")
            raise InvokeRestart("retry")
        return summarise_files(sorted(fs))
```

```
>>> summarise_dir("test")
>>> summarise_dir("test")
'one RECREATED three'
>>>
```

```
>>> summarise_dir("test")
'one RECREATED three'
>>>
>>>
>>> os.listdir("test")
['3.txt', '1.txt', '2.txt']
>>>
```

Trimming Boilerplate

Trimming Boilerplate

```
explicit error codes  =>  try-except
```

Trimming Boilerplate

explicit error codes => try-except

explicit error callbacks => withrestart

```
http://github.com/rfk/withrestart

pip install withrestart
```

http://github.com/rfk/withrestart

```
pip install withrestart
```

* pure-python implementation

```
http://github.com/rfk/withrestart

pip install withrestart
```

```
* pure-python implementation
  (but uses sys._getframe)
```

```
http://github.com/rfk/withrestart

pip install withrestart


* pure-python implementation
   (but uses sys._getframe)
* ~20 times slower than try-except
```

```
http://github.com/rfk/withrestart

pip install withrestart
```

```
* pure-python implementation
   (but uses sys._getframe)
* ~20 times slower than try-except
   (working on it, promise)
```

http://github.com/rfk/withrestart

```
pip install withrestart
```

```
* pure-python implementation
   (but uses sys._getframe)
* ~20 times slower than try-except
   (working on it, promise)
* no bytecode hackery
```

```
http://github.com/rfk/withrestart

pip install withrestart


* pure-python implementation
   (but uses sys._getframe)
* ~20 times slower than try-except
   (working on it, promise)
* no bytecode hackery
   (...yet:  http://github.com/rfk/withhacks)
```