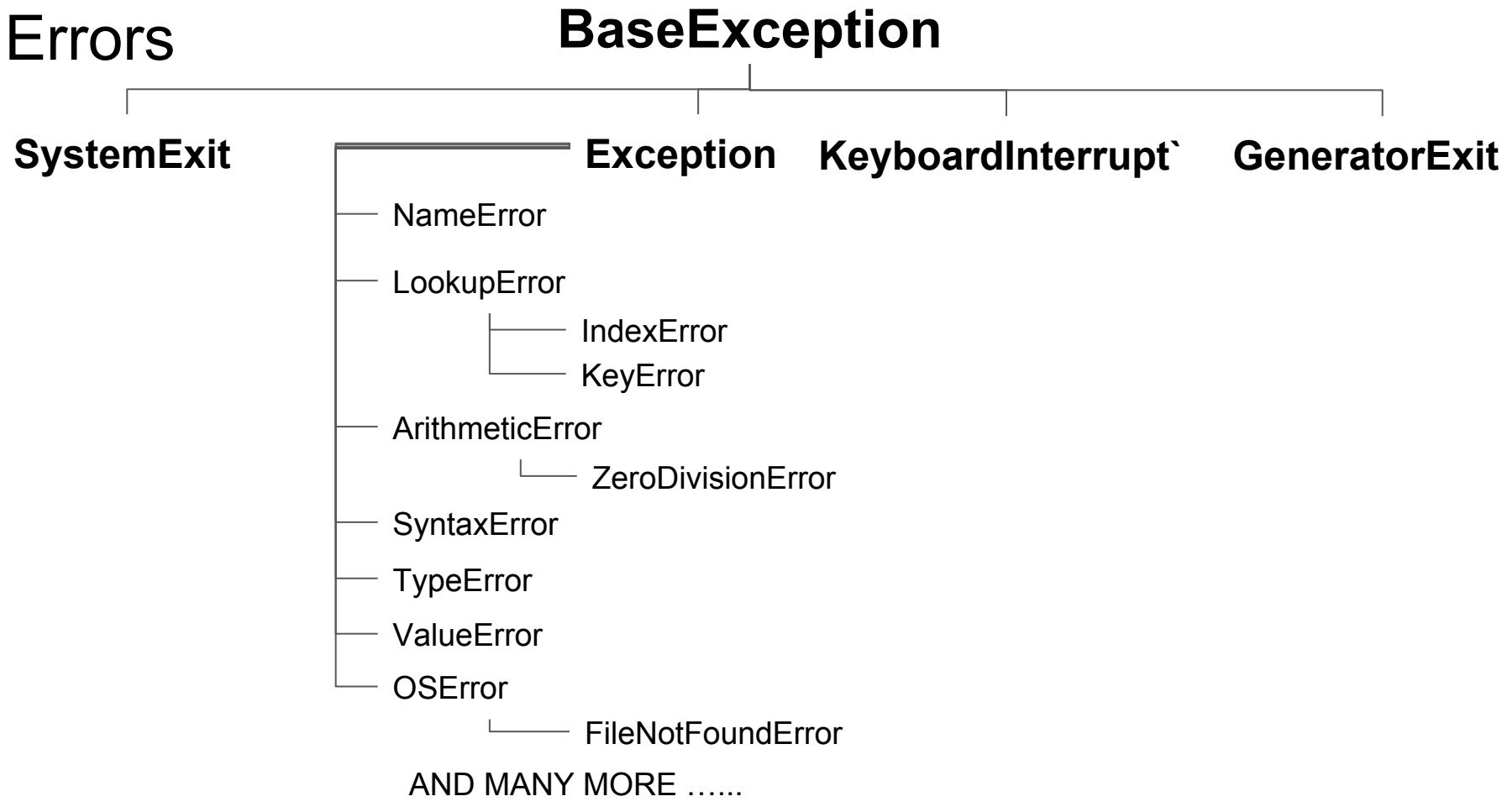
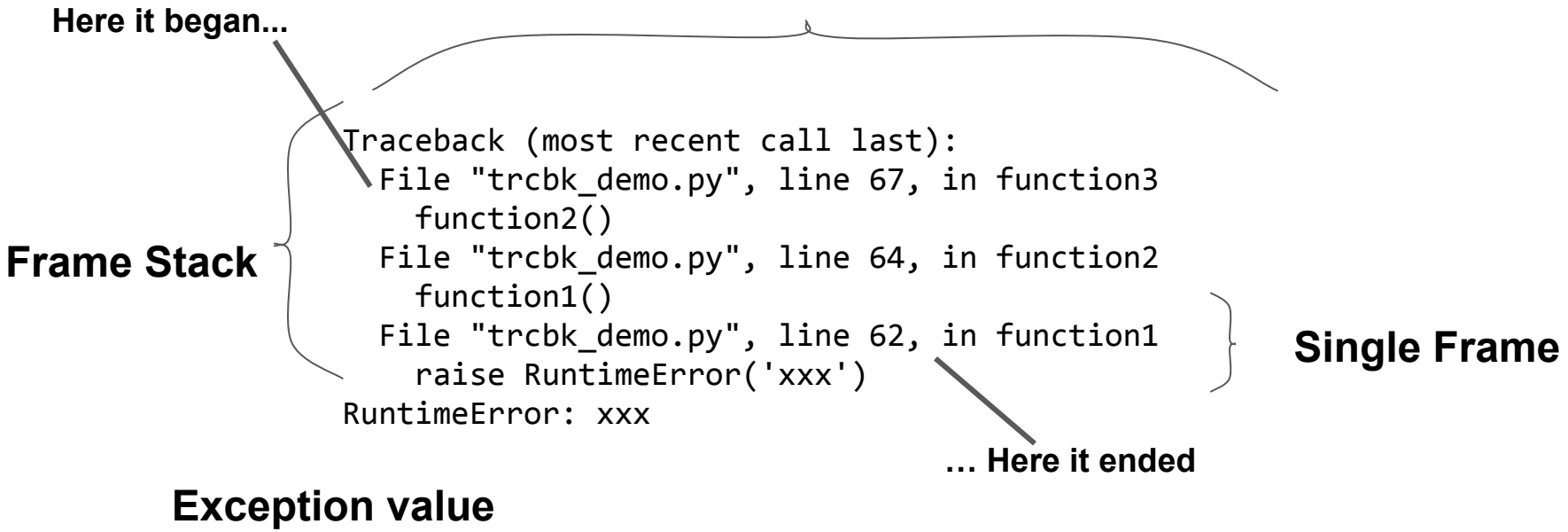
Traceback Module


Errors
BaseException
SystemExit
Exception
KeyboardInterrupt`
GeneratorExit
NameError
LookupError
IndexError
KeyError
ArithmeticError
ZeroDivisionError
SyntaxError
TypeError
ValueError
OSError
FileNotFoundError
AND MANY MORE ……

Handling Errors

```
try:
     # my code

except NameError
     #handling NameError

except TypeError
     #handling TypeError

except (OtherError,AnotherError)…:
     #handling all the listed

except:
     # any other exception
else:
     # if no error occured

finally:
     # executed always
```

Traceback


Here it began...
Traceback (most recent call last):
File "trcbk_demo.py", line 67, in function3
function2()
File "trcbk_demo.py", line 64, in function2
function1()
File "trcbk_demo.py", line 62, in function1
raise RuntimeError('xxx')
RuntimeError: xxx
Frame Stack
Single Frame
… Here it ended
Exception value

Traceback Module == Inspect Stack and Traceback

We are interested in

Exception Class

Exception Value

Traceback

Catching

```
sys.exc_info()
```

```
e = traceback.format_exception_only(*sys.exc_info()[:2])
e = traceback.format_exception(*sys.exc_info())
```

```
traceback.extract_stack()
traceback.extract_tb(sys.exc_info()[2])
```

List of FrameSummary objects

Printing

```
traceback.print_exception(sys.exc_info()
traceback.print_stack()
```

Generating

```
te = traceback.TracebackException(*sys.exc_info())
print(te.__dict__)
```

Try it out

```
import sys
def function1():
    raise RuntimeError('xxx')

def function2():
    function1()

def function3():
     try:
               function2()
     except Exception:
          print(*sys.exc_info())
          # here go your calls
          traceback.print_exception(*sys.exc_info())
          print('-------------')
          traceback.print_stack()
def function4():
    function3()

def function5():
    function4()

function5()
```

My faulty function

```
def magic_ball(n):
    personalities={2:'You are relaxed',
                   3: 'You are pretty angry',
                   5: 'I feel tension here'}
    num = int(num)
    for n in personalities:
        if num%n==0:
            print(personalities[n])
            break
    else:
        print('You are pretty strange')
```

Code continues

```
try:
    f = open('my_log.log','a')
    while True:
        num = input('Give me a number and I will tell you, how do you feel: ')
        magic_ball(num)
except:
    tb = traceback.extract_tb(sys.exc_info()[2])                # FrameSummary objects from TB in a list
    tb_list = traceback.format_list(tb)                         # formatted frame records from TB
    e = traceback.format_exception_only(*sys.exc_info()[:2])    # Only the exception message in a list
    stack = traceback.extract_stack()                           # current stack
    tb_obj = traceback.TracebackException(*sys.exc_info())
finally:
    f.close()
```

```
print(tb)
print(tb_list)
print(e)
```

Logging

Creating logger function

```
def make_logger(filename,name, level='DEBUG'):
    def logger(msg,tb):

        print('====',file=filename)
        print(level,':',name,':',msg,file=filename)
        for f in tb: # 'filename', 'line', 'lineno', 'locals', 'name'
            print('F: {} in {}| L#: {} | Code: {} |\nLocals: {}'.format(f.filename,
                                                                        f.name,
                                                                        f.lineno,
                                                                        f.line,
                                                                        f.locals),
                                                                        file=filename)

    return logger
```

Using Logger

```
try:
    fl = open('my_log.log','a')
    logger = make_logger(fl,__name__)
    logging.basicConfig(filename='my_log.logs')
    while True:
        num = input('Give me a number and I will tell you, how do
you feel: ')
        magic_ball(num)
except:
     logger('Problem with the main loop',tb)
```