Review 3

Exceptions and Try-Except Blocks
What Might You Be Asked

- Create your own Exception class
- Write code to throw an exception
- Follow the path of a thrown exception
  - Requires **understanding** of try-except blocks
  - Similar what you were asked in Prelim 1
  - But now with new `except` syntax
- Write a simply try-except code fragment
  - Think what you did with `fix_bricks` in A7
Error Types in Python

- All errors are instances of class `BaseException`
- This allows us to organize them in a hierarchy

```
BaseException
    |__init__(msg='')  __str__() ...
    |______________
    |             Exception
    |               |__init__  __str__()
    |               |______________
    | StandardError
    |               |__init__  __str__()
    |               |______________
    | AssertionError
```

→ means “extends” or “is an instance of”
Python Error Type Hierarchy

Exception

SystemExit
StandardError

AssertionError
AttributeError
ArithmeticError
IOError
TypeError
ValueError
ZeroDivisionError
OverflowError

Argument has wrong type (e.g. float([1]))
Argument has wrong value (e.g. float('a'))

We do not expect you to memorize all (or any) of these

http://docs.python.org/library/exceptions.html
Creating Your Own Exceptions

class CustomError(StandardError):
    """An instance is a custom exception"""
    pass

This is all you need
- No extra fields
- No extra methods
- No constructors
Inherit everything

Only issues is choice of parent Exception class. Use StandardError if you are unsure what.
When Do Exceptions Happen?

**Automatically Created**

```python
def void foo():
    x = 5 / 0
```

Python creates Exception for you automatically

**Manually Created**

```python
def void foo():
    raise Exception('I threw it')
```

You create Exception manually by raising it
Raising Errors in Python

- **Usage:** `raise <exp>`
  - `exp` evaluates to an object
  - An instance of Exception

- **Tailor your error types**
  - **ValueError:** Bad value
  - **TypeError:** Bad type

- **Examples:**
  - `raise ValueError('not in 0..23')`
  - `raise TypeError('not an int')`

- **Only issue is the type**

```python
def foo(x):
    assert x < 2, 'My error'
    ...
```

```python
def foo(x):
    if x >= 2:
        m = 'My error'
        raise AssertionError(m)
    ...
```

Identical
def foo():
    x = 1
    try:
        x = 2
        raise StandardError()
        x = x+5
    except StandardError:
        x = x+10
    return x
Try-Except: Possible Exam Question

def foo():
    x = 1  # executes this line normally

    try:
        x = 2  # executes this line normally
        raise StandardError()
        x = x+5  # never reaches this line

    except StandardError:
        x = x+10  # but does execute this line

    return x  # and executes this line
def foo():
    x = 1
    try:
        x = 2
        raise StandardError()
    except StandardError:
        x = x+5
    return x

What does foo() evaluate to?

Answer: 12 (2+10)
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(2)?
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(2)?

'Starting first.'
'Starting second.'
'Starting third.'
'Caught at second'
'Ending second'
'Ending first'
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(0)?
More Exception Tracing

```python
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'
```

What is the output of `first(0)`?

'Starting first.'
'Starting second.'
'Starting third.'
'Ending third'
'Ending second'
'Ending first'
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except IOError:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except AssertionError:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    if x < 0:
        raise IOError()
    elif x > 0:
        raise AssertionError()
    print 'Ending third.'

What is the output of first(-1)?
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except IOError:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except AssertionError:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    if x < 0:
        raise IOError()
    elif x > 0:
        raise AssertionError()
    print 'Ending third.'

What is the output of first(-1)?

Starting first.
Starting second.
starting third.
Caught at first.
Ending first.
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except IOError:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except AssertionError:
        print 'Caught at second'
    print 'Ending second'

What is the output of first(1)?
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except IOError:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except AssertionError:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    if x < 0:
        raise IOError()
    elif x > 0:
        raise AssertionError()
    print 'Ending third.'

What is the output of `first(1)`?

Starting first.
Starting second.
Starting third.
Caught at second.
Ending second.
Ending first.
def isfloat(s):
    """Returns: True if string s represents a float.
    False otherwise"""
    # Implement Me

float(s) returns an error if s does not represent a float
def isfloat(s):
    """Returns: True if string s represents a float. False otherwise"""
    try:
        x = float(s)
        return True
    except:
        return False

Conversion to a float might fail
If attempt succeeds, string s is a float
Otherwise, it is not