Exceptions and Files

Python exceptions are errors

What errors have you seen Python give so far?

Potentially risky code!

```
some_dict = { "a": 3, "b": 4 }

key = raw_input("Type a letter: ")
print "Value: %s" % some_dict[key]
```

Solution #1: Avoid errors!

```
some_dict = { "a": 3, "b": 4 }

key = raw_input("Type a letter: ")
if key in some_dict:
    print "Value: %s" % some_dict[key]
else:
    print "Key %s doesn't exist" % key
```

Solution #2: "catch" the exception

```
some_dict = { "a": 3, "b": 4 }

key = raw_input("Type a letter: ")
try:
    print "Value: %s" % some_dict[key]
except KeyError:
    print "Key %s doesn't exist" % key
```

More exceptions

```
def get_age():
    age = int(raw_input("What is your age? "))
    return age

user_age = get_age()
print "Your age is %i" % user_age
```

Exceptions are like "return"

```
def get_age():
    age = int(raw_input("What is your age? "))
    print "Does this line get printed?"
    return age

try:
    user_age = get_age()
    print "Your age is %i" % user_age
except ValueError as e:
    print "Invalid age: %s" % e
```

Exceptions are like "return", part 2

```
def convert_to_int(s):
    print "Debug line 1"
    value = int(s)
    print "Debug line 2"
    return value

def get_age():
    print "Debug line 3"
    age = raw_input("What is your age? ")
    age = convert_to_int(age)
    print "Debug line 4"
    return age
```

What about invalid ages?

```
def get_age():
    age = int(raw_input("What is your age? "))
    # What if the user enters a negative num?
    return age

try:
    user_age = get_age()
    # How do you know the age is invalid here?
    print "Your age is %i" % get_age()
except ValueError as e:
    print "Invalid age: %s" % e
```

Solution #1: "Magic" return number

```
def get_age():
    """ Returns an age, and -1 if invalid """
    age = int(raw_input("What is your age? "))
    if age < 0:
        return -1
    return age

age = get_age()
if age == -1:
    print "Invalid age"
else:
    print "Your age is %s" % age
```

Solution #2: Multiple return values

```
def get_age():
    """ Returns a tuple of (age, success)
    If age is valid, success is True. """
    age = int(raw_input("What is your age? "))
    if age < 0:
        return age, False
    return age, True

ret = get_age()
if ret[1]: # if success is true...
    print ret[0] # print age
```

Solution #3: Raise an exception

```
def get_age():
    """ Returns an age or throws ValueError """
    age = int(raw_input("What is your age? "))
    if age < 0:
        raise ValueError("Bad age: %i" % age))
    return age

try:
    print "Your age is %i" % get_age()
except ValueError as e:
    print "Invalid age: %s" % e
```

Generalizing try..except

```
try:
    pass # some block of code
except Exception:
    pass # some other block of code
```

Generalizing try..except

```
try:
    pass # some block of code
except Exception as e:
    pass # some other block of code
```

Generalizing try..except

```
try:
    pass # some block of code
except ValueError as e:
    pass # handle value errors
except KeyError as e2:
    pass # handle key errors
```

try : except :: if : else

```
try:
    pass # some block of code
except ValueError as e:
    pass # handle value errors
except KeyError as e2:
    pass # handle key errors
except:
    pass # any other exception
```

Spot the error in this program

```
try:
    value = raw_input(enter an int)
    print int(value)
except Exception:
    print "Invalid input"
```

Danger: don't "except Exception"

```
try:
    value = raw_input(enter an int)
    print int(value)
except Exception:
    # This catches *EVERYTHING*.
    # ...even SyntaxError.
    print "Invalid input" # A lie
```

Fix: use specific exceptions

```
try:
    value = raw_input(enter an int)
    print int(value)
except ValueError:
    print "Invalid input"
```

Changing topics: files!

Opening and closing files

```
# Open "some_file.txt" for read
some_file = open('some_file.txt', 'r')

# Put the contents of the file in
# variable text.
text = some_file.read()

# Close the file now we're done.
some_file.close()
```

Help!

```
# Print out docstrings
print file.__doc__
help(file)

You can also do this for other things:
help(open)
help(int)
```

A "file" here is a variable

```
def print_opened_file(f):
    text = f.read()
    print text

some_file = open('some_file.txt', 'r')
print_opened_file(some_file)
some_file.close()
```

Using "with" to auto-close files

```
with open('some_file.txt', 'r') as f:
    text = f.read()
    print text
    # When this block of code is done
    # f is automagically closed.
```

Using "with" to auto-close files, pt. 2

```
def file_has_word(filename, word):
    with open(filename, 'r') as f:
        text = f.read()
        return text.count(word) > 0
        # When this block of code is
        # done, f is still closed.
        # ...even if you return.
```

File IO creates exceptions

- File doesn't exist
- Can't access file (permissions)
- Disk is physically broken
- Can't write to file (locked)
- ...etc

Exception handling!

```
try:
    with open('myfile.txt', 'r') as f:
        text = f.read()
        print text
        # "with" still closes the file
        # even if there's an exception
except IOError as e:
    print e
```

Writing to files

```
with open('output.txt', 'w') as f:
    for x in range(100):
        f.write(str(x))
```

Newline separators: \n

```
with open('output.txt', 'w') as f:
    for x in range(100):
        f.write(str(x))
        f.write("\n")
```

Newline separators: \n

```
with open('output.txt', 'w') as f:
    for x in range(100):
        f.write("%i\n" % x)
```

Miscellaneous File Functions

These are all in:

http://docs.python.org/tutorial/inputoutput.html

readline(): read one line at a time

```
with open('input.txt', 'r') as f:
    line1 = f.readline()
    line2 = f.readline()
```

readlines(): get all lines as a list

```
with open('input.txt', 'r') as f:
    all_lines = f.readlines()
    for line in all_lines:
        print line
```

You can also use a for loop

```
with open('input.txt', 'r') as f:
    for line in f:
        print line
```