Practice exercises: files and exceptions

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Exercise: reading from a file

1. Define a function called **`largest`** which takes a single argument
2. The argument passed will be an opened file object
3. Read the data in the file
4. Assume that the data is separated by spaces and are all numbers
5. Find the maximum value in the file
6. Do not use **`loadtxt`**!

Solution

```
def largest(f):
    data = []
    for line in f:
        for field in line.split():
            data.append(float(field))
    return max(data)
```

Another solution

```
def largest(f):
    res = -1e20
    for line in f:
        for field in line.split():
            res = max(res, float(field))
    return res
```

Exercise: reading/writing files

1. Read the **`pendulum.txt`** file
2. Print the second column alone to another file called **`col2.txt`**
3. Remember to add a newline

Solution

```
f = open('pendulum.txt')
out = open('col2.txt', 'w')
for line in f:
    fields = line.split()
    out.write(fields[1] + '\n')
f.close()
out.close()
```

Another solution

```
f = open('pendulum.txt')
out = open('col2.txt', 'w')
for line in f:
    fields = line.split()
    print(fields[1], file=out)
f.close()
out.close()
```

Exercise: simple exceptions

1. Write a function called **`my_sum`**
2. The function is passed a single string with terms separated by spaces
3. The string contains both names and integer values in arbitrary order
4. Find the sum of all the numbers in the string

For example:

```
>>> my_sum('1 fox, 2 dogs and 3 jackals')
6
>>> my_sum('3 blind mice and 1 man')
4
```

Possible solution

```
def my_sum(s):
    total = 0
    for word in s.split():
        try:
            total += int(word)
        except ValueError:
            pass
    return total
```

Exercise: catching exceptions

1. Write a function called `safe_run(f, x)`
2. `f` is a function and `x` is a value
3. `f` can raise either `ValueError` or `TypeError`
4. Your function should return `'OK'` if no exception is raised
5. Return `'ValueError'` if `ValueError` is raised
6. Return `'TypeError'` if `TypeError` is raised

For example:

```
>>> safe_run(float, 'A')
'ValueError'
>>> def f(x): return x + 2
>>> safe_run(f, '2')
'TypeError'
>>> safe_run(float, '2')
'OK'
```

Possible solution

```
def safe_run(f, x):
    try:
        f(x)
    except ValueError:
        return 'ValueError'
    except TypeError:
        return 'TypeError'
    else:
        return 'OK'
```

Exercise: raising exceptions

1. Write a function **`func`** which takes a single integer argument **`x`**
2. If **`x`** is not a positive integer raise a **`ValueError`**
3. If **`x`** is not an integer type raise a **`TypeError`**

Possible solution

```
def func(x):
    if type(x) != int:
        raise TypeError('Expected int')
    elif x < 0:
        raise ValueError('Got negative int')
```

What next?

- Only covered the very basics
- More advanced topics remain
- Read the official Python tutorial: `docs.python.org/tutorial/`

Thank you!