CIS 180 L - Intro to Python - Fall 2006
Homework Assignment #6

Due: TUESDAY, October 24th, beginning of class

Purpose: To practice a bit with docstrings and exception handling in Python

Note: as long as you meet the specifications below, you may add additional embellishments as you wish.

How to turn this in: submit the files **help1.txt**, **help2.txt**, **pydoc1.txt**, and **hw7.py** using ~st10/180pysubmit on cs-server.

1. This will be a little different – consider the posted solutions for **hw3.py**, **hw4.py**, and **hw5.py** available from the course Moodle site. Select one of them (or select a module that you have written that includes at least two functions within it).

Copy this module to the directory you want to work from for this homework, and add appropriate docstrings to it – add one at the beginning describing the module overall, and add one as the first line within each function.

Now, test those docstrings: inside either python or IDLE,

* import your modified module,

* run the help command for your module (**help(***modname***))** -- copy and paste the help-results from the screen to a file **help1.txt**. (It should show your new docstrings to nice effect!)

* and, run the help command for a function within your module (**help(***modname.functname***)** if you imported it using **import** *modname,* and **help(***functname***)** if your imported it using **from** *modname* **import ***) -- copy and paste the help-results from the screen to a file **help2.txt**. (And this should highlight the docstring you used for that function.)

Finally, practice with the **pydoc** command. This will NOT be in python or IDLE – this would be at a command line (perhaps in cs-server). In UNIX, you can save the output of a command in a file by **redirecting** it --- that means to put a **>** and then the name of the desired output file.

Try these commands at a UNIX prompt in a directory containing your module:

```
cs-server> pydoc modname
```

...and you should see your docstrings within a nice manual page. To write this to a file **pydoc1.txt**, then:

```
cs-server> pydoc modname > pydoc1.txt
```

Submit **help1.txt**, **help2.txt**, and **pydoc1.txt** on cs-server using **~st10/180pysubmit** .

2. Now, a small experiment: what happens if you try to open a file that does not exist?

Try it --- try to open a file **nonexistent** for reading within Python or IDLE. (That is, **open('nonexistent', 'r')**). What exception does it throw? Note that exception.

Now, let's use that to write a small function. Create a plain-text file **hw7.py**. Start it with a **docstring** containing at least a brief description of this module, your name, and the last modified date.

Now write a function **try_read**. It should take a string as its argument, intended to be a file name. It should:

* try to open its argument for reading. If it can, it should then read and return the first line read.

* if it CANNOT open its argument for reading, however, it should not crash. It should use **try** and **except** to instead return **No such file** if it cannot open its argument because that file does not exist.

* include a docstring as the first thing within **try_read**.

Examples:
Assume that **file1** is readable and has as its first line **Here is the first line.** Assume that **file2** does not exist. Then:

```
hw7.try_read('file1') == "Here is the first line.\n"
hw7.try_read('file2') == "No such file"
```

3. What exception is thrown if you try to index a string or a list or a tuple with an index outside of that string or list or tuple? Try it and see... (**'name'[27]** for example)

Add a function **get_many** that accepts a string, an index, and a quantity as its three arguments.

* It tries to grab the character at that index in the string, repeat it quantity times, and return the result.

* If that index is NOT in the string, however, return an empty string. (And, yes, I know this can be done with **len** in an if-statement! But for THIS homework, use a **try** and **except** instead, for exception-handling practice.)

* include a docstring as the first thing within **get_many**

Examples:

```
ruler = 'King Arthur"
hw7.get_many(ruler, 5, 6) == "AAAAAA"
hw7.get_many(ruler, 27, 18) == ""
```

Now you can also submit **hw7.py** using ~st10/180pysubmit on cs-server (joining **help1.txt**, **help2.txt**, and **pydoc1.txt** submitted previously).