Computer Science 125
6 October 2020

1. What exceptions could occur from the following Python code?

```
a = float(input("Please enter a number: "))
b = float(input("Please enter another number: "))
fraction = a/b
F = open("myfile.txt", "w")
F.write(fraction)
F.close()
```

input might not be a number
ValueError

b could be zero
ZeroDivisionError

file might not exist, or might not be writable
PermissionError

needs to be a string
TypeError

2. How could you handle the exceptions in the code above?

```
try:
    a = float(input("Please enter a number: "))
    b = float(input("Please enter another number: "))
    fraction = a/b
except ValueError:
    print("Sorry, cannot convert that to a number.")
except ZeroDivisionError:
    print("Sorry, cannot divide by zero")

try:
    F = open("myfile.txt", "w")
    F.write(fraction)
    F.close()
except PermissionError:
    print("Sorry, cannot write file.)
```

3. What exceptions could occur from the following Python code? How could you handle them?

```
import random

def mayCauseError():
    L = [5, 12]
    r = random.randrange(0, 3)
    n = L[r]

    s = random.randrange(0, 3)
    a = n/s

    return a
```

IndexError

ZeroDivisionError

another solution:

```
def mayCauseError():
    L = [5, 12]
    r = random.randrange(0, 3)
    if r < len(L):
        n = L[r]
    else:
        print("Sorry!")
```

one solution:

```
import random

def mayCauseError():
    L = [5, 12]
    r = random.randrange(0, 3)
    n = L[r]

    s = random.randrange(0, 3)
    a = n/s

    return a

def main():
    try:
        mayCauseError()
    except IndexError:
        print("Sorry, an index error occurred.")

main()
```

```
            return None

        s = random.randrange(0, 3)
        a = n/s

        return a

    mayCauseError()
```

4. What does the `raise` statement do in Python?

raise allows you to declare that an exception has occurred

```
import random

class myNewError(Exception):
    pass

def mayCauseError():
    raise myNewError
    L = [5, 12]
    r = random.randrange(0, 3)
    n = L[r]

    s = random.randrange(0, 3)
    a = n/s

    return a

def main():
    try:
        mayCauseError()
    except IndexError:
        print("Sorry, an index error occurred.")
    except myNewError:
        print("Your error occurred.")

main()
```

PRACTICE WITH EXCEPTIONS – SOLUTIONS

1. The following line of code might raise a `ValueError`:

   ```
   def inputInteger():
       n = int(input("Enter an integer: "))
       return n
   ```

   Call this function from within a `try` statement and use `except` to catch any `ValueError` that occurs. If a `ValueError` occurs, then display a helpful error message.

   ```
   def inputInteger():
       n = int(input("Enter an integer: "))
       return n

   try:
       inputInteger()
   except ValueError:
       print("Sorry, that input does not seem to be an integer.")
   ```

2. Modify your previous code so that, if the user does not enter an integer, the program asks the user for another integer. The program should continue asking until the user enters an integer.

   ```
   repeat = True
   while repeat:
       try:
           n = inputInteger()
           repeat = False
       except ValueError:
           print("Sorry, that input does not seem to be an integer.
                 Please try again.")

   print("You entered: ", n)
   ```

3. Write a function called `oops` that intentionally raises an `IndexError` exception when called. Then call your function from within a `try` statement and use `except` to catch the error.

   ```
   def oops():
      raise IndexError

   print("PRACTICE PROBLEM 3:")
   try:
      oops()
   except IndexError:
      print("An IndexError occurred.")
   ```

4. **Bonus:** The following code retrieves a file from a web server:

```
import urllib.request
result = urllib.request.urlopen( someURL )
```

However, the function `urllib.request.urlopen()` can raise various exceptions. For example, it raises a `ValueError` if it cannot interpret its argument as a valid URL. It also raises a `urllib.error.HTTPError` if the server cannot find the requested file. To see this for yourself, run the following code:

```
import urllib.request
urllib.request.urlopen( "test" )  # this line raises a ValueError
urllib.request.urlopen( "http://www.mlwright.org/nofile.txt" )
   # this line above raises a urllib.error.HTTPError
```

Write a program that asks the user for a URL and then uses `urllib.request.urlopen()` to request that URL. If `urllib.request.urlopen()` raises a `ValueError` or `urllib.error.HTTPError`, then your program should handle the exception and print a helpful message for the user.

If no error occurs, you can (usually) print the HTML file returned like this:

```
byteData = result.read()
strData = byteData.decode("utf8")
print(strData)
```

Once you have the data from the file stored in a Python string, you can extract information from it. Experiment with using Python to read web pages!

```
import urllib.request

try:
   f =
urllib.request.urlopen("http://www.mlwright.org/teaching/cs121s17/in
dex.php")
   print(f)
   print("URL:", f.geturl())   # optional
   print("INFO:", f.info())    # optional
   print("CODE:", f.getcode()) # optional
   bytedata = f.read()
   strdata = bytedata.decode("utf8")
   print(strdata)
except ValueError:
        print("Invalid URL.")
except urllib.error.HTTPError:
   print("A HTTP Error occurred")
except urllib.error.URLError:
   print("URL not found")
```