Slide Type: Slide

1 ICS 104 - Introduction to Programming in Python and C

1.1 Files and Exceptions - Lab

Slide Type: Slide

2 Lab Objectives

- To read and write text files
- To process collections of data
- To raise and handle exceptions

Slide Type: Slide

3 Worked Example

- **Problem Statement** Read two country data files, `worldpop.txt` and `worldarea.txt`. Both files contain the same countries in the same order. Write a file `world_pop_density.txt` that contains country names and population densities (people per square km), with the country names aligned left and the numbers aligned right.

Slide Type: -

© Oksana Perkins/iStockphoto.

Singapore is one of the most densely populated countries in the world.

Slide Type: Fragment

- **Step 1:** Understand the processing task
  - We need to read each file, line by line, and then compute the density (The countries are stored in the same order).

```
While there are more lines to be read
   Read a line from each file.
   Extract the country name.
   population = number following the country name in the line from the first file
   area = number following the country name in the line from the second file
   If area != 0
      density = population / area
   Print country name and density.
```

Slide Type: Slide

- **Step 2:** Determine which files you need to read and write.

Slide Type: Fragment

-
  - Input Files: `worldpop.txt` and `worldarea.txt`
  - Output Files: `world_pop_density.txt`

Slide Type: Fragment

- **Step 3:** Choose a mechanism for obtaining the file names.
  - Hard-coding the filenames.

```
filename = "worldpop.txt"
```

  - Asking the user for inputting the filename.

```
filename = input("Enter filename: ")
```

- Hard-coded file names are included for simplicity.

Slide Type: Slide

- **Step 4:** Choose between iterating over the file or reading individual lines.
- We will read individual lines using a `while` loop.

Slide Type: Fragment

- **Step 5:** Extract the required data.

Slide Type: Fragment

- **Step 6:** Use functions to factor out common tasks.
- Because both input files have the same format, the name of the country followed by a value, we can use a single function to extract a data record.

In [2]:

Slide Type:

```
##
#  This program reads data files of country populations and areas and prints the
#  population density for each country.
#

POPULATION_FILE = "worldpop.txt"
```

```
POPULATION_FILE = "worldpop.txt"
AREA_FILE = "worldarea.txt"
REPORT_FILE = "world_pop_density.txt"

def main() :
   # Open the files.
   popFile = open(POPULATION_FILE, "r")
   areaFile = open(AREA_FILE, "r")
   reportFile = open(REPORT_FILE, "w")

   # Read the first population data record.
   popData = extractDataRecord(popFile)
   while len(popData) == 2 :
      # Read the next area data record.
      areaData = extractDataRecord(areaFile)

      # Extract the data components from the two lists.
      country = popData[0]
      population = popData[1]
      area = areaData[1]

      # Compute and print the population density.
      density = 0.0
      if area > 0 :   # Protect against division by zero.
            density = population / area
            reportFile.write("%-40s%15.2f\n" % (country, density))

      # Read the next population data record.
        popData = extractDataRecord(popFile)

   # Close the files.
   popFile.close()
   areaFile.close()
   reportFile.close()

## Extracts and returns a record from an input file in which the data is
#  organized by line. Each line contains the name of a country (possibly
#  containing multiple words) followed by an integer (either population
#  or area for the given country).
#  @param infile the input text file containing the line oriented data
#  @return a list containing the country (string) in the first element
#  and the population (int) or area (int) in the second element. If the end of
#  file was reached, an empty list is returned.
#
def extractDataRecord(infile) :
   line = infile.readline()
   if line == "" :
      return []
   else :
      parts = line.rsplit(" ", 1)
      parts[1] = int(parts[1])
      return parts

# Start the program.
main()
```

Slide Type

4 Reading the Entire File

- There are two methods for reading an entire file.
  - The call `inputFile.read()` returns a string with all characters in the file.
  - The `readlines` method reads the entire contents of a text file into a list:
    - `inputFile = open("sample.txt", "r")`
    - `listOfLines = inputFile.readlines()`
    - `inputFile.close()`
  - Each element in the list returned by the `readlines` method is a string containing a single line from the file (including the newline character). Only the last line does not contain the newline character.

Slide Type Slide

5 Exercises

Slide Type Fragment

- **Exercise # 1** Write a program that reads a file containing text. Read each line and send it to the output file, preceded by line numbers.

Slide Type -

- If the input file is

```
Mary had a little lamb
Whose fleece was white as snow.
And everywhere that Mary went,
The lamb was sure to go!
```

Slide Type -

- then the program produces the output file:

```
/* 1 */ Mary had a little lamb
/* 2 */ Whose fleece was white as snow.
/* 3 */ And everywhere that Mary went,
/* 4 */ The lamb was sure to go!
```

Slide Type -

- Prompt the user for the input and output file names.

In [9]:

Slide Type Fragment

```
# Exercise # 1 - Source Code
try:
    fileInput = open(input("Enter input file name:"),"r")
    fileOutput = open(input("Enter output file name:"),"w")
    numLine = 1
    for line in fileInput:
        fileOutput.write("/* %d */ %s" % (numLine,line))
        numLine += 1
    fileOutput.close()

except IOError:
    print("Error: file not found")
```

```
Enter input file name:inputex1.txt
Enter output file name:outputex1.txt
```

Slide Type Slide

- **Exercise # 2** Write a program that asks the user to input a set of floating-point values. When the user enters a value that is not a number, give the user a second chance to enter the value. After two chances, quit reading input. Add all correctly specified values and print the sum when the user is done entering data. Use exception handling to detect improper inputs.
- assume that reading the input ends only when the user enters 2 consecutive non-valid values

- assume that reading the input ends only when the user enters 2 consecutive non valid values
- Sample run is included below:

```
Enter a floating point value: 12.6
Enter a floating point value: 9
Enter a floating point value: -10.2
Enter a floating point value: 3rx
You did not enter a valid value
Enter a floating point value: 4
Enter a floating point value: 1.2
Enter a floating point value: .2
Enter a floating point value: 1x
You did not enter a valid value
Enter a floating point value: y
reading input will terminate
sum of all values entered correctly = 16.80
```

In [5]:

Slide Type Fragment

```python
# Exercise # 2 - Source Code
total = 0
while True:
    Value = input("Enter a floating point value: ")
    errors = 0
    try:
        FolatValue=float(Value)
        total += FolatValue
    except ValueError:
        errors = 1
        print("you did not enter a valid value ")
    if errors == 1:
        Value = input("Enter a floating point value: ")
        try:
            FolatValue=float(Value)
            total += FolatValue
        except:
            break
print("reading input will terminate")
print("sum of all values entered correctly = %.2f" % total)
```

```
Enter a floating point value: 1
Enter a floating point value: 2
Enter a floating point value: 3
Enter a floating point value: t
you did not enter a valid value
Enter a floating point value: 6
Enter a floating point value: g
you did not enter a valid value
Enter a floating point value: t
reading input will terminate
sum of all values entered correctly = 12.00
```

Slide Type Slide

- **Exercise # 3** Write a program that reads each line in a file, reverses its lines, and writes them to another file. For example, if the file input.txt contains the lines:

Slide Type Fragment

```
Mary had a little lamb
Its fleece was white as snow
And everywhere that Mary went
The lamb was sure to go.
```

- You may assume that the input file is called `inputex3.txt` and the output file is `outputex3.txt` .

Slide Type Fragment

- then output.txt contains

```
The lamb was sure to go.
And everywhere that Mary went
Its fleece was white as snow
Mary had a little lamb
```

In [3]:

Slide Type -

```python
# Exercise # 3 - Source Code
fileInput = open("inputex3.txt","r")
fileOutPUT = open("outputex3.txt","w")
LINES = fileInput.readlines()
for line in reversed(LINES):
    line = line.rstrip()
    fileOutPUT.write(line+"\n")
fileOutPUT.close()
```

Slide Type Slide

- **Exercise # 4** A hotel salesperson enters sales in a text file. Each line contains the following, separated by semicolons:
  - The name of the client, the service sold (such as Dinner, Conference, Lodging, and so on),
  - The amount of the sale, and the date of that event.
- Write a program that reads such a file and displays the total amount for each service category. Display an error if the file does not exit or the format of the record is incorrect. No need to display anything else in this regard. Note that you can assume that the dates are entered correctly (no need to validate them). You may also assume that the name of the input text file is `inputex4.txt` .
- Make sure you use a **dictionary** in your solution.
- Sample run is included below for the given input file `inputex4.txt`

```
Summary of total sales according to services
Dinner    634.75
Conference    1133.25
Lodging    550.25
Dinner and Lodging    625.14
```

In [10]:

Slide Type Fragment

```python
# Exercise # 4 - Source Code
try:
    infile = open("inputex4.txt","r")
    SERVEICES = {}
    for line in infile:
        ServiceList = line.split(";")
        if not ServiceList[1] in SERVEICES:
            SERVEICES[ServiceList[1]] = float(ServiceList[2])
```

```
        else:
            SERVEICES[ServiceList[1]] =SERVEICES[ServiceList[1]] +float(ServiceList[2])
    for serveice in SERVEICES:
        print("%-20s %5.2f"%(serveice+":",SERVEICES[serveice]))
except FileNotFoundError:
    print("file not found ")
except ValueError:
    print("WRONG FORMAT")
```

```
Dinner:              634.75
Conference:          1133.25
Lodging:             550.25
Dinner and Lodging:  625.14
```

In [ ]:

Slide Type

```
1
```