Flow Control

Objectives

To use the flow control structures of Python, and to gain familiarity in coding based on indentation! That does take a little practice. We will also be using a couple of modules from the Python standard library.

Reference Material

Chapter 3 Flow Control. The Python online documentation for the `glob` and `os` modules may also be useful. The final question uses the built-in `xrange()`, the syntax is given in the list of built-ins at the end of Chapter 01 Introduction to Python.

Questions

1. Using a **`for`** loop, display the files in your home directory, with their size
   a) Use the skeleton file **Ex3.py**
   b) Get the directory name from the environment using os.environ, HOMEPATH on Windows HOME on Linux (we have done that part for you, notice the test of `system.platform`).
   c) Construct a portable wildcard pattern using `os.path.join` (we have done that part for you as well)
   d) Use the `glob.glob()` function to obtain the list of filenames
   e) Use `os.path.getsize()` to find each file's size
   f) Add a test to only display files that are not zero length
   g) Use `os.path.basename()` to remove the leading directory name(s) from each filename before you print it.

2. Write a Python program that emulates the high-street bank mechanism for checking a PIN. Keep taking input from the keyboard (see below) until it is identical to a password number which is hard-coded by you in the program.

   To output a prompt and read from the keyboard:
   ```
   supplied_pin = raw_input("Please enter your PIN:")
   ```

   Restrict the number of attempts to 3 (be sure to use a variable for that, we may wish to change it later), and output a suitable message for success and failure. Be sure to include the number of attempts in the message.

3. Write a Python program to display a range of numbers by steps of -2.
   a) Prompt the user at the keyboard for a positive integer using:
   ```
   var = raw_input("Please enter an integer: ")
   ```

b) Validate the input (`var`) to make sure that the user entered an integer using the `isdigit()` method. If the user entered an invalid value, output a suitable error message and exit the program.

c) Use a loop to count down from this integer in steps of 2, displaying each number on the screen until either 1 or 0 is reached. For example, if the integer 16 (validated) is entered, the output would be:

```
16
14
12
10
8
6
4
2
0
```

and if 7 is entered, the output would be:

```
7
5
3
1
```

You will need to look-up the **`xrange()`** built-in in the online documentation, pay particular attention to the *stop* parameter.

.

If time allows…

4. If a year is exactly divisible by 4 but not by 100, the year is a leap year. There is an exception to this rule. Years exactly divisible by 400 are leap years. The year 2000 is a good example.

   Write a program that asks the user for a year and reports either a leap year or *not* a leap year. (*Hint*: x % y is zero if x is exactly divisible by y.) Test with the following data:

1984	is a leap year	1981	is NOT a leap year
1904	is a leap year	1900	is NOT a leap year
2000	is a leap year	2010	is NOT a leap year

   Use the following to ask the user for a year:

   ```
   year = int(input ('Please enter a year :'))
   ```

5. Take a look at the code template provided in **weekday.py**. Complete this program, to ask for a date in DD/MM/YYYY format and print out the day of the week for this date.

   There is a formula, called *Zeller's Congruence*, which calculates the day of the week from a given day, month and year. Zeller's congruence is defined as follows:

   ```
   z=(1+d+(m*2)+(3*(m+1)/5)+y+y/4-y/100+y/400) % 7
   ```

   where d, m and y are day, month, year and z is an integer (0 = Sun .. 6 = Sat).

   But with the following adjustments *before* use in the formula:

   If month is 1 or 2 and year is a leap year, subtract 2 from day.
   If month equals 1 or 2 and year is not a leap year, subtract 1 from day.
   If month is 1 or 2, add 12 to month.
   Your program should print out the name of the day (e.g. Monday), e.g.:

1/1/1980	Tuesday	9/8/1982	Monday
25/12/1983	Sunday	31/5/1989	Wednesday
2/2/1990	Friday	29/2/1992	Saturday

Solutions

Question 1

Here is our portable solution:

```
import sys
import glob
import os

# Get the directory name
if sys.platform == 'win32':
    dir = os.environ['HOMEPATH']
else:
    dir = os.environ['HOME']

# Construct a portable wildcard pattern
pattern = os.path.join(dir,'*')

# Use the glob.glob() function to obtain the list of
filenames
for filename in glob.glob(pattern):

   # Use os.path.getsize to find each file's size
   size = os.path.getsize(filename)

   # Only display files that are not zero length
   if size > 0:
       print os.path.basename(filename), size, 'bytes'
```

There are a number of solutions. If your solution differs from ours do not worry, provided it gives correct results!

Question 2

There are several solution to this question as well. Here is ours:

```
pin     = '0138'
limit   = 3
counter = 0

while counter < limit:
    supplied_pin = raw_input(
                        "Please enter your PIN:")
    counter += 1
    if supplied_pin == pin:
      break

if supplied_pin != pin:
    print "You had",limit,"tries and failed!"
else:
    print "Well done, you remembered it!"
    print "... and only after",counter,"attempts"
```

Question 3

Validate the input (`var`) to make sure that the user entered an integer using the `isdigit()` method. If the user entered an invalid value, output a suitable error message and exit the program.

```
var = raw_input("Please enter an integer: ")

if not var.isdigit():
    print "Invalid integer:",var
    exit(1)
```

Use a loop to count down from this integer in steps of 2, displaying each number on the screen until either 1 or 0 is reached.

```
for var in xrange(int(var),-1,-2):
    print var
```

If time allows…

Question 4

```
y = int(raw_input ('Please enter a year :'))

if y % 4 == 0 and (y % 400 == 0 or y % 100 != 0):
    print "Leap Year"
else:
    print "NOT a leap year"
```

Question 5

```
date = raw_input("Please enter date (DD/MM/YYYY): ")
d,m,y = date.split('/')
d = int(d)
m = int(m)
y = int(y)

if m == 1 or m == 2:
    m += 12
    if y % 4 == 0 and (y % 400 == 0 or y % 100 != 0):
        d -= 2
    else:
        d -= 1

z = 1 + d + (m * 2) + (3 * (m + 1) // 5) + y + y//4 - \
    y//100 + y//400

z %= 7

days =
["Sun","Mon","Tues","Wednes","Thurs","Fri","Satur"]

print days[z]+'day'
```

If you are interested in validating dates, checking days, months, or years, then look at the **calendar** module in the Python standard library.