#!/usr/bin/env python

# -\*- coding: utf-8; mode: python; py-indent-offset: 4; indent-tabs-mode: nil -\*-

# vim: fileencoding=utf-8 tabstop=4 expandtab shiftwidth=4

# (C) COPYRIGHT © Preston Landers 2010

# Released under the same license as Python 2.6.5

# Aron L. Hertendi

# Updated for python 3

import sys, os, traceback, types

def isUserAdmin():

if os.name == 'nt':

import ctypes

# WARNING: requires Windows XP SP2 or higher!

try:

return ctypes.windll.shell32.IsUserAnAdmin()

except:

traceback.print\_exc()

print("Admin check failed, assuming not an admin.")

return False

elif os.name == 'posix':

# Check for root on Posix

return os.getuid() == 0

else:

raise RuntimeError("Unsupported operating system for this module: %s" % (os.name,))

def runAsAdmin(cmdLine=None, wait=True):

if os.name != 'nt':

raise RuntimeError("This function is only implemented on Windows.")

import win32api, win32con, win32event, win32process

from win32com.shell.shell import ShellExecuteEx

from win32com.shell import shellcon

python\_exe = sys.executable

if cmdLine is None:

cmdLine = [python\_exe] + sys.argv

elif type(cmdLine) not in (types.TupleType,types.ListType):

raise ValueError("cmdLine is not a sequence.")

cmd = '"%s"' % (cmdLine[0],)

# XXX TODO: isn't there a function or something we can call to massage command line params?

params = " ".join(['"%s"' % (x,) for x in cmdLine[1:]])

cmdDir = ''

showCmd = win32con.SW\_SHOWNORMAL

#showCmd = win32con.SW\_HIDE

lpVerb = 'runas' # causes UAC elevation prompt.

# print "Running", cmd, params

# ShellExecute() doesn't seem to allow us to fetch the PID or handle

# of the process, so we can't get anything useful from it. Therefore

# the more complex ShellExecuteEx() must be used.

# procHandle = win32api.ShellExecute(0, lpVerb, cmd, params, cmdDir, showCmd)

procInfo = ShellExecuteEx(nShow=showCmd,

fMask=shellcon.SEE\_MASK\_NOCLOSEPROCESS,

lpVerb=lpVerb,

lpFile=cmd,

lpParameters=params)

if wait:

procHandle = procInfo['hProcess']

obj = win32event.WaitForSingleObject(procHandle, win32event.INFINITE)

rc = win32process.GetExitCodeProcess(procHandle)

#print "Process handle %s returned code %s" % (procHandle, rc)

else:

rc = None

return rc

def test():

rc = 0

if not isUserAdmin():

print("You're not an admin.", os.getpid(), "params: ", sys.argv)

#rc = runAsAdmin(["c:\\Windows\\notepad.exe"])

rc = runAsAdmin()

else:

print("You are an admin!", os.getpid(), "params: ", sys.argv)

rc = 0

os.system('pip install pygame')

os.system('pip install pynput')

os.system('pip install pillow')

x = input('Press Enter to exit.')

return rc

if \_\_name\_\_ == "\_\_main\_\_":

sys.exit(test())