# Author: Trevor Perrin

# See the LICENSE file for legal information regarding use of this file.

"""Miscellaneous functions to mask Python version differences."""

import sys

import os

import math

import binascii

if sys.version\_info >= (3,0):

def compat26Str(x): return x

# Python 3 requires bytes instead of bytearrays for HMAC

# So, python 2.6 requires strings, python 3 requires 'bytes',

# and python 2.7 can handle bytearrays...

def compatHMAC(x): return bytes(x)

def raw\_input(s):

return input(s)

# So, the python3 binascii module deals with bytearrays, and python2

# deals with strings... I would rather deal with the "a" part as

# strings, and the "b" part as bytearrays, regardless of python version,

# so...

def a2b\_hex(s):

try:

b = bytearray(binascii.a2b\_hex(bytearray(s, "ascii")))

except Exception as e:

raise SyntaxError("base16 error: %s" % e)

return b

def a2b\_base64(s):

try:

b = bytearray(binascii.a2b\_base64(bytearray(s, "ascii")))

except Exception as e:

raise SyntaxError("base64 error: %s" % e)

return b

def b2a\_hex(b):

return binascii.b2a\_hex(b).decode("ascii")

def b2a\_base64(b):

return binascii.b2a\_base64(b).decode("ascii")

def readStdinBinary():

return sys.stdin.buffer.read()

else:

# Python 2.6 requires strings instead of bytearrays in a couple places,

# so we define this function so it does the conversion if needed.

if sys.version\_info < (2,7):

def compat26Str(x): return str(x)

else:

def compat26Str(x): return x

# So, python 2.6 requires strings, python 3 requires 'bytes',

# and python 2.7 can handle bytearrays...

def compatHMAC(x): return compat26Str(x)

def a2b\_hex(s):

try:

b = bytearray(binascii.a2b\_hex(s))

except Exception as e:

raise SyntaxError("base16 error: %s" % e)

return b

def a2b\_base64(s):

try:

b = bytearray(binascii.a2b\_base64(s))

except Exception as e:

raise SyntaxError("base64 error: %s" % e)

return b

def b2a\_hex(b):

return binascii.b2a\_hex(compat26Str(b))

def b2a\_base64(b):

return binascii.b2a\_base64(compat26Str(b))

import traceback

def formatExceptionTrace(e):

newStr = "".join(traceback.format\_exception(sys.exc\_type, sys.exc\_value, sys.exc\_traceback))

return newStr