**The differences between os and sys modules, and list the commonly**

 OS: This module provides a portable way of using operating system dependent functionality.

SYS:This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

Os:

os.remove () # delete file

os.rename () # Rename file

os.walk () # Generate all file names under the directory tree

os.chdir () # change directory

os.mkdir / makedirs # Create directory / multi-layer directory

os.rmdir / removedirs # delete directory / multi-layer directory

os.listdir () # List files in the specified directory

os.getcwd () # Get the current working directory

os.chmod () # Change directory permissions

os.path.basename () # Remove the directory path and return the file name

os.path.dirname () # Remove the file name and return the directory path

os.path.join () # Combine the separated parts into a path name

os.path.split () # returns (dirname (), basename ()) # tuple

os.path.splitext () # return (filename, extension) tuple

os.path.getatime \ ctime \ mtime # return the most recent access, creation, modification time

os.path.getsize () # return file size

os.path.exists () # whether exists

os.path.isabs () # Whether it is an absolute path

os.path.isdir () # Whether it is a directory

os.path.isfile () # Whether it is a file

Sys:

sys.argv # Command line parameter List, the first element is the path of the program itself

sys.modules.keys () # return a list of all imported modules

sys.exc\_info () # Get the exception class currently being handled, exc\_type, exc\_value, exc\_traceback details of the exception currently being handled

sys.exit (n) # Exit the program, exit (0) when exiting normally

sys.hexversion # Get the version value of the Python interpreter, in hexadecimal format: 0x020403F0

sys.version Get version information of Python interpreter

sys.maxint # Maximum Int value

sys.maxunicode # Maximum Unicode value

sys.modules # returns the module field imported by the system, key is the module name, value is the module

sys.path # returns the search path of the module, using the value of the PYTHONPATH environment variable during initialization

sys.platform # returns the operating system platform name

sys.stdout # standard output

sys.stdin # standard input

sys.stderr # error output

sys.exc\_clear () # used to clear the current or recent error message of the current thread

sys.exec\_prefix # returns the location where platform-independent python files are installed

sys.byteorder # local byte rule indicator, the value of big-endian platform is 'big', the value of little-endian platform is 'little'

sys.copyright # record python copyright related things

sys.api\_version # interpreter C API version

9\*9 multiplication：

def multiplication():

for i in range(1,10):

for j in range(1,i+1):

print(str(j) + str('✖') + str(i)+'=' + str(i\*j),end="\t")

print()

def main():

multiplication()

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

main()

trailingzeros：

import math

def trailingzeros(n):

total = 0

while (n):

n = int(n / 5)

total += n

return total

def main():

n = eval(input("Please enter the number to be calculated:"))

if (n >= 0):

print('{}! = {},The number of zeros is {}'.format(n, math.factorial(n), trailingzeros(5)))

else:

print("Erros,Please enter a natural number")

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

main()

cal numbers：

import math

def calnumbers():

numbers = []

for i in range(-100, 1000000):

a = math.sqrt(i + 100)

b = math.sqrt(i + 100 + 168)

if (int(a) == a) & (int(b) == b):

numbers.append(i)

return numbers

def main():

print("There is not only one such number, these numbers are:",calnumbers())

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

main()

comparenumbers：

def comparenumbers():

l = list(map(int, input('Please enter three numbers, separated by spaces').split()))

list.sort(l)

print(l)

def main():

comparenumbers()

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

main()