Standard modules**¶**

In [ ]:

from IPython.core.display import HTML  
css = "table td:nth-child(1) {font-weight:bold; background-color: #fec;}"  
HTML('<style>{}</style>'.format(css))

sys : System-specific parameters and functions**¶**

|  |
| --- |
| Attribute |
| sys.argv | list of cmd line args passed to a script; argv[0] is the script path |
| sys.ps1 | Primary shell prompt |
| sys.ps2 | Secondary shell prompt |
| sys.stdin | Standard Input file object; mutable |
| sys.stdout | Standard Output file object; mutable |
| sys.stderr | Standard Error file object; mutable |
| sys.\_\_stdin\_\_ | Original value of sys.stdin |
| sys.\_\_stdout\_\_ | Original value of sys.stdout |
| sys.\_\_stderr\_\_ | Original value of sys.stderr |
| sys.executable | Path to the python executable |
| sys.path | List of search paths for python modules; initialized from PYTHONPATH env var |
| sys.platform | Platform identifier string |

|  |  |
| --- | --- |
| System | platform value |
| Linux (2.x and 3.x) | 'linux2' |
| Windows | 'win32' |
| Windows/Cygwin | 'cygwin' |
| Mac OS X | 'darwin' |
| OS/2 | 'os2' |
| OS/2 EMX | 'os2emx' |
| RiscOS | 'riscos' |
| AtheOS | 'atheos' |

In [ ]:

import sys  
sys.stdout = open("output","w")

In [ ]:

sys.platform

In [ ]:

import os  
os.uname()

os**¶**

Python provides a wide variety of operating system interfaces

* Basic system calls
* Operating environment
* Processes
* Timers
* Signal handling
* Error reporting
* Users and passwords
* A large portion of this functionality is contained in the os module.
* The interface is based on POSIX.
* Not all functions are available on all platforms (especially Windows/Mac).

os.path : File and path manipulation**¶**

In [ ]:

abspath(path) # Returns the absolute pathname of a path  
basename(path) # Returns filename component of path  
dirname(path) # Returns directory component of path  
normcase(path) # Normalize capitalization of a name  
normpath(path) # Normalize a pathname  
split(path) # Split path into (directory, file)  
splitdrive(path)# Split path into (drive, pathname)  
splitext(path) # Split path into (filename, suffix)  
expanduser(path)# Expands ~user components  
expandvars(path)# Expands environment vars ’$name’ or ’${name}’  
join(p1,p2,...) # Join pathname components

In [ ]:

from os.path import abspath, basename, dirname, normpath, split, splitext  
abspath("../foo")

In [ ]:

basename("/usr/bin/python")

In [ ]:

dirname("/usr/bin/python")

In [ ]:

normpath("/usr/./bin/python")

In [ ]:

split("/usr/bin/python")

In [ ]:

splitext("index.html")

os.path : Functions for portable filename inquires**¶**

In [ ]:

exists(path) # Test for existence  
isabs(path) # Return 1 if path is an absolute pathname  
isfile(path) # Return 1 if path is a regular file  
isdir(path) # Return 1 if path is a directory  
islink(path) # Return 1 if path is a symlink  
ismount(path) # Return 1 if path is a mountpoint  
getatime(path) # Get access time  
getmtime(path) # Get modification time  
getsize(path) # Get file size in bytes  
samefile(path1,path2) # Return 1 if path1 and path2 are the same file  
sameopenfile(f1,f2) # Return 1 if file objects f1 and f2 are same file.

File and dir operations**¶**

In [ ]:

dir(os)

In [ ]:

chdir()  
chown()  
chroot()  
getcwd()  
getenv()  
listdir()  
mkdir()  
rmdir()  
rename()  
popen()  
popen2()

In [ ]:

In [ ]:

Low-level File and Directory Manipulation**¶**

Portable way to access OS file and dir operations

In [ ]:

os.access(path,accessmode) # Checks access permissions on a file  
os.chmod(path,mode) # Change file permissions  
os.chown(path,uid,gid) # Change owner and group permissions  
os.link(src,dst) # Create a hard link  
os.listdir(path) # Return a list of names in a directory  
os.mkdir(path [,mode]) # Create a directory  
os.remove(path) # Remove a file  
os.rename(src,dst) # Rename a file  
os.rmdir(path) # Remove a directory  
os.stat(path) # Return file information  
os.statvfs(path) # Return filesystem information  
os.symlink(src,dst) # Create a symbolic link  
os.unlink(path) # Remove a file (same as remove)  
os.utime(path,(atime,mtime)) # Change access and modification times

In [ ]:

In [ ]:

In [ ]:

Misc**¶**

StringIO**¶**

In [ ]:

import StringIO  
f = StringIO.StringIO()  
f.write("Hello World\n")  
f.getvalue()

* cStringIO is a C-implementation of the same lib
  + much faster
  + can't be sub-classed
* StringIO is python; can be sub-classed
* StringIO objects support most of the normal file operations

time**¶**

In [ ]:

time.clock() # Current CPU time in seconds  
time.time() # Current time (GMT) in seconds since epoch  
time.localtime(secs)# Convert time to local time (returns a tuple).  
time.gmtime(secs) # Convert time to GMT (returns a tuple)  
time.asctime(tuple) # Creates a string representing the time  
time.ctime(secs) # Create a string representing local time  
time.mktime(tuple) # Convert time tuple to seconds  
time.sleep(secs) # Go to sleep for awhile

In [ ]:

import time  
t = time.time()  
t

In [ ]:

tp = time.localtime(t)  
tp

In [ ]:

time.asctime(tp)

Exercises**¶**