CURSOS INTERSEMESTRALES

PROTECO

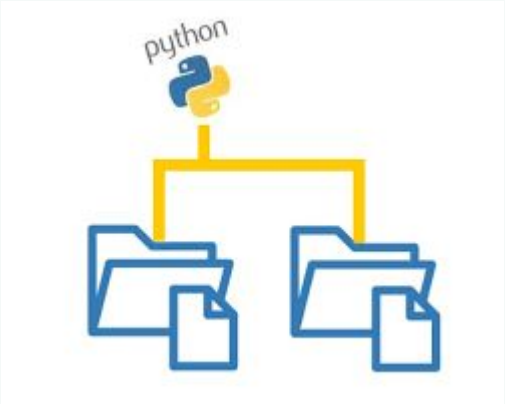
Programación del Sistema

Python intermedio

Subtemas

Programación del sistema

a. Sys
b. Os



PROTECO

Sys y Os

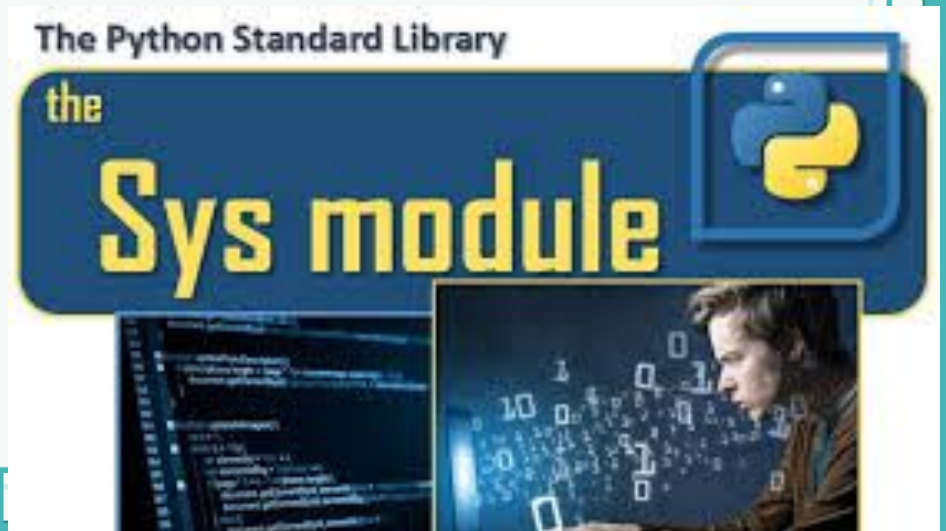
Entre los módulos de sistema que Python nos provee a través de su librería estándar, podemos destacar dos: os y sys.

sys y os forman el núcleo del arsenal de herramientas del sistema incorporado de Python.

PROTECO

Sys y Os

Los módulos os y sys proporcionan numerosas herramientas para tratar nombres de archivos, rutas y directorios,etc.



PROTECO

Sys

Este módulo proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete y a funciones que interactúan fuertemente con el intérprete. Ejemplos: Versión del intérprete, S.O. de la máquina que ejecuta el programa,

PROTECO

dir(sys)

La función dir simplemente devuelve una lista que contiene los nombres de todos los atributos en cualquier objeto con atributos. Es válido para os.

```
>>> dir(sys)

['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__', '__interactivehook__', '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__stdin__', '__stdout__', '_clear_type_cache', '_current_frames', '_debugmallocstats', '_enablelegacywindowsfsencoding', '_framework', '_getframe', '_git', '_home', '_xoptions', 'api_version', 'argv', 'base_exec_prefix', 'base_prefix', 'breakpointhook', 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyright', 'displayhook', 'dllhandle', 'dont_write_bytecode', 'exc_info', 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags', 'float_info', 'float_repr_style', 'get_asyncgen_hooks', 'get_coroutine_origin_tracking_depth', 'get_coroutine_wrapper', 'getallocatedblocks', 'getcheckinterval', 'getdefaultencoding', 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval', 'gettrace', 'getwindowsversion', 'hash_info', 'hexversion', 'implementation', 'int_info', 'intern', 'is_finalizing', 'last_traceback', 'last_type', 'last_value', 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'set_coroutine_wrapper', 'setcheckinterval', 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'version', 'version_info', 'warnoptions', 'winver']
```

Sys

len(dir(sys))

Muestra el número de atributos que posee el módulo. El contenido de estos dos módulos puede variar según la versión y plataforma de Python. Es válido para os.

```
>>> len(dir(sys))
81
>>> len(dir(os))
328
```

```
>>> len(dir(sys))

91
>>> len(dir(os))

150
```

PROTECO

Sys

help(sys)

Muestra la documentación del módulo en formato legible. Es válido para os.

sys.platform

Muestra el Sistema operativo en el que se ejecuta el código

Sistema	`platform` valor
Linux	`'linux'`
Windows	`'win32'`
Windows / Cygwin	`'cygwin'`
Mac OS X	`'darwin'`

```
>>> sys.platform
'linux'
```

Sys

sys.version

Muestra la versión del intérprete de Python que ejecuta nuestro código

sys.path

permite inspeccionar la ruta de búsqueda del módulo tanto de forma interactiva como dentro de un programa Python. sys.path es una lista de cadenas de nombres de directorios que representan la ruta de búsqueda verdadera en un intérprete de Python en ejecución.

PROTECO

Sys

sys.modules

Es un diccionario que contiene los nombres de los módulos de Python existentes que el shell actual ha importado.

sys.argv

Es una lista donde almacenan los parámetros introducidos.

Variable	Descripción
`sys.argv`	Retorna una lista con todos los argumentos pasados por línea de comandos. Al ejecutar `python modulo.py arg1 arg2`, retornará una lista: `['modulo.py', 'arg1', 'arg2']`
`sys.executable`	Retorna el path absoluto del binario ejecutable del intérprete de Python
`sys.maxint`	Retorna el número positivo entero mayor, soportado por Python
`sys.platform`	Retorna la plataforma sobre la cuál se está ejecutando el intérprete
`sys.version`	Retorna el número de versión de Python con información adicional

PROTECO

Sys

sys.argv[0] es el nombre del script
sys.argv[1] es el primer argumento
sys.argv[2] es el segundo argumento
len(sys.argv) da la cantidad de argumentos que se recibieron
len(sys.argv[1]) es la longitud del primer argumento que se recibió, sin considerar el nombre del script
len(sys.argv[2]) es la longitud del segundo argumento que se recibió, sin considerar el nombre del script

PROTECO

Os

El módulo **os** nos permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y nos permiten manipular la estructura de directorios (para leer y escribir archivos).

PROTECO

Os

Operacionalmente, os sirve como una interfaz en gran parte portátil para las llamadas al sistema de la computadora: **los scripts escritos con os y os.path generalmente pueden ejecutarse sin cambios en cualquier plataforma.**

PROTECO

Os

Tasks	Tools
Shell variables	os.environ
Running programs	os.system, os.popen, os.execv, os.spawnv
Spawning processes	os.fork, os.pipe, os.waitpid, os.kill
Descriptor files, locks	os.open, os.read, os.write
File processing	os.remove, os.rename, os.mkfifo, os.mkdir, os.rmdir
Administrative tools	os.getcwd, os.chdir, os.chmod, os.getpid, os.listdir, os.access
Portability tools	os.sep, os.pathsep, os.curdir, os.path.split, os.path.join
Pathname tools	os.path.exists('path'), os.path.isdir('path'), os.path.getsize('path')

dir(os)

```
>>> dir(os)

['DirEntry', 'F_OK', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXC
L', 'O_NOINHERIT', 'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LI
VED', 'O_TEMPORARY', 'O_TEXT', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT', 'P
_NOWAITO', 'P_OVERLAY', 'P_WAIT', 'PathLike', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'S
EEK_SET', 'TMP_MAX', 'W_OK', 'X_OK', '_Environ', '__all__', '__builtins__', '__c
ached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spe
c__', '_execvpe', '_exists', '_exit', '_fspath', '_get_exports_list', '_putenv',
 '_unsetenv', '_wrap_close', 'abc', 'abort', 'access', 'altsep', 'chdir', 'chmod
', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath', 'device_encoding', '
devnull', 'dup', 'dup2', 'environ', 'error', 'execl', 'execle', 'execlp', 'execl
pe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fdopen', 'fsdecode', 'fs
encode', 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_exec_path', 'get_handle_i
nheritable', 'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'geten
v', 'getlogin', 'getpid', 'getppid', 'isatty', 'kill', 'linesep', 'link', 'listd
ir', 'lseek', 'lstat', 'makedirs', 'mkdir', 'name', 'open', 'pardir', 'path', 'p
athsep', 'pipe', 'popen', 'putenv', 'read', 'readlink', 'remove', 'removedirs',
'rename', 'renames', 'replace', 'rmdir', 'scandir', 'sep', 'set_handle_inheritab
le', 'set_inheritable', 'spawnl', 'spawnle', 'spawnv', 'spawnve', 'st', 'startfi
le', 'stat', 'stat_result', 'statvfs_result', 'strerror', 'supports_bytes_enviro
n', 'supports_dir_fd', 'supports_effective_ids', 'supports_fd', 'supports_follow
_symlinks', 'symlink', 'sys', 'system', 'terminal_size', 'times', 'times_result'
```

PROTECO

Os

Además de todo esto, el módulo anidado **os.path** exporta aún más herramientas, la mayoría de las cuales están relacionadas con el procesamiento de nombres de archivos y directorios de manera portátil.

PROTECO

Os

En caso de que esas listas no sean suficientes para continuar, veamos algunas de las herramientas de sistema operativo más utilizadas. Al igual que sys, el módulo os viene con una colección de herramientas informativas y administrativas:

os.getpid()

Proporciona el ID del proceso del proceso de llamada (un identificador único definido por el sistema para un programa en ejecución, útil para el control de procesos y la creación de nombres únicos),

```
>>> os.getpid()
2728
```

PROTECO

os.getcwd()

Devuelve el directorio de trabajo actual. El directorio de trabajo actual es donde se asume que los archivos abiertos por su script viven, a menos que sus nombres incluyan rutas de directorio explícitas.

```
>>> os.getcwd()
'/home/solanita/Documentos'
```

PROTECO

os.path.isdir('ruta')

Muestra si la ruta es un directorio.

os.path.isfile('ruta')

Muestra si la ruta es un archivo

Ambas devuelven True o False.

```
>>> import os
>>> os.path.isdir('/home/solanita/Documentos')
True
```

PROTECO

os.path.join(*ruta* , * *rutas*)

Une uno o más componentes de la ruta de forma inteligente. El valor de retorno es la concatenación de la ruta y cualquier miembro de * rutas con exactamente un separador de directorio (os.sep) después de cada parte no vacía excepto la última, lo que significa que el resultado solo terminará en un separador si la última parte está vacía.

```
>>> import os

>>> import os.path

>>> print(os.path.join(os.sep, 'home', 'user', 'work'))

\home\user\work
```

PROTECO

OJO!

Las carpetas en UNIX están separadas por una diagonal "/"

```
>>> os.path.join(os.sep,'home', 'solana', 'Documentos')
'/home/solana/Documentos'
```

Mientras que en Windows las carpetas están separadas por una antidiagonal "\"

```
>>> print(os.path.join(os.sep, 'home', 'user', 'work'))

\home\user\work
```

PROTECO

os.system(‘comando’)

Ejecute el comando (una cadena) en una subshell.

Ejemplos de comandos:

clear -limpiar la pantalla

ls - enlistar el contenido de la carpeta

mkdir - crear carpeta

top - ver los procesos actuales

PROTECO

Descripción	Método
Saber si se puede acceder a un archivo o directorio	`os.access(path, modo_de_acceso)`
Conocer el directorio actual	`os.getcwd()`
Cambiar de directorio de trabajo	`os.chdir(nuevo_path)`
Cambiar al directorio de trabajo raíz	`os.chroot()`
Cambiar los permisos de un archivo o directorio	`os.chmod(path, permisos)`
Cambiar el propietario de un archivo o directorio	`os.chown(path, permisos)`
Crear un directorio	`os.mkdir(path[, modo])`
Crear directorios recursivamente	`os.mkdirs(path[, modo])`
Eliminar un archivo	`os.remove(path)`
Eliminar un directorio	`os.rmdir(path)`
Eliminar directorios recursivamente	`os.removedirs(path)`
Renombrar un archivo	`os.rename(actual, nuevo)`
Crear un enlace simbólico	`os.symlink(path, nombre_destino)`

Ejercicio

Con lo aprendido en los módulos Sys y Os, crear un menú con las siguientes opciones:

1.-Crear una carpeta

2.-Ver contenido de la carpeta

3.-Indicar qué sistema operativo es y saludar al usuario.

PROTECO

Referencias

https://docs.python.org/3/library

PROTECO