



Python

Clase 9-6

Módulos del Sistema

PROGRAMACION 2

Esc. Sup. De Comercio N° 49 Cap. Gral. J.J. de Urquiza

Técnico Superior en Desarrollo de Software



Módulos de sistema

En la librería estándar, existe un módulo especializado en el tratamiento de fechas: **datetime**.

Que los permite trabajar con fechas y horas

Elegir en el editor coding **UTF-8** para no tener problemas con vocales acentuadas y ñ para que permita caracteres del español

Pues bien, entre lo mucho que tiene *datetime* nos ofrece una clase especial de objeto: **date**, con todos sus atributos y métodos. Uno de ellos es el método **today()**, que nos devuelve la fecha actual.

```
File Edit Format Run Options Window Help
from datetime import date, time
print("Fecha actual:", date.today())
|
```

```
print("Día actual:", date.today().day)    nos devuelve el número del día
```

```
| print("Mes actual:", date.today().month)    nos devuelve el número del mes
```

```
| print("Fecha actual:", date.today().year)
| print("Día actual:", date.today().weekday())
```

weekday() nos devuelve el nro de día de la semana (0 a 6) 0 representa Lunes y así sucesivamente

```
print("Fecha y Hora Día actual:", datetime.now())
```

Modulo io

Operaciones de entrada y salida basadas en archivos



El primer paso para introducirnos es saber cómo abrir un determinado archivo. Para esto se utiliza la función `open`, que toma como primer argumento el nombre de un fichero.

```
from io import open
archivo_texto=open("archivo.txt","w")
```

La función `open()` se encuentra dentro del módulo estándar `io`. Acá estamos creando el archivo vacío

→ **tipo de apertura**

lo podemos abrir en modo lectura(`r`), escritura (`w`) o append (`a`)

nombre del archivo, por su extensión `.txt` será un archivo de texto

Para manejarnos con las funciones/métodos que manipulan los archivos siempre utilizamos la sintaxis del punto, es decir el nombre del objeto.nombre de la función o método

Con estas líneas en las que utilizamos el método **`write()`** agregamos contenido

```
frase="Estupendo día para estudiar Python \n el miércoles"
archivo_texto.write(frase)
```

`archivo_texto.close()` Para cerrar el archivo utilizamos el método `close()`

```
from io import open
archivo_texto=open("archivo.txt","r")
texto=archivo_texto.read()
archivo_texto.close()
print(texto)
```

Se queremos leer lo que hay escrito en el archivo lo abrimos de modo lectura "`r`" y usamos la función/método `read()`

Luego mostramos el contenido de la variable `texto` a la cual le hemos asignado el contenido del archivo.

De esta forma leemos el contenido total del archivo, pero si quisiéramos leer línea a línea podríamos usar:

```
from io import open
archivo_texto=open("archivo.txt","r")
lineas_texto=archivo_texto.readlines()
archivo_texto.close()
print(lineas_texto)
```

`Readlines()` lee línea a línea y almacena en una lista esto nos va a permitir realizar operaciones de búsqueda.

Al realizar el `print(lineas_texto)` obtenemos lo siguiente:

```
['Estupendo día para estudiar Python \n', ' el miércoles']
```

Los `[]` nos indican que es una lista

Si queremos acceder, por ejemplo, al primer elemento pondremos `print(lineas_texto[0])`.

```
from io import open

archivo_texto=open("archivo.txt","a")

archivo_texto.write("\n siempre es una buena ocasión para estudiar Python")

archivo_texto.close()
```

Acá estamos agregando contenido al archivo.

Recordemos que los caracteres `\n` indican nueva línea

Cuando leemos el archivo sino indicamos nada por defecto el puntero se sitúa al comienzo del archivo y cuando terminamos de leer la posición del puntero es la última

```
from io import open

archivo_texto=open("archivo.txt","r")

print(archivo_texto.read())
print(archivo_texto.read())
```

Si queremos leer dos veces no nos va a mostrar dos veces la misma información ya que luego del primer `read` el cursor se sitúa al final del archivo.

Cómo podemos hacer para modificar la posición del puntero.

Podemos hacerlo con el método `seek()`, este método nos va a pedir el nro de carácter en donde queremos que se posicione. Por ejemplo `seek(5)` se posicionará en la posición 5

```
from io import open

archivo_texto=open("archivo.txt","r")

print(archivo_texto.read())

archivo_texto.seek(0)

print(archivo_texto.read())
```

```
from io import open

archivo_texto=open("archivo.txt","r")

archivo_texto.seek(11) ←

print(archivo_texto.read())
```

`Seek` posiciona el puntero y luego con el `read` leemos a partir de ahí

El `read` tiene la opción de poder indicarle hasta que carácter queremos que lea, en el ejemplo leera desde donde está posicionado a la posición 11. Si hacemos una segunda lectura se hará a partir de la posición 11

```
print(archivo_texto.read(11))
```

```
from io import open

archivo_texto=open("archivo.txt","r")

archivo_texto.seek(len(archivo_texto.read())/2)
```

En este ejemplo situaremos el puntero en la mitad del texto.

`len` nos da la longitud de caracteres que contiene el archivo y al dividirlo por 2 será la mitad ese valor lo toma el `seek` para posicionarse

```
archivo_texto=open("archivo.txt","r+") @ lectura y escritura
```

Si abrimos el archivo como r+ podremos leer y escribir al mismo tiempo.

Además de poner leer por línea readlines también abriendo de esta forma podremos escribir por línea

```
archivo_texto.writelines(lista_texto)
```

En el siguiente ejemplo creamos un archivo html que contiene la estructura básica y el mensaje hola mundo.

```
# r : lectura de ficheros

# creamos el manejador para crear un archivo HTML
manejador = open("index.html", "a")

html = "<!DOCTYPE HTML>\n"
html += "<html>\n"
html += "<head>\n"
html += "<title>Hola mundo</title>\n"
html += "</head>\n"
html += "<body>\n"
html += "<h1>Hola mundo</h1>\n"
html += "</body>\n"
html += "</html>\n"

#Escribimos en el fichero
```

Módulo 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 [Referencia oficial](https://docs.python.org/3/tutorial/stdlib.html)).

<https://docs.python.org/3/tutorial/stdlib.html>

```
import os
```

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

Módulo `os.path`

El módulo `os.path` nos permite gestionar diferentes opciones relativas al sistema de ficheros como pueden ser ficheros, directorios, etc.

El módulo `os` también nos provee del submódulo `path` (`os.path`) el cual nos permite acceder a ciertas funcionalidades relacionadas con los nombres de las rutas de archivos y directorios.

Descripción	Método
Ruta absoluta	<code>os.path.abspath(path)</code>
Directorio base	<code>os.path.basename(path)</code>
Saber si un directorio existe	<code>os.path.exists(path)</code>
Conocer último acceso a un directorio	<code>os.path.getatime(path)</code>
Conocer tamaño del directorio	<code>os.path.getsize(path)</code>
Saber si una ruta es absoluta	<code>os.path.isabs(path)</code>
Saber si una ruta es un archivo	<code>os.path.isfile(path)</code>
Saber si una ruta es un directorio	<code>os.path.isdir(path)</code>
Saber si una ruta es un enlace simbólico	<code>os.path.islink(path)</code>
Saber si una ruta es un punto de montaje	<code>os.path.ismount(path)</code>

Vamos a ver algunos ejemplos

```
import os

ruta=os.getcwd()
print("La ruta actual es: ", ruta)

nueva_ruta=input("Introduzca nueva ruta: ")
if os.path.isdir(nueva_ruta):
    os.chdir(nueva_ruta)
else:
    print("Ruta No Valida")

ruta=os.getcwd()
print("Ahora la ruta es: ", ruta)
```

Este código nos muestra la ruta o path actual, permite ingresar la ruta donde quiero posicionarme y realizar el cambio hacia la misma, luego muestra para verificar que ha cambiado.

Si quisiéramos crear un archivo en ese lugar podríamos utilizar lo que vimos en el módulo system

```
archivo=input("Indique el nombre del Archivo: ")
manejador=open(ruta+archivo, "w")
manejador.close()

carpeta=input("Indique el nombre de la carpeta" )
os.mkdir(ruta+carpeta)
```

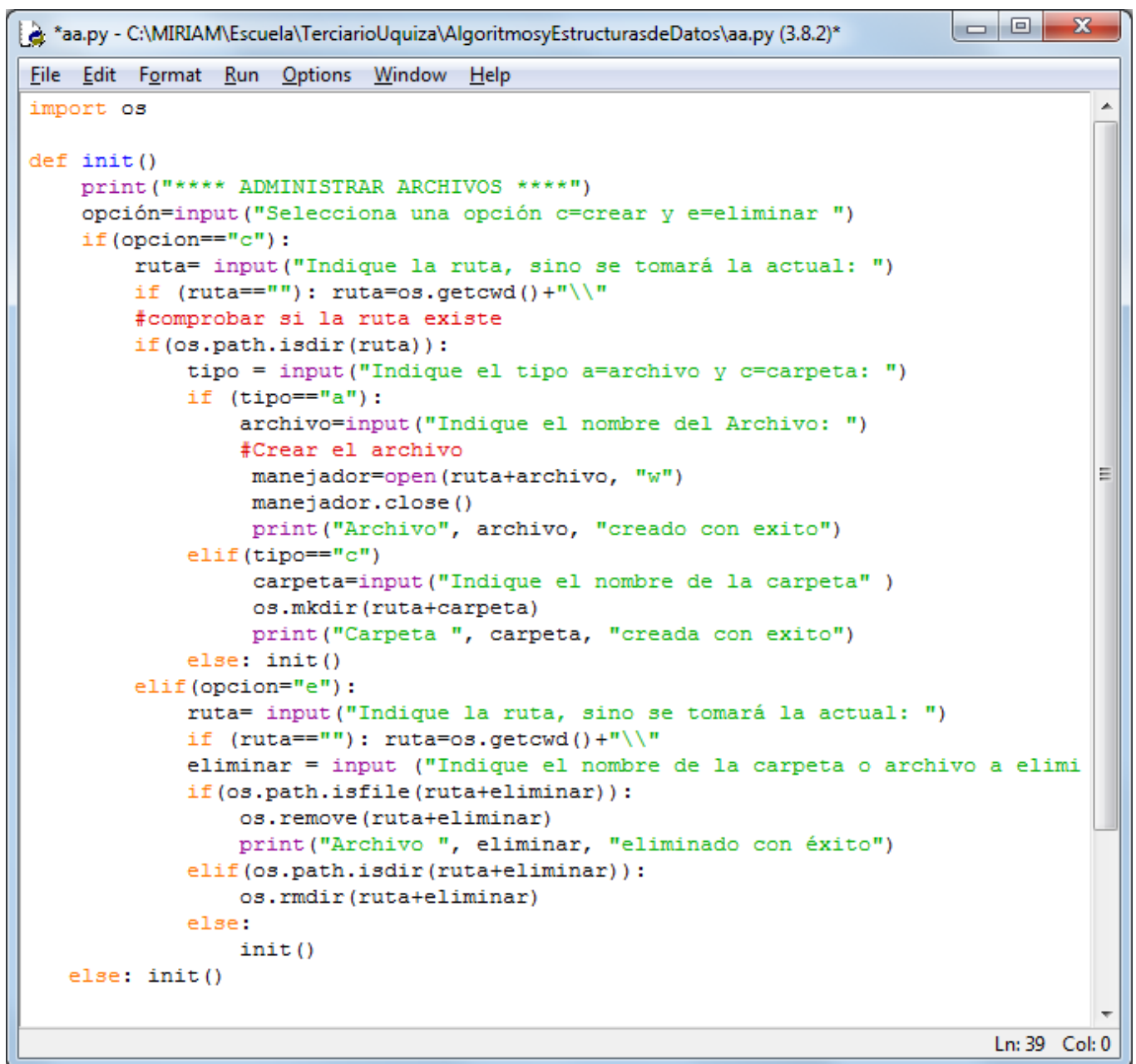
Si quisiéramos crear una carpeta

Verificamos que existe la carpeta y luego la eliminamos

```
eliminar=input("Indique el nombre de la carpeta a eliminar: ")
if (os.path.isdir(ruta+eliminar)):
    os.rmdir(ruta+eliminar)
```

Verificamos que existe el archivo y luego lo eliminamos

```
eliminar=input("Indique el nombre del archivo a eliminar: ")
if (os.path.isfile(ruta+eliminar)):
    os.remove(ruta+eliminar)
```

```
*aa.py - C:\MIRIAM\Escuela\TerciarioUquiza\AlgoritmosyEstructurasdeDatos\aa.py (3.8.2)*
File Edit Format Run Options Window Help

import os

def init():
    print("**** ADMINISTRAR ARCHIVOS ****")
    opción=input("Selecciona una opción c=crear y e=eliminar ")
    if(opción=="c"):
        ruta= input("Indique la ruta, sino se tomará la actual: ")
        if (ruta==""): ruta=os.getcwd()+"\\"
        #comprobar si la ruta existe
        if(os.path.isdir(ruta)):
            tipo = input("Indique el tipo a=archivo y c=carpeta: ")
            if (tipo=="a"):
                archivo=input("Indique el nombre del Archivo: ")
                #Crear el archivo
                manejador=open(ruta+archivo, "w")
                manejador.close()
                print("Archivo", archivo, "creado con éxito")
            elif(tipo=="c"):
                carpeta=input("Indique el nombre de la carpeta" )
                os.mkdir(ruta+carpeta)
                print("Carpeta ", carpeta, "creada con éxito")
            else: init()
        elif(opción=="e"):
            ruta= input("Indique la ruta, sino se tomará la actual: ")
            if (ruta==""): ruta=os.getcwd()+"\\"
            eliminar = input ("Indique el nombre de la carpeta o archivo a elimi
            if(os.path.isfile(ruta+eliminar)):
                os.remove(ruta+eliminar)
                print("Archivo ", eliminar, "eliminado con éxito")
            elif(os.path.isdir(ruta+eliminar)):
                os.rmdir(ruta+eliminar)
            else:
                init()
    else: init()

Ln: 39 Col: 0
```

Módulo sys

```
import sys
```

El módulo sys es el encargado de proveer variables y funcionalidades, directamente relacionadas con el intérprete.

Módulo subprocess

El módulo subprocess es aquel que nos permite trabajar de forma directa con órdenes del sistema operativo.

Entradas y salidas que pueden ser capturadas con Popen

- **stdout**: nomenclatura correspondiente a la salida estándar en sistemas UNIX-Like. Es la encargada de almacenar la salida de un programa.
- **stdin**: nomenclatura correspondiente a la entrada estándar en sistemas UNIX-like. Es la encargada de enviar información a un programa.
- **stderr**: al igual que las anteriores, se utiliza como referencia a los errores producidos en la salida de un programa.

Utilizando tuberías para capturar la salida

Popen nos permite capturar tanto la entrada como la salida estándar o su error. Para efectuar dicha captura, tanto `stdout` como `stdin` y/o `stderr` se pasan como argumentos clave a Popen. El valor de dichos argumentos, deberá ser un archivo o una tubería que funcione como tal. Y para esto, Popen, también nos provee de una tubería para capturar dichas entradas y salidas, llamada PIPE.

De esta forma, si quisiéramos capturar la salida estándar o error de nuestro código, debemos pasarle a Popen, `stdout` y `stderr` como argumentos claves, con PIPE como valor de cada uno de ellos, para lo cual, también debemos importar PIPE:

os.system()

Podemos ejecutar un comando del Sistema usando la función **os.system()**

Limpiar la pantalla nos resulta muy práctico si los resultados van a ser mostrados por **consola**. 'clear' en Linux y 'cls' en Windows como argumento.

```
os.system('cls')
```

```
os.system('clear')
```

```

import os
def menu():
    """
    Función que limpia la pantalla y muestra nuevamente el menu
    """
    os.system('clear') # NOTA para windows tienes que cambiar clear por cls
    print ("SISTEMA DE DIAGNOSTICO RÁPIDO DE FALLAS EN RED")
    print ("*****")
    print ("***** MENU *****")
    print ("HERRAMIENTA PING SELECCIONA (1)")
    print ("HERRAMIENTA TRACERT SELECCIONA (2)")
    print ("HERRAMIENTA HOSTNAME SELECCIONA (3)")
    print ("HERRAMIENTA IPCONFIG SELECCIONA (4)")
    print ("HERRAMIENTA DNS SELECCIONA (5)")
    print ("HERRAMIENTA GATEWAY SELECCIONA (6)")
    print ("SALIR (7)")
while True:
    menu()
    opcionMenu = input("inserta un numero valor >> ")
    if opcionMenu=="1":
        print ("SE EJECUTARA PING")
        os.system("ping 192.168.3.1")
    elif opcionMenu=="2":
        print ("SE EJECUTARA TRACEROUTE")
        os.system("tracert 200.48.225.130")
    elif opcionMenu=="3":
        print ("SE EJECUTARA HOSTNAME")
        os.system("hostname")
    elif opcionMenu=="4":
        print ("SE EJECUTARA IPCONFIG COMPLETO")
        os.system("ipconfig /all > ip.txt")
    elif opcionMenu=="5":
        print ("SE EJECUTARA ANALISIS DE DNS")
        os.system("ping 200.48.225.130")
    elif opcionMenu=="6":
        print ("SE EJECUTARA ANALISIS DE GATEWAY")
        os.system("ping 192.168.3.1")
    elif opcionMenu=="7":
        break
    else:
        print ("")input("No has pulsado ninguna opción correcta...\npulsa una tecla para continuar")

```

módulos cgi http.server http.client

La librería estándar de Python incluye los módulos `http.server`, `http.client` y `cgi`.

Servidor

Un servidor HTTP mínimo puede hacerse con el módulo `http.server`

CGI (**C**ommon **G**ateway **I**nterface, Interfaz de entrada común) es un estándar de programación web que consiste en ejecutar un programa en el servidor y desplegar su resultado hacia el cliente (o navegador web).

Por lo general, dicho programa o programas ejecutados en el servidor web están desarrollados en algún lenguaje de script (PHP, Perl, Python), por cuestiones de portabilidad entre las distintas plataformas; por lo tanto, se los denomina *scripts CGI* o simplemente *CGIs*.

En el área de desarrollo web con Python se pueden mencionar dos métodos distintos: vía CGI y vía un framework web como Django, Pyramid, web2py, entre otros

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.
4.  from http.server import HTTPServer, CGIHTTPRequestHandler
5.
6.
7.  class Handler(CGIHTTPRequestHandler):
8.      cgi_directories = ["/"]
9.
10.
11. httpd = HTTPServer(("", 8000), Handler)
12. httpd.serve_forever()
```

Para poder dar comienzo a la programación web vía CGI primero es necesario crear un servidor web para desarrollo local. Es decir, será éste el encargado de ejecutar el script CGI que solicita el navegador web y, al finalizar, retornar su contenido. Por lo tanto, crea un nuevo archivo `cgiserver.py` y añade lo siguiente:

Realmente no es necesario comprender el código anterior, aunque tampoco es muy difícil de entender. Simplemente se crea un nuevo servidor HTTP utilizando la librería estándar y se le añade soporte para ejecutar scripts CGI. Reitero, se trata únicamente de un servidor para desarrollo local, no para producción. Nótese la línea número 8, en la que se indican los directorios que contendrán archivos CGIs (en este caso con extensión `.py`). Por último, la línea número 11 indica que el servidor escuchará peticiones vía el puerto `8000`.

Una vez guardado el código anterior, ejecútalo.

Vamos a comenzar creando un archivo `script.py`, en el mismo directorio en donde tenemos corriendo nuestro servidor `cgiserver.py`. Luego, inserta el siguiente código:

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.
4.  import cgi
5.
6.  # Headers
7.  print("Content-Type: text/plain")
8.  print()
9.
10. # Contenido (texto plano, como lo hemos especificado)
11. print("""Hola mundo!""")
```