Lenguaje de Programación Python

Archivos

Dr. Mario Marcelo Berón
Argentina Programa
Universidad Nacional de San Luis

Archivos: Apertura, Lectura y Escritura de un Archivo

- Abrir Archivos:
  - open(): recibe como parámetro un string que indica el archivo que se desea abrir. Este string puede contener un paso absoluto o uno relativo. Esta función retorna un objeto archivo. También recibe como parámetro un modo el cual permite leer, escribir o ambas actividades.

Función Open

- La función open abre un archivo en modo lectura. Ese es el valor por defecto.
- Se puede especificar un parámetro adicional para cambiar el modo de acceso del archivo.

Archivos: Apertura, Lectura y Escritura de un Archivo

- Lectura del Contenido de Archivos:
  - *read()*: lee el contenido completo del archivo. Este método retorna como resultado el contenido del archivo completo como un string.
  - *readline()*: lee una línea del archivo.
  - *readlines()*: este método retorna una lista de strings. Un string por cada línea de texto.

Archivos: Apertura, Lectura y Escritura de un Archivo

Contenido del Archivo

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
python
```

Ejemplo

```
f=open("Datos.dat")
contenido= f.read()
print(contenido)
f.close()
```

Salida

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
python
```

Archivos: Apertura, Lectura y Escritura de un Archivo

Contenido del Archivo

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
python
```

Ejemplo

```
f=open("Datos.dat")
líneas=f.readline()
print(listaDeLineas)
f.close()
```

Salida

```
'Bienvenido'
```

Archivos: Apertura, Lectura y Escritura de un Archivo

Contenido del Archivo

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
python
```

Ejemplo

```
f=open("Datos.dat")
listaDeLineas=
        f.readlines()
print(listaDeLineas)
f.close()
```

Salida

```
['Bienvenido\n', 'Al curso de\n',
'Introducción al lenguaje\n',
'de programación\n', 'python']
```

Archivos: Apertura, Lectura y Escritura de un Archivo

- Escritura en Archivos: para escribir un archivo el mismo se tiene que abrir en modo escritura 'w' (sobreescribe el contenido del archivo) o en modo append 'a' (agrega al final del contenido del archivo). Si el archivo no existe y el modo es escritura (w) o append (a) el mismo se crea.
  - *write()*: este método recibe como parámetro un string que es el que se va a grabar en el archivo.

Lectura y Escritura

- Después de leer o escribir un archivo el mismo se debe cerrar utilizando la función *close()*
- El método *write* no agrega un caracter nueva línea se tiene que agregar explícitamente.

Archivos: Apertura, Lectura y Escritura de un Archivo

Contenido del Archivo

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
python
```

Ejemplo

```
f=open("Datos.dat","a")
f.write("Agrego una linea")
```

Salida

```
Bienvenido
Al curso de
Introducción al lenguaje
de programación
Agrego una línea
```

Otras funcionalidades Útiles de Archivo

Archivos: Paths

El módulo *pathlib* provee diferentes funciones para manejar los pasos en diferentes sistemas operativos.
Path recibe como parámetro strings que representan nombres de carpetas y archivos y retorna un string con el paso en la notación del sistema operativo que se está usando.

Ejemplo

```
from pathlib import Path
Path("Datos","Personas","Carlos.txt")
WindowsPath('Datos/Personas/Carlos.txt')
str(Path("Datos","Personas","Carlos.txt"))
'Datos\\Personas\\Carlos.txt'
```

Archivos: Operador /

El operador / se utiliza para unir paths y paths y strings. / tiene asociatividad de izquierda a derecha.

Ejemplo

```
from pathlib import Path
Path("Datos","Personas")
WindowsPath('Datos/Personas')
Path("/Docentes","Carlos.txt")
WindowsPath('/Docentes/Carlos.txt')
Path("Datos","Personas") / Path("/Doc","Carlos.txt")
WindowsPath('Datos/Personas/Doc/Carlos.txt')
```

Archivos: El Directorio de Trabajo Actual

Todo programa tiene su directorio corriente (*current working directory*) *cwd* por sus siglas en inglés. La función del módulo *pathlib cwd* permite recuperar el directorio corriente como un string y la función del módulo *os chdir* permite cambiar de directorio.

Ejemplo

```
from pathlib import Path
import os
Path.cwd()
WindowsPath('C:/Users/Programs/Python/Python37')'
os.chdir('C:\\Windows\\System32')
Path.cwd()
WindowsPath('C:/Windows/System32')
```

Archivos: Funciones Útiles

- *makedirs*: se encuentra en el módulo *os* y su función es crear carpetas. Recibe como parámetro un string que representa un paso. Esta función crea todas las carpetas intermedias que figuran en el string de forma tal de asegurarse de que exista el paso.

Ejemplo

```
>>> os.makedirs("./Documentos/Personas")
```

Archivos: Funciones Útiles

- *mkdir*: se encuentra en el módulo *os* y su función es crear una carpeta. A diferencia de la función anterior *mkdir* crea un directorio a la vez.

Ejemplo

```
>>> os.makedirs("Empleados")
```

Archivos: Funciones Útiles

- *is_absolute (path)*: se encuentra en el módulo *pathlib* y su función es determinar si el paso recibido como parámetro es absoluto. La función retorna *True* si el paso es absoluto y *False* en otro caso.

Ejemplo

```
>>> paso=Path.cwd()
>>> Path.is_absolute(paso)
```

Archivos: Funciones Útiles

- *abspath (path)*: se encuentra en el módulo *os* y su función es convertir un paso relativo en uno absoluto.

Ejemplo

```
>>> os.path.abspath(".")
```

Archivos: Funciones Útiles

- *isabs*: se encuentra en el módulo *os* y su función es determinar si el paso recibido como parámetro es absoluto. Retorna *True* si el paso es absoluto y *False* en otro caso.

Ejemplo

```
>>> os.path.isabs(".")
False
>>> os.path.isabs(Path.cwd())
True
```

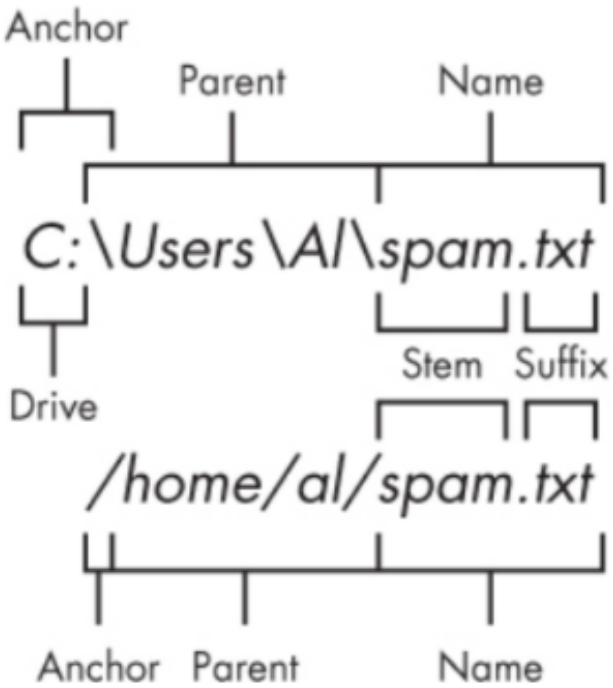
Archivos: Funciones Útiles

- *relpath(path, start)*: se encuentra en el módulo *os* y su función es retornar un paso relativo desde el paso *start* a *path*. En caso de que *start* no se provea se usa el directorio actual.

Ejemplo

```
>>> path="/home/user/Desktop/file.txt"
>>> start="/home/user"
>>> rp=os.path.relpath(path,start)
>>> rp
'Desktop/file.txt'
```

Archivos: Partes de un Paso a un Archivo



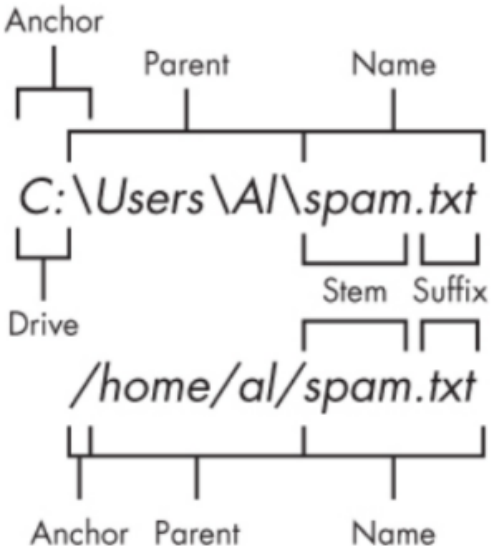
Un objeto *Path* provee diferentes atributos que permiten sacar las partes de un path.

Archivos: Partes de un Paso a un Archivo

Ejemplo

```
p = Path('C:/Users/Al/spam.txt')
p.anchor
'C:\\'
p.parent
WindowsPath('C:/Users/Al')
p.name
'spam.txt'
p.stem
'spam'
p.suffix
'.txt'
p.drive
'C:'
```

Archivos: Funciones Útiles

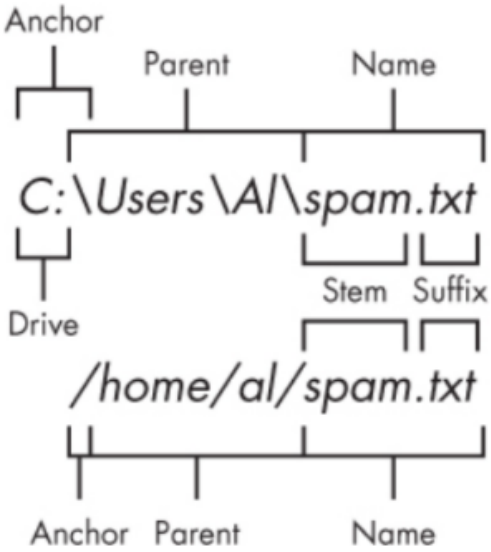


- *getSize (path)*: retorna el tamaño en bytes del archivo que se encuentra especificado en *path*.

Ejemplo

```
>>> os.path.getsize("/home/usuario/Computadora.py")
335
```

Archivos: Funciones Útiles



- *listdir(path)*: retorna una lista que contiene los nombres de archivos que se encuentran en *path*.

Ejemplo

```
>>> os.listdir("/home/usuario")
['.mozilla', '.bash_logout', '.bash_profile',....]
```

Archivos: Patrones Glob

- Los objetos Path tienen un método denominado glob() que permite listar el contenido de carpetas de acuerdo a un *Patrón Glob*.
- Un *Patrón Glob* es como una expresión regular simplificada que se usa en la línea de comandos.
- El *Patrón Glob* retorna un objeto glob el cual se pasa a lista para visualizar el contenido.

Ejemplos

```
list(p.glob('*'))
list(p.glob('*.txt')
list(p.glob('project?.docx')
list(p.glob('*.?x?')
```

Archivos: Validez de un Path

- p.exists(): retorna *True* si el path existe *False* en otro caso.

Ejemplo

```
>>> p
PosixPath('/home/usuario/usuario.dat')
>>> p.exists()
False
>>>
```

Archivos: Validez de un Path

- p.is_file(): retorna *True* si el path existe y es un archivo *False* en otro caso.

Ejemplo

```
>>> np=Path.cwd()/"Ejemplo1.py"
>>> np.is_file()
True
>>>
```

contenidos...

Archivos: Validez de un Path

- p.is_dir(): retorna *True* si el path existe y es un directorio *False* en otro caso.

Ejemplo

```
>>> Path.cwd().is_dir()
True
```