David Gopar Morales

Programación Avanzada.
Tarea 10:

Hacer un resumen de los siguientes métodos del tipo file de python: close, closed, encoding, errors, fileno, flush, isatty, mode, name, newlines, next, read, readinto, readline, readlines, seek, softspace, tell, truncate, write, writelines, xreadlines.

close: (self, /)Cierra el archivo, no tiene efecto si el archivo ya está cerrado. Ejemplo:

```
>>> f = open(r'C:\aiw.txt')
>>> f.readline()
"               ALICE'S ADVENTURES IN WONDERLAND\n"
>>> f.close()
>>> f.readline()
ValueError: I/O operation on closed file
```

closed: file.closed Nos regresa un boleano que establece si el archivo ha sido cerrado. Ejemplo:

```
>>> f = open(r'C:\test.txt')
>>> f.closed
False
>>> f.close()
>>> f.closed
True
```

encoding: Codificación del flujo de texto. Ejemplo:

```
>>> f = io.open(r'C:\aiw.txt', encoding='utf-8')
>>> f.encoding
'utf-8'
```

errors: La configuración de error del decodificador o codificador. Las subclases deben anularse. Ejemplo:

```
>>> f = io.open(r'C:\aiw.txt', errors='ignore')
>>> f.errors
'ignore'
```

fileno: (self, /) Devuelve el descriptor de archivo subyacente, si existe(es un número). Se genera OSError si el objeto IO no utiliza un descriptor de archivo. Ejemplo:

```
>>> f = open(r'C:\aiw.txt')
>>> f.fileno()
3
>>> f1 = open(r'C:\aiw.txt')
>>> f1.fileno()
4
```

flush: (self, /) Vacía los búferes de escritura, si corresponde. Esto no está implementado para transmisiones de solo lectura y sin bloqueo. Ejemplo:

```
>>> fw = open(r'C:\test.txt', "w")
>>> fw.write('foobar')
>>> fr = open(r'C:\test.txt')
```

>>> fr.readlines() # even though we just wrote a line to the fr file it appears empty until close() method is called
[]
>>> fw.flush() # flushing forces the buffer content into the file without closing it
>>> fr.readlines()
['foobar']

isatty: Devuelve si el flujo de archivos se trata de una secuencia 'interactiva'. Devuelve Falso si no se puede determinar. Ejemplo:
>>> f = open(r'C:\aiw.txt')
>>> f.isatty()
False

mode: file.mode, nos regresa el I/O mode de file. Si el archivo fue abierto usando open() como contructor de la función, mode será el valor del parámetro mode. Ejemplo:
```
>>> f = open(r'C:\test.txt', 'w')
>>> f.mode
'w'
```

name: file.name. Nos regresa el nombre del archivo. Ejemplo:
```
>>> f = open(r'C:\test.txt')
>>> f.name
'C:\\test.txt'
```

newlines: Terminaciones de línea traducidas hasta ahora. Solo se consideran los finales de línea traducidos durante la lectura. (Return type of newlines encountered while reading the file.)

next: __next__(self, /) Implementa nex(self). Ejemplo:
>>> f = open(r'C:\aiw.txt')
>>> f.next()
" ALICE'S ADVENTURES IN WONDERLAND\n"
>>> f.next()
'\n'
>>> f.next()
' Lewis Carroll\n'

read: read(self, size=-1, /) lee a lo más n caracteres de la cadena. Se lee desde el búfer subyacente hasta que tengamos n caracteres o presionemos EOF. Si n es negativo o se omite, lea hasta EOF. Ejemplo:
>>> f.read(8)
'   '
>>> f.read(128)
" ALICE'S ADVENTURES IN WONDERLAND\n\n Lewis Carroll\n\n THE MILLENNIUM FULCRUM EDITION"
>>> f.read() # this returns contents of the whole file

readinto: (b, /) Lee bytes en un lugar prelocalizado, escribe bytes como objetos en b, y nos regresa la cantidad de bytes leidos. Por ejemplo b pude ser un arreglo de bytes.

readline: readline(self, size=-1, /) nos regresa una linea del archivo. Ejemplo:

```
>>> f = open(r'C:\aiw.txt')
>>> f.readline()
"               ALICE'S ADVENTURES IN WONDERLAND\n"
>>> f.readline(32)
'\n'
>>> f.readline(64)
'                       Lewis Carroll\n'
```

readlines: nos regresa una lista de líneas del archivo. Ejemplo:

```
>>> f = open(r'C:\aiw.txt')
>>> f.readlines()  # this will return the list of all the lines in the file
```

seek: seek(self, cookie, whence=0, /) cambia la posición del archivo. Los valores para whence son

* 0 -- start of stream (the default); offset should be zero or positive
* 1 -- current stream position; offset may be negative
* 2 -- end of stream; offset is usually negative

nos regresa la nueva posición absoluta.
Ejemplo:

```
>>> # test.txt contents:
>>> # ABCDE
>>> f = open(r'C:\test.txt')
>>> f.seek(3)
>>> f.read()   # starts reading from the 3rd character
'DE'
```

Ejemplo:

```
>>> f = open(r'C:\test.txt')
>>> f.seek(2)    # move two characters ahead
>>> f.seek(2, 1) # move two characters ahead from the current position
>>> f.read()
'E'
```

Ejemplo:

```
>>> f = open(r'C:\test.txt')
>>> f.seek(-3, 2) # move to the 3rd character from the end of the file
>>> f.read()
'CDE'
```

softspace: Nos regresa un boleano que indica cuando un caracter debe ser impreso antes de otro valor cuando se usa print.
Ejemplo:

```
>>> f = open(r'C:\test.txt')
>>> f.softspace
0
```

tell: (self, /) nos regresa la posición actual del archivo. Ejemplo:

```
>>> # test.txt contents:
>>> # 0000 0000
>>> f = open(r'C:\test.txt')
```

```
>>> f.tell()
0L
>>> f.seek(3)  # changes position by 3
>>> f.tell()
3L
```

truncate : (self, pos=None, /) Redefine el tamaño del archivo a un tamaño específico. Truncar el archivo al tamaño de bytes. El puntero del archivo se deja sin cambios. El tamaño predeterminado es el IO actual posición según lo informado por tell(). Devuelve el nuevo tamaño. Ejemplo:

```
>>> # test.txt contents:
>>> # ABCDE
>>> f = open(r'C:\test.txt', 'r+')
>>> f.truncate(2)
>>> f.read()
'AB'
```

write: write(self, text, /) Escribe una cadena en el archivo, y nos regresa la cantidad de caracteres que se escribieron en el archivo. Ejemplo:

```
>>> f = open(r'C:\test.txt', 'w')
>>> f.write('foo')
>>> f.close()
>>> f = open(r'C:\test.txt')
>>> f.read()
'foo'
```

writelines: writelines(self, lines, /) Escribe una lista de líneas en un archivo. El separador de líneas no se agrega, por lo cual es usual que cada línea a agregar tenga un separador de líneas al final. Ejemplo:

```
>>> f = open(r'C:\test.txt', 'w')
>>> f.writelines(['foo', 'bar'])
>>> f.close()
>>> f = open(r'C:\test.txt')
>>> f.read()
'foobar'
```

xreadlines: file.xreadlines() nos regresa un iterador sobre las lineas del archivo, se supone que nos regresa lo mismo que iter(file) y que el uso de este método ya no es tan útil pues desde Python 2.3 los objetos File son iteradores por default. Ejemplo:

```
>>> # test.txt contents:
>>> #ABC AB A
>>> #ABC AB
>>> #ABC
>>> f = open(r'C:\test.txt')
>>> fx = f.xreadlines()
>>> fx.next()
'ABC AB A\n'
>>> fx.next()
'ABC AB\n'
>>> fx.next()
'ABC\n'
>>> fx.next()
StopIteration
```

Referencias:
https://python-reference.readthedocs.io/en/latest/docs/file/index.html
help(‘objeto tipo file’)
https://www.w3schools.com/python/python_ref_file.asp