CLASE ARCHIVO

Programación Concurrente y Paralela

Isai Samir Hernandez Lira

Liga del código

https://github.com/isai-samir/Clase-Archivo.git

Constructor

Se deben de importar las librerías a utilizar, como se utilizará la función exit se importará de la carpeta sys.

En el constructor se abre el archivo con el tipo 'r' que es solo lectura, con el nombre que recibe de parámetro el constructor, también se crea o abre un nuevo archivo(copia.txt) que es el que utilizara para la función de copiar a un archivo nuevo, en este caso envés de usar el tipo 'r' se utiliza el tipo de 'w' que es para escribir, aparte que crea una archivo en caso de no existir. Las partes del código en donde se abre y se crea el archivo están en un try para manejar los errores que existan (como que no exista), el nombre de la opción es de FileNotFoundError.

```
def __init__(self,nombre):
        try:
                self.archivo = open(nombre,'r')
                self.nonbre = nombre
                self.copia = open("copia.txt",'w')
        #El nombre de la excepcion es la siguiente
        except FileNotFoundError:
                print("No se puede abir el archivo ",nombre)
                exit()
```

Método encontrar

Como existen varios métodos en la que su función es de contar cuantas veces aparece algo dentro del archivo, se creó este método estático para que los métodos lo utilicen y así no tener un código muy grande y repetir las cosas en los métodos.

Este método al ser estático no lleva la palabra self es sus parámetros. Los parámetros que tiene son 2. el primero es la cadena en donde se tiene que buscar, y el segundo lo que se tiene que buscar en las cadenas. Al final retorna el número de apariciones que hay.

```
    @staticmethod
    def encuentra(cadena,conjunto):
            contador = 0
            for i in range(len(cadena)):
                    if cadena[i] in conjunto:
                            contador += 1
            #se retorna la cantidad de veces que aparece en la cadena
            return contador
```

Método muestra

Lo que hace es mostrar el contenido del archivo, para ello se usa un for que valla de line en línea del archivo, eso lo hace automáticamente en Python al colocar self.Archivo. AL imprimir se uso un .format

en el print, esto para poder darle estilo a la forma de escribir. Por ejemplo. Se uso {:3} para indicar que se empiece a escribir a partir del tercer espacio.

Al finalizar el for es importante de reiniciar el apuntador del archivo al inicio (0) esto para que cuando los demás métodos lo usen empiecen a leer desde el inicio, de lo contrario los demás métodos al intentar leerlo les dirá que no hay nada. Esto se hará al finalizar cada método.

```
def muestra(self):
        i = 1
        for linea in self.archivo:
                print("{:3}:{}".format(i,linea),end = "")
                i += 1
        #cada vez que se lee el archivo se tiene que pocici
        self.archivo.seek(0)
```

Método cuentaVocales, cuentaConsonantes, cuentaSignosPuntuacion, cuentaEspacios, cuentaMayusculas y cuentaMinusculas

Como se dijo al inicio estos métodos se parecen en algo (cuentan la cantidad de apariciones de algún símbolo en la cadena), lo único que los diferencia es lo que van a contar (los símbolos que buscan como las vocales, espacios etc.).

Se usa un for como el de mostrar, y en cada línea se manda a llamar a la función de encontrar con los parámetros de la línea que es la que leyeron del archivo y una cadena que contengan todos los elementos que quieren contar su aparición.

El conjunto para buscar (dependiente de cada caso) es de las letras en mayúsculas y minúsculas, así como con acento o sin acento, ya que, por ejemplo, una vocal puede ser a,A,Á o á, por lo que se toman todos los casos posibles.

El conjunto de los métodos es:

cuentaVocales: aeiouAEIOUáéíóúÁÉÍÓÚ.

cunetaConsonates: bcdfghjklmnñpqrstvwxyzBCDFGHJKLMNÑPQRSTVWXYZ y las que tienen acento ćǵḱĺḿńṕŕśẃýźĆǴḰĹḾŃṔŔŚẂÝŹ.

cuentaSignosPuntuacion: .:,;\'\"¿?!¡-ćǵḱĺḿńṕŕśẃýźĆǴḰĹḾŃṔŔŚẂÝŹáéíóúÁÉÍÓÚ. Para este caso se tomaron los signos de puntuación como las letras con signos de puntuación

cuentaEspacios: “ ”. Es un espacio en blanco.

cuentaMayusculas: ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁĆÉǴÍḰĹḾŃÓṔŔŚÚŰẂÝŹ.

cuentaMinusculas: abcdefghijklmnñopqrstuvwxyzáćéǵíḱĺḿńóṕŕśúűẃýź.

```
def cuentaVocales(self):
        contador = 0
        # con el for se lee linea por linea
        for linea in self.archivo:
                #para cada linea se busca las vocales com acento y sin acento y mayusculas y minusculas.
                contador += Archivo.encuentra(linea,"aeiouAEIOUáéíóúÁÉÍÓÚ")
        self.archivo.seek(0)
        return contador

def cuentaConsonantes(self):
        contador = 0
        for linea in (self.archivo):
                contador += Archivo.encuentra(linea,"bcdfghjklmnñpqrstvwxyzBCDFGHJKLMNÑPQRSTVWXYZćǵḱĺḿńṕŕśẃýźĆǴḰĹḾŃṔŔŚẂÝŹ
        self.archivo.seek(0)
        return contador

def cuentaSignosPuntuacion(self):
        contador = 0
        for linea in self.archivo:
                contador += Archivo.encuentra(linea,".:,;\'\"¿?!¡-ćǵḱĺḿńṕŕśẃýźĆǴḰĹḾŃṔŔŚẂÝŹáéíóúÁÉÍÓÚ")
        self.archivo.seek(0)
        return contador

def cuentaMayusculas(self):
        contador = 0
        for linea in self.archivo:
                contador += Archivo.encuentra(linea,"ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁĆÉǴÍḰĹḾŃÓṔŔŚÚẂÝŹ")
        self.archivo.seek(0)
        return contador

def cuentaMinusculas(self):
        contador = 0
        for linea in self.archivo:
                contador += Archivo.encuentra(linea,"abcdefghijklmnñopqrstuvwxyzáćéǵíḱĺḿńóṕŕśúẃýź")
        self.archivo.seek(0)
        return contador
```

Otra forma de hacer los métodos cuentaMayusculas y cuentaMinusculas con ASCII.

Otra forma de hacer estos metodos es utilizando los ASCII de cada símbolo. El ASCII de la a es 97 y de la z es de 122, así que para que sea letra solo debe de ser menor que 97 y mayor a 122, al igual que con mayúsculas, el ASCII de A es 65 y de la Z es122. Con esta comparación evitamos hacer todas las comparaciones que usa el método de encontrar, lo que lo hace mejor.

```
def cuentaMayusculas2(self):
        contador = 0
        for linea in self.archivo:
                for i in range(len(linea)):
                        if ord(linea[i]) >= 65 and ord(linea[i]) <= 90:
                                contador += 1
        self.archivo.seek(0)
        return contador

def cuentaMinusculas2(self):
        contador = 0
        for linea in self.archivo:
                for i in range(len(linea)):
                        if ord(linea[i]) >= 97 and ord(linea[i]) <= 122:
                                contador += 1
        self.archivo.seek(0)
        return contador
```

Método cuenta Palabras

En este método no podemos usar la función encontrar, ya que hay demasiadas palabras y formas de usarlas. Lo que hace es aumentar el contador después de haber recorrido todos los espacios en blanco que existan, ya que puede que haya más de un espacio en la cadena, así que no es lo mismo “que haces” que “qué haces”, ya que si contamos los espacios en blanco la primera oración tendrá 2 palabras mientras que la segunda 3.

```
def cuentaPalabras(self):
        palabras = 0
        for linea in self.archivo:
                i = 0
                if len(linea) != 0 and linea[0] != " ":
                        palabras += 1
                #se tiene que recorrer todos los espacios en blanco por qu
                while i < len(linea):
                        if linea[i] == " ":
                                while i < len(linea) and linea[i] == " ":
                                        i += 1
                                palabras += 1
                        i += 1
        self.archivo.seek(0)
        return palabras
```

Método cuentaLineas

En este método se usa el for que se explicó en el método muestra solo que cada por cada línea se incrementa un contador.

```
def cuentaLineas(self):
        contador = 0
        for linea in self.archivo:
                contador += 1
        self.archivo.seek(0)
        return contador
```

Método copiaArchivo

Para este método se usa el archivo que se creó en el constructor(copia.txt) en el que se puede escribir. Se hace un for que valla línea por línea el archivo a copiar y se escribe con la función write en el archivo copia. Al finalizar se cierra el archivo con close para evitar errores.

```
def copiaArchivo(self):
        try:
                #se usa la funcion write para escribir por lineas la copia del archivo
                for linea in self.archivo:
                        self.copia.write(linea)
                self.archivo.seek(0)
                #al terminar de usarse se cierran los archivos
                self.copia.close()
                print("Se creo una copia de su archivo, su nombre es copia.txt\nSu contenido es")
                for linea in self.archivo:
                        print("  ",linea,end = "")
                self.archivo.seek(0)
        except FileNotFoundError:
                print("No se puede crear el archivo ",nombre)
```

Método convierteAMayusculas y convierteAMinusculas

Para estos métodos se usan funciones que ya están Python. upper() convierte en mayúsculas una cadena y lower() las convierte a minúscula.

```
def convierteAMayusculas(self):
        print("La cadena convertida a mayusculas es ")
        for linea in self.archivo:
                #para convertir a mayusculas se usa la funcion upper
                print("   ",linea.upper(),end = "")
        self.archivo.seek(0)

def convierteAMinusculas(self):
        print("La cadena convertida a minusculas es ")
        for linea in self.archivo:
                #para convertir a minusculas se usa la funcion lower
                print("   ",linea.lower(),end  = "")
        self.archivo.seek(0)
```

Método muestraHexadeciamal

Para convertir a hexadecimal se convierte primero a ASCII y luego a hexadecimal. Para convertir a ASCII se utiliza la función ord() y a hexadecimal hex(). Se utilizan 2 for, uno es el que va línea por línea, y el otro va carácter por carácter de la línea.

```
def muestraHexadecimal(self):
        print("La cadena convertida a hexadecimal es ")
        for linea in self.archivo:
                print("   ",linea,end = "")
                for i in range(len(linea)):
                        #para mostrar a hexadecimal se usa la funcion hex y la funcion ord
                        print(hex(ord(linea[i])),end = "")
                print("")
        self.archivo.seek(0)
```