Ayudantía Programación

Trabajo de Textos y Archivos

Gonzalo Fernández

IWI-131 14/11/2017

Temas de la Ayudantía de Hoy

- Lectura y Escritura de Archivos
- Funciones de Trabajo de Textos
- Ejercicios

Trabajando Archivos

En Python, cualquier cosa que se desee realizar con un archivo, debe comenzar con la **apertura** de este mediante la función **open()**.

```
a = open("nombre_archivo", "letra")
```

Esta función pose 2 parámetros, el nombre del archivo, y, la acción a realizar.

- w: Write – Sobrescribe el archivo y permite escribir en él.
- r: Read – Abre el archivo y permite leerlo.
- a: Append – Abre el archivo y permite escribir desde el final de este.

Una vez terminado de usar el archivo, siempre hay que **CERRARLO**, con la función **close()**.

Lectura de Archivos

La lectura de archivos en Python se realiza haciendo uso de la letra “a” en la función **open**(), luego, se **itera** sobre esa variable, lo que equivale a leer el archivo línea por línea (no es necesario incluír siempre **“r”**, ya que si no se indica la letra, Python asume que es **“r”** por defecto.

```
archivo = open(“links_mangas_mega.txt”, “r”)
for i in archivo:
    print i
archivo.close()
```

Escritura de Archivos

Para escribir en un archivo, primero es necesario abrirlo con la opción adecuada, en este caso existen 2:

- Escribir un archivo nuevo (sobrescribirá el archivo en caso de existir):
  `file = open("packs_mega.txt", "w")`

- Escribir texto al final del archivo (no lo sobrescribe):
  `file = open("packs_mega.txt", "a")`

Una vez abierto el archivo, podemos ingresar texto en él mediante la función write():

```
file.write("https://mega.nz/#!gzIH1SgK!6yoBPT0Uki1EQw7AUMb")
file.close() # Recuerden siempre cerrar el archivo cuando
             terminan de usarlo!
```

Ejemplo de Lectura y Escritura

```
a = open("ayudantia1.txt")
b = ""
for i in a:
        if i[:5] == "Fecha":
                i = "Fecha: 14/11/2017 \n"
        b += i
a.close()

c = open("ayudantia2.txt", "w")
c.write(b)
c.close()

c = open("ayudantia2.txt", "a")
c.write("Cualquier parecido con la ayudantía anterior es mera coincidencia")
c.close()
```

Funciones de Trabajo de Texto

- texto.strip()
  - Borra los caracteres especiales de una cadena de texto, desde ambos lados, una vez que encuentre un carácter que no es especial, se detiene por el lado que está buscando.
  - Ejemplo:

    `"\t Holi \n".strip() => "Holi"`

- texto.split("separador")
  - Recorre una cadena de texto y lo separa cada vez que encuentra el "separador" en él, esta función retorna una lista de cada parte separada del string, además borra cada ocurrencia del separador.
  - Ejemplo:

    `"Ohayou Gozaimasu".split(" ") => ["Ohayou", "Gozaimasu"]`

Funciones de Trabajo de Texto

- texto.join(lista) # Lista de strings
  - Une una lista de strings agregando **texto** entre cada elemento
  - Ejemplo:

```
lista = [“Tan”, “Ga”, “Na”, “Ni”, “Ca”]
“-”.join(lista) => “Tan-Ga-Na-Ni-Ca”
```

- texto.replace(“cadena a reemplazar”, “cadena nueva”, **x**)
  - Reemplaza una cadena de texto por otra **x** veces, si no se indica **x**, reemplazará todas las ocurrencias.
  - Ejemplo:

“Holiwi”.replace(“iwi”, “a”) => “Hola”

Funciones de Trabajo de Texto

- map(función, lista)
  - Aplica una **función** a cada elemento de una **lista** y devuelve una lista con el resultado de aplicar la función a cada elemento.
  - Ejemplo:

    a = [“4”, “5”, “123”, “92138”]
    map(int, a) => [4, 5, 123, 92138]

- format()
  - Da formato a un texto basándose en indicadores específicos de este
  - Ejemplo:

    a = “Oli, soy {**0**}, tengo {**1**} años y {**2**}”
    a.format(“Karo”, “13”, “me encanta programar”)
    >>> “Oli, soy Karo, tengo 13 años y me encanta programar”

    a = “Oli, soy {**nombre**}, tengo {**edad**} años y {**hobby**}”
    a.format(nombre=“Karo”, edad=“13”, hobby=“me encanta programar”)
    >>> “Oli, soy Karo, tengo 13 años y me encanta programar”

Funciones de Trabajo de Texto

- texto.upper()
  - Devuelve una cadena de texto en mayúsculas
  - Ejemplo:
    a = “Omae wa, mou shindeiru”
    a.upper() => “OMAE WA, MOU SHINDEIRU”

- texto.lower()
  - Devuelve una cadena de texto en mayúsculas
  - Ejemplo:
    a = “NANI!?”
    a.lower() => “nani!?”

- texto.swapcase()
  - Intercambia las mayúsculas por minúsculas y viceversa
  - Ejemplo:
    a = “omae wa, MOU SHINDEIRU”
    a.swapcase() => “OMAE WA, mou shindeiru”

[20 %] Considere el archivo `cuentas.txt` que contiene la dirección de correo electrónico (*email*) de usuarios, sus contraseñas y las fechas de caducidad de las cuentas en formato `dd/mm/aaaa`. Los siguientes son **ejemplos**.

`cuentas.txt`

```
mharvey715@hotmail.com:swimming1:28/11/2016
mathwzrd@hotmail.com:buzzman:02/12/2016
manoela@hotmail.com:tutinho:30/11/2016
```

`vence.txt`

```
mharvey715@hotmail.com
manoela@hotmail.com
```

Ordenamiento

Es de suma urgencia ordenar (con la indentación respectiva) la función `prox_venc(cuentas, fecha)`, la que recibe el nombre del archivo de `cuentas` y una fecha en formato `mm/aaaa`, escribiendo en el archivo `vence.txt` el *email* de todos los usuarios cuyas cuentas caducarán en el mes y año indicado en la `fecha`.

```
>>> prox_venc('cuentas.txt', '11/2016')
>>>
```

```
arch.close()
f = datos[-1].split('/')
dest.close()
arch = open(cuentas)
dest = open('vence.txt', 'w')
datos = li.strip().split(':')
if mm == m and aa == a:
def prox_venc(cuentas, fecha):
for li in arch:
mm, aa = f[1], f[-1]
m, a = fecha.split('/')
dest.write(datos[0] + '\n')
```

Análisis de algoritmo

En el contexto del problema anterior, indique en palabras qué realiza la siguiente función:

```
def verificar(cuentas, mi):
    lista = list()
    arch = open(cuentas)
    for li in arch:
        da = li.strip().split(':')
        safe = False
        for i in range(10):
            if str(i) in da[1]:
                safe = True
        if safe and len(da[1]) >= mi:
            lista.append(da[0])
    arch.close()
    return lista
```

2. [40 %] Dado el éxito que ha tenido el popular juego pokémon GO, la UTFSM no ha querido quedarse atrás y le ha solicitado el desarrollo del juego pukamon GOES (el cual no es una copia del pokémon).

El juego funciona utilizando el GPS del celular para ubicar al jugador. La información de los pukamones se almacena en un archivo, donde cada línea tiene el formato `nom_pukamon;coord_x;coord_y`. El archivo `pukamones.txt` es un **ejemplo** de lo anteriormente expuesto.

A usted le ha tocado desarrollar el sistema que lista los pukamones cercanos al jugador, para ello debe desarrollar las siguientes funciones:

`pukamones.txt`

```
Pukachu1;40.02;-74.03
Bulmasaur;41.5;-74.0
NewTwo;50.3;-73.2
Pukachu;40.0;-30.0
Escuartul;40.4;-74.6
```

a) Desarrolle la función `leer_pukamones(nomarch)` la que recibe el nombre del archivo con la información de los pukamones. La función retorna una lista de tuplas, donde cada tupla tiene la siguiente estructura: `(nombre_pukamon, coordenadaX, coordenadaY)`.

Importante: usted debe respetar los tipos de datos del ejemplo.

```
>>> leer_pukamones('pukamones.txt')
[('Pukachu1', 40.02, -74.03), ('Bulmasaur', 41.5, -74.0), ('NewTwo',
   50.3, -73.2), ('Pukachu', 40.0, -30.0), ('Escuartul', 40.4, -74.6)]
```

b) Desarrolle la función `calcular_cercania(pos, radio, archivo)` la que recibe una tupla (`pos`) con la posición del jugador, un entero que corresponde al `radio` del radar pukamon y el nombre del archivo donde se encuentran todos los pukamones. La función retorna un diccionario cuya llave es el grado de cercanía y el valor una lista con los nombres de los pukamones que se encuentran en dicho grado. Si el pukamon se encuentra a una distancia menor al `radio` del radar, entonces el grado de cercanía es 1. Si el pukamon se encuentra a una distancia menor a dos veces el `radio` del radar, el grado es 2 y en cualquier otro caso el grado es 3.

```
>>> calcular_cercania((39, -74), 2, 'pukamones.txt')
{1: ['Pukachu1', 'Escuartul'], 2: ['Bulmasaur'], 3: ['NewTwo', 'Pukachu
   ']}
```

c) Desarrolle la función `itinerario_pukamones(pos, radio, archivo, final)` la que recibe la posición del jugador (`pos`), el `radio` del radar pukamon, el nombre del archivo con las posiciones de los pukamones y el nombre de un archivo a crear (`final`). La función debe crear el archivo indicado, con el contenido mostrado en el ejemplo (*'result.txt'*).

```
>>> itinerario_pukamones((39,-74), 2, 'pukamones.txt', 'result.txt')
>>>
```

result.txt

```
Itinerario a seguir
Pukamones encontrados a 1 grados de cercania
Pukachu1 - Escuartul

Pukamones encontrados a 2 grados de cercania
Bulmasaur

Pukamones encontrados a 3 grados de cercania
NewTwo - Pukachu
```

[40 %] La Universidad de Molterra tiene un avanzado sistema de control de sus estudiantes.

Para llevar control sobre las notas, cada alumno tiene un archivo cuyo nombre es su rol más la extensión `.txt`. La primera línea del archivo dice "Resumen academico", luego una línea en blanco y a continuación por cada nueva línea se lista cada uno de los cursos con nota. Los cursos deben estar ordenados según el semestre al que corresponden por malla y los cursos de un mismo semestre pueden estar en cualquier orden. Finalmente hay una línea que muestra el promedio de las notas en los cursos. A la derecha se muestra un **ejemplo** de archivo.
Para actualizar las notas un profesor indica el nombre de archivo donde puso las notas finales. Los 6 primeros caracteres del nombre del archivo indican la sigla del curso.

```
Resumen Academico

01 MAT021 56
01 IWG101 98
01 FIS100 64
01 QUI010 79
02 FIS110 60
02 IWI131 90
03 FIS120 40

PROMEDIO 69.57
```

Por ejemplo, para programación (código: IWI131) un nombre de archivo podría ser `IWI131201602.txt`. Cada línea del archivo contiene el rol de un alumno, el semestre al que corresponde el curso según su malla curricular y la nota que obtuvo. Los datos están separados por un espacio en blanco. Por ejemplo, una persona que está tomando el curso por malla en este semestre tendría una línea del tipo `201610001-k 02 90`, donde el `02` indica segundo semestre según su malla y el `90` es la nota que obtuvo.

Escriba un programa que pregunte al usuario (profesor) el nombre del archivo desde el cual leer notas, actualice el resumen académico de los respectivos alumnos en sus archivos y repita todo este proceso hasta que el nombre de archivo ingresado sea `0`. Notar que al actualizar un Resumen Académico el curso puede o no haber estado ingresado previamente, de estarlo, debe reemplazar la nota por la nueva. En cualquier caso, se debe actualizar el promedio general en la última línea del archivo.

Nota: puede utilizar la función `os.rename(nom_arch1, nom_arch2)`, la que renombra el archivo llamado `nom_arch1` con el nombre `nom_arch2`. Ambos parámetros son strings.

Restricción: Está prohibido el uso de cualquier función de ordenamiento usando alguna estructura de datos.