Análisis de Datos y Toma de Decisiones en Computación
Facultad de Ingeniería de Sistemas Computacionales
Grupo 1IL-131
1er Semestre 2024

Laboratorio #2 – Python Esencial

Asignada: 6 de abril de 2024
Fecha de Entrega: 14 de abril de 2024 (11:59pm).

Nota: El laboratorio puede ser enviado en grupos de *máximo 3 personas* a través de la plataforma TEAMS (en un archivo .zip si son múltiples archivos). Provea sus respuestas en formato digital (formato Word o en código Python, o en un Python notebook, formato .ipynb). En cuanto sea posible, sea claro con sus respuestas, e identifique su código en la celda inicial con su nombre y cedula.

Parte I. Python Esencial

Problema 1. Lectura de argumentos del usuario en Python (40 puntos).

1. Escriba un script llamado *nparesab.py* que reciba dos argumentos desde la línea de comandos, usando *input()*. El primer parámetro será nuestro número inferior y el segundo será nuestro número superior.
    - Primero verifique que los argumentos pasados sean números enteros, luego verifique que el número superior sea mayor que el número inferior.
    - Escriba una tenga una función llamada *pares* que reciba el número inferior y el número superior y devuelva una lista de los números pares entre inferior y superior inclusive.

2. Escribe un script llamado *funcionesvarias.py* que tenga las siguientes funciones:
    a. Una función llamada *cadenareversa()* que reciba como argumento una cadena (lista) y devolverá el reverso de esa cadena (lista).
    b. Una función llamada *combinalista()* que toma dos listas de números como entradas y devuelve una sola lista con los valores de ambas listas manera ordenada.

    Sugerencia: Lea los parámetros desde la línea de comandos o desde entrada del usuario usando *input*.

3. Escriba un script llamado *verifica_anagrama.py* que tomará dos cadenas y devolverá *True* si son anagramas de cada una otro y *False* de lo contrario. Por ejemplo, si tenemos dos palabras '*paso'* y *'sopa'* esto se evaluaría a *True*, ya que uno puede formar una palabra mezclando las letras en el otra.

    Sugerencia: Lea las cadenas como entrada del usuario usando *input*. Puede ser el comando *in* que vimos en clase.

4. Escriba un script llamado *enromano.py* que tome un número del usuario entre 1 y 1000 (verifique que el numero esta en este rango) y lo transforme a su notación en números romanos. Recuerdo que los números romanos se simbolizan con los siguientes caracteres:

Numero en Arábigo	Numero En Romano
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

    Sugerencia: Lea las cadenas como entrada del usuario usando *input*. No utilice ninguna librería estándar de Python, utilice ciclos recursivos y manejo de cadenas para realizar el problema. Recordar que para representar un número menor al número en romano, por ejemplo, el numero 46, se concatena desde la izquierda así: XLVI

Problema 2. Juego de adivinanzas de números aleatorios (15 puntos). En el módulo *random* existe una función llamada *randint*, *random.randint(x, y),* esta función devuelve un numero entero aleatorio mayor que o igual a *x*, y menos que o igual a *y*.

Escriba un script llamado *AdivinaRandom.py* que: 1) elige un número aleatorio entre 1 y 99 (1 y 99 inclusive), 2) le pregunta al usuario la cantidad de veces que quiere adivinar, a continuación, procesar cada de los intentos con *input.*

La secuencia de comandos debe ser como sigue: imprimir "¡*increíble, su suposición fue correcta! NUM es el número correcto!* y detener la ejecución del script si el argumento es igual al número aleatorio elegido, e imprimir "*Su suposición es falsa. Por favor, inténtelo de nuevo, usted tiene X turnos restantes para averiguar el numero correcto* " y continuar pidiendo otra suposición. Si el usuario agota sus oportunidades, se debe imprimir, "Usted agotado sus XX conjeturas. El número real era NUM ".

En su última línea, se imprime el número real elegido. Tenga en cuenta que tendrá que importar el módulo *random*. También puede llamar a *random.seed(43)* antes de llamar a *random.randint(),* para que el número aleatorio elegido será el mismo cada vez que se ejecuta el script. Una corrida de ejemplo se vería como sigue:

$Python *AdivinaRandom.py*
¿Cuántas veces usted quiere jugar hoy? 3

¿Cuál cree usted que es el numero? 15
Su suposición es falsa. Por favor, inténtelo de nuevo; Tiene 2 turnos restantes para averiguar.
¿ Cuál cree usted que es el numero? 33
Su suposición es falsa. Por favor, inténtelo de nuevo; Tiene 1 turnos restantes para averiguar.
¿ Cuál cree usted que es el numero? 77
"Increíble, su suposición era correcta! 77 fue el número correcto! "

Otra opción seria si el usuario no llega a obtener la respuesta:
¿ Cuál cree usted que es el numero? 30
Su suposición es falsa. Por favor, inténtelo de nuevo; Tiene 1 turno restantes para averiguar.
¿ Cuál cree usted que es el numero (final)? 9
Usted agotó sus 5 intentos. El número real era 77.

Sugerencia: haga uso de *input*, formato de texto, y los comandos *if*, *else*, *elif*, *break*, *continue* y *pass* para romper los bucles.

Parte II. Python Manejo de Archivos, Funciones

Problema 3. Filtrado de Texto con CSV (30 puntos). Para este ejercicio usarás el archivo *dresses.csv*, el archivo esta delimitado por comas. Es una base de datos de vestidos ordenados por precios de estilo y recomendación.

Escribamos un script que lea el archivo utilizando el módulo CSV Reader y realice las siguientes operaciones:

a. Obtenga todos los vestidos que tiene una puntuación > 0, a continuación, e imprima la calificación promedio de estos vestidos.

b. Basándose en la columna *Recomendation*, imprima el *Dress ID* que se ha recomendado (Recomendación = 1). Para aquellos vestidos que no fueron recomendados (Recomendación = 0), averigüe cuántos tenían un tipo de patrón de animales (animal pattern).

c. Imprima toda la información para todos los vestidos de verano que tengan un *Style* del tipo “sexy”.

d. Utilizando la columna *Price*, cree un diccionario que contenga todos los *Dress ID* para cada uno de los valores “Bajo”, “Alto”, “Promedio” y “Medio”. Para cada llave, imprima los códigos asociados y el número total de códigos asociados para cada una de las llaves.

Problema 4. Manejo de Archivos JSON y Conexión a APIs (30 puntos).

En este problema utilizaremos los módulos *datetime*, *json* y *requests*. Las dos primeras liberarías son para manejo de tiempo y para manejar archivos JSON respectivamente. *Requests* es una librería que se utiliza para hacer peticiones a servidores HTTP, FTP y APIs REST/SOAP (en el caso de que no esté instalada en su sistema la puede instalar con el comando $*pip install requests* en su Terminal).

Se requiere que realice un script que realice las siguientes funciones:

- Tenga una función llamada *descargajson()* que reciba como parámetro una dirección web.
    - En este problema haremos una consulta al API REST de Github, de modo que la dirección web nos devolverá a una lista de los *commits* hechos al repositorio de Github de un proyecto de software.
    - Por ejemplo, su función debe recibir y procesar la información de los siguientes proyectos (sus repositorios):
        - D3 – Data Driven Documents (https://api.github.com/repos/d3/d3/stats/commit_activity)
        - Tensorflow (https://api.github.com/repos/tensorflow/tensorflow/stats/commit_activity)
        - *Usted elija un tercer proyecto que puede ser de su preferencia.*

- Esta función debe salvar el contenido de la página utilizando *requests.get(),* verifique que la petición le da un código de *status 200* y proceda salvarlo en un archivo JSON.

- Sobre el archivo json, verifique que la estructura de las entradas sean 3 campos “days”, “total” y “week”

- Se requiere que verifique:
    - El día de la semana (lunes a domingo) se dan más *commits*.
    - La semana en que se han dado mayor número de commits. Recuerde que el campo “week” está dado en tiempo “timestamp” de UNIX de modo que tiene que convertirlo a tiempo “normal” utilizando la función *fromtimestamp()*

Verifique si el número de *commits* que usted calculo es el mismo que se genera en la página de *activity* del proyecto usted decidió investigar, por ejemplo, en: https://github.com/flutter/flutter/graphs/commit-activity. HAGA UNA CAPTURA DE PANTALLA y la comenta.