INF-253 Lenguajes de Programación
Tarea 1: Python

Profesor: José Luis Martí, Jorge Díaz Matte, Rodrigo Salas Fuentes
Ayudante Cátedras: Gonzalo Severín Muñoz, Jhossep Martínez Velásquez
Ayudante Coordinador: Álvaro Rojas Valenzuela
Ayudante Tareas: Ricardo Barrida Vera, Bryan González Ramírez, Bastían Salomón Ávalos, Cristian Tapia Llantén, Cristóbal Tirado Morales

1 de abril de 2024

1. Cadena de Producción

El registro de empresas y producción (R.E.P) de Pythonia es una importante entidad que administra la información de las empresas de la nación. En esta ocasión, necesitan rescatar datos de la red de producción entre las empresas industriales. Pero se enfrenta a un problema de escala, entre más fábricas agregan, más compleja crece la red. Siendo imposible completarse a mano. Es por eso que se acercan a usted.

Se le piden crear un programa en *Python3* junto con la librería RegEx que automatice la creación de la red de producción y se puedan realizar consultas sobre las empresas y la cadena de producción.

2. La Red de Producción

La Red de Producción es un lenguaje que permite generar un grafo dirigido, donde los nodos poseen información sobre las empresas, es decir: Nombre, Rol Único Tributario y una lista de producción. Mientras que sus conexiones son a cuáles empresas les vende sus productos.

2.1. Empresas

Las empresas, representados por los nodos, entregan información sobre su nombre, Rut y lista de producción separados por un punto, usando un formato estándar proporcionado por la R.E.P.

- **Nombre**: El nombre de la empresa debe comenzar con mayúscula. Seguido por alguna combinación de letras, números, espacios o guiones (−).
  - *Correcto:* Empresa-sana-10
  - *Incorrecto:* vicuzeta

  Nota El nombre debe ser único para el funcionamiento de las búsquedas por nombre.

- **Rut**: El Rut de la empresa es único para cada uno. El primer número siempre debe ser entre 1 y 9. Seguido por otros números del 0 al 9. Además, los últimos dos caracteres son un guion, seguido por un dígito o k. Por último, debe ser válido, descrito en la página: https://es.wikipedia.org/wiki/Rol Único Tributario
  - *Correcto:* 11985602-7
  - *Incorrecto:* 01121-1

  Nota: Un Rut inválido se considera como error de sintaxis.

- **Lista de Productos**: Es una lista de productos que la empresa ofrece. La lista debe venir encerrado por corchetes ([‘*producto*’]) y cada elemento separados por una coma.

- **Productos**: Cada producto posee tres características, cada una separada por un guion (−):
  - **ID**: Inicia con un símbolo ‘#’, seguido por una combinación de letras en mayúsculas y dígitos del 0 al 9, No puede iniciar con un 0.
    Nota: Puede haber ID repetidos.
  - **Precio**: Inicia con un símbolo ‘$’, seguido por un número sin ceros al inicio y separados en grupos de a tres por un punto de derecha a izquierda. También incluye el 0 como un único número.
  - **Nombre**: El nombre debe empezar por alguna letra (solo minúsculas) o por un número, seguido por cualquier combinación de letras, números, guiones y puntos.

- **Ejemplo:**
  - *Correcto:* Empresa A.24146524-1.[#A0B-$0-tierra-version-1.0,#AB96-$9.999.999-tierra-version-2.0]
  - *Correcto:* Empresa-sospechosa 10.9220781-1.[#8CASD-$200.000-plastico-falso-2.0]

- *Incorrecto:* piramidades-9999.111-k.[#mc90-$10.56-estafazan.01]
  (Se puede ver que el nombre empieza con minúscula, el Rut es invalido, la ID está en minúsucla y el precio debería ser $1.056)

Todo lo anterior se puede ver en la siguiente EBNF:

```
empresa ::= nombre_empresa '.' rut '.' lista_produccion

nombre_empresa ::= (A-Z) {(a-z | A-Z | digito_o_cero | '-' | ' '}

rut ::= digito {digito_o_cero} '-' (digito_o_cero | 'k')

lista_produccion ::= '[' producto {',' producto} ']'

producto ::= id '-' precio '-' nombre_producto

id ::= '#' (A-Z | digito) {(A-Z | digito_o_cero)}

precio ::= '$' (digito (digito_o_cero | digito_o_cero digito_o_cero | {'.'
    digito_o_cero digito_o_cero digito_o_cero}) {'.' digito_o_cero digito_o_cero
    digito_o_cero}) | '0'

nombre_producto ::= (a-z | digito_o_cero) {a-z | digito_o_cero |'.' | '-'}

digito_o_cero ::= '0' | digito

digito ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

2.2. Linea de Producción

Para construir la red, se a recopilado información de la siguiente forma (Nombre A.Producto A -> Nombre B.x Unidades). Permitiendo decir que la empresa de nombre "A" les vende el producto "A" a la empresa de nombre "B" una cantidad de X unidades representado por un número decimal, describiendo una conexión en la red final. Además también se pueden identificar la empresa por su Rut y el producto por su ID. Todo lo anterior se puede ver en la siguiente EBNF:

```
venta ::= '(' (nombre_empresa | rut ) '.' (nombre_producto | ID) '->' (
    nombre_empresa | rut) '.x' (numero_decimal) ')'
numero_decimal ::= (digito {digito_o_cero} ('.' {digito_o_cero} digito | '')) |
    ('0.' {digito_o_cero} digito)
```

Un ejemplo de una sintaxis correcta sería:

```
(20464360-1.terra.nova->Plantas INC.x5.2)
```

Esto representaría que la empresa de Rut 20464360-1 le vende terra.nova a Plantas.INC 5.2 unidades de terra.nova.

Cabe resaltar que una empresa no puede venderse a si mismo y el resultado final de la red debe ser un grafo dirigido sin ciclos. (Lo mas cercano es a una grafo de fluidos, pero con la posibilidad de múltiples entradas y salidas)

3. Comandos y Consultas en la Red

El R.E.P. te pide además programar las funciones de consultas para la Red de producción. Cada comando son para construir la red o permiten realizar consultas sobre la red. Estos comandos siempre terminan con un salto de línea. Los posibles comandos que se pueden realizar serán los siguientes:

- *empresa*: crea un nodo representando la empresa.
  Si al ingresar una empresa, algún nombre de empresa, rut o Id de productos ya se encuentra en la red, este debe ser ignorado y se debe escribir "nombre/rut/Id ya se encuentra en la linea de producción"

- *venta*: crea una conexión entre las empresas visto anteriormente.
  Si alguna de las dos empresa o producto no existe en momento de ejecutar, se debe escribir "No se pudo crear conexión, no existe tal nombre/Rut/producto".
  Además, si la venta permite la existencia de un **ciclo** en la red, este debe ser ignorado y escribir "La venta *venta* genera un ciclo".

  Nota: Se concidera una venta repetida en el caso de que una empresa intente vender el mismo producto dos veces a la misma empresa. Este debe ser ignorado y son libre de escribir algo o no en el archivo de salida.

- *ver_empresa nombre_empresa o rut*: Se puede buscar tanto por nombre o por rut de la empresa, y se debe escribir en un archivo de salida los datos de la empresa con el siguiente formato.

```
VER EMPRESA:
- nombre_empresa
- rut
- [
producto1,
producto2,
...
]
```

  En caso de no existir tal empresa, se debe escribir "No existe la empresa con el nombre/Rut dado".

- *ver_ventas nombre_empresa o rut*: Se puede buscar tanto por nombre o por rut de la empresa. Se debe calcular el **precio final**, siendo la multiplicación del producto por la cantidad a que se le vende redondeado hacia arriba. Se debe escribir en un archivo de salida con el siguiente formato:

```
VER VENTAS:
- nombre_empresa
- rut
- [
producto1 cantidad1 precio_fianl1 -> nombre_empresa_compradora1,
producto2 cantidad2 precio_fianl2 -> nombre_empresa_compradora2,
...
]
```

Producto, cantidad, precio_final llevan la misma sintaxis que las EBNF *producto,* x*(numero_decimal)* y *precio* respectivamente.
En caso de no existir tal empresa, se debe escribir "No existe la empresa con el nombre/Rut dado".

- *buscar_MP nombre_empresa o rut*: Se define *MP* como la materia prima, es decir aquellas empresas que son la primeras en proveer a la linea de producción. Por ejemplo, Si A les provee a B, y B les provee a C y D. Además si F les provee a D. Significa que *buscar_MP D* debería de mostrar los productos de A y de F participantes en la cadena. Mientras que *buscar_MP C* debería salir solo aquellos de A. Esta información se debe escribir en un archivo de salida con el siguiente formato:

```
BUSCAR MP:
- nombre_empresa, rut
- [
productoMP1 nombre_empresa_fabricante1,
productoMP2 nombre_empresa_fabricante2,
...
]
```

En caso de no existir tal empresa se debe escribir "No existe la empresa con el nombre/Rut dado". Si ya es productor de materia prima, se debe escribir "La empresa nombre/Rut ya es productor de MP".

4. Ejecución del Programa

La entrada del programa será ingresado por un archivo llamado *input.txt* y la salida por otro archivo llamado *output.txt*. No se espera que las entradas estén bien escritas. En ese caso, todo comando que no tenga una sintaxis correcta, debe ser escrito en otro archivo de texto con nombre *errores.txt*. Si no hay comandos mal escritos, *errores.txt* debe estar vacío.
Se puede ver el siguiente ejemplo:

Input.txt

```
Manufacter-1.9141434-1.[#5AB-$50.301-circuitos-verdes,#78C-$6.320-chips-md60]
Manufacter-2.18683024-5.[#78Z-$9.990-circuitos-rojos-ver2.0]
(9141434-1.#5AB->Manufacter-2.x5.6)
(18683024-5.#78Z->Manufacter-1.x0.8)
hola
Manufacter-3.20012447-2.[#79TR-$900-plastico,#9CTR-$800.690-envases-plasticos]
M 4.8688823-8.[#620C-$891.001-centellas]
M5.19001277-8.[#901A-$600-estrellitas]
M-6.24932356-k.[#YIPI1-$9.999.990-cubo-rubik]
M-7-0.12893004-3.[#1IPIY-$1-rubic-1x1]
M-8-0.12893004-3.[#CR2-$82-rubic-12x12]
m9-0.22485756-k.[#MAT1-$666.001-rubic-1x1]
(Manufacter-3.plastico->Manufacter-2.x4)
```

```
(18683024-5.#78Z->M 4.x1)
(18683024-5.#78Z->M5.x2)
(Manufacter-1.chips-md60->M-6.x9)
(M-6.cubo-rubik->12893004-3.x0.2)
(M-8.#CR2->22485756-k.x1)
(M9.#MAT1->M 4.x1.0)
ver_empresa M5
ver_empresa 20012447-2
buscar_MP M 4
buscar_MP M-7-0
buscar_MP 9141434-1
VER EMPRESA Manufacter-3
```

Output.txt

```
La venta (18683024-5.#78Z->Manufacter-1.x0.8) genera un ciclo
12893004-3 ya se encuentra en la linea de produccion
No se pudo crear conexion, no existe tal M-8
VER EMPRESA:
- M5
- 19001277-8
- [
#901A-$600-estrellitas
]
VER EMPRESA:
- Manufacter-3
- 20012447-2
- [
#79TR-$900-plastico
#9CTR-$800.690-envases-plasticos
]
BUSCAR MP:
- M 4, 8688823-8
- [
#79TR-$900-plastico Manufacter-3,
#5AB-$50.301-circuitos-verdes Manufacter-1
]
BUSCAR MP:
- M-7-0, 12893004-3
- [
#78C-$6.320-chips-md60 Manufacter-1
]
La empresa 9141434-1 ya es productor de MP
```

Errores.txt

```
hola
m9-0.22485756-k.[#1IPIY-$666.001-rubic-1x1]
(M9.#MAT1->M 4.x1.0)
VER EMPRESA Manufacter-3
```

5. Datos Relevantes

- Al imprimir listas de productos, no importa el ordern.
- Al escribir en el archivo *errores.txt* los comandos incorrectos, si importa el orden, esto debido a que los comandos se agregan en tiempo de ejecución.
- Al no hacer uso de expresiones regulares, **la tarea no será revisada**.
- **Toda consulta se deben realizar en el foro en la sección Tareas, disponible en Aula**
- No se permite el uso de otras librerías para la construcción de grafos.

Nota para el lector: Se agregaron las notas en algunas partes del enunciado para aclarar algunos casos ambiguos después de la publicación inicial de la tarea. Disculpa por las molestias que pudieron haber generado.

6. Sobre la Entrega

- Se deberá entregar un programa en Python 3 llamado *red.py*, este es el archivo a ejecutar.
- Se puede agregar otros archivos *.py*.
- De no existir orden en el código, se realizaran descuentos.
- Las funciones implementadas deben ser comentadas de la siguiente forma. **Habrán descuentos por función no comentada**.

```
def fun(para1, para2, ...):
'''
***
Parametro 1 : Tipo
Parametro 2 : Tipo
...
***
Tipo de Retorno o None
***
Breve descripción de la función y el retorno
'''
```

- El trabajo es individual obligatoriamente.
- **La entrega debe realizarse en un archivo comprimido en tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz**.
- **El archivo README.txt debe contener nombre y rol del alumno, e instrucciones detalladas para la correcta utilización**.
- La entrega será vía aula y el plazo máximo de entrega es hasta el **05 de abril a las 23:55 hora aula**.

- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro la primera hora).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

7. Clasificación

7.1. Entrega

Para la clasificación de su tarea, sebe realizar una entrega con requerimientos mínimos que otorgarán 30 pts base, luego se entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

7.1.1. Entrega Mínima

- Crea diferentes expresiones regulares y las utiliza de manera correcta para obtener la información toda la de las empresas y sus ventas, aprovechándose de la modularización generada. *Nota: No se permite el uso de string.split()*
- El programa es capaz de generar los nodos y las conexiones correspondientes, con la posibilidad de recorrer la red de producción del problema descrito. Esto incluye:
  - Alguna estructura de datos ordenada que contenga las conexiones de la red. (Una matriz de adyacencia u otro).
  - Alguna estructura de datos que almacene todos los datos de las empresas y sus ventas, y que es sea posible ser recorrido a partir del punto anterior.

7.1.2. Entrega

- Ejecución del Programa (total 50 pts):
  1. El programa escribe en "*output.txt*" el resultado de las consultas, según el formato establecido (Max 10 pts).
  2. El programa salta las *empresas* y *ventas* que se encuentren repetidas, ausentes o generan un ciclo según las condiciones establecidas (Max 10 pts).
  3. El programa ejecuta "*ver_empresa*" encontrando el nodo correspondiente, escribiendo la información pedida y detecta cuando no existe tal empresa en la red (Max 10 pts).
  4. El programa ejecuta "*ver_ventas*" encontrando el nodo correspondiente, entregando la información correcta y detecta cuando este no existe en la red (Max 10 pts).
  5. El programa ejecuta "*buscar_MP*" encontrando el nodo correspondiente, muestra aquellos *productos* y *empresas* con las condiciones establecidos y detecta cuando este no existe en la red (Max 10 pts).
- Detección de Errores (total 20 pts):
  1. El programa no detecta ningún tipo de error (Max 0 pts).
  2. El programa detecta errores generales de sintaxis, por ejemplo al escribir mal un nombre de una empresa (Max 10 pts).

3. El programa detecta todos los errores de sintaxis para todos los tipos de comandos (Max 5 pts).
4. El programa reporta todos los errores detectados según el formato establecido (Max 5 pts)

- **Nota**: Puede existir puntaje parcial.

7.2. Descuentos

- Falta de comentarios (-10 pts c/u, Max 30 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el REAMDE (-5 pts c/u)
- Falta de orden (-20 pts)
- Día de atraso (-20 pts por día, -10 dentro de la primera hora)
- Mal nombre en algún archivo entregado (-5 pts c/u)

7.3. Re-corrección

Una vez publicada las notas de esta tarea en Aula, todas las personas tendrán derecho para una recorreción de su nota. Para esto, se coordinará con los ayudantes de tareas una semana para que el ayudante, junto a usted, revisará los puntos a discutir.
Más información se publicará en aula una vez coordinado con los ayudantes.