INF-253 Lenguajes de Programación
Tarea 1 2025-2 : Juez de Freestyle

Profesores: José Luis Martí Lara, Wladimir Ormazábal Orellana, Ricardo Salas Letelier
Ayudante coordinador: Bastián Ríos Lastarria
Ayudantes: Diego Duarte, Andrés Jablonca, Cristobal Tirado,
Martin Pardo, Carlos Bravo, Martín Palacios, Javier Alarcon,
Blas Olivares, Sofía Ramirez, Benjamín Echeverria

6 de agosto de 2025

Índice

1. Puntos a Evaluar y Objetivos de aprendizaje

1.1. Objetivo principal

Desarrollar un **Juez de Rimas** en `Python` que, empleando únicamente expresiones regulares y lógica propia, evalúe estrofas de cuatro versos y asigne una nota final entre 0 y 10.

1.2. Ítems de corrección

1. **Detección de rima**: consonante, asonante, misma terminación, gemela, etc.
2. **Detección de esquema**: Una estrofa es válida si y solo si no existen símbolos no pertenecientes al lenguaje tradicional español, La estrofa tiene exactamente 4 versos y los versos respetan algún tipo de rima entre ellos.
3. **Asignación de puntaje**: aplicación correcta del puntaje y escalado lineal al rango 0–10.
4. **Verificación de formato**: rechazo de símbolos prohibidos y ausencia total del patrón `\w` en el código.

2. Contexto

La prestigiosa organización musical internacional **BlueBull** es conocida por sus espectaculares batallas de freestyle, eventos donde miles de aficionados asisten para ver a sus raperos favoritos participar. Sin embargo, la empresa atraviesa una profunda crisis: los aficionados reclaman que los jueces humanos son cada vez más parciales y sus veredictos dependen más del carisma que de la calidad real de las rimas. Las polémicas han puesto a BlueBull al borde de la ruina.
En un intento desesperado por recuperar la credibilidad del torneo, la directiva de BlueBull firma un convenio con la **Universidad Técnica Federico Santa María (UTFSM)** para recuperar sus días de gloria. Tu misión, como estudiante del curso de Lenguajes de Programación, es desarrollar un programa que actúe como juez imparcial: un algoritmo capaz de puntuar estrofas de cuatro versos con criterios claros y repetibles. ¿Estás listo para salvar esta compañia de la quiebra?

3. Descripción de la tarea

Se creará un programa en Python llamado 'Juez.py' el cual recibirá de entrada un archivo 'estrofas.txt' y generará un archivo de salida 'decision.txt'.

3.1. Criterios de evaluación del juez

Se puntúa cada estrofa según las rimas de cada pareja de versos mas un posible bonus por utilizar una palabra clave (entregadas al principio del archivo de entrada).

3.1.1. Tipos de rima y puntaje por pareja (máx. 8 pts)

- **Consonante**: mismo sufijo con vocales y consonantes.
  - coincidencia de 3 a 4 letras: **5 pts**
  - coincidencia de 5 letras o más: **8 pts**

- **Asonante**: sólo vocales coinciden.
  - 1 vocal final igual: **3 pts**
  - 2 vocales finales iguales: **4 pts**
  - 3 vocales finales iguales o más: **8 pts**
- **Misma terminación**: comparten 2 o más letras finales pero no cumple con asonante ni consonante: **2 pt**.
- **Gemela**: palabra final idéntica: **1 pt**.
- **Sin rima**: **0 pt**.

Hay seis parejas distintas en una estrofa de cuatro versos; por tanto, el *puntaje bruto máximo por rimas* es **48 pts** (6×8).

3.1.2. Esquema global

«El esquema es válido si todos los versos comparten al menos algún tipo de rima entre ellos (consonante, asonante, misma terminación o gemela). Se resta 2 pts al puntaje bruto en caso de algún verso que no logre rimar con nada.» La penalización se aplica una sola vez sobre el puntaje bruto de la estrofa.

3.1.3. Bonus por palabra clave

Se define una lista de palabras clave en el programa a partir de la primera línea del archivo de entrada. Si *exactamente una* de ellas aparece y forma parte de la rima, se añaden **+2 pts** (máximo 2 pts).

Máximo teórico: 48 (rimas) + 2 (bonus) = **50 pts**.
El puntaje final se calcula como $PuntajeObtenido/5$ redondeado a un decimal.

3.2. Verificación de símbolos válidos

Cada verso solo puede contener:

- letras españolas (mayúsculas/minúsculas, con tildes);
- espacios simples;
- signos «.», «,», «¿», «¡», «!», «?», etc.

Cualquier otro carácter invalida la estrofa. (ver EBNF)

3.3. Entrada

- El programa debe leer un archivo llamado '`estrofas.txt`' ubicado en el mismo directorio de ejecución.
- El archivo de entrada contiene **una o más estrofas**, cada una de **exactamente cuatro líneas**. Las estrofas se separan con *una* línea en blanco.
- La primera palabra del archivo 'estrofas.txt' SIEMPRE será la o las palabra(s) bonus, se separan con una coma (,)

Ejemplo de estructura

```
<palabra-bonus>        ← línea 1 (obligatoria, palabra o lista de palabras separadas por ",")
                       ← 1 línea en blanco
V1                     ← verso 1
V2
V3
V4                     ← verso 4
                       ← 1 línea en blanco
V1
V2
V3
V4
...
```

3.4. Salida

El programa debe generar un archivo llamado '`decision.txt`' en el mismo directorio de ejecución. Cada línea del archivo corresponde a una estrofa, entregando el número de estrofa, la evaluación de ésta, si es que obtuvo el bonus por palabra clave y las rimas que incluye, con el siguiente formato:

```
Estrofa <N>: <puntaje>/10 (BONUS)
Rimas: <Rimas>
```

donde `<puntaje>` es el puntaje entero ya escalado a 0–10, `<Rimas>` son las rimas que fueron detectadas (consonante, asonante, etc) y <N>el número de la estrofa evaluada.

Ejemplo de `decision.txt`

```
Estrofa 1: 8.8/10
Rimas: consonante, asonante
Estrofa 2: 7.1/10
Rimas: misma terminación, consonante, gemela
Estrofa 3: 9.5/10 (BONUS)
Rimas: consonante
```

3.5. Restricciones de implementación

- Usar solo la biblioteca estándar de Python; sin paquetes externos ni acceso a red.
- Uso obligatorio de la libreria RegEx (re).
- La detección de rimas y la validación de símbolos **deben implementarse usando expresiones regulares** y únicamente las funciones del módulo estándar `re`.

4. Ejemplos de uso

estrofas.txt

```
fresco, mangos, melón, cielo

Salgo con lo puesto,
te ofrezco,
esto que es fresco
Y hago sin mirar al resto,
no es que no sea modesto

Sale el sol, se baila malambo
Pintó el calorcito, por eso traje unos mangos
Y eso de ahogar las penas, lo hacemos escabiando
Nunca nos sale bien, por eso terminan flotando

Que ahora nada malo va a poder pasarte
Que encuentro que algo lindo en esto de extrañarte
Siento que estás ahí, aunque no pueda hablarte
Tu calor me acompaña, siempre va pa todas parte

Espero
Entre la línea que hacen el mar y el cielo
Sereno
Hay tiempo pa que se nos derritan los hielos
```

Salida: decision.txt

```
Estrofa 1: Inválida (5 versos)
Estrofa 2: 5.4/10 (BONUS)
Rimas: consonante, asonante
(consonantes [-ando]: v3-v4. {5 pts}
asonantes [a-o]: v1-v2, v1-v3, v1-v4, v2-v3, v2-v4. {4x5 = 20 pts}
25 puntos + 2 por uso de "mangos" = 27 totales obtenidos, dividido en 5 => 5.4 puntaje final)

Estrofa 3: 6/10
Rimas: consonante
(consonantes [-arte]: 6x5 = 30 pts)

Estrofa 4: 6.8/10 (BONUS)
Rimas: asonante
(asonante [e-e-o]: v1-v3 {8 pts}
asonante [e-o]: v1-v2, v1-v4, v2-v3, v3-v4 {4x4 = 16}
asonante [i-e-o]: v2-v4 {8 pts} + 2 bonus por "cielo"
34 puntos finales)
```

Considerar que lo que se encuentra entre los paréntesis no es parte de la entrega, sino un desglose de los cálculos hechos para obtener cada puntaje. lo que se les solicita que entregue su programa sería simplemente:

```
Estrofa 1: Inválida
Estrofa 2: 5.4/10 (BONUS)
Rimas: consonante, asonante
```

```
Estrofa 3: 6/10
Rimas: consonante
Estrofa 4: 6.8/10 (BONUS)
Rimas: asonante
```

5. EBNF General

digitos ::= ([0-9])
vocal ::= (AEIOU|aeiou)
vocal_acentuada ::= (ÁÉÍÓÚ|áéíóú)
letras ::= ([A-Z|a-z|Ññ])
string ::= (<vocal>|<vocal_acentuada>|<letras>)
caracteres_permitidos ::= ('¿'|'?'|'¡'|'!'|','|'.'|';'| ' |'-'|'('|')'|':'| ")
palabra ::= (<string>|<caracteres_permitidos>|<digitos>)

6. Sobre la entrega

- El código debe venir indentado y ordenado.
- Se debe entregar los siguientes archivos:
  - Juez.py
  - README.txt

▶ **El Uso de IA está prohibido**. Si se detecta, la tarea recibirá nota 0 sin derecho a apelación. (aplica para la totalidad de la entrega)

▶ Está prohibido emplear la expresión regular `\w` y sus variantes. Su uso podría significar nota 0 sin derecho a reclamo.

▶ El uso de expresiones regulares es **obligatorio**; soluciones que eviten regex (por ejemplo, comparaciones carácter-a-carácter sin patrones) obtendrán 0 pts en el ítem de detección de rima.

- Si no existe orden en el código habrá descuento.
- Todas las funciones deben ir comentadas con el siguiente formato:
  '''
  ***
  Parametro 1 : Tipo
  Parametro 2 : Tipo
  ...
  ***
  Tipo de Retorno o None
  ***
  Breve descripción de la función y el retorno
  '''
  Se harán descuentos por funcion no comentada

- La entrega debe realizarse en un archivo comprimido en tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz.
  Ej.: Tarea1LP_202473000-k.tar.gz.
- El archivo README.txt debe contener el nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa.
- La entrega será vía aula y el plazo máximo de entrega es hasta el día **Miércoles 20 de agosto, 23:59 horas, vía aula.**
- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro de la primera hora)
- Las copias y Uso de código generado por IA serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Solo se pueden realizar consultas respecto a la tarea hasta 2 días antes de la entrega.
- Las consultas del enunciado se debe realizarse mediante el foro de la tarea disponible en AULA.

7. Calificación

7.1. Entrega

Para la calificación de su tarea, debe 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 (total 30 pts)

- Definir e implementar patrones de expresiones regulares fieles al EBNF utilizando RegEx. (10 pts)
- Reconocimiento de estructura y validación de las palabras y símbolos permitidos dentro de las estrofas. (10 pts)
- Evaluar correctamente una estrofa que contenga únicamente rimas consonantes. (10 pts)

De obtener los 30 puntos de la entrega mínima, se procederá a la revisión de las siguientes secciones de la tarea.

7.1.2. Asignación de puntajes general (total 70 pts)

1. Detección de rimas (35 pts)
- Clasificación de rima asonante y correcta diferenciación de longitudes: 15 pts.
- Clasificación de rima consonante y correcta diferenciación de longitudes: 10 pts.
- Clasificación de misma terminación: 5 pts.
- Clasificación de gemela y sin rima: 5 pts.

2. Cálculo de puntaje y escalado (20 pts)
- Correcta distinción en tipos de rima dentro de una misma estrofa: 5 pts.
- Suma de los puntajes de las 6 parejas de versos: 5 pts.
- Aplicación del bonus: 5 pts.

- Penalización de "sin rima"(–2 pts) y normalización a escala 0–10: 5 pts.

3. Formato de salida (10 pts)

- Escritura de `decision.txt` con formato solicitado: 10 pts.

4. Uso riguroso de regex (5 pts)

- Todas las detecciones de rima y validaciones de símbolos implementadas exclusivamente con patrones de `re` respetando restricciones.

7.2. Descuentos

- Falta de comentarios sobre cada predicado (–5 pts c/u, máx. –20 pts)
- Falta de README (–20 pts)
- Falta de información obligatoria en el README (nombre, rol, instrucciones) (–5 pts c/u)
- Falta de orden o mala indentación en el código (–5 a –20 pts según gravedad)
- Mal nombre en los archivos entregados (–5 pts c/u; –10 pts si es el `.tar.gz`)
- Uso de código generado por IA, nota máxima 30 y en casos graves implicará un 0 y se le informará a las respectivas autoridades.
- Entrega tardía (–10 pts si es dentro de la primera hora; –20 pts por cada día o fracción de retraso)