GENERACIÓN AUTOMÁTICA DE ORACIONES

(Septiembre de 2020)

Ivan Fernando Mujica Mamani  
Universidad Catolica Boliviana

**Resumen**—En años recientes el campo de la ciencia de datos ha tomado especial importancia en la toma de decisiones de las empresas e instituciones, una parte vital de este campo es el dominio de un lenguaje con una amplia implementación de librerías y paquetes para el manejo e interpretación de datos; Python es uno de los lenguajes más populares para la aplicación de ciencia de datos, en parte gracias a que tiene una de las curvas de aprendizaje menos pronunciadas. En el presente artículo se implementa un programa que resuelve el problema de la generación automática de oraciones mediante la conjugación aleatoria de elementos de un conjunto de listas de artículos, sujetos, verbos, preposiciones y adverbios, el resultado del programa es el stdout del sistema operativo residente.

**Índice de términos**—Python, Ciencia de Datos, stdout

—————————— ◆ ——————————

# 1 Introduccion

En el presente artículo explicamos cómo se implementa un programa para el problema generación automática de oraciones mediante el uso de Python en su versión 3.8, para tal efecto se usó la librería *random* perteneciente al core del lenguaje. Debido a que este lenguaje implementa tipado dinámico y programación funcional es mucho más sencillo y la curva de aprendizaje es menos pronunciada en relación a otros lenguajes como Java o C++[1]. Es necesario contar con un ambiente de desarrollo con Python y algún editor de código.

Se necesita un programa que genere N oraciones aleatorias, una oración básica debe estar compuesta por una secuencia similar a articulo, sujeto, verbo y adverbio; siendo el adverbio opcional ya que sólo enfatiza la acción del verbo, nuestro programa genera oraciones con cualquiera de estas dos estructuras como indica el enunciado.

# 2 Resolución **del problema**

## 2.1 Selección aleatoria

La selección aleatoria está implementada gracias al paquete random desde el core de Python, mediante el método *choice,* técnicamente lo que hace es elegir al azar un elemento de una listay la función *randint,* este último recibe dos argumentos que serían el límite inferior y superior del número entero aleatorio que se desea generar [3].

## 2.2 Conjugación **de palabras**

Para la conjugación de palabras necesitamos una estructura tipo array para contener la oración generada o simplemente podríamos enviarlo directamente al stdout del sistema operativo en una sola cadena formateada que contenga los elementos elegidos de cada array, la función print recibe una cadena que puede ser formateada con variables gracias al uso de llaves {} como comodín del template[2].

## 2.3 Construcción **del programa**

Primeramente necesitamos un conjunto de listas, estas pueden contener cuantas palabras necesitemos. En el programa declaramos 4 arrays (Artículos, Sujetos, Verbos y Adverbios) .

# declaring arrays

articles = ['the', 'an', 'a', 'those', 'that', 'these', 'this']

subjects = [ 'cat', 'puppy','friend', 'lover', …..]

verbs = ['eat', 'loves', 'run away', 'shows', …. ]

adverbs = ['loudly', 'well', 'badly','always', 'quietly', '']

Una vez declarado los arrays necesitamos una bifurcación para preguntar cual de las dos estructuras se va a generar, para eso utilizamos la funciones *randint y choice* .

for n in range(int(inp)) :

if random.randint(0,1) == 1 :

print(f'{n + 1} {random.choice(articles)} {random.choice(subjects)} {random.choice(verbs)} {random.choice(adverbs)}')

else :

print(f'{n + 1} {random.choice(articles)} {random.choice(subjects)} {random.choice(verbs)}')

# Conclusion

Por lo expuesto en el presente documento podemos apreciar la flexibilidad del lenguaje de programación en cuanto a la mensajería con el Sistema Operativo se refiere, así también un poco de la amplia colección de librerías implementadas en Python. Es importante profundizar en las funciones que ya tiene incorporado para el máximo de provecho al lenguaje. La implementacion final de este problema se encuentra en <https://github.com/ifmm87/introduction-data-science/blob/master/exercises-1109/word-generator.py>

**References**

1. M. Summerfield, *Programming in Python 3 A complete introduction to the Python language, 2nd ed.* Boston, MA.: Pearson Education, pp. 2-3, 2010.
2. Python.org “Python Documentation”[online] Available: <https://docs.python.org/3/tutorial/inputoutput.html>. 2020.
3. Python.org “Python Documentation” [online] Available: <https://docs.python.org/3/library/random.html> 2020.