TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO DE NUEVO LEÓN

ISC

Lenguajes y Autómatas 1

Unidad 1

Reporte

Marlene Garza Guzmán

16480135

Guadalupe N.L. Septiembre 2018

Reporte

Primeramente, se realizó una investigación de los atributos que se pudieran realizar en Python, ¿cómo funcionaban?, ¿cómo se podían agregar?, etc.

Posteriormente se realizó la búsqueda de ejemplos similares para dar una idea de su creación y su funcionamiento, esto con el objetivo de poder entender mejor las funciones y el algoritmo de los programas, algunas de las funciones que se investigaron y fueron utilizadas fueron:

- **Imput:** permite obtener texto escrito por el teclado
- **In range:** esta propiedad genera una lista de números, que generalmente se usa para iterar con forbucles
- **Random:** proporciona un generador de números aleatorios
- **Randint:** genera un numero aleatorio entero
- **Join:** convierte una lista en una cadena formada por los elementos de la lista separados por comas.
- **Append:** agrega un solo elemento al final de la lista. Este no devuelve una nueva lista, simplemente modifica la original.
- **Choice:** Devuelve un elemento aleatorio de la secuencia no vacía

El programa con la instrucción de "Dado un alfabeto definir un número n de palabras de ese alfabeto", este programa pide que se le dé un alfabeto donde posteriormente se puedan crear palabras o combinaciones partiendo de este mismo, dándole un límite ya que si se le deja hacer muchas combinaciones estas pueden llegar a ser muy grandes y en ocasiones traba la computadora y fue realizado de la siguiente manera:

Primeramente, declaramos *alfabeto* con el 'input' donde se pueda realizar la acción de ingresar el alfabeto que estaremos utilizando en el programa, en esta parte se notó un error de que el alfabeto tiene que ir todo pegado sin espacios ni comillas ya que los tomara como un elemento al momento de hacer las combinaciones.

Posteriormente declaramos las variables que estaríamos utilizando más adelante en el programa, indicándoles longitud o dejándolos como cadena vacía.

Indicamos un ciclo a través de un ‘while’ donde estará checando que la cantidad de letras sea menor a la indicada. Por el ‘for’ avanza entre cada una de las letras del alfabeto. Después de que lee las letras que lo conforman se elige una letra al azar, cada vez que vuelve a dar la vuelta se coloca una de las letras del alfabeto.

Con la función de ‘append’ gurda la cadena creada, y posteriormente se elimina la cadena formada, esto es para que al momento de que se esté generando una nueva no tenga elementos sobre ella y así se puede evitar que las combinaciones sean muy largas ya que se van combinando las cadenas anteriores en las nuevas si no se vacía. Sigue siguiendo el ciclo hasta que sea igual al número de vueltas que se le indico y posteriormente la imprime.

Programa 1 Dado un alfabeto definir un número n de palabras de ese alfabeto.

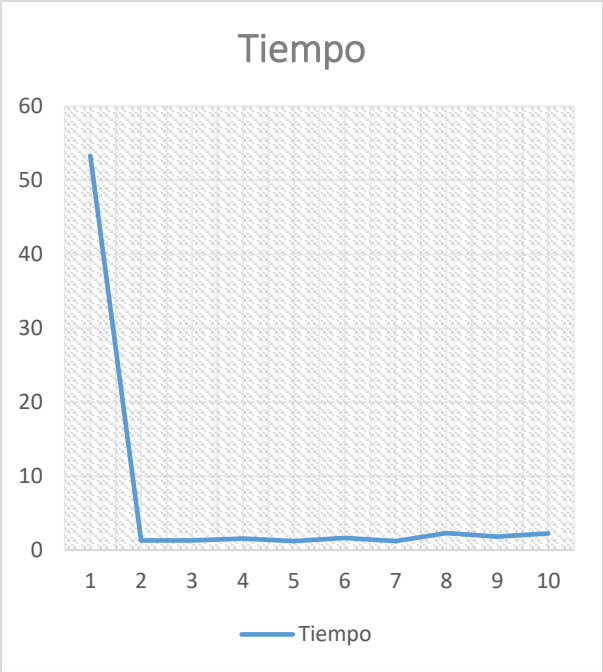
Código

```
import random

def clase1():
    alfabeto=input("ingresa alfabeto")
    n=5
    m=0
    letra=""
    lista=[]

    while  m<=n:#checa que la cantidad de letras sea menor
        for x in range (0,len(alfabeto)): #avanza cada una de las letras del alfabeto

            letras=random.randint(1,5) #una letra aleatoria del alfabeto
            letra=letra.join(random.choice(alfabeto)for _ in range(letras))#en cada vuelta se vaya realizando se agregara una letra

            lista.append(letra)#append guarda
            letra="" #elimina la cadena que se tenia guardada dejandola vacia para que ingrese
            # una nueva si no se vacia la cadena se ara mas grande lo cual si son muchas combinaciones puede trabar la computadora
            m=m+1

    print (lista)#imprime la lista

from time import time#tiempo que tardo en compilar
start =time()
clase1()
end=time()-start
print(end)
```

En esta tabla comparativa se utilizó el mismo abecedario en distintas ocasiones para la generación de palabras o combinaciones de las mismas, sin embargo, se notó que la primera vez que se realizó tenía un mayor número de tiempo en que se tardó realizando la operación, después disminuyo considerablemente y donde estuvo dentro de un margen no muy grande como el anterior.

Tabla y Grafica de tiempo en el que se ingresó una misma cadena

N° Prueba (“abcd1234kjhg9876”)	Tiempo que tardo
1	53.279603481292725
2	1.2832844257354736
3	1.3042066097259521
4	1.5549366474151611
5	1.2110364437103271
6	1.6643636226654053
7	1.2318320274353027
8	2.2740039825439453
9	1.8267695903778076
10	2.2296531200408936



El programa con la instrucción de “Dado un conjunto de cadenas de un lenguaje, definir cuál es el alfabeto de este lenguaje”, en este programa se pide que se realice la identificación del alfabeto partiendo de una cadena de texto que se ingresara con anterioridad.

Igual que en el primer programa de ingresa primeramente el ‘input’ donde el usuario podrá ingresar una cadena de texto. Se declararán las variables que se vayan a utilizar.

Posteriormente se declara el ‘por i in’ donde se identifica cada uno de los valores de la lista. Se toma el primer valor y si el primer elemento de la lista no está en la nueva cadena se guardará, se irán eliminado los caracteres que se tengan repetidos, dejando que solo se imprima una vez cada elemento.

Tanto en este programa como en el anterior la cadena o el alfabeto se tienen que escribir todos juntos para que no tome ningún elemento vacío.

Programa 2 “Dado un conjunto de cadenas de un lenguaje, definir cuál es el alfabeto de este lenguaje”

Código

```
def clase2():
    lista=input("escribe una cadena de texto")#input para que se pueda obtener texto escrito por el teclado
    cadena=""
    lista2=[]#lista donde se guardaran los valores que se imprimiran
    for i in (lista): #es cada uno de los valores de la lista
        m=i #toma el primer valor
        for x in (m): #primer valor
            if x not in lista2:#not in separa los valores de el primer valor
                lista2.append(x)#si el elemnto todavia no se encuentra en la lista se guarda

    print (lista2)#se iempime la lista

from time import time #es el que tomara el tiempo en que tarda en ejectutarce
start =time()
clase2()
end=time()-start
print(end)
```

Como se vio en el primer programa, la primera vez que se ingresó la cadena de texto se tardó más que las que fueron posteriores exactamente con la misma cadena, mas sin embargo no fue tan grande el rango en el que tardo en ejecutarse como el primer programa, la varianza de tiempo se pudo notar más en este programa.

Tabla y Grafica de tiempo en el que se ingresó una misma cadena

N° Prueba (“prueba del programa 2”)	Tiempo que tardo
1	13.114821434020996
2	1.6568455696105957
3	1.4249203205108643
4	3.496532440185547
5	1.8537476062774658
6	1.428074836730957
7	1.4078037738800049
8	1.9596378803253174
9	1.084688425064087
10	0.9501259326934814