INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MATERIA: Teoría Computacional.

PRÁCTICA: Práctica 1. Introducción a Python.

ALUMNO: Meza Vargas Brandon David.

PROFESOR: Jorge Luis Rosas Trigueros.

FECHA PRÁCTICA: 16-oct-2020

FECHA DE ENTREGA: 23-oct-2020

MARCO TEÓRICO

Cadena:

Es una secuencia finita de símbolos yuxtapuestos. Se utilizan las letras minúsculas del final del alfabeto (w, x, y, z), para designar a las cadenas.

Ejemplos: w = abc, x = 321abc

Longitud de cadena:

Se denota como |w| (donde w es una cadena) y se define como el número de símbolos que componen la cadena.

Ejemplos: Sean las cadenas w = abcd, x = 0123a, entonces |w| = 4, |x| = 5

Prefijos y sufijos de una cadena:

Los prefijos están formados por los primeros símbolos de la cadena; y los sufijos, por los últimos. Un prefijo o sufijo de una cadena que no sea la misma cadena es un prefijo o sufijo propios.

Ejemplo: Sea la cadena w = abc; los prefijos y sufijos, son Prefijos: ϵ, a, ab, abc. Sufijos: ϵ, c, bc, abc.

Cadenas en Python

Una cadena (str) al igual que una lista (lista), una tupla (tuple) o una cadena de bytes (bytes) es un conjunto de elementos ordenados (caracteres UTF-8) e indizables. Cada elemento es numerado empezando desde 0:

cadena: HOLA
índices: 0123

de forma que puedes acceder a un carácter dado usando su índice y la sintaxis objeto[índice]. Python además permite usar índices negativos, de forma que -1 hace referencia al último elemento, -2 el antepenúltimo, etc. Un ejemplo de lo anterior lo podemos ver en la imagen 1:

```
>>> cadena = "ESCOM"
>>> cadena[2]
'C'
>>> cadena[-1]
'M'
```

Imagen 1. Ejemplo de acceso a un carácter de una cadena en Python.

El indizado extendido lleva a cabo un slicing o rebanado del iterable, el cual permite obtener una nueva cadena con el contenido de parte de la cadena original (o toda). Es decir, en vez de obtener un solo carácter usando su índice, obtenemos un conjunto de caracteres de la cadena usando un rango de índices. La sintaxis general es: *[indice_inicial: indice_final: paso].* Podemos ver un ejemplo en la imagen 2:

```
>>> cadena[1:]
'SCOM'
>>> cadena[::]
'ESCOM'
>>> cadena[0:4:2]
'EC'
>>> cadena[-4:-2]
'SC'
>>> cadena[:-1]
'ESCO'
```

Imagen 2. Ejemplo de indizado extendido en Python.

Función range() en Python:

La función range() retorna una secuencia de números, empezando en 0 por defecto e incrementa de 1 en 1 por defecto hasta un numero especificado.

Sintaxis: range*(inicio, final, paso*)

Función len() en Python:

La función len() retorna el número de elementos en una cadena, es decir, nos da la longitud de una cadena, tupla, lista o rango.

Sintaxis: len(cadena, bytes, tupla, lista, rango o una colección)

En la imagen 3 podemos ver un ejemplo de uso de la función range() y len():

```
>>> cadena = "hola"
>>> len(cadena)
4
>>> len(range(1, 100, 7))
15
```

Imagen 3. Ejemplo de las funciones len() y range().

MATERIAL Y EQUIPO

Para la realización de esta práctica son necesarias las siguientes herramientas:

- Un equipo de cómputo que cumpla con los requerimientos para el uso del lenguaje de programación Python.
- Tener instalado el lenguaje de programación Python.
- Contar con un IDE para programar con Python, cualquiera es útil.

DESARROLLO DE LA PRÁCTICA

En primera instancia se comenzó la práctica con la investigación de las preguntas siguientes:

¿Quién desarrolló Python?

Guido van Rossum ideó el lenguaje Python a finales de los 80 y comenzó a implementarlo en diciembre de 1989. En febrero de 1991 publicó la primera versión pública, la versión 0.9.0. La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008.

Van Rossum nació en 1956 en los Países Bajos y allí recibió un título de maestría en matemática y ciencias de la computación de parte de la Universidad de Amsterdam en 1982. Esto le abrió las puertas a varios puestos de trabajo que ocupó durante los próximos años, en el Centrum Wiskunde & Informatica (CWI) en esa misma ciudad, donde creó Python como un sucesor del lenguaje ABC.

¿Por qué se llama "Python"?

el nombre del lenguaje proviene de la afición de su creador original Guido van Rossum, por los humoristas británicos Monty Python.

Monty Python fue un grupo británico de humoristas que sintetizó en clave de humor la idiosincrasia británica de los años 60 y 70

Versión actual de Python.

La versión actual de Python es la 3.9.0, siendo lanzada el 5 de octubre del presente año 2020.

Explicar el término "Pythonico".

Código Pythonico es un código escrito en Python, que no solo es sintácticamente correcto, sino que está escrito de forma tal que aglutina las mejores prácticas del lenguaje Python, cumpliendo con las convenciones establecidas por la propia comunidad de Pythonistas.

Diferencia entre lista, tupla y diccionario.

Tupla: es una variable donde se pueden almacenar múltiples datos, se caracteriza y se diferencia de las listas porque una vez se ha creado o llenado sus valores ya luego no pueden modificarse. Otro dato importante a mencionar es que en una tupla podemos almacenar distintos tipos de datos, pueden ser texto, numéricos entre otros, todo dentro de una misma tupla. Véase un ejemplo de estas en la imagen 4.

```
tupla=('hola','32','34.2')
print(tupla[0])
```

Imagen 4; ejemplo de una tupla en Python

Lista: en estas a diferencia de las tuplas puede cambiarse sus valores y al momento de declarar una se encierran sus valores en corchetes, si existe la posibilidad de llegar a necesitar modificar algún valor siempre es recomendado usar las listas. Véase un ejemplo en la imagen 5.

```
lista = ['hola','32','34.2']
print(lista[0])      #Imprimiendo el elemento 0
lista[0] = 'adios'   #Cambiando el valor del elemento 0
print(lista[0])

hola
adios
```

Imagen 5; ejemplo de una lista en Python y su salida en la terminal

Diccionarios: los diccionarios son muy similares a las tuplas, pero tienen el beneficio de utilizar un nombre llamado clave, en lugar de solo utilizar un numero de índice, de esta forma nos es más fácil de recordar estas claves que recordar los números de índices. Véase un ejemplo en la imagen 6.

```
diccionario = {'edad':20,'nombre':'pedro'}
print(diccionario['edad']) #Imprimira la edad; 20
```

Imagen 6; ejemplo de un diccionario en Python.

Posteriormente comenzamos a revisar algunos comandos básicos que podemos usar para cadenas en el lenguaje de programación Python (estas están desarrollados en el marco teórico del presente reporte), para así poder resolver el programa propuesto por el profesor; encontrar los sufijos y prefijos de una cadena, así como contar cuantos hay, además de cuantas subcadenas tiene una cadena ingresada.

La primera parte del problema es generar los prefijos de una cadena, esto lo logramos con un ciclo for que va hasta la longitud de la cadena, cada prefijo que encontremos lo iremos guardando en una lista, esto gracias a la propiedad append, esto lo hacemos para poder cuantos prefijos y prefijos propios tiene la cadena ingresada, podemos ver la parte de este código en la imagen 7:

```
cadena="perrera"
prefijos=[]
sufijos=[]
subcadenas=[]

for n in range(len(cadena)+1):
    print("Prefijo ", n, "=", cadena[:n], end="    ") #
    prefijos.append(cadena[:n])
    if cadena[:n] != cadena:
        print("Prefijo propio")

print('\nHay',len(prefijos),' prefijos, ',len(prefijos)-1, 'propios\n')
```

Imagen 7. parte que genera los prefijos de una cadena, además de cuantas hay.

Para verificar esto ejecutamos el programa con F5 en la terminal IDLE y la salida la vemos en la imagen 8;

```
Prefijo  0 =     Prefijo propio
Prefijo  1 = p    Prefijo propio
Prefijo  2 = pe    Prefijo propio
Prefijo  3 = per    Prefijo propio
Prefijo  4 = perr    Prefijo propio
Prefijo  5 = perre    Prefijo propio
Prefijo  6 = perrer    Prefijo propio
Prefijo  7 = perrera
Hay 8  prefijos,  7 propios
```

Imagen 8. Prefijos de la cadena.

Posteriormente generamos los sufijos de la cadena, de igual forma los podemos generar con un ciclo for haciendo unas pequeñas modificaciones a la indización de la cadena, de igual forma los guardamos con append para poder saber su cantidad, lo podemos ver en la imagen 9:

```
for n in range(len(cadena)+1):
    if n==0:
        print("Sufijo ", n, "=     sufijo propio", end="    \n")
        sufijos.append('')
    else:
        print("Sufijo ", n, "=", cadena[-n:], end="    ") #
        sufijos.append(cadena[-n:])
        if cadena[-n:] != cadena:
            print("Sufijo propio")
        else:
            print("\n")

print('\nHay',len(sufijos),' sufijos, ',len(prefijos)-1, 'propios\n')
```

Imagen 9. parte que genera los sufijos de una cadena, además de su cantidad.

Al ejecutar esta parte la salida que tenemos es la siguiente (véase imagen 10):

```
Sufijo  0 =      sufijo propio
Sufijo  1 = a     Sufijo propio
Sufijo  2 = ra     Sufijo propio
Sufijo  3 = era     Sufijo propio
Sufijo  4 = rera     Sufijo propio
Sufijo  5 = rrera     Sufijo propio
Sufijo  6 = errera     Sufijo propio
Sufijo  7 = perrera


Hay 8  sufijos,  7 propios
```

Imagen 10. Sufijos de la cadena.

En esta parte me surgió un problema, el sufijo 0 que es la cadena vacía no se generaba y los saltos de línea no salían de manera adecuada, esto lo logre resolver implementando una condición para cuando n sea igual a 0, de esta manera aseguras que la cadena vacía se incluya siempre en los sufijos de una cadena, esto se puede ver en el código de la imagen 9.

Pala la ultima parte del programa realice el código para imprimir todas las subcadenas generadas por una cadena dada, esta parte fue de pensar más, use un for anidado para generar las subcadenas, aquí también me surgió una problemática; la cadena vacía se repetía cada que incrementaba n en el ciclo, para resolver esto implemente una condición, además de una variable de control que no afecta a nuestro programa en el segundo for para así asegurar que la cadena vacía se almacenara solo una vez, esta parte del código lo vemos en la imagen 11

```
cont=0
for n in range(len(cadena)+1):
    for m in range(n, len(cadena)+1):
        if(cadena[n:m] == "" and n>=1):
            cont=0;
        else:
            print(cadena[n:m])
            subcadenas.append(cadena[n:m])
print('\nHay',len(subcadenas),' subcadenas\n')
```

Imagen 11. Código que genera las subcadenas de una cadena, además de cuantas hay.

De esta forma el código completo lo podemos ver en la imagen 12, este código será anexado junto con este reporte:

```
cadena="perrera"
prefijos=[]
sufijos=[]
subcadenas=[]

for n in range(len(cadena)+1):
    print("Prefijo ", n, "=", cadena[:n], end="    ") #
    prefijos.append(cadena[:n])
    if cadena[:n] != cadena:
        print("Prefijo propio")

print('\nHay',len(prefijos),' prefijos, ',len(prefijos)-1, 'propios\n')

for n in range(len(cadena)+1):
    if n==0:
        print("Sufijo ", n, "=     sufijo propio", end="    \n")
        sufijos.append('')
    else:
        print("Sufijo ", n, "=", cadena[-n:], end="    ") #
        sufijos.append(cadena[-n:])
        if cadena[-n:] != cadena:
            print("Sufijo propio")
        else:
            print("\n")

print('\nHay',len(sufijos),' sufijos, ',len(prefijos)-1, 'propios\n')

cont=0
for n in range(len(cadena)+1):
    for m in range(n, len(cadena)+1):
        if(cadena[n:m] == "" and n>=1):
            cont=0;
        else:
            print(cadena[n:m])
            subcadenas.append(cadena[n:m])
print('\nHay',len(subcadenas),' subcadenas\n')
```

Imagen 12. código completo del programa.

La salida de este programa lo vemos en la imagen 13:

```
Prefijo  0 =     Prefijo propio
Prefijo  1 = p    Prefijo propio
Prefijo  2 = pe    Prefijo propio
Prefijo  3 = per    Prefijo propio
Prefijo  4 = perr    Prefijo propio
Prefijo  5 = perre    Prefijo propio
Prefijo  6 = perrer    Prefijo propio
Prefijo  7 = perrera
Hay 8  prefijos,  7 propios

Sufijo  0 =     sufijo propio
Sufijo  1 = a    Sufijo propio
Sufijo  2 = ra    Sufijo propio
Sufijo  3 = era    Sufijo propio
Sufijo  4 = rera    Sufijo propio
Sufijo  5 = rrera    Sufijo propio
Sufijo  6 = errera    Sufijo propio
Sufijo  7 = perrera


Hay 8  sufijos,  7 propios


p
pe
per
perr
perre
perrer
perrera
e
er
err
erre
errer
errera
r
rr
rre
rrer
rrera
r
re
rer
rera
e
er
era
r
ra
a

Hay 29  subcadenas
```

Imagen 13. Salida del código mostrado en la imagen 12.

Otra problemática que me surgió fue que tenía un for que no hacia lo que le pedía, pero esto era por que al no haber llaves en Python que indican el final de un ciclo, el final de este lo indicas con la tabulación de tu código, por lo que solucione este problema dándole la tabulación adecuada a mi código.

CONCLUSIONES Y RECOMENDACIONES

Esta practica me introdujo a un mundo nuevo, Python, esta fue la primera vez que interactúe con este lenguaje de programación, el profesor explico de manera muy concisa y clara los elementos requeridos por nosotros para realizar la práctica, creo que la manera en la que se realizo esta practica es muy adecuada para la modalidad actual que es en línea, aunque una recomendación que haría es que el profesor codifique algunos ejemplos completos para que las personas que no hemos interactuado tanto con Python nos vayamos familiarizando.

Con los ejercicios realizados me aventure a experimentar con la sentencia if e investigue su uso en Python, es muy similar al if de otros lenguajes de programación, pero aprendí a usarlo en Python, al menos una noción general de este. También experimente con el ciclo for e investigue un poco mas sobre este, igual funciona de manera muy similar al for de otros lenguajes de programación.

Aprendí algunos elementos básicos de Python, más que nada sobre las cadenas y algunas de sus funciones, el uso correcto del if y del ciclo for, además de la importancia de la tabulación en Python.

BIBLIOGRAFIA

-Barrueto Luis Eduardo. (2020). “Desarrollador de Python”. Recuperado de: http://www.maestrosdelweb.com/introspectiva-guido-van-rossum-python/

-Sevilla. J (2019). “Cadenas en Python”. Recuperado de: https://es.stackoverflow.com/questions/280955/manejo-de-cadenas-en-python

-Bahit Eugenia. (2011). “Listas, tuplas y diccionarios en Python”. Recuperado de: https://www.codigazo.com/python/diferencia-entre-tupla-lista-diccionario-en-python

-Pérez. I (2020). “Lenguajes y Autómatas”. Recuperado de: https://www.uaeh.edu.mx/docencia/P_Presentaciones/icbi/asignatura/lenguajes_y_automatas.pdf