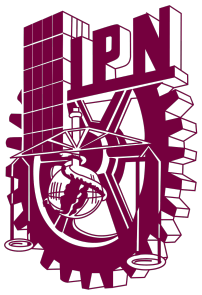

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Práctica 2

Alumno: Meléndez Padilla Mauricio

Profesor: Rosas Trigueros Jorge Luis

Fecha de realización de la práctica

31/AGO/18

Fecha de entrega del reporte

6/SEP/18

Marco Teórico

Funciones con cadenas en Python.

Concatenar

Este término significa juntar cadenas de caracteres. El proceso de concatenación se realiza mediante el operador de suma (+). Ten en cuenta que debes marcar explícitamente dónde quieres los espacios en blanco y colocarlos entre comillas.

En este ejemplo, la cadena de caracteres “mensaje1” tiene el contenido “Hola Mundo”

mensaje1 = 'Hola' + ' ' + 'Mundo' print(mensaje1) -> Hola Mundo

Multiplicar

Si quieres varias copias de una cadena de caracteres utiliza el operador de multiplicación (*). En este ejemplo, la cadena de caracteres mensaje2a lleva el contenido “Hola” tres veces, mientras que la cadena de caracteres mensaje2b tiene el contenido “Mundo”. Ordenemos imprimir las dos cadenas.

mensaje2a = 'Hola ' * 3 mensaje2b = 'Mundo' print(mensaje2a + mensaje2b) -> Hola Hola Hola Mundo

Añadir

¿Qué pasa si quieres añadir material de manera sucesiva al final de una cadena de caracteres? El operador especial para ello es compuesto (+=).

mensaje3 = 'Hola' mensaje3 += ' ' mensaje3 += 'Mundo' print(mensaje3) -> Hola Mundo

Métodos para cadenas de caracteres: buscar, cambiar

En adición a los operadores, Python trae preinstalado docenas de métodos que te permiten hacer cosas con las cadenas de caracteres. Solos o en combinación, los métodos pueden hacer casi todo lo que te imagines con las cadenas de caracteres. Puedes usar como referencia la lista de métodos de cadenas de caracteres (String Methods) en el sitio web de Python, que incluye información de cómo utilizar correctamente cada uno. Para asegurar que tengas una comprensión básica de métodos para cadenas de caracteres, lo que sigue es una breve descripción de los utilizados más comúnmente. Extensión Puedes determinar el número de caracteres en una cadena utilizando el método len. Acuérdate que los espacios en blanco cuentan como un carácter.

mensaje4 = 'hola' + ' ' + 'mundo' print(len(mensaje4)) -> 10

Encontrar

Puedes buscar una sub-cadena en una cadena de caracteres utilizando el método find y tu programa te indicará el índice de inicio de la misma. Esto es muy útil para procesos que veremos más adelante. Ten en mente que los índices están numerados de izquierda a derecha y que el número en el que se comienza a contar la posición es el 0, no el 1.

mensaje5 = "Hola Mundo" mensaje5a = mensaje5.find("Mundo") print(mensaje5a) -> 5

Si la sub-cadena no está presente el programa imprimirá el valor -1.

mensaje6 = "Hola Mundo" mensaje6a = mensaje6.find("ardilla") print(mensaje6a) -> -1

Minúsculas

A veces es útil convertir una cadena de caracteres a minúsculas. Para ello se utiliza el método lower. Por ejemplo, al uniformar los caracteres permitimos que la computadora reconozca fácilmente que “Algunas Veces” y “algunas veces” son la misma frase.

mensaje7 = "HOLA MUNDO" mensaje7a = mensaje7.lower() print(mensaje7a) -> hola mundo

Convertir las minúsculas en mayúsculas se logra cambiando .lower() por upper(). Reemplazar

Si necesitas cambiar una sub-cadena de una cadena se puede utilizar el método replace.

mensaje8 = "HOLA MUNDO"

mensaje8a = mensaje7.replace("L", "pizza") print(mensaje8a) -> HOpizzaA MUNDO

Cortar

Si quieres cortar partes que no quieras del principio o del final de la cadena de caracteres, lo puedes hacer creando una sub-cadena. El mismo tipo de técnica te permite separar una cadena muy larga en componentes más manejables.

mensaje9 = "Hola Mundo" mensaje9a = mensaje9[1:8] print(mensaje9a) -> ola Mun

Puedes sustituir las variables por números enteros como en este ejemplo:

mensaje9 = "Hola Mundo" startLoc = 2 endLoc = 8 mensaje9b = mensaje9[startLoc: endLoc] print(mensaje9b)

-> la Mun

Esto hace mucho más simple usar este método en conjunción con el método find como en el próximo ejemplo, que busca la letra “d” en los seis primeros caracteres de “Hola Mundo” y correctamente nos dice que no se encuentra ahí (-1). Esta técnica es mucho más eficaz en cadenas largas -documentos enteros, por ejemplo. Observa que la ausencia de un número entero antes de los dos puntos significa que queremos empezar desde el principio de la cadena. Podemos usar la misma técnica para decirle al programa que pase hasta el final de la cadena de caracteres dejando vacío después de los dos puntos. Y recuerda que la posición del índice empieza a contar desde 0, no desde 1.

mensaje9 = "Hola Mundo" print(mensaje9[:5].find("d")) -> -1

Existen más, pero los métodos para cadenas de caracteres anteriores son un buen comienzo. Fíjate que en el ejemplo anterior utilizamos corchetes en vez de paréntesis. Esta diferencia en los símbolos de la sintaxis es muy importante. Los paréntesis en Python son utilizados generalmente para llevar un argumento a una función. De tal manera que cuando vemos algo como:

print(len(mensaje7))

quiere decir que se lleva la cadena de caracteres “mensaje7” a la función len y entonces enviar el valor resultante de esa función a la declaración print para ser impresa. Una función

puede ser llamada sin un argumento, pero de todas formas tienes que incluir un par de paréntesis vacíos después del nombre de la función. Vimos un ejemplo de ello también.

mensaje7 = "Hola Mundo" mensaje7a = mensaje7.lower() print(mensaje7a) -> Hola Mundo

Esta declaración le dice a Python que aplique la función lower a la cadena mensaje7 y guarde el valor resultante en la cadena mensaje.

Material y equipo.

El material utilizado en la práctica es el siguiente:

Herramientas de software:

- Mac OS X 10.13.6
- Python 2.7.15
- VIM - Vi IMproved 8.1
- Terminal

Herramientas de hardware:

- Computadora personal.

Desarrollo de la práctica.

1. Escriba un programa de Python para calcular la longitud de una cadena.

```
cadena=raw_input("inserte una cadena: ")
print ("La longitud de la cadena es: " + str(len(cadena)))
```

2. Escriba un programa de Python para contar el número de caracteres (frecuencia de caracteres) en una cadena.

Cadena de ejemplo: ’google.com '
Resultado previsto: {'o': 3, 'g': 2, '': 1, 'e': 1, 'l': 1, 'm': 1, 'c': 1}

```
cadena=raw_input("inserte una cadena: ")
miDiccionario = {}
for caracter in cadena:
    if miDiccionario.has_key(caracter):
        miDiccionario[caracter] += 1
    else:
        miDiccionario[caracter]= 1
print miDiccionario
```

3. Escriba un programa de Python para obtener una cadena hecha de los 2 primeros y los 2 últimos caracteres de una determinada cadena. Si la longitud de cadena es menor que 2, devuelve la cadena vacía.

```
# -*- coding: utf-8 -*-

cadena=raw_input("inserte una cadena: ")
if len(cadena)>=4:
    print cadena[:2]+cadena[-2:]
else:
    print "є"
```

4. Escriba un programa de Python para obtener una cadena de una cadena dada, donde todas las apariciones de su primer carácter se han cambiado a '$', excepto el propio primer carácter.
Cadena de ejemplo: 'regresar'
Resultado esperado: 'regesa'

```
from UserString import MutableString

cadena=raw_input("inserte una cadena: ")
cadenanueva = MutableString(cadena.replace(cadena[0],'$'))
cadaux = MutableString(cadena)
cadenanueva[0] = cadaux[0]
print cadenanueva
```

5. Escriba un programa de Python para obtener una sola cadena de dos cadenas dadas, separadas por un espacio, además de intercambiar los dos primeros caracteres de cada cadena.
Cadenas de ejemplo: 'abc', 'xyz'
Resultado Esperado: 'xycabz'

```
cadena1=raw_input("inserte la primer cadena: ")
cadena2=raw_input("inserte la segunda cadena: ")
print(cadena2[:2]+cadena1[2:]+cadena1[:2]+cadena2[2:])
```

6. Escribe una función de Python que tome una lista de palabras y devuelva la longitud de la más larga.

```
miLista = ["Uno", "Dos", "Tres", "Cuatro", "Cinco", "Seis"]
r = ''
for i in miLista:
    if len(i)>len(r):
        r = i
print r + ' =', len(r)
```

7. Escriba un programa de Python para quitar el carácter de una posición dada de una cadena no vacía.

```
from UserString import MutableString

cadena=raw_input("inserte una cadena: ")
indice=input("introduzca el indice: ")
cadena = MutableString(cadena)
cadena[indice]=''
print cadena
```

8. Escriba un programa de Python para eliminar los caracteres que tienen valores de índice impar de una cadena dada.

```
from UserString import MutableString

cadena=raw_input("inserte una cadena: ")
cadena = MutableString(cadena)
for i in range(len(cadena)):
    if i%2 == 0:
        cadena[i]=' '
print cadena
```

Diagramas, gráficas y pantallas

```
MacPro:Práctica 2 berry$ python prac2-1.py
inserte una cadena: holamundo
La longitud de la cadena es: 9
```

Imagen 2.1 Ejercicio 1

```
MacPro:Práctica 2 berry$ python prac2-2.py
inserte una cadena: www.google.com.mx
{'c': 1, 'e': 1, 'g': 2, 'm': 2, 'l': 1, 'o': 3, '.': 3, 'w': 3, 'x': 1}
```

Imagen 2.2 Ejercicio 2

```
MacPro:Práctica 2 berry$ python prac2-3.py
inserte una cadena: holamundo
hodo
MacPro:Práctica 2 berry$ python prac2-3.py
inserte una cadena: q
ϵ
```

Imagen 2.3 Ejercicio 3

```
MacPro:Práctica 2 berry$ python prac2-4.py
inserte una cadena: otorrinolaringologo
ot$rrin$laring$l$g$
```
Imagen 2.4 Ejercicio 4

```
MacPro:Práctica 2 berry$ python prac2-5.py
inserte la primer cadena: hola
inserte la segunda cadena: mundo
mulahondo
```
Imagen 2.5 Ejercicio 5

```
MacPro:Práctica 2 berry$ python prac2-6.py
Cuatro = 6
```
Imagen 2.6 Ejercicio 6

```
MacPro:Práctica 2 berry$ python prac2-7.py
inserte una cadena: holamundo
introduzca el indice: 5
holamndo
```
Imagen 2.7 Ejercicio 7

```
MacPro:Práctica 2 berry$ python prac2-8.py
inserte una cadena: alguienquierepensarenlosninos?
 l u e q i r p n a e l s i o ?
```
Imagen 2.8 Ejercicio 8

Conclusiones y recomendaciones

Python se caracteriza por su sencillez y elegancia, me resultó sorprendente el como Python requiere de pocas líneas de programación en comparación con otros lenguajes que es necesario escribir una gran cantidad de líneas, en Python es posible realizar funciones en una única línea sin perder nada de claridad.

En cuanto a los nuevos aprendizajes adquiridos sobre VIM, puedo decir que VIM resulta ser una eficiente herramienta de desarrollo debido a su sencillez que esta maneja, lo que se traduce a un desarrollo ágil.

Bibliografía

- [1]"Python (programming language)", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language). [Accessed: 30- Aug- 2018].
- [2]"Learn python in Y Minutes", Learnxinyminutes.com, 2017. [Online]. Available: https://learnxinyminutes.com/docs/python/. [Accessed: 30- Aug- 2018].
- [3]"Vim", Es.wikipedia.org, 2018. [Online]. Available: https://es.wikipedia.org/wiki/Vim. [Accessed: 19- Aug- 2017].
- [4]"welcome home : vim online", Vim.org, 2018. [Online]. Available: http://www.vim.org/. [Accessed: 30- Aug- 2018].
- [5]R. Python, "VIM and Python - a match made in heaven - Real Python", Realpython.com, 2018. [Online]. Available: https://realpython.com/blog/python/vim-and-python-a-match-made-in-heaven/. [Accessed: 30- Aug- 2018].