Practica 1

Alumno: *Ángel López Manríquez*
Tema: *Operaciones con cadenas*
Fecha: *17 de junio de 2019*
Grupo: *2CV1*

M. en C. Luz María Sánchez García

Instituto Politecnico Nacional
Escuela Superior de Cómputo

Operaciones sobre cadenas

Lopez Manriquez Angel 2CV1

Febrero 2018

1. Introducción

En esta práctica implementaremos en el lenguaje de programación `python3` las operaciones definidas para cadenas. En la teoria computacional, se define una cadena w como una secuencia finita de caracteres pertenecientes a algún alfabeto Σ. En este programa se manejan $u, v \in \Sigma$ (el alfabeto se obvia) que el usuario introducirá al principio, para realizar con ellas una serie de operaciones. Tendremos ocho operaciones principales:

1. **Longitud:** $|u|$ denota el no. de caracteres usados en la palabra (tambien se cuentan caracteres repetidos).
2. **Concatenación:** Denotada por $u \cdot v$.
3. **Inverso:** Denotada por u^{-1}.
4. **Potencia:** Devolveremos u^n donde $n \in \mathbb{Z}^+$, si $n < 0$ nos podemos ayudar de la propiedad $w^{-n} = (w^{-1})^n$.
5. **Subcadenas:** Las subcadenas de u son los elemntos de $\{w \mid u = awb, a, b \in \Sigma^*\}$.
6. **Prefijos:** Los prefijos de u son los elementos de $\{w \mid u = wb, b \in \Sigma^*\}$.
7. **Sufijos:** Los sufijos de u son los elementos de $\{w \mid u = aw, a \in \Sigma^*\}$.
8. **Palindromo:** Decimos que u es palindroma $\iff u = u^{-1}$.

2. Planteamiento del problema

Realizar un programa que calcule las siguientes operaciones con dos cadenas de un determinado alfabeto, en donde el usuario podrá seleccionar con cuales cadenas va a trabajar y qué tipo de operación va a seleccionar:

1. **Longitud:** Usaremos la funcion sobrecargada `len()` la cual puede ser usada para un tipo `str` para obtener la longitud de la cadena deseada.
2. **Concatenación:** Usaremos el operador binario `+` sobre los tipos `string` para concatenar dos cadenas. Es decir, la operación $u \cdot v$ se representara en el codigo como `u+v`.
3. **Inverso:** En python, podremos invertir la mayoria de colecciones poniendo `[::-1]` despues del objeto `str`.
4. **Potencia:** Sea $p \in \mathbb{Z}$ Consideraremos tres casos:

- Cuando $p = 0$: devolvemos ϵ.
- Cuando $p > 0$: Expresaremos $u * p$ a nuestro resultado, es decir, devolveremos $u^p = \prod_{j=1}^{p} u = \underbrace{u \cdot u \ldots u}_{p \text{ veces}}$.
- Cuando $p < 0$: hacemos $u \leftarrow u^{-1}$ y hacemos $u^{|p|}$. Aquí usamos la propiedad que dice que $u^{-p} = (u^{-1})^p$.

5. **Subcadenas:** Para construir el conjunto antes dicho, leeremos la cadena u caracter por caracter, para efectos visuales, tambien mostraremos elemntos repetidos.

6. **Prefijos:** Para construir este conjunto (que tendrá cardinalidad $|u|+1$), lo llenaremos primero con cadenas vacías, y por cada $i-$ésimo caracter de u, asignaremos a la $(i+1)-$ésima cadena del conjunto la concatenación de la $i-$ésima cadena con el $i-$ésimo caracter de u.

7. **Sufijos:** Para construir este conjunto (que tendrá cardinalidad $|u|+1$), lo llenaremos primero con cadenas vacías, y por cada $i-$ésimo caracter de u (leídos de reversa), asignaremos a la $i-$ésima cadena del conjunto la concatenación del $i-$ésimo caracter de u con la $(i+1)-$ésima cadena.

8. **Palindromo:** Solo verificaremos la proposicion $u = u^{-1}$.

3. Implementación de la solución

```
"""
    Practica 1: Operaciones con cadenas
    autor: Angel Lopez Manriquez
    grupo: 2CV1
    fecha: 23/02/2018
    ================================================================================
        Programa de consola que permite realizar las siguientes operaciones:

        -longitud
        -concatenacion
        -inverso
        -potencia
        -subcadenas
        -prefijos
        -sufijos
        -palindromo

    Caracteristicas ------------------------------------------------------------
        Se hace uso de los metodos de string de python, tanto de slice (denotada por
    corchetes [inicio:final:pasos]) como len que retorna la longitud de objeto.

        La mayoria de los metodos piden al usuario una o dos cadenas para trabajar,
    dependiendo de la situacion.

    Uso ------------------------------------------------------------------------
    python3 practice1.py
```

```
    (usar python practice1.py si no funciona la anterior o bien de doble clic)
"""

def length():
    """ Imprime la longitud de una cadena dada, no retornamos nada. """
    u = input('ingrese la cadena: ')
    print('longitud: ', len(u))

def concat():
    """ Concatena dos cadenas dadas y las muestra, no regresa nada. """
    u = input('ingrese la primer cadena: ')
    v = input('ingrese la segunda cadena: ')
    print('la cadena concatenada es', u + v)

def inverse():
    """ Mostramos una cadena dada al reves. No retorna nada.

        Caracteristicas:
            Si w es una palabra entonces w[::-1] retornara w ^ {-1}
            aqui hacemos uso de el "extended slice syntax" """
    u = input('ingrese la cadena: ')
    print('La cadena invertida es: ', u[::-1])

def potency():
    """ Mostramos la potencia de una cadena dada, no retorna nada.

        Caracteristicas:
            Sea w una palabra, n un entero, entonces el interprete de
            python deducira que w * n = w + w + ... + w (n-veces)
            para un n >= 0"""
    u = input('ingrese la cadena: ')
    n = int(input('ingrese el entero: '))
    if n < 0:
        u = u[::-1]
        n = -n
    if n == 0:
        print('La cadena es epsilon')
    else:
        print('La cadena potencia es: ', u * n)

def sliding(w, n):
    """ Brinda todas las secuencias de longitud n de una palabra

        Caracteristicas:
            Con el metodo append agregamos elementos a una lista, si w es una palabra
            entonces w[a:b] con a <= b, ambos enteros, se retorna una subcadena
            empezando por a (incluyendo) y terminando hasta b (excluyendo).

        Ejemplos:
            >>> 'hola'[1:3]
```

```
77              'ol'
78              >>> sliding("hola", 1)
79              [ 'h', 'o', 'l', 'a' ]
80              >>> sliding("hola", 2)  [ 'ho', 'ol', 'la' ]
81
82          parametros:
83              w <- string
84              n <- int
85
86          retorna:
87              una lista de subcadenas de w de longitud n """
88      words = list()
89      for i, j in enumerate(range(n, len(w) + 1)):
90          words.append(w[i:j])
91      return words
92
93  def substring():
94      """ Muestra todas las subcadenas de una cadena dada, no retorna nada
95
96          Caracteristicas
97              Hacemos uso de la funcion sliding """
98      w = input('ingrese la cadena: ')
99      print('epsilon') # epsilon es subcadena para cualquier cadena
100     for i in range(1, len(w) + 1):
101         for u in sliding(w, i):
102             print(u)
103
104 def prefixes():
105     """ Muestra todos los prefijos de una cadena dada """
106     w = input('ingrese la cadena: ')
107     print('epsilon')
108     for j in range(len(w) + 1):
109         print(w[0:j])
110
111 def sufixes():
112     """ Muestra todos los sufijos de una cadena dada """
113     w = input('ingrese la cadena: ')
114     print('epsilon')
115     for i in range(len(w) + 1):
116         print(w[i:len(w) + 1])
117
118 def is_palindrome():
119     """ Mostramos si dos cadenas dadas son iguales """
120     u = input('ingrese la primer cadena: ')
121     if u == u[::-1]:
122         print('La cadena es palindroma')
123     else:
124         print('La cadena no es palindroma')
125
126 def main(sel = 'y'):
```

```
127     """ Funcion en donde preguntamos al usuario alguna accion a realizar sobre una
128         cadena dada.
129 
130         Caracteristicas
131             Mediante recursion nos mantendremos preguntando al usuario
132             hasta que ya quiera detener el script
133 
134         parametros
135             sel: string = Variable bandera, mientras no sea 'n' seguimos corriendo
136                 el programa.
137 
138         Retorna
139             None """
140     if sel == 'y' or sel == 'Y':
141         options = [ 'Longitud', 'concatenacion', 'inverso', \
142                     'potencia', 'subcadenas', 'prefijos', \
143                     'sufijos', 'palindromo' ]
144         print() # se imprime un salto de linea
145         # enumerate retorna una tupla enumerada de una coleccion
146         for i, option in enumerate(options):
147             print(i + 1, option)
148         sel = input("\nSeleccione: ")
149         { # emulacion de un switch
150             '1' : length,
151             '2' : concat,
152             '3' : inverse,
153             '4' : potency,
154             '5' : substring,
155             '6' : prefixes,
156             '7' : sufixes,
157             '8' : is_palindrome
158         # Se mostrara 'caracter invalido' si no se escoge [1-8]
159         }.get(sel, lambda: print('caracter invalido'))()
160         sel = input('Desea continuar? (Y/n): ')
161         main(sel)
162 
163 main('y') # Arrancamos la funcion principal
```

4. Funcionamiento

1. **Longitud:**

```
PS C:\Users\ANGEL\Documents\codes\python\theory-of-computation> python .\practice1.py

1 Longitud
2 concatenacion
3 inverso
4 potencia
5 subcadenas
6 prefijos
7 sufijos
8 palindromo

Seleccione: 1
ingrese la cadena: hola
longitud:  4
Desea continuar? (Y/n):
```

2. **Concatenación:**

```
Seleccione: 2
ingrese la primer cadena: hola
ingrese la segunda cadena: mundo
la cadena concatenada es holamundo
Desea continuar? (Y/n):
```

3. **Inverso:**

```
Seleccione: 3
ingrese la cadena: Inverso
La cadena invertida es:  osrevnI
Desea continuar? (Y/n):
```

4. **Potencia (positiva):**

```
Seleccione: 4
ingrese la cadena: pongame diez
ingrese el entero: 10
La cadena potencia es:  pongame diez pongame diez pongame diez pongame diez pongame diez pongame diez ponga
me diez pongame diez pongame diez pongame diez
Desea continuar? (Y/n): y
```

5. **Potencia (donde el exponente es 0):**

```
Seleccione: 4
ingrese la cadena: limite
ingrese el entero: 0
La cadena es epsilon
Desea continuar? (Y/n):
```

6. **Potencia (negativa):**

```
Seleccione: 4
ingrese la cadena: inverso
ingrese el entero: -4
La cadena potencia es:   osrevni osrevni osrevni osrevni
Desea continuar? (Y/n):
```

7. **Subcadenas:**

```
Seleccione: 5
ingrese la cadena: queso
epsilon
q
u
e
s
o
qu
ue
es
so
que
ues
eso
ques
ueso
queso
Desea continuar? (Y/n):
```

8. **Prefijos:**

```
Seleccione: 6
ingrese la cadena: rafaga
epsilon

r
ra
raf
rafa
rafag
rafaga
Desea continuar? (Y/n):
```

9. **Sufijos:**

```
Seleccione: 7
ingrese la cadena: computadora
epsilon
computadora
omputadora
mputadora
putadora
utadora
tadora
adora
dora
ora
ra
a

Desea continuar? (Y/n):
```

10. **Palindromo:**

```
Seleccione: 8
ingrese la primer cadena: anitalavalatina
La cadena es palindroma
Desea continuar? (Y/n):
```

```
Seleccione: 8
ingrese la primer cadena: python
La cadena no es palindroma
Desea continuar? (Y/n): n
PS C:\Users\ANGEL\Documents\codes\python\theory-of-computation>
```

5. Instrucciones de ejecución

`python practice.py` o bien darle doble clic al archivo.

6. Conclusiones

Pude reforzar mis conocimientos respecto a los fundamentos del algebra de cadenas y un mejor uso del tratamiento de cadneas en python para futuras practicas. Tambien se concluye que una cadena w tiene a lo mucho $|w|(|w|+1)/2+1$ subcadenas.