Clase 8 – Strings

IIC1103 Sección 9 – 2018-2

Profesor: Felipe López

26-09-2018

Contenidos

1. Concatenación
2. Secuencias de caracteres
3. Función `len()`
4. *Slice*
5. `for` sobre strings
6. Ejercicios

Strings

- Recordemos qué es un *string.*
- Como vimos anteriormente, hay variables que pueden almacenar texto, por ejemplo:

```
variable_de_texto = “texto de ejemplo”
```

- Estas variables tienen tipo de dato texto, o como se le denomina en Python, *strings*.
- Para variables de tipo entero o decimal (*int* o *float* respectivamente), existen gran variedad de operadores (+,-,/,*,librería *math*, etc.).
- La pregunta es entonces ¿Cómo podemos operar con variables de texto *strings?*

Concatenación

- Para poder concatenar dos *strings* (unir), se puede ocupar el operador +. Por ejemplo:

```
variable_de_texto1 = "Podemos unir"
variable_de_texto2 = "dos textos distintos"

concatenacion = variable_de_texto1 + variable_de_texto2
print(concatenacion)
```

```
Podemos unirdos textos distintos
```

- El resultado de la concatenación es “Podemos unirdos textos distintos”. Notemos que la concatenación no agrega ningún espacio o algo similar entre dos *strings* distintos.

Concatenación

- Se puede unir más de dos *strings distintos*. Por ejemplo

```
variable_de_texto1 = "Podemos unir"
variable_de_texto2 = "dos textos distintos"

concatenacion = variable_de_texto1 + " " + variable_de_texto2
print(concatenacion)
```

```
Podemos unir dos textos distintos
```

- En este ejemplo, unimos tres *strings* distintos con el operador +.

Strings: Secuencias de caracteres

- Digamos que queremos saber cuál es la primera letra de un *string* ¿Cómo podríamos hacerlo?
- En realidad, un *string* es una secuencia de caracteres. Por ejemplo, el *string* "hola" en realidad es una concatenación de los caracteres "h", "o", "l", "a".
- No solo es la concatenación de estos caracteres, sino que además cada uno de ellos tiene una posición en específico.

Caracter →	h	o	l	a
Posición →	0	1	2	3

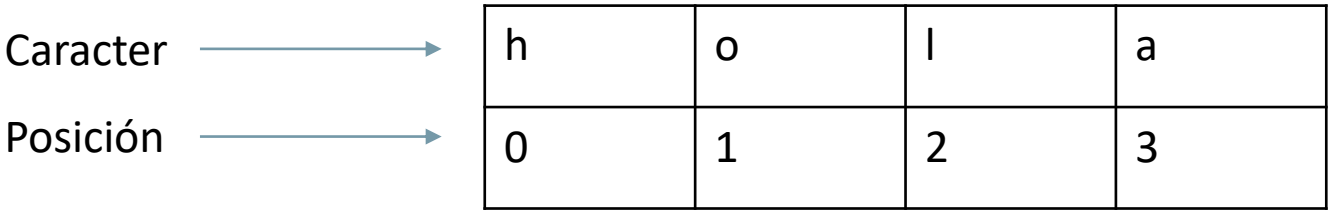
Strings: Secuencias de caracteres

- Digamos que queremos saber cuál es la primera letra de un *string* ¿Cómo podríamos hacerlo?
- En realidad, un *string* es una secuencia de caracteres. Por ejemplo, el *string* “hola” en realidad es una concatenación de los caracteres “h”,”o”,”l”,”a”.
- No solo es la concatenación de estos caracteres, sino que además cada uno de ellos tiene una posición en específico.

Caracter →	h	o	l	a
Posición →	0	1	2	3

Notemos que los caracteres se empiezan a contar desde el 0, y no desde el 1.

Strings: Secuencias de caracteres



- Por lo tanto, para saber cuál es la primera letra de este *string*, necesitamos saber qué letra está en la posición 0.

- O si quisiéramos saber cuál es la tercera letra de la palabra, simplemente tenemos que saber qué letra está en la posición 2.

Strings: Secuencias de caracteres

Caracter →	h	o	l	a
Posición →	0	1	2	3

- Para hacer esto en Python, veamos el siguiente ejemplo:

```
s = "hola"
print(s[0])
```

```
h
```

En este caso, nuestra variable *string* que tiene el texto “hola” se llama s. Podemos extraer la primera letra en la posición 0 ocupando [0], y así para cualquier posición.

- Notemos que para extraer la letra de cualquier posición, ocupamos la siguiente notación:

```
nombre_variable_texto[posición]
```

Strings: Secuencias de caracteres

- Veamos un ejemplo. Queremos juntar la cuarta y segunda letra de dos textos distintos e imprimirlas en la consola. Las palabras son:
  - “Pelota”
  - “Perla”

```
texto1 = "Pelota"
texto2 = "Perla"

print(texto1[1] + texto2[3])
```

```
el
```

Strings: Secuencias de caracteres

- Veamos un ejemplo. Queremos juntar la cuarta y segunda letra de dos textos distintos e imprimirlas en la consola. Las palabras son:
  - “Pelota”
  - “Ala”

```
texto1 = "Pelota"
texto2 = "Perla"

print(texto1[2] + texto2[4])
```

```
el
```

- Podemos ocupar el operador + para unir ambas letras. Esto quiere decir que para Python, una letra, o un carácter, **también se considera como texto**.

Strings

Ahora, queremos crear un programa que tome el texto ingresado por un usuario e imprima la última letra.

Strings

Ahora, queremos crear un programa que tome el texto ingresado por un usuario e imprima la última letra.

El problema es que no sabemos a prior cuál es el largo del texto que ingresará el usuario (porque puede ingresar el texto que él desee), por lo tanto no sabemos cuál es la posición de la última letra.

Strings

Ahora, queremos crear un programa que tome el texto ingresado por un usuario e imprima la última letra.

El problema es que no sabemos a prior cuál es el largo del texto que ingresará el usuario (porque puede ingresar el texto que él desee), por lo tanto no sabemos cuál es la posición de la última letra.

Para poder saber el largo de cualquier variable de tipo *string*, ocuparemos la función *len.* Esta función no pertenece a ningún paquete ni tampoco hay que definirla, sino que Python ya la tiene incorporada.

Función `len()`

- La función `len()` nos indica cuál es la cantidad de caracteres que contiene un texto. Se ocupa de la siguiente manera:

```
s=“Texto”
largo_del_texto = len(s)
```

Donde la variable largo_del_texto tendrá el valor entero 5.

Es decir, podemos ocupar la función incluyendo cualquier variable dentro del paréntesis y el valor de retorno será la cantidad de caracteres de ese *string.*

Función `len()`

- Veamos un ejemplo:

```
string_de_ejemplo = "Hola"
print(len(string_de_ejemplo))

string_de_ejemplo2 = "¿Hola cómo estás?"
print(len(string_de_ejemplo2))
```

```
4
17
```

Ejercicio ejemplo

- Volvamos al ejercicio propuesto anteriormente. Queremos imprimir en consola la última letra de un texto ingresado por el usuario, y nuestro problema era que no sabíamos a priori cuál sería el largo de este texto.
- Una solución posible sería la siguiente:

```
texto_ingresado_por_el_usuario = input("Ingrese un texto cualquiera. Python imprimirá el último caracter de este texto\n")

largo_texto = len(texto_ingresado_por_el_usuario)

ultimo_caracter = texto_ingresado_por_el_usuario[largo_texto-1]

print(ultimo_caracter)
```

Ejercicio ejemplo

- Analicemos la respuesta:

```
Ingrese un texto cualquiera. Python imprimirá el último caracter de este texto
 PALABRA
A
```

- Es importante notar que la posición del último carácter está dada por largo_texto menos 1. Esto es porque las posiciones se empieza a contar desde el 0.

Carácter	P	A	L	A	B	R	A
Posición	0	1	2	3	4	5	6

- El largo de la palabra en Python es 7. Por lo tanto, si queremos extraer la última letra, tenemos que buscar el carácter que está el posición 6 (largo de la palabra menos 1).

Slice

Digamos que tenemos el siguiente *string*, que representa el nombre completo de una persona:

"Andrés José Pérez Rojas"

No obstante, solo queremos saber el primer apellido ¿Cómo podríamos resolverlo? Una posible solución es extraer letra a letra del apellido, pero es muy ineficiente.

Para eso podemos ocupar el concepto de *slice*. Esta acción nos permite extraer una secuencia de caracteres de un *string*.

Slice

Esta acción permite extraer una secuencia de caracteres de un *string.* Se ocupa de la siguiente manera. Digamos que tenemos un string cualquier en la variable *s*:

```
s[i:j]
```

Donde esta acción extrae los caracteres que se encuentran desde la posición *i* (posición inicial) hasta la posición *j-1* (posición final).

Slice

Veamos un ejemplo:

```
string_de_ejemplo = "texto de ejemplo"
primera_palabra = string_de_ejemplo[0:5]
print(primera_palabra)

segunda_palabra = string_de_ejemplo[6:8]
print(segunda_palabra)

ultima_palabra = string_de_ejemplo[9:16]
print(ultima_palabra)
```

```
texto
de
ejemplo
```

Slice

Veamos un ejemplo:

```
string_de_ejemplo = "texto de ejemplo"
primera_palabra = string_de_ejemplo[0:5]
print(primera_palabra)

segunda_palabra = string_de_ejemplo[6:8]
print(segunda_palabra)

ultima_palabra = string_de_ejemplo[9:16]
print(ultima_palabra)
```

```
texto
de
ejemplo
```

T	E	X	T	O		D	E		E	J	E	M	P	L	O
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Slice

Volvamos al ejemplo propuesto inicialmente. el siguiente *string*, que representa el nombre completo de una persona:

"Andrés José Pérez Rojas"

Queremos extraer el primer apellido ¿Cómo lo podemos hacer?

```
nombre = "Andrés José Pérez Rojas"
print(nombre[12:17])
```

```
Pérez
```

`for` sobre *strings*

Ahora nos enfrentamos al siguiente problema: Queremos imprimir todos los caracteres de un *string*. Nuevamente, podríamos imprimir carácter según su posición, pero es muy ineficiente.

Para eso, podemos ocupar un `for`. Python nos permite ocupar un `for` para poder "recorrer" todos los caracteres de un *string*.

`for` sobre *strings*

Por ejemplo, digamos que tenemos el siguiente código:

```
s="ejemplo"
for i in s:
        print(i)
```

```
e
j
e
m
p
l
o
```

En el *for* la variable *i* representa a cada uno de los caracteres del *string* s. Por lo tanto, con `print(i)` imprimimos en consola cada uno de los caracteres del *string* s.

`for` sobre *strings*

No solo podemos crear un `for` que recorra cada uno de los caracteres de un *string* como mostramos anteriormente, sino que también podemos hacerlo por el índice. Por ejemplo:

```
s="ejemplo"
for i in range(0,len(s)):
    print(s[i])
```

En este caso, `i` no representa al carácter mismo, sino que a las posiciones que toma cada uno de los caracteres del *string*.

break

5 minutos

Ejercicio 1

Escriba una función que en base a un string s1 y un caracter s2, determine la posición del caracter s2 dentro del string s1. En caso de que s2 no se encuentre dentro de s1, entonces la función debe devolver -1.

Ejercicio 1

Escriba una función que en base a un string s1 y un caracter s2, determine la posición del caracter s2 dentro del string s1. En caso de que s2 no se encuentre dentro de s1, entonces la función debe devolver -1.

```
s1 = "palabra"
s2 = "b"

def contiene(s1,s2):
  for i in range(0,len(s1)):
    if s1[i] == s2[0]:
      return i
  return -1

print(contiene(s1,s2))
```

Ejercicio 2

Escriba una función que determine si un string es palíndromo. Es decir, que sea lea de izquierda a derecha igual que de derecha a izquierda

Ejercicio 2

Escriba una función que determine si un string es palíndromo. Es decir, que sea lea de izquierda a derecha igual que de derecha a izquierda

```
s1 = "anitalavalatina"

def esPalindromo(s1):
  for i in range(0,len(s1)):
     if s1[i] != s1[len(s1)-i-1]:
      return False
  return True

print(esPalindromo(s1))
```

Ejercicio 3

Escriba una función que intercale dos strings distintos y devuelva un string con la palabra intercalada. En caso que uno sea más largo que el otro, entonces se deben intercalar las palabras hasta que se acaben los caracteres del string más corto y luego añadir lo que queda de la palabra más grande.

Ejercicio 3

```
def intercalar(s1,s2):
  str_aux = ""
  if len(s1) == len(s2):
    for i in range(0,len(s1)):
      str_aux += s1[i] + s2[i]
  elif len(s1) < len(s2):
    for i in range(0,len(s1)):
      str_aux += s1[i] + s2[i]
    str_aux += s2[i+1:len(s2)]
  elif len(s1) > len(s2):
    for i in range(0,len(s2)):
      str_aux += s1[i] + s2[i]
    str_aux += s1[i+1:len(s1)]
  return str_aux
```

Resumen de la clase

- **Concatenar:** Para unir dos o más *strings* distintos, se ocupa el operador "+".

```
string s1
string s2
concatenacion = s1 + s2
```

Los *strings* pueden ser literales, no solo variables. Notemos que textos de un solo caracter y con caracteres especiales (!,#,@, etc.) **también son *strings*.**

- **Secuencias de caracteres:** Un *string* es una secuencia de caracteres. Cada caracter tiene una posición específica.

Caracter →	h	o	l	a
Posición →	0	1	2	3

Para un *string* `s1`, podemos extraer cualquier carácter de la siguiente manera:

```
s1[k]
```

Siendo k una posición dentro del string.

Resumen de la clase

- **Largo de un string:** Para poder saber cuál es el largo de un *string,* ocupamos la función `len()`.

Ejemplo:

```
s1 = “hola”
len(s1) #igual a 4
```

- ***Slice*:** Es posible extraer “un pedazo” de un string mediante la operación *slice*.

Para un *string* `s1`, podemos extraer una secuencia de caracteres dentro de este string, de la siguiente manera:

```
s1[i:j]
```

Tomando los caracteres entre `i` y `j-1`.

Resumen de la clase

- **For sobre un *string*:** Podemos “recorrer” todos los caracteres de un string mediante un `for`.

Ejemplo:

```
s1 = “hola”
for i in s1:
    print(i) #i representa a cada uno de los caracteres de
             #s1 y en este caso se imprimen en la
             #consola.
```

Resumen de la clase

- **For sobre las posiciones de un *string*:** Podemos “recorrer” todas las posiciones de un string

```
s="ejemplo"
for i in range(0,len(s)):
    print(s[i])
```

Donde la variable i representa a todas las posiciones del string s

Bibliografía

- http://runest.ing.puc.cl/string.html
- A. B. Downey. Think Python: How to think like a computer scientist. Green Tea Press, 2013 -> Capítulo 8

Links

- https://repl.it/@FelipeLopez/IIC1103StringsI que contiene todos los ejemplos de la clase.

Clase 8 – Strings

IIC1103 Sección 9 – 2018-2

Profesor: Felipe López

26-09-2018