Universidad Católica

San Pablo

Cadenas y Listas

Programación Básica con Python

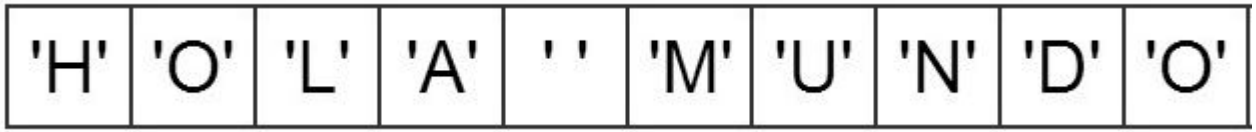
Graciela Meza Lovon, Yessenia Yari Ramos, Alvaro Mamani Aliaga

Contenido Cadenas

- Cadenas
  - Un tipo de dato compuesto
  - Recorridos en cadenas
  - Segmentos de cadenas

Cadenas: Un tipo de dato compuesto

- Las cadenas son diferentes de los otros tipos int y float porque están compuestas de piezas más pequeñas llamadas caracteres y por eso se llaman tipos de datos compuestos.

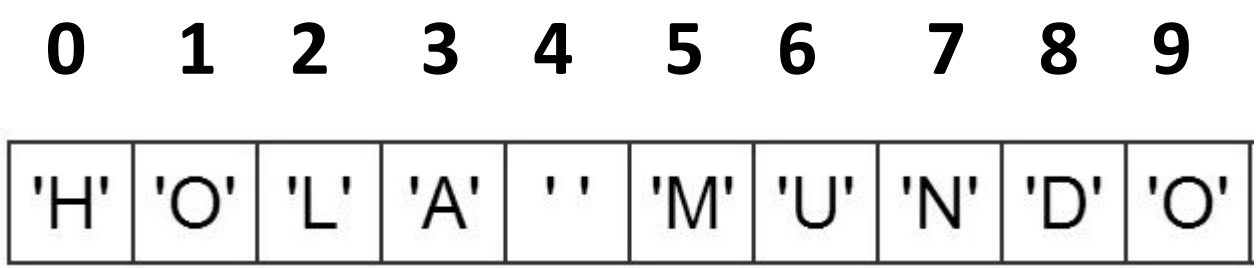



```
cadena ="HOLA MUNDO"
```

Cadenas: Un tipo de dato compuesto

- Dependiendo de lo que hagamos podemos tratar un tipo compuesto como unidad o podemos acceder a sus partes.
  - Si se desea acceder a un carácter de una cadena se usa el **operador corchete []** y el **índice** deseado.
  - Un **índice** especifica un elemento de un conjunto ordenado, en este caso el conjunto de caracteres de la cadena.

Cadenas: Un tipo de dato compuesto




```
cadena ="HOLA MUNDO"
print(cadena)
letra =cadena[3]
print(letra)
```

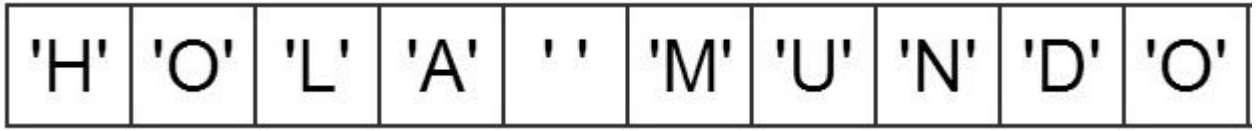
```
Salida:
HOLA MUNDO
A
```

Cadenas: Un tipo de dato compuesto

- Longitud de una cadena:
  - La función **len** retorna el número de caracteres en una cadena y nos ayuda a encontrar la última letra de la cadena.

Cadenas: Un tipo de dato compuesto

Índices: 0 1 2 3 4 5 6 7 8 9




```
cadena ="HOLA MUNDO"
longitud =len(cadena)
print(longitud)
print(cadena[longitud-1])
print(cadena[len(cadena)-1])
```

```
Salida:
10
O
```

Cadenas: Recorridos en cadenas

- Se llama recorrido al procesamiento de una cadena caracter por caracter
- Para ello se pueden usar:
  - La sentencia **for**
  - La sentencia **while**

Cadenas: Recorridos en cadenas

- Usando la sentencia while.
  - Ejemplo: Imprimir los caracteres de una cadena.

```
fruta = "fresa"
indice = 0
while indice < len(fruta):
    letra = fruta[indice]
    print (letra)
    indice = indice + 1
```

```
Salida:
f
r
e
s
a
```

Cadenas: Recorridos en cadenas

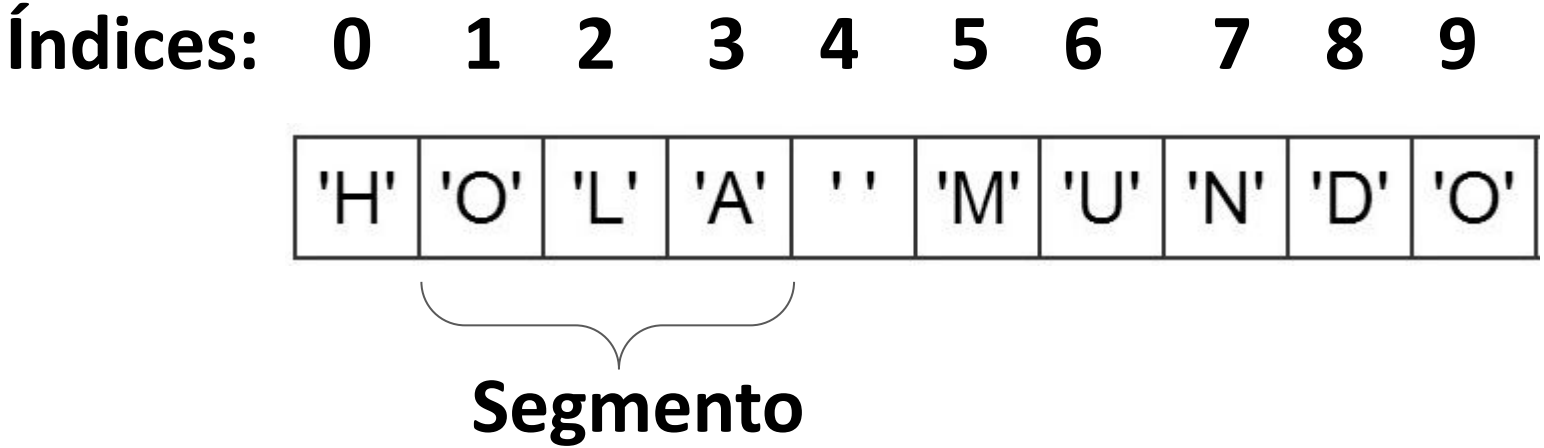
- Usando la sentencia for
  - Ejemplo: Imprimir los caracteres de una cadena.

```
fruta = "fresa"
for car in fruta:
    print (car)
```

```
Salida:
f
r
e
s
a
```

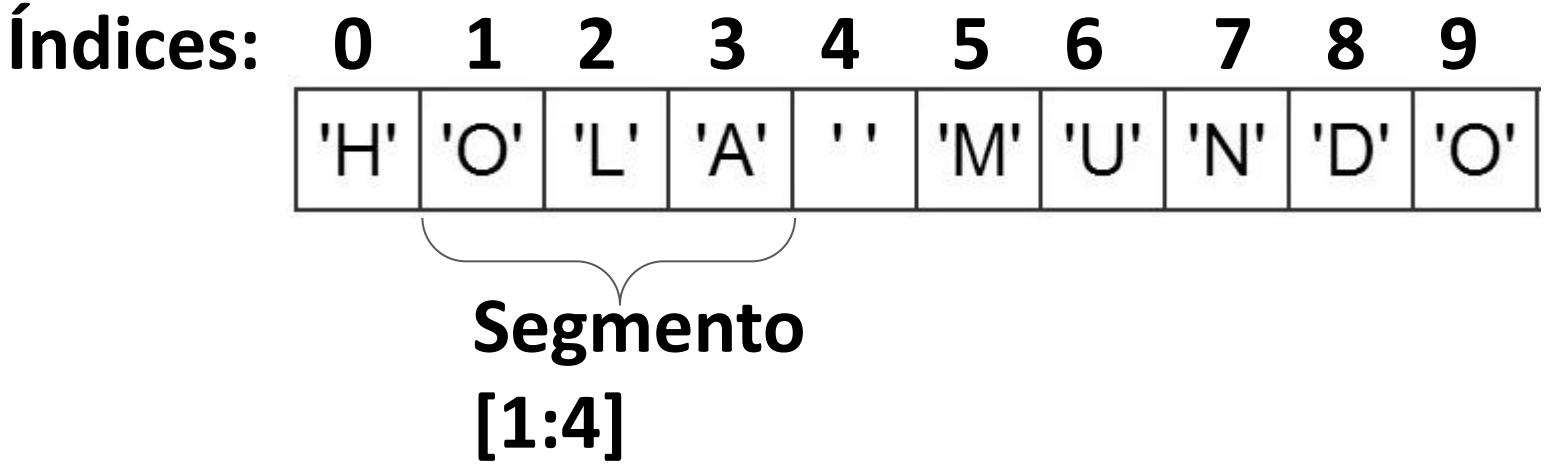
Cadenas: Segmentos de cadenas

- Un segmento de una cadena es una porción de una cadena



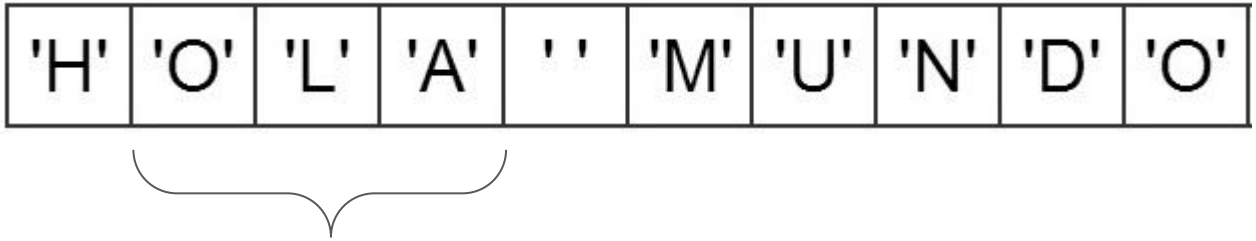
Cadenas: Segmentos de cadenas

- Para extraer un segmento se usa el operador corchete[] con el siguiente formato[n:m], que significa que el segmento extraído va desde del índice n al índice m-1.



Cadenas: Segmentos de cadenas

Índices: 0 1 2 3 4 5 6 7 8 9



Segmento [1:4]

```
cadena ="HOLA MUNDO"
print(cadena[1:4])
Salida:
OLA
```

Cadenas: Las cadenas son inmutables

- Las cadenas creadas de la forma explicada hasta ahora, son constantes es decir sus caracteres no pueden cambiar.

Cadenas: Las cadenas son inmutables

- Las cadenas creadas de la forma explicada hasta ahora, son constantes es decir sus caracteres no pueden cambiar.

```
saludo = "Hola mundo"
saludo[0] = 'J'
print(saludo)
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-35-6a2a37672926> in <module>
      1 saludo = "Hola mundo"
----> 2 saludo[0] = 'J'
      3 print(saludo)

TypeError: 'str' object does not support item assignment
```

Cadenas: Las cadenas son inmutables

- Solución: Concatenar y extraer

```
saludo = "Hola mundo"
nuevo_saludo = 'J' + saludo[1:len(saludo)]
print(nuevo_saludo)
```

```
Salida:
Jola Mundo
```

Contenido Listas

- Cadenas
  - Creación de listas
  - Acceso a sus elementos
  - Recorrido de una lista
  - Concatenación y segmentos
  - Agregar, eliminar y modificar elementos

Listas: Concepto

- Son similares a las cadenas pero pueden obtener elementos de cualquier tipo de dato.

  - Cadena:

    ```
    mensaje = "Hola mundo”
    ```

  - Lista:

    ```
    mi_lista = ["hola", 2.0, 5, [10, 20]]
    ```

Listas: Creación listas

- Para crear una lista es necesario usar “[ ]”.

```
a = [100, 2, 3, 4, 5]
b = ["hola", "como", "estas"]
c = ["hola", 2.0, 5, [10, 20]]
d = [] #lista vacía
```

Listas: Acceso a sus elementos

- Acceso: Similar a las cadenas, i.e., solo usando índices enteros
  - Si escribimos a[2.0], se produce un error
- Longitud de una lista: Similar a las cadenas, i.e., con la función *len(nombreLista)*

```
            0      1   2   3
lista = ["hola", 2.0, 5, [10, 20]]
print(lista[0])
print(lista[2])
print(len(lista))
print(lista[len(lista)-1])
```

```
Salida:
hola
5
4
[10,20]
```

Listas: Recorriendo una lista

- Recorridos: Similar a las cadenas:
  - Usando while
  - Usando for

```
lista = ["hola", 2.0, 5, [10, 20]]

# Usando for
for i in range(len(lista)):
    print(lista[i])

# Usando while
i = 0
while i < len(lista):
    print(lista[i])
    i = i + 1
```

```
hola
2.0
5
[10, 20]
```

Listas: Concatenación

- Concatenación: al igual que las cadenas se pueden concatenar listas.

```
18  lista_1 = [1, 2, 3, 4, 5, 6, 7]
19  lista_2 = [8, 9, 10]
20
21  lista = lista_1 + lista_2
22  print(lista)
23
24
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D
/Codigo2020II/cla.py
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Listas: Segm

- Segmentos d
  similar que e
  cadenas

```
26  lista = [1, "Hola", 0.3, 'a', 5, 25, 70, True]
27
28  segmento_1 = lista[2:5]
29  segmento_2 = lista[:5]
30  segmento_3 = lista[5:]
31  segmento_4 = lista[:]
32  print(segmento_1)
33  print(segmento_2)
34  print(segmento_3)
35  print(segmento_4)
36
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D:/Pyth
/Codigo2020II/cla.py
[0.3, 'a', 5]
[1, 'Hola', 0.3, 'a', 5]
[25, 70, True]
[1, 'Hola', 0.3, 'a', 5, 25, 70, True]
```

Listas: Cambiar elementos en una lista

- Las listas son mutables, por lo tanto se puede modificar un valor ya establecido.

```
39  lista = [1, "Hola", 0.3, 'a', 5, 25, 70, True]
40
41  print("Lista antes de la modificación")
42  print(lista)
43  lista[0] = 1000
44  lista[len(lista)-1] = False
45  lista[2] = 333
46  print("Lista después de la modificación")
47  print(lista)
48
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D:/Pyth
/Codigo2020II/cla.py
Lista antes de la modificación
[1, 'Hola', 0.3, 'a', 5, 25, 70, True]
Lista después de la modificación
[1000, 'Hola', 333, 'a', 5, 25, 70, False]
```

Listas: Agregar elementos

- Las listas son mutables, por lo tanto se puede agregar elementos.

```
50  lista = [1, "Hola", 0.3, 'a', 5, 25, 70, True]
51
52  print(lista)
53  lista[0:0] = ["Nuevo_1"]
54  print(lista)
55  lista[3:3] = ["Nuevo_2"]
56  print(lista)
57
58
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D:/Python3-
/Codigo2020II/cla.py
[1, 'Hola', 0.3, 'a', 5, 25, 70, True]
['Nuevo_1', 1, 'Hola', 0.3, 'a', 5, 25, 70, True]
['Nuevo_1', 1, 'Hola', 'Nuevo_2', 0.3, 'a', 5, 25, 70, True]
```

Listas: Eliminar elementos

- Las listas son mutables, por lo tanto se puede eliminar elementos.

```
50  lista = [1, "Hola", 0.3, 'a', 5, 25, 70, True]
51
52  print(lista)
53  lista[0:1] = []
54  print(lista)
55  lista[3:4] = []
56  print(lista)
57
58
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D:/Python
/Codigo2020II/cla.py
[1, 'Hola', 0.3, 'a', 5, 25, 70, True]
['Hola', 0.3, 'a', 5, 25, 70, True]
['Hola', 0.3, 'a', 25, 70, True]
```

Listas Anidadas

- Una lista anidada aparece como elemento dentro de otra lista.
- Las matrices se pueden implementar como listas anidadas, es decir, como una lista de listas.

2	300	4
5	6	7
8	9	0
-1	0	1

```
matriz=[[2,300,4],[5, 6, 7],[8, 9, 0],[-1, 0, 1]]
```

Matrices

- Es una lista de listas, es decir una lista anidada.
- Se pueden realizar operaciones como:
  - Número de filas: *len(nombreMatriz)*
  - Número de columnas: *len(nombreMatriz[0])*
  - Acceder a un dato : *nombreMatriz[Fila][Columna]*

```
matriz=[[2,300,4],[5, 6, 7],[8, 9, 0],[-1, 0, 1]]

for i in range(len(matriz)):
    for j in range(len(matriz[i])):
        print(matriz[i][j], end='\t')
    print()
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
PS D:\ACursosUcsp\ProgVideoJuegos\Codigo2020II> & D:/P
/Codigo2020II/cla.py
2       300     4
5       6       7
8       9       0
-1      0       1
```

¡Gracias!