IIC1103 - TAV

Clase 6 módulo 1: Manejo de strings

Prof. Francisca Cattan

Universidad Católica de Chile

11 de enero, 2023

¿Que sabemos hasta ahora?

1. Variables y expresiones.
2. Control condicional
3. Control de flujo
4. Funciones
5. Recursión
6. Strings

Strings

Strings ó `str`

Un string es una cadena de caracteres.

Strings

Strings ó `str`

Un string es una cadena de caracteres.

¿Qué sabemos de strings hasta ahora?

Strings

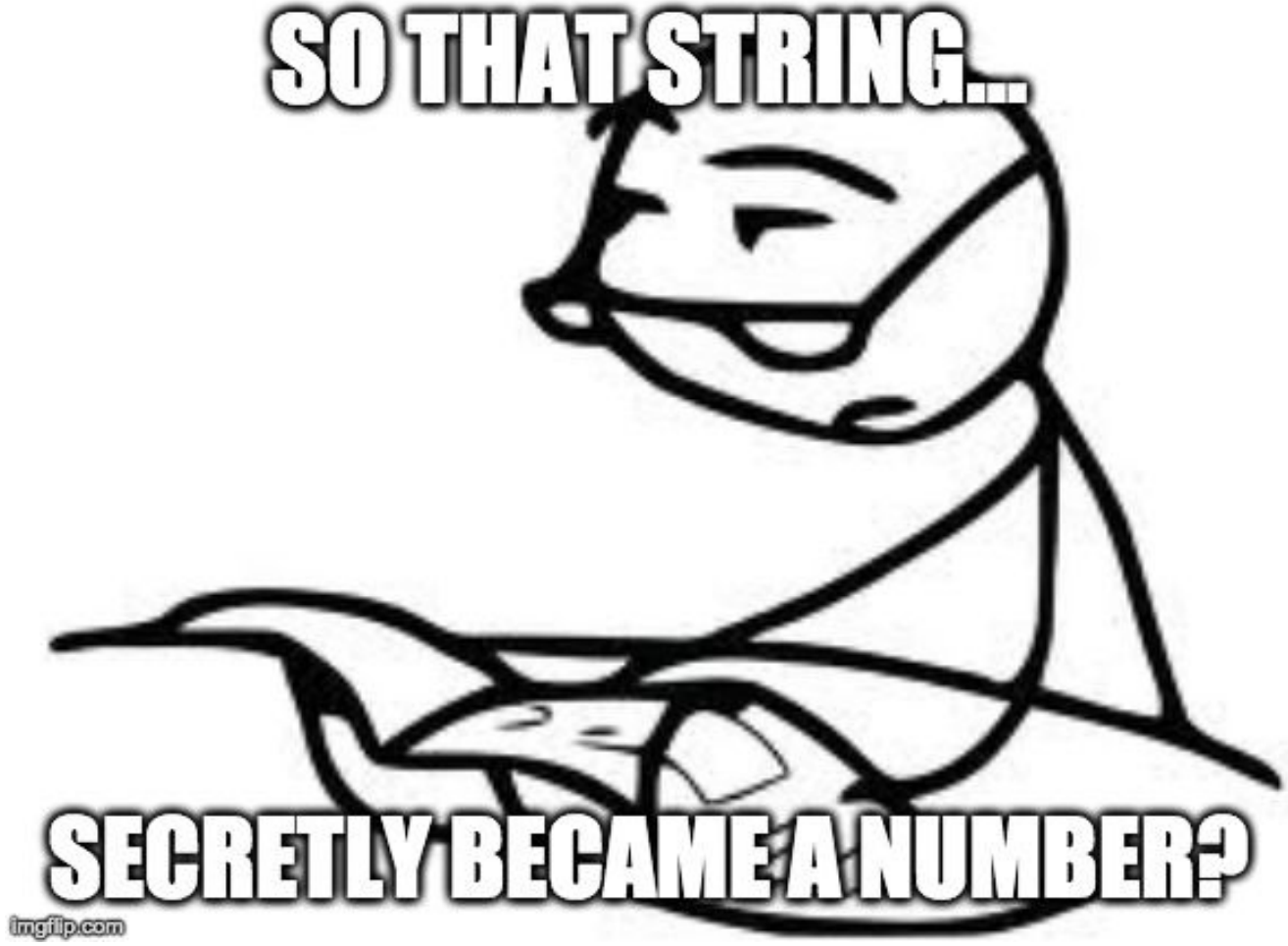
Strings ó `str`

Un string es una cadena de caracteres.

¿Qué sabemos de strings hasta ahora?

- Definir strings.
- Imprimir strings.
- Concatenar strings.
- Conversión de strings.




SO THAT STRING...
SECRETLY BECAME A NUMBER?
imgflip.com

Definir strings

```
1 # Comillas simples dentro de comillas dobles
2 print("Homero: 'Hable mas fuerte que tengo una toalla'")
3 # Comillas dobles dentro de comillas simples
4 print('Homero: "Hable mas fuerte que tengo una toalla"')
5 # Caracter de escape \'
6 print('Homero: \'Hable mas fuerte que tengo una toalla\'')
7 # Caracter de escape \"
8 print("Homero: \"Hable mas fuerte que tengo una toalla\"")
```

```
Homero:  'Hable mas fuerte que tengo una toalla'
Homero:  "Hable mas fuerte que tengo una toalla"
Homero:  'Hable mas fuerte que tengo una toalla'
Homero:  "Hable mas fuerte que tengo una toalla"
```

Definir strings

El caracter "\" permite poner comillas, saltos de línea y tabs dentro de un string.

Secuencia	Significado
\'	Comilla simple
\"	Comilla doble
\n	Salto de línea
\t	Tabulador
\\	Backslash

Definir strings

```
1 s = "Luke ... \n\t \"yo soy tu padre\""
2 print(s)
```

```
Luke ...
        "yo soy tu padre"
```

Formatear strings

¿Cómo paso un número a string con formato? Es muy frecuente en el caso de imprimir datos de tipo decimal.

```
1 a = 3.141526535          # a es un dato tipo float
2 a_string = str(a)        # a_string es un dato tipo str
3 print(a_string)           # que se imprime???
```

```
3.141526535
```

Formatear strings

Python permite convertir un número a un string en diversos formatos. El método `format` permite completar los {} de un `str`.

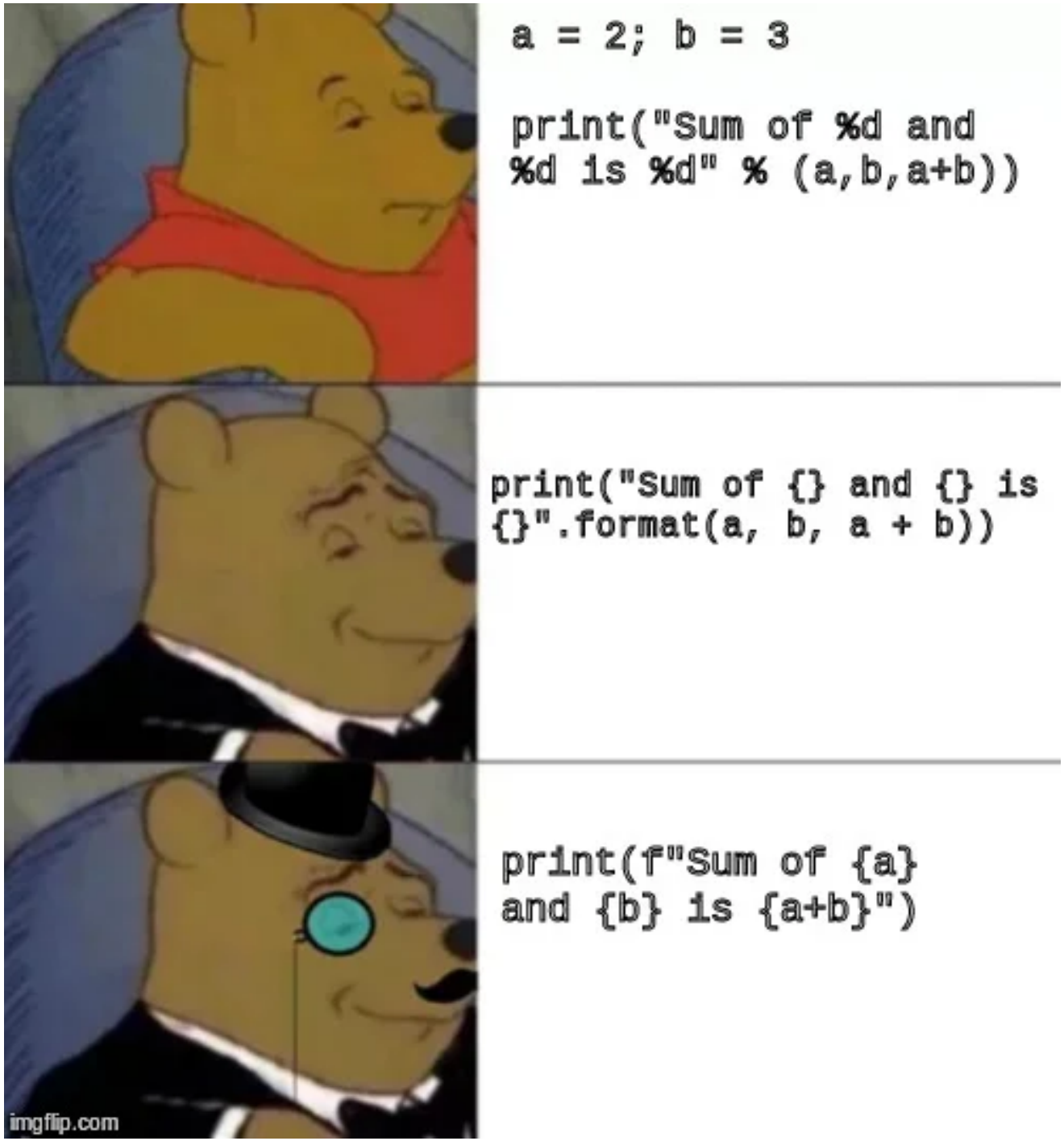
```
1  s1 = "En primer lugar queda {} y le sigue {}".format("
   Leclerc", "Perez")
2  print(s1)
3
4  t = "{0} gana con {1} puntos y {3} con {2} puntos"
5  s2 = t.format("Leclerc",26,18,"Perez")
6  print(s2)
7
8  dif = 20.526312
9  s3 = "La diferencia fue de {:.2f}s".format(dif)
10 print(s3)
```

Formatear strings

Python permite convertir un número a un string en diversos formatos. El método `format` permite completar los {} de un `str`.

```
1  s1 = "En primer lugar queda {} y le sigue {}".format("
   Leclerc", "Perez")
2  print(s1)
3
4  t = "{0} gana con {1} puntos y {3} con {2} puntos"
5  s2 = t.format("Leclerc",26,18,"Perez")
6  print(s2)
7
8  dif = 20.526312
9  s3 = "La diferencia fue de {:.2f}s".format(dif)
10 print(s3)
```

```
En primer lugar queda Leclerc y le sigue Perez
Leclerc gana con 26 puntos y Perez con 18 puntos
La diferencia fue de 20.53s
```


a = 2; b = 3
print("Sum of %d and %d is %d" % (a,b,a+b))
print("Sum of {} and {} is {}".format(a, b, a + b))
print(f"Sum of {a} and {b} is {a+b}")
imgflip.com

Formatear strings

Un mecanismo muy nuevo de formateo de string es f-string, que simplifica la interpolacion. Pare crear un f-string antepone una letra `f` al inicio del string y se completa el contenido a formatear dentro de los {}.

```
1 n1 = 'Leclerc'
2 n2 = 'Perez'
3 p = 18.05
4 print(f"En primer lugar queda {n1} y le sigue {n2}")
5
6 s = f"{n1} gana con {26} puntos y {n2} con {p:0.1f} puntos"
7 print(s)
```

```
En primer lugar queda Leclerc y le sigue Perez
Leclerc gana con 26 puntos y Perez con 18.1 puntos
```

Strings como secuencias

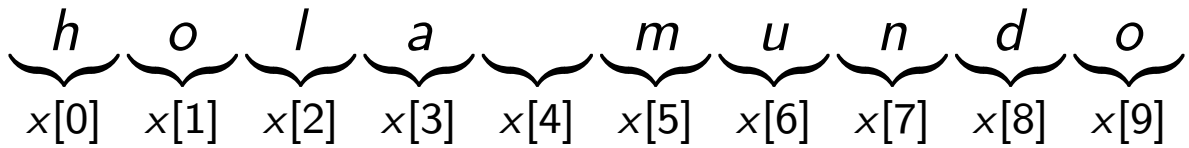
Es válido preguntarse cómo acceder a los elementos de un string.

Strings como secuencias

Es válido preguntarse cómo acceder a los elementos de un string. Supongamos que tenemos x = "hola mundo". Podemos ver los caracteres pensando al string como un vector:

Strings como secuencias

Es válido preguntarse cómo acceder a los elementos de un string. Supongamos que tenemos x = "hola mundo". Podemos ver los caracteres pensando al string como un vector:

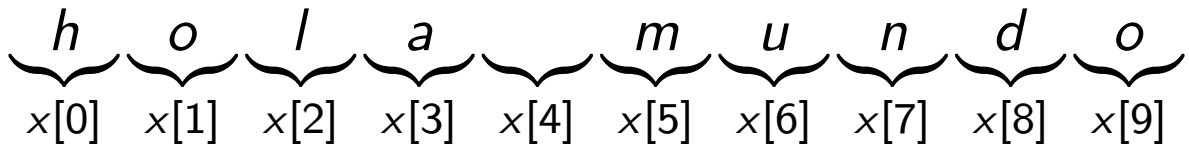


Pregunta 1: ¿Cuál es el caracter que se encuentra en la posición 6?

Pregunta 2: ¿Cómo accedemos al caracter que se encuentra en la posición 6?

Strings como secuencias

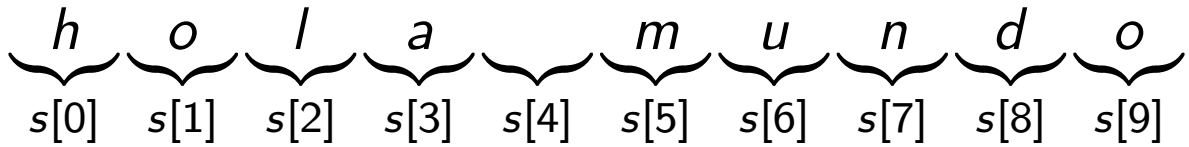
Es válido preguntarse cómo acceder a los elementos de un string. Supongamos que tenemos x = "hola mundo". Podemos ver los caracteres pensando al string como un vector:



Pregunta 1: ¿Cuál es el caracter que se encuentra en la posición 6?

Pregunta 2: ¿Cómo accedemos al caracter que se encuentra en la posición 6? R: x[6]

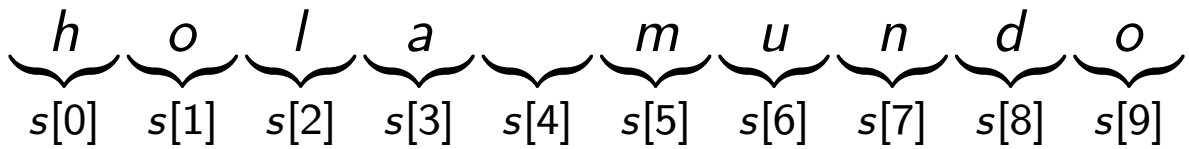
Strings como secuencias



Ahora que sabemos que podemos encontrar caracteres específicos, ¿qué pasa si usamos índices negativos?

```
1 s = "hola mundo"
2 print(s[-2])
3 print(s[-3])
4 print(s[-4])
5 print(s[-5])
```

Strings como secuencias

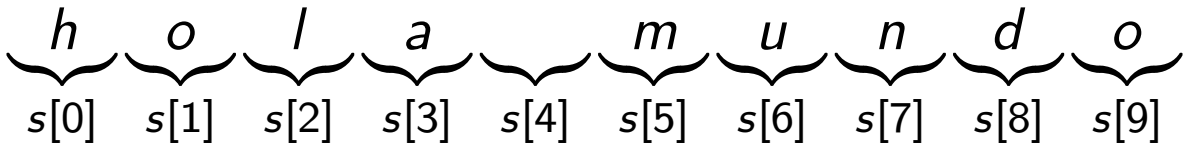


Ahora que sabemos que podemos encontrar caracteres específicos, ¿qué pasa si usamos índices negativos?

```
1 s = "hola mundo"
2 print(s[-2])
3 print(s[-3])
4 print(s[-4])
5 print(s[-5])
```

```
d
n
u
m
```

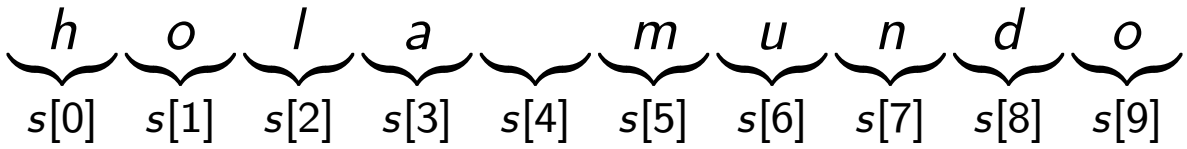
Strings como secuencias



Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

Strings como secuencias

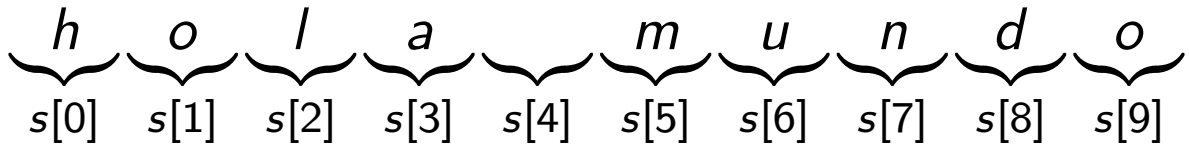


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

- o

Strings como secuencias

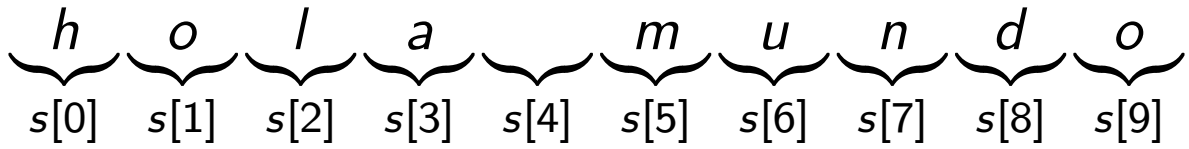


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
```

Strings como secuencias

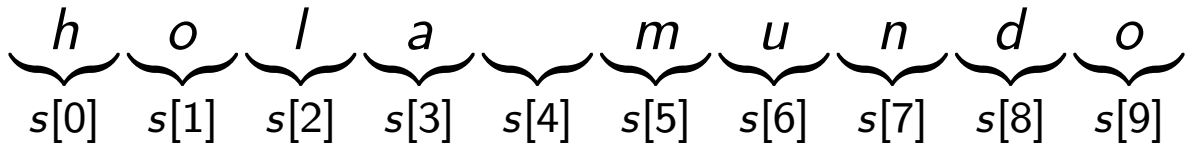


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
hola
```

Strings como secuencias

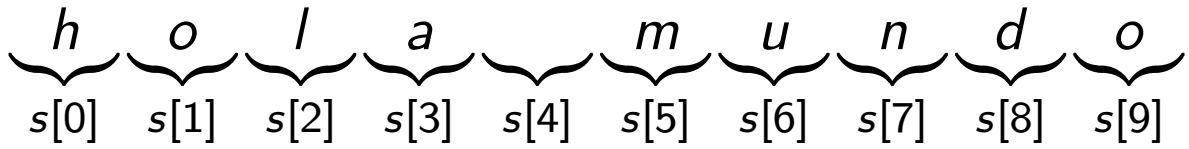


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
hola
do
```

Strings como secuencias

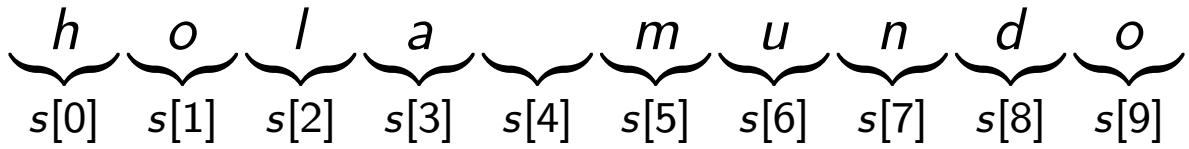


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
hola
do
au
```

Strings como secuencias

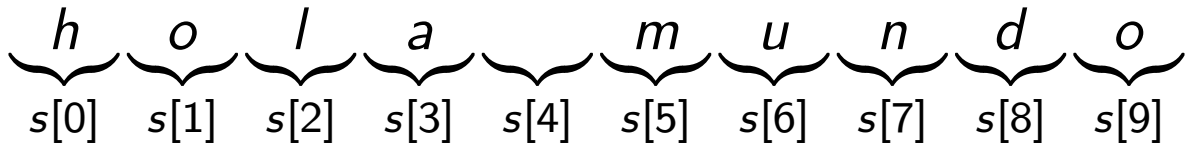


Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
hola
do
au
dnum al
```

Strings como secuencias



Probemos obtener trozos o *slices* del string.

```
1 s = "hola mundo"
2 print(s[1])
3 print(s[3:8])
4 print(s[:5])
5 print(s[8:])
6 print(s[3:8:3])
7 print(s[8:1:-1])
8 print(s[::-1])
```

```
o
a mun
hola
do
au
dnum al
odnum aloh
```

Strings como secuencias

Podemos encontrar el largo del string usando la función **len()**.

```
1 s = "paralelepipedo"
2 print(len(s))
```

```
14
```

Strings como secuencias

Podemos recorrer los caracteres de un string dentro de un for. Por ejemplo, `for i in x`, la variable `i` recorrerá todos los caracteres del string `x`. Es decir, i=h, i=o, . . . , i=d, i=o.

```
1 s = "hola mundo"
2 for i in range(len(s)):
3     print(s[i])
```

Strings como secuencias

Podemos recorrer los caracteres de un string dentro de un for. Por ejemplo, `for i in x`, la variable `i` recorrerá todos los caracteres del string `x`. Es decir, i=h, i=o, . . . , i=d, i=o.

```
1 s = "hola mundo"
2 for i in range(len(s)):
3     print(s[i])
```

```
h
o
l
a

m
u
n
d
o
```

Strings como secuencias

Podemos recorrer los caracteres de un string dentro de un for. Por ejemplo, `for i in x`, la variable `i` recorrerá todos los caracteres del string `x`. Es decir, i=h, i=o, . . . , i=d, i=o.

```
1 s = "hola mundo"
2 for i in s:
3     print(i)
```

```
h
o
l
a

m
u
n
d
o
```

Strings como secuencias

Usando el comando `in` podemos recorrer el string. También podemos verificar si un caracter pertenece a un string o no.

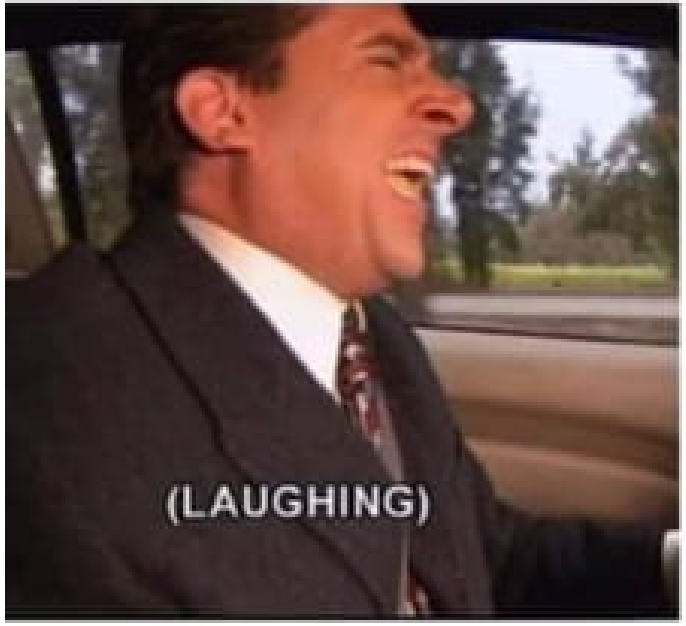
```
1 s = "hola mundo"
2 for i in s:
3     if i in 'hln':
4         print('encontrado!!')
```

```
1 a = 'hola'
2 b = 'chao'
3 print('ol' in a)
4 print('ol' not in b)
5 print('x' in a or 'y' in b)
```

True

True

False



Looking at programming memes



Actually coding

Strings

Veamos algunos ejemplos:

- Escriba un programa que retorne el largo de un string (sin usar len).
- Escriba una función que cuente la cantidad de vocales en un string.
- Escriba una función que quite los caracteres de posición impar de un string y retorne el nuevo string.
- Escriba una función que quite las vocales de un string y retorne el nuevo string.
- Escriba una función que revierta un string.