

---

# Python

## Tipo de secuencia:

# **STRINGS**

# STRINGS

---

- Una variable tipo string es una secuencia de caracteres (hilera, cadena)
- Un dato tipo string esta encerrado entre comillas dobles o simples
- Cada caracter de un string es un elemento del mismo

# INDEXACIÓN EN UN STRING

Cada carácter en el string es un elemento de dicha secuencia.

0	1	2	3	4	5	6	7	8	9	10	11
H	o	l	a	,		m	u	n	d	o	.
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> var = "Hola, mundo."  
>>> letra = var[1]  
>>> print (letra)  
o  
>>> letra = var[0]  
>>> print (letra)  
H  
>>> print(var[0],var[6], var[-9])  
H m a
```

# MEMBRESÍA EN UN STRING

---

```
>>> carreras = "E PI IA IM"
```

```
>>> "E" in carreras
```

```
True
```

```
>>> "C" in carreras
```

```
False
```

```
>>> "C" not in carreras
```

```
True
```

# TROZO DE UN STRING

```
>>> var = "Hola mundo!"
```

```
>>> var [5:10]
```

```
'mundo'
```

```
>>> var [:4]
```

```
'Hola'
```

```
>>> var [5:]
```

```
'mundo!'
```

```
>>> print(var[:])
```

```
Hola mundo!
```

```
>>> print(var[2:-3])
```

```
la mun
```

```
>>> print(var[-1:-3])    # empieza de izquierda a derecha
```

```
>>> print(var[-3:-1])
```

```
do
```

# LARGO, MÍNIMO, MÁXIMO

- Largo de un string: función **len**
- Elemento de menor valor en un string: función **min**
- Elemento de mayor valor en un string: función **max**
- Para aplicar **min** y **max** los elementos deben ser homogéneos: mismo tipo de datos, además las secuencias deben tener al menos un elemento

## ■ Ejemplos

```
>>> letras="eabczm"
```

```
>>> len(letras)
```

```
7
```

```
>>> min(letras)
```

```
'a'
```

```
>>> max(letras)
```

```
'z'
```

```
>>>
```

■ String vacío: "" → len("") → 0

# OPERADORES PARA STRINGS

---

- Concatenar strings (unir): "+"

```
>>> fruta = "Fresas"
```

```
>>> merienda = " con helado"
```

```
>>> x = fruta + merienda
```

```
>>> print (x)
```

```
Fresas con helado
```

```
>>>
```



## ■ Repetición de strings: "\*"

```
>>> "25"*4
```

```
'25252525'
```

```
>>> "JaJa"*3
```

```
'JaJaJaJaJaJa'
```

# RECORRIDO DE STRINGS

---

- Recorrer una variable de tipo string se refiere a acceder cada uno de sus caracteres
- Se pueden usar los estatutos de ciclos

## ■ Ejemplo con for-in

```
>>> prefijo = "Grupo-"
```

```
>>> sufijo = "123"
```

```
>>> for letra in sufijo:  
    print (prefijo+letra)
```

Grupo-1

Grupo-2

Grupo-3

## ■ Ejemplo con while

```
>>> carreras = "ICPIMI"
>>> i = 0
>>> largo = len(carreras)
>>> while i < largo:
    print (carreras[i:i+2])
    i = i + 2
```

IC  
PI  
MI

# STRING SON INMUTABLES

---

- Un elemento específico de una variable tipo string no puede ser cambiado, no puede estar a la izquierda en una asignación

```
>>> var = "Hola mundo!"
```

```
>>> var [10] = "."
```

Traceback (most recent call last):

File "<pyshell#46>", line 1, in <module>

var [10] = "."

TypeError: 'str' object does not support item assignment

## ■ Alternativa para hacer modificaciones:

- Sustituir la variable con un nuevo valor o concatenado strings

```
>>> var = "Hola mundo!"
```

```
>>> var = "Saludos mundo!"
```

```
>>> var = "Hola mundo!"
```

```
>>> var = "Saludos"+var[4:]
```

```
>>> print (var)
```

```
Saludos mundo!
```

# COMPARACIÓN DE STRING

---

- Los operadores de comparación (operadores relacionales) trabajan sobre strings

- Cada carácter tiene un código numérico interno asociado el cual se puede obtener con la función "ord". La comparación se basa en este código.

(blanco / números / mayúsculas / minúsculas)

- Ejemplo:

```
for c in [" ", "0", "1", "2", "A", "B", "C", "a", "b", "c"]:  
    print (ord(c), end=" ")
```

32 48 49 50 65 66 67 97 98 99



---

Ejemplos:

var= "pax"

var1= "pasa"

¿ cuál valor es mayor ?

var= "p"

var1= "papa"

¿ cuál valor es mayor ?

Función chr(expresión): da el caracter asociado al número en la expresión

>>> chr(97)

'a'

# MÉTODO “CONTAR” (“count”)

- Cuenta las veces que un substring aparece dentro de un string
- Sintaxis
  - `objetoString.count(sub[,start[,end]])`
- Ejemplo:

```
>>> var = "Hola mundo!"  
>>> print (var.count("o"))  
2
```

# MÉTODO “DIVIDIR” (“split”)

- Toma un string y lo divide en una lista de palabras
- Sintaxis
  - `objetoString.split([delimiter])`
  - Sino se da el delimitador de las palabras, Python asume que ese delimitador es el espacio en blanco
    - ✓ El delimitador no se incluye en las palabras resultantes

## ■ Ejemplos

```
>>> frase="La lluvia en Cartago"
```

```
>>> lp=frase.split()
```

```
>>> print(lp)
```

```
['La', 'lluvia', 'en', 'Cartago']
```

```
>>> lp=frase.split("a")
```

```
>>> print(lp)
```

```
['L', ' lluvi', ' en C', 'rt', 'go']
```

# MÉTODO “UNIR” (“join”)

---

- Toma una lista de strings y los une creando un solo string
- Sintaxis
  - `delimitador.join(objetoLista)`
  - El delimitador es un string que se agrega entre cada elemento del `objetoLista` para obtener el nuevo string

## ■ Ejemplos:

```
>>> lp
```

```
['La', 'lluvia', 'en', 'Cartago']
```

```
>>> frase2=" ".join(lp)
```

```
>>> frase2
```

```
'La lluvia en Cartago'
```

```
>>> frase2="--".join(lp)
```

```
>>> frase2
```

```
'La--lluvia--en--Cartago'
```

- Ejercicio: Haga la función solo\_dígitos que reciba un string y retorne otro string con solo los dígitos que contenga. Ejemplos:

```
>>> solo_dígitos("aTj823ski9")  
'8239'
```

```
>>> solo_dígitos("sdjnd")  
''
```

- 
- Otros métodos aplicados a los strings: libro de texto.