

Clase N° 6

Cadenas de Caracteres Segunda Parte

© Lic. Ricardo Thompson

Métodos para cadenas

- Los nombres de los siguientes métodos se caracterizan por comenzar con la partícula **is**.
- Se trata de métodos interrogativos, que permiten obtener información acerca de una cadena y no llevan parámetros.
- Todos los métodos que comienzan con **is** devuelven **True** o **False**.

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.isalpha()**: Devuelve True si todos los caracteres de <str> son alfabéticos (letras). Devuelve False en caso contrario. Reconoce caracteres regionales.

```
cad = 'Hola'  
print(cad.isalpha( )) # True
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.isdigit()**: Devuelve True si todos los caracteres de <str> son dígitos numéricos.

```
cad1 = '1234'  
cad2 = '3.1416'  
print(cad1.isdigit( ), cad2.isdigit( ))  
True False
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.isalnum()**: Devuelve True si todos los caracteres de <str> son alfabéticos o numéricos.

```
cad1 = '-1234'
```

```
cad2 = 'XR150'
```

```
print(cad1.isalnum( ), cad2.isalnum( ))
```

False True

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.isupper()**: Devuelve True si todos los caracteres alfabéticos de <str> están en mayúscula. Ignora los caracteres no alfabéticos.

```
cad = 'LIMA 717'
```

```
if cad.isupper( ):
```

```
    print('Está en mayúsculas') ←
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.islower()**: Devuelve True si todos los caracteres alfabéticos de <str> están en minúscula. Ignora los caracteres no alfabéticos.

```
cad = 'azúcar: 150 gramos'  
print(cad.islower( )) # True
```

© Lic. Ricardo Thompson

Métodos para cadenas

- Todos estos métodos pueden ser aplicados a una variable o a una constante de cadena de caracteres.

```
print("Hola!".isalpha( )) # False  
print("UADE".isupper( )) # True
```

- Lo mismo ocurre con los métodos siguientes.

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.upper()**: Devuelve <str> convertida a mayúsculas. Los caracteres no alfabéticos no resultan modificados.

```
cad = 'hola 123'.upper( )  
print(cad) HOLA 123
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.lower()**: Devuelve <str> convertida a minúsculas. Los caracteres no alfabéticos no resultan modificados.

```
cad = 'HOLA 123'  
print(cad.lower( )) hola 123
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.capitalize()**: Devuelve <str> convirtiendo a mayúscula el primer carácter de la cadena, y todo lo demás a minúsculas.

```
cad = 'lunes MARTES'  
cad = cad.capitalize( )  
print(cad) Lunes martes
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.title()**: Devuelve <str> convirtiendo a mayúscula el primer carácter de cada palabra, y todo lo demás a minúsculas.

```
cad = 'lunes MARTES'  
cad = cad.title( )  
print(cad) Lunes Martes
```

© Lic. Ricardo Thompson

Ejemplo 1

Ingresar por teclado el nombre de una entidad o institución y generar la sigla correspondiente, tomando la inicial de cada una de sus palabras.

**"Automóvil Club Argentino" → ACA
"Yacimientos Petrolíferos Fiscales" → YPF**

© Lic. Ricardo Thompson

```
cad = input("Ingrese una cadena: ")  
listadepalabras = cad.split()  
sigla = ""  
for palabra in listadepalabras:  
    sigla = sigla + palabra[0].upper( )  
print(sigla)
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.center(<ancho>[,<relleno>]):**
Devuelve <str> centrada en el ancho especificado. El resto de la cadena se rellena con espacios o con el carácter de relleno, si está presente.

```
cad1 = 'Hola'
cad2 = cad1.center(10,'-')
print(cad2) # ---Hola---
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.ljust(<ancho>[,<relleno>]):**
Devuelve <str> alineada a la izquierda en el ancho especificado. El final de la cadena se rellena con espacios o con el carácter de relleno, si está presente.

```
cad1 = 'Hola'
cad2 = cad1.ljust(10,'-')
print(cad2) # Hola-----
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.rjust(<ancho> [, <relleno>])**: Devuelve <str> alineada a la derecha en el ancho especificado. El comienzo de la cadena se rellena con espacios o con el carácter de relleno, si está presente.

```
cad1 = 'Hola'
cad2 = cad1.rjust(10, '-')
print(cad2) # -----Hola
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.zfill(<ancho>)**: Devuelve <str> alineada a la derecha en el ancho especificado. El comienzo de la cadena se rellena con ceros.

```
n = 3
cad = str(n).zfill(5)
print(cad) # 00003
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.lstrip([<cad>])**: Elimina cualquiera de los caracteres de <cad> del comienzo de <str>. Si <cad> se omite se eliminarán los espacios.

```
a = '--> Salida'
```

```
b = a.lstrip('->') # 'Salida'
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.rstrip([<cad>])**: Elimina cualquiera de los caracteres de <cad> del final de <str>. Si <cad> se omite se eliminarán los espacios.

```
a = 'Continuará...'
```

```
b = a.rstrip('.') # 'Continuará'
```

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.strip([<cad>]):** Elimina cualquiera de los caracteres de <cad> de ambos extremos de <str>. Si <cad> se omite se eliminarán los espacios.

```
lista = [1, 2, 3, 4]
```

```
cad1 = str(lista) # [1, 2, 3, 4]
```

```
cad2 = cad1.strip("[ ]") # 1, 2, 3, 4
```

© Lic. Ricardo Thompson

Ejemplo 2

Desarrollar un programa para imprimir una factura de venta para una empresa alimenticia.

© Lic. Ricardo Thompson

```
print("-" * 40)
print("Industrias Alimenticias S.A.".center(40))
print( )
print("Avenida del Campo 279".ljust(20), end="")
print("Tel. 46019-1146".rjust(20))
print( )
numero = 12
print("Factura N°", str(numero).zfill(8))
print( )
articulo = "94156 Aceite de oliva premium"
print(articulo.lstrip("0123456789 ").ljust(30, '.'), end="")
precio = 416
print(str(precio).rjust(10, '.'))
print("-" * 40)
```

© Lic. Ricardo Thompson

Resultado de la ejecución:

Industrias Alimenticias S.A.

Avenida del Campo 279 Tel. 46019-1146

Factura N° 00000012

Aceite de oliva premium.....416

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.find(<cad> [, <inicio> [, <fin>]]):**
Busca la primera aparición de <cad> dentro de <str>. Devuelve la posición donde se encontró. A diferencia de index, find no provoca un error si no se encontró, sino que devuelve -1.
- Pueden indicarse los subíndices donde comenzará y terminará la búsqueda.

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.rfind(<cad> [, <inicio> [, <fin>]]):**
Similar a *find*, pero busca la última aparición de <cad> dentro de <str>.
- Pueden indicarse los subíndices donde comenzará y terminará la búsqueda. Si no se los detalla se asumen los extremos de <str>.

© Lic. Ricardo Thompson

Ejemplo 3

Separar las oraciones de una frase, imprimiéndolas en líneas separadas.

La presencia de un punto marca el final de cada oración.

© Lic. Ricardo Thompson

```
frase = input("Ingrese una frase: ")
cont = 1
inicio = 0
fin = frase.find(".")
while fin != -1:
    print(str(cont).rjust(2), end=" ")
    print(frase[inicio:fin+1].strip( ).rstrip('.'))
    cont = cont + 1
    inicio = fin + 1
    fin = frase.find(".", inicio)
```

© Lic. Ricardo Thompson

Resultado de la ejecución:

Ingrese una frase: Lunes. Martes. Miércoles.

1 Lunes

2 Martes

3 Miércoles

© Lic. Ricardo Thompson

Métodos para cadenas

- **<str>.format(<datos>):** El método format es la nueva manera ofrecida por Python para darle formato a la salida impresa.
- Tiende a reemplazar a la antigua manera de formatear datos con especificadores de conversión y el operador %, brindando mayores posibilidades.

© Lic. Ricardo Thompson

Método format

- La cadena <str> debe contener **marcadores de posición**, representados a través de llaves. Éstos serán los lugares donde se insertarán los datos.

día = 20

mes = 'Abril'

a = 'Hoy es {} de {}'.format(día, mes)

Hoy es 20 de Abril

© Lic. Ricardo Thompson

Método format

- Los marcadores de posición pueden contener números dentro de las llaves, para indicar qué dato se colocará allí. Sin números se usa el orden posicional.

día = 20

mes = 'Abril'

a = 'En {1} llovió {0} días'.format(día, mes)

En Abril llovió 20 días

© Lic. Ricardo Thompson

Método format

- Para alinear números y cadenas se puede colocar un número dentro de las llaves para representar el ancho. Es necesario escribir "dos puntos" antes del número.

```
a = 15
```

```
cad = '*{:5}*'.format(a)
```

```
print(cad) # * 15*
```

© Lic. Ricardo Thompson

Método format

- En forma predeterminada los números se alinean a la derecha y las cadenas a la izquierda. Ésto puede alterarse escribiendo los símbolos <, > o ^ entre los "dos puntos" y el ancho.

<: Fuerza alineación a la izquierda

>: Fuerza alineación a la derecha

^: Fuerza alineación en el centro

© Lic. Ricardo Thompson

Método format

Alinear un número a la izquierda:

```
c = "{:<8}".format(4) # *4 *
```

Alinear una cadena a la derecha:

```
c = "{:>8}".format("Hola") # * Hola*
```

Centrar una cadena

```
c = "{:^8}".format("Hola") # * Hola *
```

© Lic. Ricardo Thompson

Método format

- Cuando se utilizan los símbolos anteriores es posible indicar cuál será el carácter de relleno.
- Este carácter se escribe entre los "dos puntos" y el símbolo de alineación.

```
c = "{:=^8}".format("Hola") # *==Hola==*
```

© Lic. Ricardo Thompson

Método format

- Para regular la cantidad de decimales de un número real se escribe un punto, la cantidad de decimales deseada y una letra f detrás de los "dos puntos".

```
c = "{:.2f}".format(3.1416)
print(c) # 3.14
```

© Lic. Ricardo Thompson

Ejemplo 4

Imprimir un triángulo con números enteros correlativos:

```
0
1 2
3 4 5
6 7 8 9
10 11 12 13 14
```

© Lic. Ricardo Thompson

```
k = 0  
for i in range(5):  
    for j in range(i+1):  
        print("{:3}".format(k), end="")  
        k += 1  
print( )
```

© Lic. Ricardo Thompson

Ejercitación

Práctica 4: Ejercicios 10 a 16

© Lic. Ricardo Thompson