

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 no llevan parámetros y permiten obtener información acerca de una cadena.
- 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), o 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

Ejemplo 1

Escribir una función para separar una palabra de los signos de puntuación que pueda tener, devolviendo tres strings: prefijo, palabra, sufijo.

```
cad = "(ingeniero),"  
limpiarpalabra(cad) → '(', 'ingeniero', ')',  
limpiarpalabra("increíble!") → '"', 'increíble', '!'
```

© Lic. Ricardo Thompson

```
def limpiarpalabra(palabra):
```

```
    # Separa la palabra de los signos de puntuación anteriores y  
    # y posteriores y devuelve tres cadenas.
```

```
    i = 0
```

```
    while i < len(palabra) and not palabra[i].isalpha( ):
```

```
        i += 1
```

```
    inicio = palabra[ :i]
```

```
    j = len(palabra) - 1
```

```
    while j > i and not palabra[j].isalpha( ):
```

```
        j -= 1
```

```
    final = palabra[j+1: ]
```

```
    palabra = palabra[i:j+1]
```

```
    return inicio, palabra, final
```

© 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 2

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: ")
```

```
# Estrategia 1
```

```
listadepalabras = cad.split( )
```

```
sigla = ""
```

```
for palabra in listadepalabras:
```

```
    sigla = sigla + palabra[0].upper( )
```

```
print(sigla)
```

```
# Estrategia 2
```

```
sigla = "".join([p[0] for p in cad.split()]).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 3

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

Nota: A los efectos del ejemplo los datos para confeccionar la factura están prefijados en el código. En una situación real estos datos se obtendrían de otra fuente, por ejemplo el teclado o mediante una consulta con una base de datos.

© 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 Nº 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

f-strings

- Las ***f-strings*** fueron introducidas a partir de la versión 3.6 de Python, que fue presentado el 23/12/2016.
- Su nombre proviene de *formatted strings*, es decir cadenas formateadas.
- Proporcionan una manera simplificada para darle formato a los datos, muy superior al operador %.

© Lic. Ricardo Thompson

f-strings

- Consisten en cadenas normales precedidas por la letra f.
- En los lugares donde deben insertarse los datos se escriben las variables correspondientes, encerradas entre llaves. A estas llaves se las denomina *marcadores de posición*.

© Lic. Ricardo Thompson

f-strings

```
dia = 16
mes = "Enero"
cad = f"Sucedió el {dia} de {mes}."
print(cad) # Sucedió el 16 de Enero.
```

© Lic. Ricardo Thompson

f-strings

- También pueden escribirse expresiones en lugar de variables.

```
print(f"{2 * 37}") # 74
```

- O invocarse funciones y métodos.

```
dia = "Lunes"  
print(f"{dia.lower()}") # lunes
```

© Lic. Ricardo Thompson

f-strings

- Para alinear números y cadenas se puede colocar un número luego del nombre de la variable, separado por "dos puntos". Este número representará el ancho.

```
num = 15  
cad = f"*{num:5}*"   
print(cad) # * 15*
```

© Lic. Ricardo Thompson

f-strings

- Escribiendo un 0 delante del ancho se rellena con ceros en lugar de espacios. Sólo funciona con el carácter 0.

```
num = 15  
cad = f"*{num:05}*" 05  
print(cad) # *00015*
```

© Lic. Ricardo Thompson

f-strings

- 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

f-strings

Ejemplos:

```
num = 15  
pal = "Hola"
```

Alinear un número a la izquierda:

```
cad = f"*{num:<8}*"          # *15      *
```

Alinear un string a la derecha:

```
cad = f"*{pal:>8}*"          # *   Hola*
```

Centrar un string

```
cad = f"*{pal:^8}*"          # *   Hola   *
```

© Lic. Ricardo Thompson

f-strings

- 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.

```
pal = "Hola"
```

```
cad = f"*{pal:=^8}*"        # *==Hola==*
```

© Lic. Ricardo Thompson

f-strings

- 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".

```
pi = 3.1416
```

```
cad = f"{pi:.2f}"
```

```
print(cad) # 3.14
```

© Lic. Ricardo Thompson

Ejemplo 4

Imprimir un triángulo de Floyd, ingresando la cantidad de filas.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

© Lic. Ricardo Thompson

```
n = int(input("Cantidad de filas? "))  
k = 1  
for i in range(n):  
    for j in range(i+1):  
        print(f"{k:3}", end="")  
        k += 1  
    print()
```

© Lic. Ricardo Thompson

Ejercitación

Práctica 4: Ejercicios 10 a 14

© Lic. Ricardo Thompson

Trabajo Práctico 4 (2º parte)

Ejercitación por equipos

Tomar el número del grupo y calcular el resto de dividirlo por 3.

- Resto 0: Ejercicios 10 y 11
- Resto 1: Ejercicios 12 y 14
- Resto 2: Ejercicio 13