

Rebanado de cadenas

Edison Achalma

Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Este abstract será actualizado una vez que se complete el contenido final del artículo.

Tabla de contenidos

Introduction	2
0.1 Motivación	2
0.2 Intuición	2
0.3 Modelo mental	2
0.4 Desarrollo	2
0.4.1 1. Sintaxis básica	2
0.4.2 2. Slice desde el inicio	2
0.4.3 3. Slice hasta el final	2
0.4.4 4. Slice completo	3
0.4.5 5. Índices negativos	3
0.4.6 6. Slice con paso ([inicio:fin:paso])	3
0.4.7 7. Tabla de ejemplos de slicing	3
0.5 Ejemplos	4
0.6 Errores comunes	4
0.7 Buenas prácticas	5
0.8 Ejercicio	6
0.9 Síntesis	7
0.10 Qué sigue	7
1 Publicaciones Similares	7

Edison Achalma  <https://orcid.org/0000-0001-6996-3364>

El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; <https://credit.niso.org/>) de la siguiente manera: Edison Achalma: conceptualización, metodología, análisis formal, investigación, recursos, curación de datos, redacción, visualización, supervisión, administración del proyecto

La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga, Portal Independencia N° 57, Ayacucho, AYA 5001, Perú, Email: elmer.achalma.09@unsch.edu.pe

Rebanado de cadenas

0.1 Motivación

A menudo necesitas trabajar con partes de un texto: extraer el nombre de usuario de un email, obtener la extensión de un archivo, o simplemente tomar una porción de un mensaje. El slicing (rebanado) te permite hacer esto de manera elegante y eficiente, sin necesidad de bucles o lógica compleja.

0.2 Intuición

Imagina que tienes una fila de personas (string) y quieres seleccionar a las personas de las posiciones 2 a 5. El slicing es como decir: “dame desde la posición 2 hasta la posición 5”. En Python, `[2:5]` significa: empieza en el índice 2 y **detente antes** del índice 5. Es como un cuchillo que corta exactamente donde le indicas.

0.3 Modelo mental

La sintaxis de slicing es `[inicio:fin:paso]:`

- **inicio** (opcional): dónde empezar (incluido). Por defecto es el principio.
- **fin** (opcional): dónde parar (no incluido). Por defecto es el final.
- **paso** (opcional): cada cuántos caracteres tomar. Por defecto es 1.

La regla clave: **el índice final es exclusivo**, es decir, no se incluye en el resultado.

0.4 Desarrollo

0.4.1 1. Sintaxis básica

```
texto = "Hello, World!"

# [inicio:fin] - fin no incluido
print(texto[2:5])    # "llo" (índices 2, 3, 4)
print(texto[7:12])   # "World" (índices 7-11)
```

0.4.2 2. Slice desde el inicio

Omite el inicio para empezar desde el primer carácter:

```
texto = "Hello, World!"
print(texto[:5])    # "Hello" (índices 0-4)
print(texto[:7])    # "Hello, " (índices 0-6)
```

0.4.3 3. Slice hasta el final

Omite el fin para ir hasta el último carácter:

```
texto = "Hello, World!"
print(texto[7:])    # "World!" (índices 7-12)
print(texto[2:])    # "llo, World!" (índices 2-12)
```

0.4.4 4. Slice completo

```
texto = "Hello, World!"
print(texto[:])    # "Hello, World!" (copia completa)
```

0.4.5 5. Índices negativos

Empieza desde el final (último carácter es -1):

```
texto = "Hello, World!"

# Últimos 5 caracteres
print(texto[-5:])    # "orld!"
print(texto[-6:-1]) # "World" (desde -6 hasta antes de -1)
print(texto[-5:-2]) # "orl" (desde -5 hasta antes de -2)
```

0.4.6 6. Slice con paso ([inicio:fin:paso])

Controlas el salto entre caracteres:

```
texto = "Python"

# Cada 2 caracteres
print(texto[::2])    # "Pto" (índices 0, 2, 4)
print(texto[1::2])   # "yhn" (índices 1, 3, 5)

# Invertir string (paso -1)
print(texto[::-1])   # "nohtyP"

# Desde el final con paso -2
print(texto[::-2])    # "nhy"
```

0.4.7 7. Tabla de ejemplos de slicing

Slicing	Resultado (para "Python")	Explicación
[0:3]	"Pyt"	Posiciones 0, 1, 2
[2:5]	"tho"	Posiciones 2, 3, 4
[:4]	"Pyth"	Desde el inicio hasta posición 3
[3:]	"hon"	Desde posición 3 hasta el final
[-3:]	"hon"	Últimos 3 caracteres
[:-2]	"Pyth"	Hasta antes de los últimos 2
[::2]	"Pto"	Caracteres en posiciones pares
[::-1]	"nohtyP"	String invertido

0.5 Ejemplos

Mínimo — Slicing básico:

```
fruta = "Manzana"
print(fruta[0:3]) # "Man"
print(fruta[3:6]) # "ana"
print(fruta[2:5]) # "nza"
```

Progresivo — Extracción de datos:

```
email = "usuario@gmail.com"

# Extraer nombre de usuario (hasta el @)
nombre_usuario = email[:email.index("@")]
print(nombre_usuario) # "usuario"

# Extraer dominio (después del @)
dominio = email[email.index("@") + 1:]
print(dominio) # "gmail.com"

# Extraer extensión (últimos 3 caracteres)
extension = email[-3:]
print(extension) # "com"
```

Real — Procesamiento de nombres de archivo:

```
archivo = "documento_2024.pdf"

# Obtener extensión
extension = archivo.split(".")[1] # "pdf"
# Con slicing:
punto = archivo.index(".")
extension_slice = archivo[punto + 1:] # "pdf"

# Obtener nombre sin extensión
nombre_base = archivo[:archivo.index(".")] # "documento_2024"

# Año (últimos 4 dígitos antes del punto)
año = archivo[-8:-4] # "2024"

print(f"Archivo: {nombre_base}")
print(f"Extensión: {extension_slice}")
print(f"Año: {año}")
```

0.6 Errores comunes

Error	Síntoma	Solución
Índice fuera de rango	IndexError: string index out of range	Verifica que el índice sea < len(string)
Confundir [inicio:fin]	No incluir el último carácter	Recuerda: fin no se incluye
Olvidar :	Intenta acceder a un solo carácter	Usa [inicio:fin] no [inicio, fin]
Índices negativos confusos	Resultados inesperados	Recuerda: -1 es el último carácter

0.7 Buenas prácticas

1. Usa índices negativos para “desde el final”:

```
# Obtener últimos 4 caracteres
ultimos_4 = texto[-4:]

# Menos claro
ultimos_4 = texto[len(texto)-4:]
```

2. Combina slicing con len() para cortar dinámicamente:

```
texto = "Hola mundo"
# Primeros 3
texto[:3]
# Últimos 3
texto[-3:]
# Mitad
mitad = len(texto) // 2
texto[:mitad] # Primera mitad
texto[mitad:] # Segunda mitad
```

3. Usa [::-1] para invertir strings:

```
palabra = "Python"
print(palabra[::-1]) # "nohtyP"
```

4. Verifica que los índices sean válidos antes de hacer slicing si no estás seguro:

```
texto = "abc"
# Slice con índices fuera de rango es seguro (no da error)
print(texto[0:10]) # "abc" (no da error)
# Pero acceso directo sí da error
# print(texto[5]) # IndexError
```

5. Usa slicing para copiar strings:

```
original = "Hola"
copia = original[:] # Crea una copia
```

0.8 Ejercicio

Intermedio — Manipulación de nombres de archivo

Escribe un programa que:

1. Tenga un string `ruta_archivo = "C:/Users/Ana/Documents/proyecto_final_v2.py"`
2. Extraiga y muestre:
 - El nombre del archivo (última parte después de /)
 - La extensión del archivo
 - El nombre base (sin extensión)
 - La carpeta donde está el archivo (todo antes del último /)
 - Una versión con solo las iniciales de cada carpeta: `C:/U/A/D/proyecto_final_v2.py`

Pista: usa `rfind()` para encontrar la última ocurrencia de un carácter, y luego slicing.

Solución (no mires hasta haberlo intentado):

```
ruta_archivo = "C:/Users/Ana/Documents/proyecto_final_v2.py"

# 1. Nombre del archivo (último segmento)
ultimo_slash = ruta_archivo.rfind("/")
nombre_archivo = ruta_archivo[ultimo_slash + 1:]
print(f"Archivo: {nombre_archivo}") # "proyecto_final_v2.py"

# 2. Extensión
punto = nombre_archivo.rfind(".")
extension = nombre_archivo[punto + 1:]
print(f"Extensión: {extension}") # "py"

# 3. Nombre base (sin extensión)
nombre_base = nombre_archivo[:punto]
print(f"Nombre base: {nombre_base}") # "proyecto_final_v2"

# 4. Carpeta (todo antes del último /)
carpeta = ruta_archivo[:ultimo_slash]
print(f"Carpeta: {carpeta}") # "C:/Users/Ana/Documents"

# 5. Iniciales de carpetas
partes = ruta_archivo.split("/")
iniciales = "/".join([p[:1] if p else "" for p in partes[:-1]]) + "/" + partes[-1]
print(f"Con iniciales: {iniciales}") # "C:/U/A/D/proyecto_final_v2.py"
```

0.9 Síntesis

- **Sintaxis:** `[inicio:fin:paso]` donde `fin` no se incluye.
- **Inicio por defecto:** `[:fin]` empieza desde el principio.
- **Fin por defecto:** `[inicio:]` va hasta el final.
- **Índices negativos:** cuentan desde el final (-1 es el último).
- **Paso:** controla el salto entre caracteres. `[::-1]` invierte.
- **Slicing seguro:** índices fuera de rango no dan error (a diferencia de acceso directo).
- **Strings inmutables:** el slicing devuelve un nuevo string, no modifica el original.

0.10 Qué sigue

- **Relacionado:** métodos de string, indexación, f-strings
- **Siguiente tema:** Métodos de strings — aprende todas las herramientas para manipular texto.

1 Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1.  [Instalacion De Anaconda](#)
2.  [Configurar Entorno Virtual Python Anaconda](#)
3.  [01 Introducion A La Programacion Con Python](#)
4.  [02 Variables Expresiones Y Statements Con Python](#)
5.  [03 Objetos De Python](#)
6.  [04 Ejecucion Condicional Con Python](#)
7.  [05 Iteraciones Con Python](#)
8.  [06 Funciones Con Python](#)
9.  [07 Dataframes Con Python](#)
10.  [08 Prediccion Y Metrica De Performance Con Python](#)
11.  [09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12.  [10 Metodos De Machine Learning Para Regresion Con Python](#)
13.  [Visualizacion De Datos Con Python](#)

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!