

OPTATIVO 1 – PYTHON I

Prof. Ing. Aaron Zárate



Cadenas en Python

Una cadena o string en ingles es un tipo de dato que se utiliza para almacenar una secuencia de caracteres.

Las cadenas se deben encerrar entre comilla doble o comilla simple.

Los caracteres pueden ser letras, números, símbolos o espacios.

```
# Cadenas en Python
cadena1 = "Hola Mundo"
```

Memoria


Objetos

Variables

cadena1

str
"Hola Mundo"

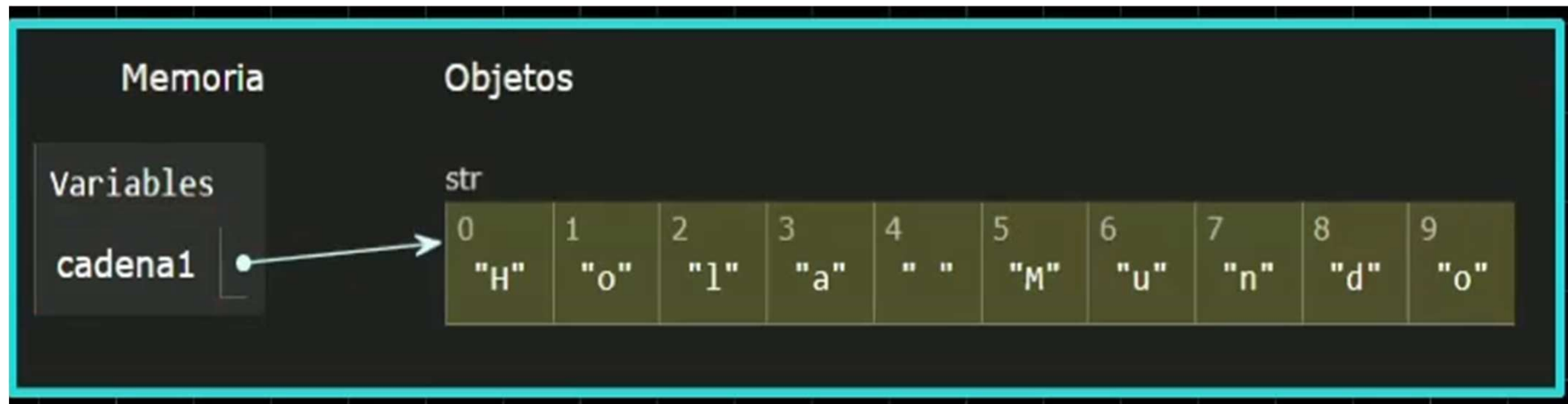
Cadenas en Python

 manejo_cadenas.py ×

```
1  #Manejo de cadenas
2  cadena1 = 'Sayonara'
3  cadena2 = "Python es Animal"
4  # noinspection PyInterpreter
5  cadena3 = """Este es un ejemplo
6  de multiples lineas en
7  una cadena de Python"""
8
9  print(cadena1)
10 print(cadena2)
11 print(cadena3)
```

Detalle de una Cadena

Los caracteres de una cadena están indexados de una manera secuencial. Por lo tanto podemos acceder cada carácter indicando el índice del carácter que deseamos recuperar.



Manejo de índices en cadena



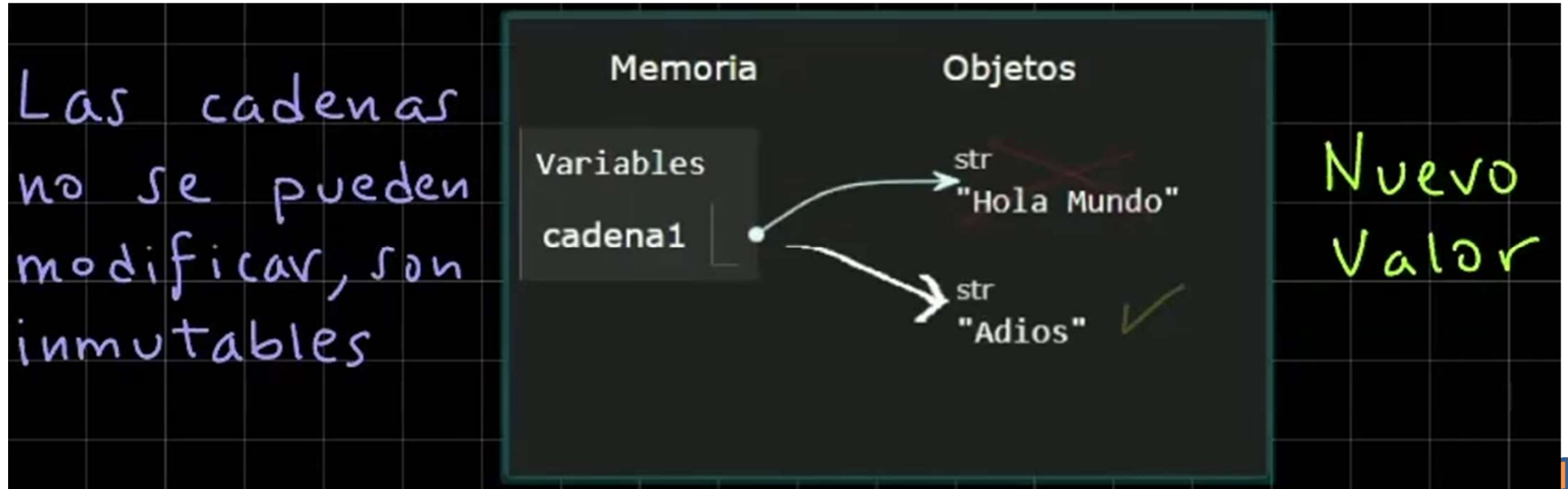
manejo_indice.py ×

```
1  # Manejo de índice en una cadena
2
3  cadena1 = "Hola Mundo"
4  print(cadena1)
5
6  #Recuperar el primer caracter
7
8  primer_caracter = cadena1[0] # Recupera 'H'
9  print(primer_caracter)
10
11 #Recuperar ultimo caracter
12
13 ultimo_caracter = cadena1[9]
14 print(ultimo_caracter) # Recupera 'o'
```

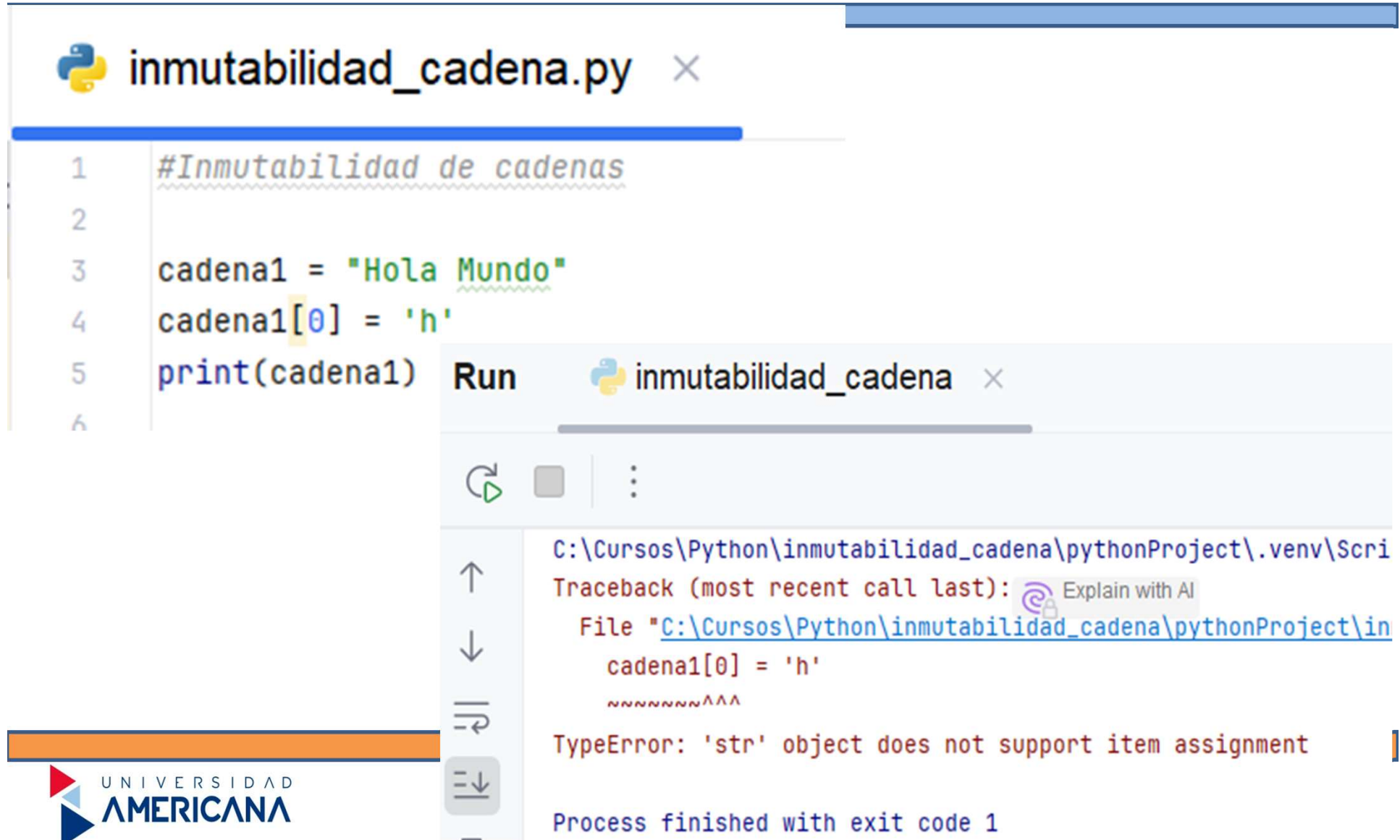
Inmutabilidad de cadenas en Python

Una vez que se crea una cadena, los caracteres dentro de ella no pueden ser modificados.

Si deseamos modificar una cadena entonces tenemos que crear una nueva cadena.



Inmutabilidad de cadenas en Python



The image shows a Python IDE window titled 'inmutabilidad_cadena.py'. The code in the editor is as follows:


```
1  #Inmutabilidad de cadenas
2
3  cadena1 = "Hola Mundo"
4  cadena1[0] = 'h'
5  print(cadena1)
6
```

Below the editor, a 'Run' window is open, showing the execution of the script. It displays a 'Traceback (most recent call last):' error message. The error details are:

- File "C:\Cursos\Python\inmutabilidad_cadena\pythonProject\inmutabilidad_cadena.py", line 4, in <module>: cadena1[0] = 'h'
- ~~~~~
- TypeError: 'str' object does not support item assignment


The window concludes with 'Process finished with exit code 1'. On the left side of the Run window, there are navigation icons: a green play button, a square stop button, a vertical ellipsis, and arrows for navigating through the traceback.

Inmutabilidad de cadenas en Python

 inmutabilidad_cadena.py ×

```
1  #Inmutabilidad de cadenas
2
3  cadena1 = "Hola Mundo"
4  #cadena1[0] = 'h' # no se puede eliminar los caracteres
5  cadena1 = "Adios"
6  print(cadena1)
7
```


Inmutabilidad de cadenas en Python

 inmutabilidad_cadena.py ×

```
1  #Inmutabilidad de cadenas
2
3  cadena1 = "Hola Mundo"
4  #cadena1[0] = 'h' # no se puede eliminar los caracteres
5  cadena2 = cadena1
6  cadena1 = "Adios"
7  print(cadena1)
8  print(cadena2)
9
```

 inmutabilidad_cadena ×

C:\Cursos\Python\inmutabilidad_caden
Adios
Hola Mundo

Process finished with exit code 0


Caracteres especiales en Python

Las cadenas pueden incluir caracteres especiales. Estos caracteres se introducen usando el carácter de diagonal invertida (“ \ ”) (ALT+92).

- Nueva línea: “ \n ” inserta un salto de línea.
- Tabulación : “ \t ” inserta un tabulador horizontal, útil para alinear texto.
- Comilla simple: “ \' ” permite incluir comillas simples en una cadena delimitada por comillas simples.
- Comilla doble: “ \" ” permite incluir comillas dobles en una cadena delimitada por comillas dobles.
- Barra invertida: “ \\ ” permite incluir una barra invertida en la cadena.

Existen mas caracteres pero estos son los esenciales.

Caracteres especiales en Python

 caracteres_especiales.py x

```
1  #Caracteres especiales
2  print("Hola \nMundo")
3  print("\t\tPython \tes animal")
4  print('Juan\'Perez\'')
5  print("Fulano \" Menelao")
6  print("Caracter \\Diagonal invertida")
```

 caracteres_especiales

```
Hola
Mundo

    Python  es animal
Juan'Perez'
Fulano " Menelao
Caracter \Diagonal invertida
```

Concatenación de cadenas

La concatenación de cadenas es una operación que permite combinar dos o mas cadenas para formar una nueva cadena.

En Python existen varias formas y algunas son:


- Uso de operador + : El operador + es el mas directo para concatenar cadenas simplemente tenemos que poner el operador + entre las cadenas que deseamos unir, ejemplo:

concatenación = "Hola" + "Mundo"

- Uso de la función join: La función join nos permite unir tantas cadenas como necesitemos. Solo necesitamos pasar cada cadena a concatenar separados por coma Ejemplo:

" ".join(["Cadena1", "Cadena2", "Cadena3"])

Concatenación

 concatenacion.py ×

```
1  #Concatenacion de cadenas
2  cadena1 = "Hola"
3  cadena2 = "Mundo"
4  concatenacion = cadena1 + " " + cadena2
5  print(concatenacion)
6  #Utilizando el metodo join
7  concatenacion = "".join([cadena1, " ", cadena2])
8  print(concatenacion)
9
```

Formateo de cadenas en Python

Python ofrece varias maneras de formatear cadenas, que incluyen la capacidad de concatenar texto, variables e incluso dar otro tipo de formato, como por ejemplo indicar el numero de decimales en el formato

- f-string (Python 3,6+): Esta es la opción mas recomendada, por ser la mas sencilla rápida y legible.

Resultado = f ' Hola {variable} '


Formateo de cadenas en Python

- Método format: Es muy versátil y poderoso, permite construir cadenas muy complejas.

resultado = 'Hola { }'.format(variable)


Veremos a seguir algunos ejemplos de formateo de cadenas:

Formateo mas sencillo y recomendable f.string

 formateo_cadenas.py ×

```
1  #Formateo de cadenas
2  nombre = "Esteban"
3  edad=30
4  # f-string
5  mensaje = f'Hola, me llamo {nombre} y tengo {edad} años'
6  print(mensaje)
```


Formateo mas sencillo y recomendable format

 formateo_cadenas.py ×

```
1  #Formateo de cadenas
2  nombre = "Esteban"
3  edad=30
4  # f-string
5  mensaje = f'Hola, me llamo {nombre} y tengo {edad} años'
6  print(mensaje)
7
8  # Metodo format
9
10 mensaje= "Hola, me llamo {} y tengo {} años".format(*args: nombre,edad)
11 print(mensaje)
12
```


! 4 ✓ 1

Métodos de cadenas

Las cadenas en Python vienen con una serie de métodos útiles que facilitan su manipulación. Por ejemplo:

- `upper()` → Cambia las letras a mayúsculas
- `lower()` → cambia las letras a minúsculas
- `strip()` → Elimina espacios en blanco al inicio y al final de la cadena.

Métodos de cadenas

 metodos_cadenas.py ×

```
1  # >Metodos de Cadenas
2  cadena1 = "Hola Mundo"
3  print(f'Cadena original: {cadena1}')
4
5  #Para convertir a mayusculas
6  mayusculas= cadena1.upper()
7  print(f"Cadena en mayusculas: {mayusculas}")
8
9  #Para convertir a minusculas
10 print(f"Cadena en minusculas: {cadena1.lower()}")
11
12 #Sin espacios en blanco inicio y al final
13 cadena2=" Rodriguez de Francia "
14 print(f"Cadena sin espacios inicial y final: {cadena2}")
15 print(f"Cadena sin espacios inicial y final: {cadena2.strip()}")
```

! 6 ✓

Obtener el largo de un cadena

Para obtener la longitud de una cadena, utilizamos la función incorporada `len()`.


La función `len()` funciona con varios tipos de datos incluyendo cadenas, listas, etc.

Cuando se calcule el largo de una cadena se toman en cuenta todos los caracteres de una cadena incluyendo, incluyendo espacios en blanco, caracteres especiales , etc.

`cadena1="Hola, Mundo!"`

`longitud=len(cadena1) → devuelve largo de 12`

Obtener el largo de un cadena

 largo_cadena.py ×

```
1  cadena="Hola, Mundo!"
2  largo_cadena= len(cadena)
3  print(f"Cadena original: {cadena}")
4  print(f"Largo de la cadena: {largo_cadena}")
5
```


Run  largo_cadena ×




C:\Cursos\Python\Formateo_cadenas\pythonPi
Cadena original: Hola, Mundo!
Largo de la cadena: 12

Subcadenas en Python

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

 manejo_subcadenas.py ×

```
1  # Manejo de subcadenas
2  cadena="Hola, Mundo!"
3  # Obtenemos la subcadena de hola [inicio:fin (sin incluirlo)]
4  subcadena_hola= cadena[0:4]
5  print(f'Subcadena de Hola: {subcadena_hola} ')
6
7  # Obtenemos la subcadena mundo
8  subcadena_mundo= cadena[6:11]
9  print(f'Subcadena de mundo: {subcadena_hola} ')
```

 manejo_subcadenas ×

```
C:\Cursos\Python\Formateo_cadenas
Subcadena de Hola: Hola
Subcadena de mundo: Hola
```

Subcadenas en Python

- Reemplazar subcadenas (replace): El método replace reemplaza una subcadena por otra dentro de la cadena principal

cadena = “Hola, mundo”

nueva_cadena = cadena.replace (“mundo”, “a todos”)

print(nueva_cadena) #Hola a todos

Subcadenas en Python

Buscar subcadenas (find): el método find() devuelve el índice de la primera de la primera aparición de la subcadena, devuelve -1


```
cadena = " Hola mundo"
```

```
posición = " cadena.find ("mundo"
```

```
print (posición) # imprime 5
```


Subcadenas en Python

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


 buscar_subcadena.py ×

```
1  # buscar subcadenas
2
3  cadena="Hola, mundo"
4  indice= cadena.find("mundo")
5  print(f"Indice de la subcadena mundo: {indice}")
6
7  #Obtener el indice de la subcadena Hola
8  indice =cadena.find("hola")
9  print(f"Indice de la subcadena mundo: {indice}")
```

```
C:\Cursos\Python\Formateo_cadenas\pyt
Indice de la subcadena mundo: 6
Indice de la subcadena mundo: -1
```

Subcadenas en Python

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

 reemplazar_subcadenas.py ×

```
1  # reemplazar cadenas
2  cadena= "Hola, Mundo!"
3  print(f"Cadena original: {cadena}")
4  nueva_cadena=cadena.replace("mundo", "a todos")
5  print(f"Nueva cadena reemplazada: {nueva_cadena}")
6
```

C:\Cursos\Python\Formateo_cadenas\pythonf

Cadena original: Hola, Mundo!

Nueva cadena reemplazada: Hola, Mundo!

Nueva cadena reemplazada: Adios, Mundo!

Subcadenas en Python


Extraer subcadenas por separados (Split): la función Split permite dividir una cadena en una lista de subcadenas basadas en un carácter separador, ejemp,

```
datos = 'Juan, 30 , México'
```


```
lista = datos.split(",")
```

```
print(lista) # ['Juan' , '30' , 'Mexico']
```

Subcadenas en Python


 separar_cadenas.py ×

```
1  # separar cadenas split
2  datos = "Hola Mundo"
3  lista = datos.split() # default separa con espacios en blanco
4  print(lista)
```

 separar_cadenas ×

C:\Cursos\Python\Formateo_cadenas\pythonPro
['Hola', 'Mundo']

Subcadenas en Python

 separar_cadenas.py ×

```
1  # separar cadenas split
2  datos = "Hola Mundo"
3  lista= datos.split()# default separa con espacios en blanco
4  print(lista)
5
6  datos = "Juan,30,Mexico"
7  lista=datos.split(",")
8  print(lista)
```

C:\Cursos\Python\Formateo_cadenas

['Hola', 'Mundo']

['Juan', '30', 'Mexico']

Subcadenas en Python



multiplicacion_cadenas.py ×

```
1  # Multiplicacion de cadenas
2  print("***Multiplicacion de cadenas**")
3  texto= "Hola"
4  veces = 4
5  resultado= texto * veces
6  print(resultado)
```

```
C:\Cursos\Python\Formateo_cadenas\p
***Multiplicacion de cadenas**
HolaHolaHolaHola
```


Generador de email

Crea un programa para generar un email a partir de los siguientes datos:

- Nombre: Francisco Solano Lopez
- Empresa: Ejercito Paraguayo
- Dominio: com.py

Resultado Final:

Email:

francisco.solano.lopez@ejercitoparaguayo.com.py

Generador de email

Generador de Emails

Nombre de Usuario: Francisco Solano Lopez

Nombre Normalizado

Nombre empresa: Ejercito Paraguayo

Extension del dominio: .com.py

Dominio del email normalizado: @ejercitoparaguayo.com.py

Email final generado: francisco.solano.lopez@ejercitoparaguayo.com.py

Resultado parte 1



generador_email.py ×

```
1  # Generador de emails
2  print("***Generador de Emails***")
3  #Nombre completo del usuario
4  nombre_completo=" Francisco Solano Lopez "
5  print(f"Nombre de Usuario: {nombre_completo}")
6  ✓ #Procesar o normalizar nombre de usuario
7  #limpiamos los espacios al inicio y al final
8  nombre_normalizado=nombre_completo.strip()
9  #Reemplazar espacios en blanco por puntos
10 nombre_normalizado=nombre_normalizado.replace(" ", ".")
11 #convertimos a minusculas
12 nombre_normalizado=nombre_normalizado.lower()
13 print(f"Nombre Normalizado")
```

Resultado parte 2

```
14  # Datos de la empresa
15  nombre_empresa= " Ejercito Paraguayo "
16  print(f"\nNombre empresa: {nombre_empresa}")
17  extension_dominio = ".com.py"
18  print(f"Extension del dominio: {extension_dominio}")
19  #Quitamos los espacios en blanco y mayusculas
20  nombre_empresa_normalizado=nombre_empresa.replace(" ", "").lower()
21  dominio_email_normalizado=f"@{nombre_empresa_normalizado}{extension_dominio}"
22  print(f"Dominio del email normalizado: {dominio_email_normalizado}")
23  #Creamos el email final
24  email=f"{nombre_normalizado}{dominio_email_normalizado}"
25  print(f"\nEmail final generado: {email}")
```