

P1

Programación 1

04
UNIDAD

05
CLASE

VARIABLES DE CADENAS DE CARACTERES



- | Variables tipo “**cadena**s de **caracteres**”
- | Función “**len**”
- | Métodos propios de las cadenas de caracteres



Al final de esta clase ya podrás:

- | Utilizar los operadores relacionales para identificar si dos cadenas son iguales, distintas o mayores entre sí.
- | Aprovechar las funciones de retorno lower, upper y capitalize.

ISSD

-Des-
Desarrollo de
Software

MÓDULO
DIDÁCTICO
2023

Segundo paso con estructuras repetitivas

¡Bienvenidos a la quinta clase de Programación 1!

En esta nueva oportunidad nos adentraremos en el trabajo con diversos tipos variables y su aplicación a través una serie de métodos que nos facilitan la creación de nuestros programas. El objetivo de la clase es que puedas valorar y aprovechar las ventajas que presentan las diferentes variables que provee el lenguaje Python de una manera más eficiente.

Esta clase se estructura en torno a tres temas:

- | El primero de ellos te introducirá en las variables tipo “**cadenas de caracteres**”.
- | En el segundo incorporaremos la función “**len**”.
- | En tercer lugar abordaremos los **métodos propios de las cadenas de caracteres**.

Como en todas las clases, te iremos proponiendo ejercicios para que puedas establecer y practicar lo que vamos aprendiendo.

Como ya lo señalamos anteriormente, esta primera materia te introduce al mundo de la Programación, por eso lo más importante es que desarrolles la lógica, que seas capaz de identificar los elementos de un problema y que logres diseñar el algoritmo necesario para su resolución.

¿Seguimos aprendiendo?

Si querés realizar las pruebas del código en tu pc, te recomendamos hacer click en el botón ⬇️. Se encuentra ubicado junto a cada código ejemplificado. De esta manera, podrás copiar y pegar el texto puro para que evitar errores en la ejecución.

Si accedes al módulo desde un navegador, te recomendamos abrir los hipervínculos en una nueva pestaña presionando la rueda central del ratón, o bien usando el botón derecho y seleccionando la opción correspondiente.



Antes de comenzar con la clase, leé el siguiente artículo.

Tienen 17 años y crearon un robot que, con IA, puede leer e interpretar Lengua de Señas

Dos adolescentes mendocinos combinaron sus conocimientos en impresión 3D y en programación para dar vida al primer robot que puede interpretar y pasar al lenguaje textual todo lo que es Lengua de Señas. Sueñan con que esté en oficinas y otros espacios para asistir a personas sordas.

Santiago Vázquez y **Juan Cruz Ledesma** tienen 17 años (ambos) y viven en Guaymallén y Maipú, respectivamente. Mientras que el primero de ellos es un **apasionado por la robótica y las impresiones 3D**, el segundo tiene facilidad y debilidad por la **programación**. Fueron estas pasiones las que, unidas y a través del **trabajo en equipo**, los llevaron a ser “*padres*” de un robot que va camino a ser un hito en todo el país.

Y es que Santiago y Juan Cruz crearon un **robot** que, por medio de la

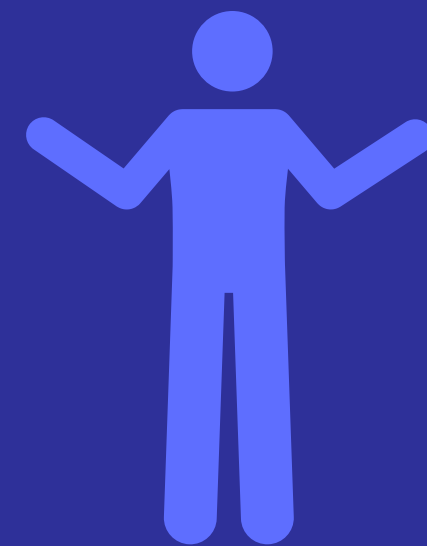
programación y la Inteligencia Artificial (IA), puede **interpretar Lengua de Señas Argentina (LSA)**. Según explicaron sus creadores, el robot –que aún no tiene nombre- **puede “ver”** a alguien que está utilizando esta lengua y **luego transcribirla**, de manera escrita, y hasta interpretar de manera **oral** lo que se ha dicho. (...)

“La idea es que el robot pueda estar en un restaurante, en un banco o en una oficina pública, siempre a disposición de personas sordas que lleguen. Muchas veces ocurre que no pueden hacerse entender cuándo van a hacer algún trámite o al médico. Con este robot, habría una herramienta más de acceso”, sueñan en voz alta sus creadores. (...)

Si bien aún resta ajustarle algunos detalles, el robot ya está programado para interpretar letras y algunas palabras específicas. “En marzo empezamos a hacer las pruebas y nos parábamos adelante para comprobar todo. **Vos le decís ‘derecha’ y mueve sus ojos para la derecha**. Así empezamos a programar todo hasta llegar al proyecto de la **Lengua de Señas**, concluyeron los jóvenes.

Los Andes - Abril de 2023

Podemos afirmar que la tecnología se volvió rápidamente muy amigable, y hoy aprender a programar, no está reservado para genios. Lo que sí es cierto es que a la programación hay que dedicarle tiempo y voluntad.



Por eso, si no terminaste de hacer todos los ejercicios de la clase anterior, estás a tiempo de completarlos antes de continuar avanzando. Aprender a programar requiere no dejar dudas para después. Recurrí a tu tutor cada vez que sea necesario. ¡Sigamos adelante!



Procesar variables de tipo cadenas de caracteres

Hasta este momento hemos visto cómo definir variables enteras y flotantes, y cómo realizar su carga por asignación y por teclado. Para iniciar las variables por asignación utilizamos el operador "="

```
#definición de una variable entera
cantidad=20
#definición de una variable flotante
altura=1.92
```

Como vemos el intérprete de Python diferencia una variable flotante de una variable entera por la presencia del carácter punto.

| Para realizar la carga por teclado utilizando la función input debemos llamar a la función int o float para convertir el dato devuelto por input:

```
cantidad=int(input("Ingresar la cantidad de personas:"))
altura=float(input("Ingresar la altura de la persona en metros ej:1.70:"))
```

A estos dos tipos de datos fundamentales (int y float) se suma un tipo de dato muy utilizado que son **las cadenas de caracteres**.

Una cadena de caracteres está compuesta por uno o más caracteres. También podemos iniciar una cadena de caracteres por asignación o ingresarla por teclado.

| Para realizar una inicialización de una cadena por asignación:

```
#definición e inicio de una cadena de caracteres  
dia="lunes"
```

| Obtenemos el mismo resultado si utilizamos la comilla simple:

```
#definición e inicio de una cadena de caracteres  
dia='lunes'
```

Para la carga por teclado de una cadena de caracteres utilizamos la función input que retorna una cadena de caracteres:

```
nombre=input("ingrese su nombre:")
```

Ejemplo 22

Identificador de mayor estatura

Realizar la carga por teclado del nombre, edad y altura de dos personas. Mostrar por pantalla el nombre de la persona con mayor altura.

| Programa: 

```
print("Datos de la primer persona")
nombre1=input("Ingrese nombre:")
edad1=int(input("Ingrese la edad:"))
altura1=float(input("Ingrese la altura Ej 1.75:"))
print("Datos de la segunda persona")
nombre2=input("Ingrese nombre:")
edad2=int(input("Ingrese la edad:"))
altura2=float(input("Ingrese la altura Ej 1.75:"))
print("La persona mas alta es:")
if altura1>altura2:
    print(nombre1)
else:
    print(nombre2)
```

| Es importante notar que cuando cargamos un entero el dato devuelto por la función input se lo pasamos a la función int que tiene por objetivo convertirlo a entero:

```
edad1=int(input("Ingrese la edad:"))
```

| Cuando cargamos un valor con decimal el dato devuelto por la función input se lo pasamos a la función float que tiene por objetivo convertirlo a float:

```
altura1=float(input("Ingrese la altura Ej 1.75:"))
```

| Finalmente cuando cargamos una cadena de caracteres como es en este caso el nombre de una persona la función input retorna directamente una cadena de caracteres.

```
nombre1=input("Ingrese nombre:")
```


Ejemplo 23

Identificador de nombres alfabéticamente mayores

Realizar la carga de dos nombres por teclado. Mostrar cual de los dos es mayor alfabéticamente o si son iguales.

| Programa: 

```
nombre1=input("Ingrese el primer nombre:")
nombre2=input("Ingrese el segundo nombre:")
if nombre1==nombre2:
    print("Ingreso dos nombre iguales")
else:
    if nombre1>nombre2:
        print(nombre1)
        print("es mayor alfabeticamente")
    else:
        print(nombre2)
        print("es mayor alfabeticamente")
```

Cuando trabajamos con cadenas de caracteres al utilizar el operador “>” estamos verificando si una cadena es mayor alfabéticamente a otra (esto es distinto a cuando trabajamos con enteros o flotantes)

Luis>Carlos>Benjamín>Agustín

Por ejemplo, 'luis' es mayor a 'carlos' porque la 'l' se encuentra más adelante que la 'c' en el abecedario.



Ejemplo 24

Identificador de nombres alfabéticamente mayores

Realizar la carga de enteros por teclado. Luego de ingresar el valor, preguntar si desea cargar otro valor debiendo el operador ingresar la cadena 'si' o 'no' por teclado.

Mostrar la suma de los valores ingresados.

Para resolver este problema hemos inicializado una *variable de tipo cadena de caracteres* (también se las llama variables de tipo **string**) con el valor "si", esto hace que la condición del while se verifique verdadera la primera vez.

Dentro del while luego de cargar el valor entero se pide la carga por teclado que confirme si desea cargar otro valor, en caso que cargue el string "si" el ciclo repetitivo while se vuelve a repetir.

El ciclo se cortará cuando el operador cargue un string distinto a "si".

Es importante notar que el string "si" es distinto al string "Si", es decir las mayúsculas no tienen el mismo valor alfabético que las minúsculas (después veremos que podemos convertir mayúsculas a minúsculas y viceversa)

| Programa: ⌵

```
opcion="si"
suma=0
while opcion=="si":
    valor=int(input("Ingrese un valor:"))
    suma=suma+valor
    opcion=input("Desea cargar otro numero (si/no):")
print("La suma de valores ingresados es")
print(suma)
```




Desempeño 35

Soluciones



| Realizar la carga de dos nombres de personas distintos.
Mostrar por pantalla luego ordenados en forma alfabética.

Variables enteras, flotantes y cadenas de caracteres

Ya hemos visto que podemos cargar una cadena de caracteres por asignación:

```
#con doble comillas
cadena1="juan"
#el resultado es igual con simple comillas
cadena2='ana'
```

| También podemos cargarla por teclado:

```
nombre=input("Ingrese su nombre:")
```

Podemos utilizar los operadores relacionales para identificar si dos cadenas son iguales, distintas o cuál es la mayor alfabética:

```
== Igualdad
!= Desigualdad
< menor
<= menor o igual
> mayor
>= mayor o igual
```

Como su nombre lo indica una cadena de caracteres está formada generalmente por varios caracteres (de todos modos podría tener solo un carácter o ser una cadena vacía)

| Podemos acceder en forma individual a cada carácter del string mediante un subíndice:

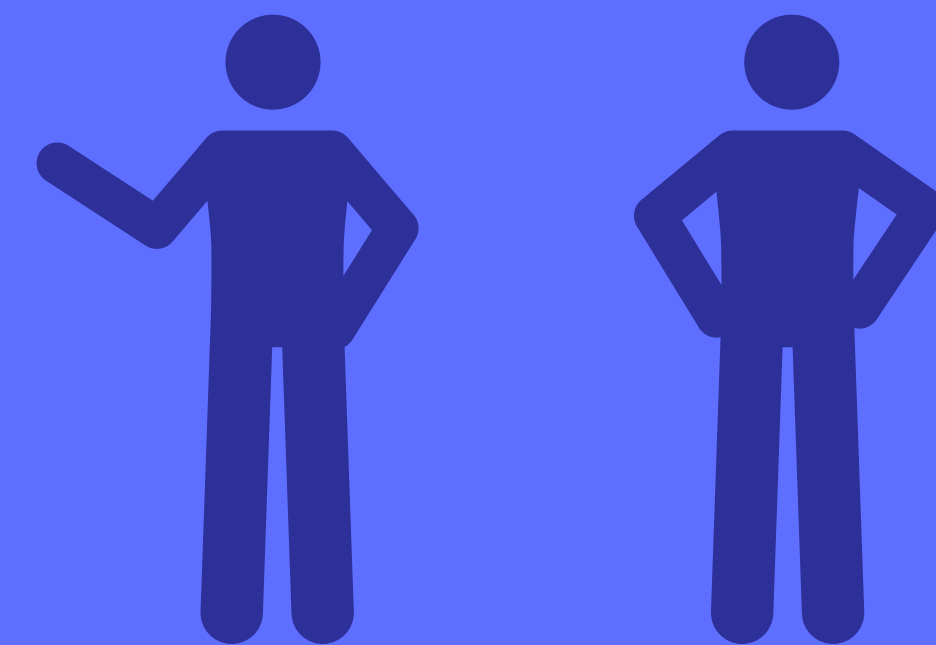
```
nombre='juan'
print(nombre[0])    #se imprime una j
if nombre[0]=="j":  #verificamos si el primer caracter del string es una j
    print(nombre)
    print("comienza con la letra j")
```

| Si queremos conocer la longitud de un string en Python disponemos de una función llamada “len” que retorna la cantidad de caracteres que contiene:

```
nombre='juan'
print(len(nombre))
```

El programa anterior imprime un 4 ya que la cadena nombre almacena 'juan' que tiene cuatro caracteres.

Los subíndices comienzan a numerarse a partir del cero.



Ejemplo 25

Mostrador de iniciales y su cantidad de caracteres

Realizar la carga del nombre de una persona y luego mostrar el primer carácter del nombre y la cantidad de letras que lo componen.

| Programa 

```
nombre=input("Ingrese su nombre:")  
print("Primer caracter")  
print(nombre[0])  
print("Cantidad de letras del nombre:")  
print(len(nombre))
```


Ejemplo 26

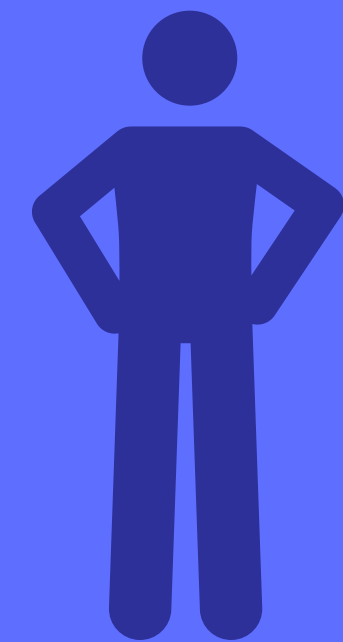
Cargador de nombres e identificador de vocales

Solicitar la carga del nombre de una persona en minúsculas. Mostrar un mensaje si comienza con vocal dicho nombre.

| Programa 

```
nombre=input("Ingrese su nombre:")
if nombre[0]=="a" or nombre[0]=="e" or nombre[0]=="i" or nombre[0]=="o" or nombre[0]=="u":
    print("El nombre ingresado comienza con vocal")
else:
    print("El nombre ingresado no comienza con vocal")
```

Con que una de las condiciones del **if** sea **verdadera** es suficiente para que se ejecute el bloque del verdadero.



Ejemplo 27

Validador de direcciones de correo

Ingresar un mail por teclado. Verificar si el string ingresado contiene solo un carácter "@".

| Programa 

```
mail=input("Ingrese un mail:")
cantidad=0
x=0
while x<len(mail):
    if mail[x]=="@":
        cantidad=cantidad+1
    x=x+1
if cantidad==1:
    print("Contiene solo un caracter @ el mail ingresado")
else:
    print("Incorrecto")
```

Para analizar cada carácter del string ingresado disponemos una estructura while utilizando un contador llamado x que comienza con el valor cero y se repetirá tantas veces como caracteres tenga la cadena (mediante la función len obtenemos la cantidad de caracteres):

```
while x<len(mail):
```

Dentro del ciclo while verificamos cada carácter mediante un if y contamos la cantidad de caracteres "@":

```
if mail[x]=="@":  
    cantidad=cantidad+1
```

Cuando sale del ciclo while procedemos a verificar si el contador tiene almacenado el valor 1 y mostramos el mensaje respectivo:

```
if cantidad==1:  
    print("Contiene solo un caracter @ el mail ingresado")  
else:  
    print("Incorrecto")
```

Los string en Python son inmutables, esto quiere decir que una vez que los inicializamos no podemos modificar su contenido: [\[1\]](#)

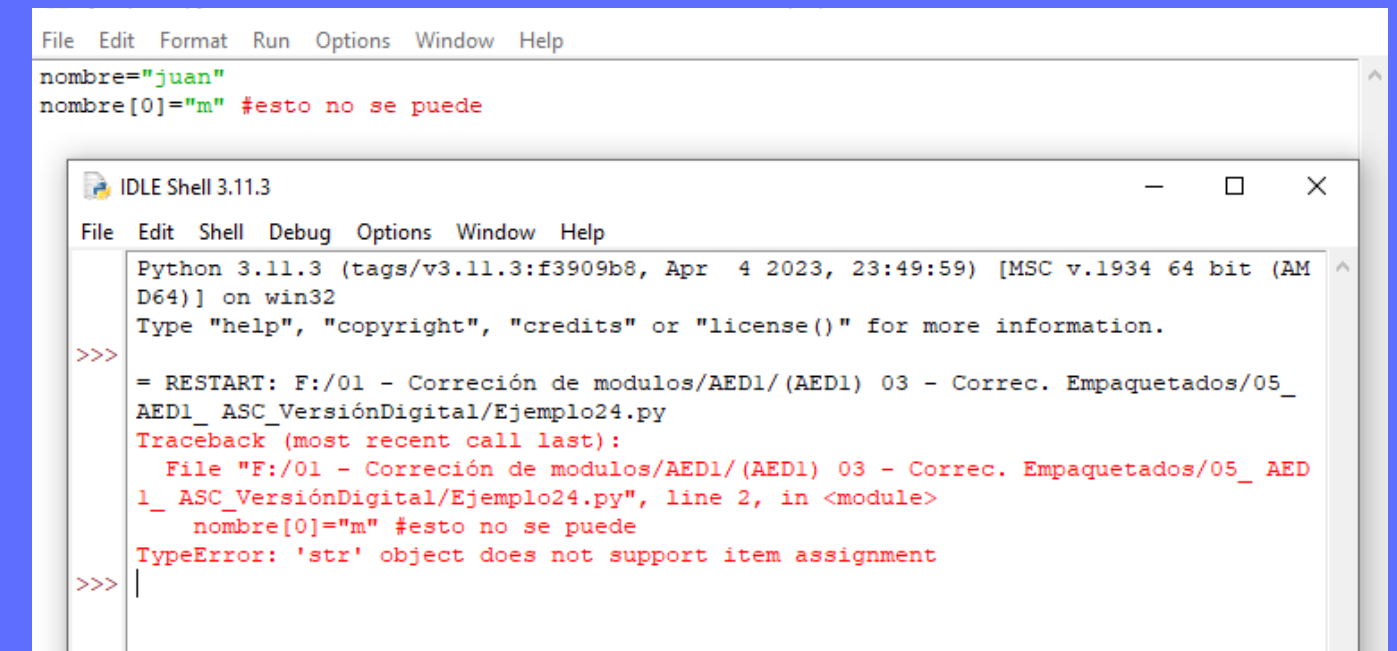
```
nombre="juan"
nombre[0]="m" #esto no se puede
```

¡Importante!

No hay que confundir cambiar parte del string con realizar la asignación de otro string a la misma variable, luego sí es correcto asignar otro valor a un string: [\[2\]](#)

```
nombre="juan"
print(nombre)
nombre="ana"
print(nombre)
```

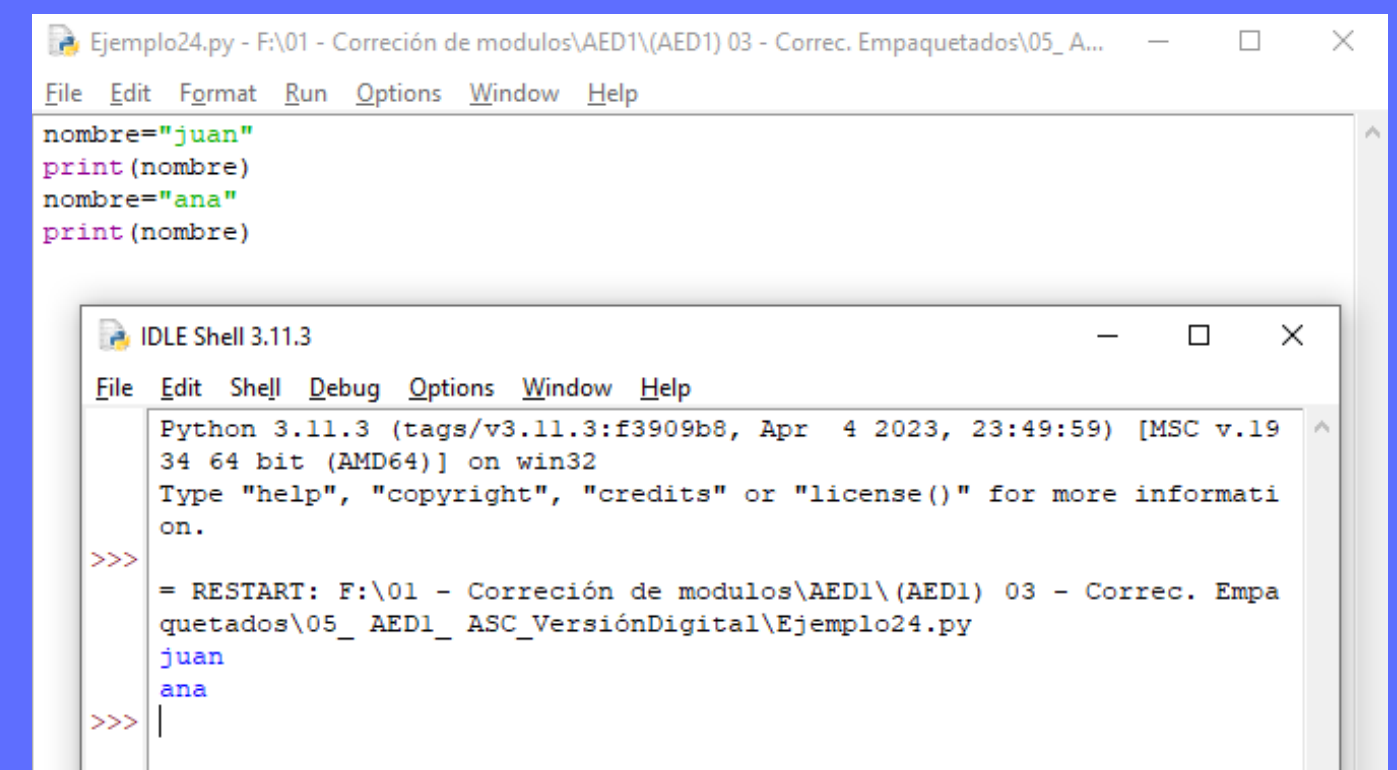
[\[1\]](#)



```
File Edit Format Run Options Window Help
nombre="juan"
nombre[0]="m" #esto no se puede

IDLE Shell 3.11.3
File Edit Shell Debug Options Window Help
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:/01 - Corrección de módulos/AED1/(AED1) 03 - Correc. Empaquetados/05_
AED1_ ASC_VersiónDigital/Ejemplo24.py
Traceback (most recent call last):
  File "F:/01 - Corrección de módulos/AED1/(AED1) 03 - Correc. Empaquetados/05_ AED
l_ ASC_VersiónDigital/Ejemplo24.py", line 2, in <module>
    nombre[0]="m" #esto no se puede
TypeError: 'str' object does not support item assignment
>>> |
```

[\[2\]](#)



```
Ejemplo24.py - F:\01 - Corrección de módulos\AED1\AED1) 03 - Correc. Empaquetados\05_ A...
File Edit Format Run Options Window Help
nombre="juan"
print(nombre)
nombre="ana"
print(nombre)

IDLE Shell 3.11.3
File Edit Shell Debug Options Window Help
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\01 - Corrección de módulos\AED1\AED1) 03 - Correc. Empaquetados\05_ AED1_ ASC_VersiónDigital\Ejemplo24.py
juan
ana
>>> |
```


Métodos propios de las cadenas de caracteres.

Los string tienen una **serie de métodos** (funciones aplicables solo a los string) que nos facilitan la creación de nuestros programas.

Los primeros tres métodos que veremos son:

| `upper()` : Devuelve una cadena de caracteres convertida todos sus caracteres a mayúsculas.

| `lower()` : Devuelve una cadena de caracteres convertida todos sus caracteres a minúsculas.

| `capitalize()` : Devuelve una cadena de caracteres convertida a mayúscula solo su primer carácter y todos los demás a minúsculas.

A continuación, te indicaremos su correcto funcionamiento a través del siguiente ejemplo.

Ejemplo 28

Métodos upper, lower y capitalize

Inicializar un string con la cadena "mAriA" luego llamar a sus métodos upper(), lower() y capitalize(), guardar los datos retornados en otros string y mostrarlos por pantalla.

| Programa: 

```
nombre1="mAriA"  
print(nombre1)  
nombre2=nombre1.upper()  
print(nombre2)  
nombre3=nombre1.lower()  
print(nombre3)  
nombre4=nombre1.capitalize()  
print(nombre4)
```

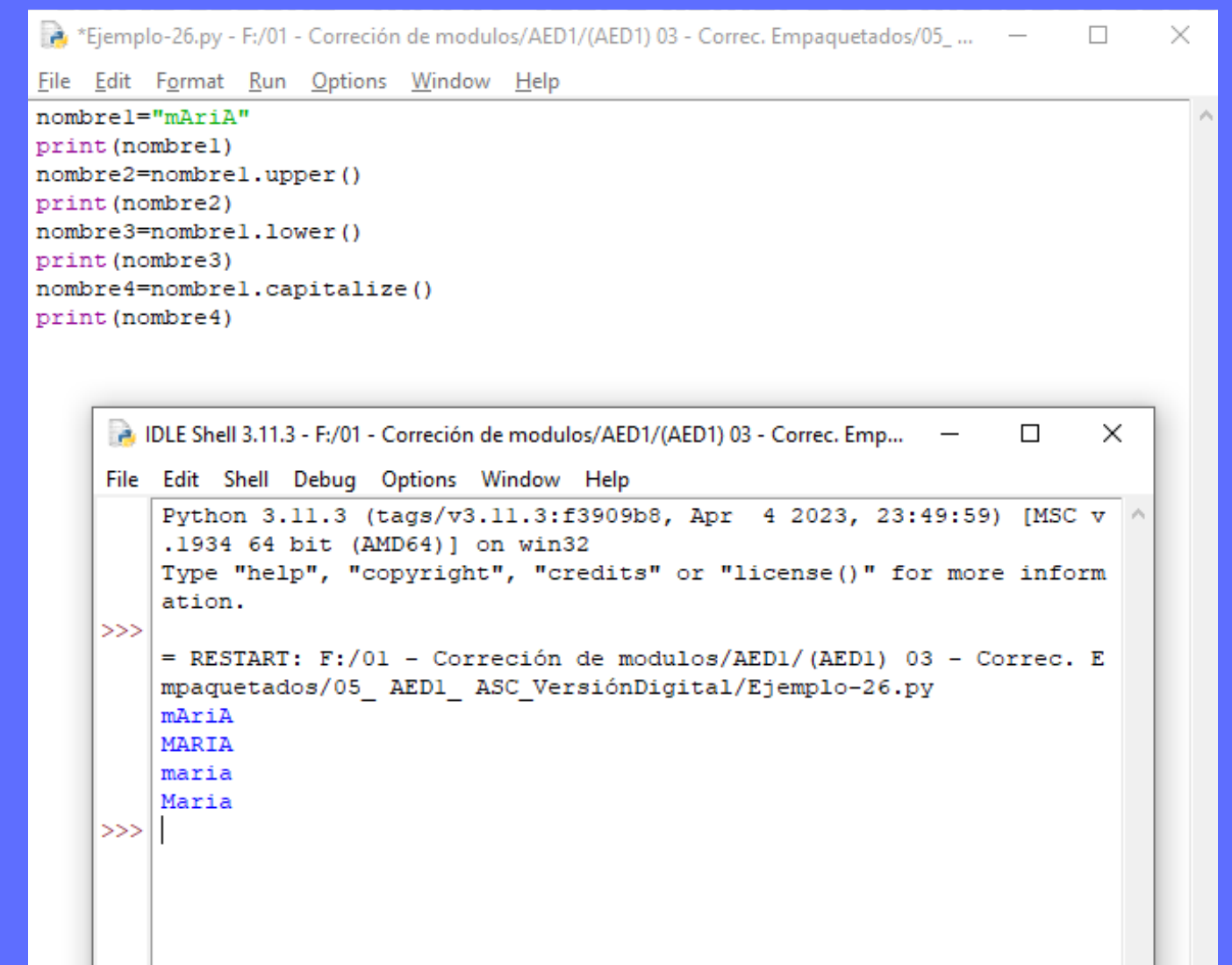
| El resultado de ejecutar este programa es: [|3|](#)

Para llamar a un método del string debemos disponer entre el nombre del string y el método el carácter punto:

```
nombre2=nombre1.upper()
```

Es importante decir que el string nombre1 no se modifica su contenido (*recordá que un string es immutable*) pero el método `upper()` retorna el contenido de la variable nombre1 para convertida a mayúsculas. El dato devuelto se almacena en la variable nombre2.

|3|



The screenshot displays two windows from a Python IDE. The top window, titled '*Ejemplo-26.py - F:/01 - Corrección de modulos/AED1/(AED1) 03 - Correc. Empaquetados/05_...', contains the following Python code:

```
nombre1="mArIA"  
print(nombre1)  
nombre2=nombre1.upper()  
print(nombre2)  
nombre3=nombre1.lower()  
print(nombre3)  
nombre4=nombre1.capitalize()  
print(nombre4)
```

The bottom window, titled 'IDLE Shell 3.11.3 - F:/01 - Corrección de modulos/AED1/(AED1) 03 - Correc. Emp...', shows the output of the script:

```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v  
.1934 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more inform  
ation.  
>>>  
= RESTART: F:/01 - Corrección de modulos/AED1/(AED1) 03 - Correc. E  
mpaquetados/05_ AED1_ ASC_VersiónDigital/Ejemplo-26.py  
mArIA  
MARIA  
maria  
Maria  
>>> |
```



Desempeño 36

| Cargar una oración por teclado. Mostrar luego cuantos espacios en blanco se ingresaron.
Tené en cuenta que un espacio en blanco es igual a " ", en cambio una cadena vacía es ""



Desempeño 37

| Ingresar una oración que puede tener letras tanto en mayúsculas como minúsculas.
Contar la cantidad de vocales.
Crear un segundo string con toda la oración en minúsculas para que sea más fácil disponer la condición que verifica que es una vocal.



Desempeño 38

Soluciones



| Solicitar el ingreso de una clave por teclado y almacenarla en una cadena de caracteres.

Controlá que el string ingresado tenga entre 10 y 20 caracteres para que sea válido, en caso contrario mostrar un mensaje de error.

Como te habrás dado cuenta, a medida que incorporamos más estructuras y funciones de programación, se logra una mayor eficiencia a la hora de diseñar la solución a un problema. Mientras más problemas resuelvas, lograrás una mayor habilidad para detectar cuál algoritmo es el que mejor se ajusta a una determinada situación problemática.

Un consejo: cada vez que se mencione un ejemplo, primero tratá de pensar cómo lo abordarías. Recién después, mirá su resolución, así vas logrando autonomía.

En la próxima clase veremos el uso de datos tipo listas, aprenderemos sus beneficios y cómo utilizar funciones que faciliten nuestro flujo de trabajo al emplear series de datos.

¡Seguimos avanzando!

Créditos

Imágenes

Encabezado: Image by Steve Buissinne from Pixabay

<https://pixabay.com/photos/pawn-chess-pieces-strategy-chess-2430046/>

Tipografía

Para este diseño se utilizó la tipografía *Source Sans Pro* diseñada por Paul D. Hunt.

Extraída de Google Fonts.

Si detectás un error del tipo que fuere (falta un punto, un acento, una palabra mal escrita, un error en código, etc.), por favor comunicate con nosotros a correcciones@issd.edu.ar e indicanos por cada error que detectes la página y el párrafo. Muchas gracias por tu aporte.

Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Solución al desempeño 35

| Programa: 

```
nombre1=input("Ingresa el primer nombre:")
nombre2=input("Ingresa el segundo nombre:")
print("Listado alfabetico:")
if nombre1<nombre2:
    print(nombre1)
    print(nombre2)
else:
    print(nombre2)
    print(nombre1)
```


[Volver a la clase](#)



Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Solución al desempeño 36

| Programa: 


```
oracion=input("Ingrese una oracion:")
cantidad=0
x=0
while x<len(oracion):
    if oracion[x]==" ":
        cantidad=cantidad+1
    x=x+1
print("La cantidad de espacios en blanco ingresado son")
print(cantidad)
```

Solución al desempeño 37

| Programa: ⬇

```
oracion=input("Ingrese una oracion:")
oracionmin=oracion.lower()
cantidad=0
x=0
while x<len(oracionmin):
    if oracionmin[x]=="a" or oracionmin[x]=="e" or oracionmin[x]=="i" or
oracionmin[x]=="o" or oracionmin[x]=="u":
        cantidad=cantidad+1
    x=x+1
print("La cantidad de vocales de la oracion son")
print(cantidad)
```

Solución al desempeño 38

| Programa: 

```
clave=input("Ingrese una clave que tenga entre 10 y 20 caracteres:")
if len(clave)>=10 and len(clave)<=20:
    print("Largo correcto")
else:
    print("Largo incorrecto")
```

[Volver a la clase](#)

