Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

**Tareas**

**Portafolio de tareas**

Nombre: **Martinez Reyes Fernando**

Profesor: **Dr. García Ruiz Alejandro Humberto**

Asignatura: **Diseño Electrónico Basado en Sistemas Embebidos**

**8**vo. Semestre – Grupo “**I**”

2022-1

**ÍNDICE**

[1.- Operaciones con cadenas de caracteres en arduino 3](#_Toc116656365)

[2.- Operaciones con cadena de caracteres en python 8](#_Toc116656366)

[Tabla de métodos de cadena de Python 9](#_Toc116656367)

[3.- Segunda capa del modelo osi 14](#_Toc116656368)

[4.- ¿Qué es una trama en el modelo OSI y su constitución? 15](#_Toc116656369)

[5.- Proceso de serialización y deserialización de cadena 16](#_Toc116656370)

[6.- Multiplicación de matrices 19](#_Toc116656371)

[7.- Instancia de datos 24](#_Toc116656372)

[8.- Cadenas de markov 24](#_Toc116656373)

# 1.- Operaciones con cadenas de caracteres en arduino

Hay diferentes funciones integradas de cadenas en Arduino y estas funciones toman la entrada de dos cadenas y después de realizar alguna función específica, devuelven la salida única. Algunas de las funciones importantes son:

**concat():** esta función se usa para unir dos cadenas para formar una sola cadena, por ejemplo, tenemos dos cadenas; cadena\_A y cadena\_B. Combinaremos ambas cadenas y el resultado de esta función se mostrará en la salida del monitor en serie usando el código:

configuración vacía ( ) { Serial.begin ( 9600 ) ;  
  
Cadena cadena\_A, cadena\_B, cadena\_C;  
  
string\_A = "Esto es LinuxHint" ;  
  
string\_B = "Estás leyendo tutoriales de Arduino" ;  
  
cadena\_A.concat ( "" ) ;  
  
cadena\_A.concat ( cadena\_B ) ;  
  
Serial.println ( cadena\_A ) ;  
  
}  
  
bucle vacío ( ) {  
  
}

Text

Description automatically generated  
En el código anterior, primero declaramos tres variables del tipo de datos de cadena. Almacene algunos valores en string\_A y string\_B y luego concatenemos string\_A y string\_B usando la función concat(). La función concat() une la cadena con otra cadena como en el código anterior, agregamos algo de espacio en string\_A y luego agregamos el valor string\_B en string\_A. Finalmente, usando comunicación serial, imprimí los resultados.

**length():** esta función se usa para calcular la longitud de la cadena específica, cuenta la cantidad de caracteres utilizados en la cadena y devuelve el valor entero. Considere un ejemplo:

String arr = "¡Bienvenido a LinuxHint!" ;configuración vacía ( ) {  
  
Serial.begin ( 9600 ) ;  
  
Serial.print ( "Longitud de la cadena:" ) ;  
  
int arr\_len = arr.longitud ( ) ;  
  
Serial.println ( arr\_len ) ;  
  
}  
  
bucle vacío ( ) {  
  
}

Text

Description automatically generated  
El código anterior muestra que la longitud de la cadena es 21.

**toUpperCase() y toLowerCase():** podemos cambiar la distinción entre mayúsculas y minúsculas de la cadena usando estas funciones. Consideraremos un ejemplo, en el que cambiaremos la distinción entre mayúsculas y minúsculas de la cadena a mayúsculas usando la función toUpperCase():

configuración vacía ( ) { Serial.begin ( 9600 ) ;  
  
Cadena cadena\_A;  
  
string\_A = "Esto es LinuxHint" ;  
  
cadena\_A.toUpperCase ( ) ;  
  
Serial.println ( cadena\_A ) ;  
  
}  
  
bucle vacío ( ) {  
  
}

Text

Description automatically generated

De manera similar, podemos cambiar el valor de la cadena a minúsculas usando la función “toLowerCase()”.

**compareTo():** ​​esta función se utiliza para averiguar la longitud de la cadena y compara las cadenas en función de su carácter. Se consideran los valores ASCII de los caracteres, lo que significa que sabemos que "a" viene antes de "b", pero viene después de "A" porque el código ASCII de "a" es 97 y "A" es 65. Considere un ejemplo:

configuración vacía ( ) { Serial.begin ( 9600 ) ;  
  
Cadena cadena\_A, cadena\_B, cadena\_C;  
  
string\_A = "Esto es LinuxHint" ;  
  
string\_B = "Estás leyendo tutoriales de Arduino" ;  
  
string\_C = "Esto es LinuxHint" ;  
  
Serial.println ( "Resultado de la comparación de string\_A con string\_B: " ) ;  
  
Serial.println ( cadena\_A.compareTo ( cadena\_B ) ) ;  
  
Serial.println ( "Resultado de la comparación de string\_A con string\_C: " );  
  
Serial.println ( cadena\_A.compareTo ( cadena\_C ) ) ;  
  
}  
  
bucle vacío ( ) {  
  
}

A partir del resultado que se muestra en la salida, se sabe que string\_A y string\_c son iguales, por lo que devuelve el "0" y en comparación con string\_A y string\_B, "-5" significa que el valor de string\_A viene antes que el valor de string\_B.

Hay algunas otras funciones que se utilizan con las cadenas:

|  |  |
| --- | --- |
| **Funciones** | **Explicación** |
| **carácter(n)** | Esta función se utiliza para acceder a cualquier carácter particular de la cadena. |
| **termina con()** | Si ambas cadenas terminan con los mismos caracteres, devuelve verdadero |
| **es igual ()** | Si ambas cadenas son iguales, devuelve verdadero. Cabe señalar que esta función distingue entre mayúsculas y minúsculas, lo que significa que "hammad" y "HAMMAD" no son iguales |
| **igualsignorecase()** | Funciona de manera similar a la función equals() pero no distingue entre mayúsculas y minúsculas, lo que significa que "hammad" y "HAMMAD" son iguales |
| **reemplazar()** | Reemplaza todas las instancias de la cadena A con la cadena B |
| **comienza con()** | Devuelve verdadero si ambas cadenas comienzan con el mismo carácter |
| **retirar()** | Se utiliza para eliminar cualquier carácter de la cadena de cualquier índice en particular. |
| **reserva()** | Reserva un búfer para la manipulación de la cadena en la memoria. |
| **aCharArray** | Copia los caracteres completos de la cadena en el búfer. |
| **toInt** | Se utiliza para convertir una cadena en un número entero. |
| **recortar()** | Se utiliza para recortar cualquier parte de la cuerda. |

**Conclusión**

Las cadenas se usan en Arduino para almacenar la serie de caracteres que se pueden usar para muchos propósitos, como mostrar la salida en la interfaz LCD con Arduino. En este artículo, hemos explicado ambos tipos de cadenas con ejemplos. Además, discuta en detalle algunas funciones importantes utilizadas en Object Strings.

# 2.- Operaciones con cadena de caracteres en python

[**La cadena de Python**](https://www.geeksforgeeks.org/python-strings/) es una secuencia de caracteres Unicode que está encerrada entre comillas. En este artículo, discutiremos la función incorporada, es decir, las funciones proporcionadas por Python para operar en cadenas.

**Nota:** cada método de cadena no cambia la cadena original, sino que devuelve una nueva cadena con los atributos modificados.

**Cambio de caja de cuerdas**

Las siguientes funciones se utilizan para cambiar el caso de las cadenas.

* [**lower():**](https://www.geeksforgeeks.org/python-string-lower/) Convierte todos los caracteres en mayúsculas de una cadena en minúsculas
* [**upper ():**](https://www.geeksforgeeks.org/python-string-upper/) convierte todos los caracteres en minúsculas en una cadena en mayúsculas
* [**title ():**](https://www.geeksforgeeks.org/title-in-python/) convierte la cadena en mayúsculas y minúsculas

## Tabla de métodos de cadena de Python

| Nombre de la función | Descripción |
| --- | --- |
| [capitalizar()](https://www.geeksforgeeks.org/string-capitalize-python/) | Convierte el primer carácter de la cadena en una letra mayúscula (mayúscula) |
| [casefold()](https://www.geeksforgeeks.org/casefold-string-python/) | Implementa la coincidencia de cadenas sin mayúsculas y minúsculas |
| [centro()](https://www.geeksforgeeks.org/string-center-python/) | Rellene la cadena con el carácter especificado. |
| [contar()](https://www.geeksforgeeks.org/python-string-count/) | Devuelve el número de ocurrencias de una subcadena en la cadena. |
| [codificar()](https://www.geeksforgeeks.org/python-strings-encode-method/) | Codifica cadenas con el esquema codificado especificado |
| [termina con()](https://www.geeksforgeeks.org/string-endswith-python/) | Devuelve "Verdadero" si una cadena termina con el sufijo dado |
| [expandir pestañas ()](https://www.geeksforgeeks.org/python-expandtabs-method/) | Especifica la cantidad de espacio que se sustituirá con el símbolo "\t" en la cadena |
| [encontrar()](https://www.geeksforgeeks.org/python-string-find/) | Devuelve el índice más bajo de la subcadena si se encuentra |
| [formato()](https://www.geeksforgeeks.org/python-format-function/) | Formatea la cadena para imprimirla en la consola |
| [formato\_mapa()](https://www.geeksforgeeks.org/python-string-format_map/) | Formatea valores especificados en una cadena usando un diccionario |
| [índice()](https://www.geeksforgeeks.org/python-string-index-applications/) | Devuelve la posición de la primera aparición de una subcadena en una cadena |
| [isalnum()](https://www.geeksforgeeks.org/python-string-isalnum/) | Comprueba si todos los caracteres de una cadena dada son alfanuméricos o no |
| [isalfa()](https://www.geeksforgeeks.org/python-string-isalpha-application/) | Devuelve "Verdadero" si todos los caracteres de la cadena son alfabetos |
| [isdecimal()](https://www.geeksforgeeks.org/python-string-isdecimal/) | Devuelve verdadero si todos los caracteres de una cadena son decimales |
| [esdigito()](https://www.geeksforgeeks.org/python-string-isdigit-application/) | Devuelve "Verdadero" si todos los caracteres de la cadena son dígitos |
| [esidentificador()](https://www.geeksforgeeks.org/python-string-isidentifier/) | Comprobar si una cadena es un identificador válido o no |
| [es bajo()](https://www.geeksforgeeks.org/python-string-islower-method/) | Comprueba si todos los caracteres de la cadena están en minúsculas |
| [isnumeric()](https://www.geeksforgeeks.org/python-string-isnumeric-application/) | Devuelve "Verdadero" si todos los caracteres de la cadena son caracteres numéricos |
| [es imprimible()](https://www.geeksforgeeks.org/isprintable-python-application/) | Devuelve "Verdadero" si todos los caracteres de la cadena son imprimibles o si la cadena está vacía |
| [esespacio()](https://www.geeksforgeeks.org/python-string-isspace-application/) | Devuelve "Verdadero" si todos los caracteres de la cadena son caracteres de espacio en blanco |
| [estitulo()](https://www.geeksforgeeks.org/python-string-istitle/) | Devuelve "Verdadero" si la cadena es una cadena en mayúsculas y minúsculas |
| [essuperior()](https://www.geeksforgeeks.org/python-string-isupper-method/) | Comprueba si todos los caracteres de la cadena están en mayúsculas |
| [unirse()](https://www.geeksforgeeks.org/join-function-python/) | Devuelve una cadena concatenada |
| [solo](https://www.geeksforgeeks.org/string-rjust-ljust-python/) () | La izquierda alinea la cadena de acuerdo con el ancho especificado |
| [más bajo()](https://www.geeksforgeeks.org/python-string-lower/) | Convierte todos los caracteres en mayúsculas de una cadena en minúsculas |
| [lstrip()](https://www.geeksforgeeks.org/python-string-lstrip-method/) | Devuelve la cadena con los caracteres principales eliminados |
| [hacer trans](https://www.geeksforgeeks.org/python-maketrans-translate-functions/) () | Devuelve una tabla de traducción. |
| [dividir()](https://www.geeksforgeeks.org/string-partition-python/) | Divide la cadena en la primera aparición del separador |
| [reemplazar()](https://www.geeksforgeeks.org/python-string-replace/) | Reemplaza todas las apariciones de una subcadena con otra subcadena |
| [encontrar()](https://www.geeksforgeeks.org/python-string-rfind/) | Devuelve el índice más alto de la subcadena |
| [rindex()](https://www.geeksforgeeks.org/string-rindex-python/) | Devuelve el índice más alto de la subcadena dentro de la cadena |
| [ajustar()](https://www.geeksforgeeks.org/string-rjust-ljust-python/) | Alinea a la derecha la cadena de acuerdo con el ancho especificado |
| [rpartición()](https://www.geeksforgeeks.org/python-string-rpartition/) | Dividir la cadena dada en tres partes |
| [rsplit()](https://www.geeksforgeeks.org/python-string-rsplit/) | Divida la cadena desde la derecha por el separador especificado |
| [rstrip()](https://www.geeksforgeeks.org/python-string-rstrip/) | Elimina los caracteres finales |
| [líneas divididas()](https://www.geeksforgeeks.org/python-string-splitlines/) | Dividir las líneas en los límites de línea |
| [comienza con()](https://www.geeksforgeeks.org/python-string-startswith/) | Devuelve "Verdadero" si una cadena comienza con el prefijo dado |
| [banda()](https://www.geeksforgeeks.org/python-string-strip/) | Devuelve la cadena con caracteres iniciales y finales |
| [intercambio de casos ()](https://www.geeksforgeeks.org/python-string-swapcase/) | Convierte todos los caracteres en mayúsculas a minúsculas y viceversa |
| [título()](https://www.geeksforgeeks.org/title-in-python/) | Convertir cadena a mayúsculas y minúsculas |
| [traducir()](https://www.geeksforgeeks.org/python-string-translate/) | Modifique la cadena de acuerdo con las asignaciones de traducción dadas |
| [superior()](https://www.geeksforgeeks.org/python-string-upper/) | Convierte todos los caracteres en minúsculas de una cadena en mayúsculas |
| [zfill()](https://www.geeksforgeeks.org/python-string-zfill/) | Devuelve una copia de la cadena con los caracteres '0' en el lado izquierdo de la cadena. |

# 3.- Segunda capa del modelo osi

**2. Capa de enlace de datos (DLL) (Capa 2):**

La capa de enlace de datos es responsable de la entrega del mensaje de nodo a nodo. La función principal de esta capa es asegurarse de que la transferencia de datos esté libre de errores de un nodo a otro, sobre la capa física. Cuando llega un paquete a una red, es responsabilidad de DLL transmitirlo al Host utilizando su dirección MAC.   
La capa de enlace de datos se divide en dos subcapas:

1. Control de enlace lógico (LLC)
2. Control de acceso a medios (MAC)

El paquete recibido de la capa de red se divide en tramas según el tamaño de trama de la NIC (tarjeta de interfaz de red). DLL también encapsula la dirección MAC del remitente y el destinatario en el encabezado.

La dirección MAC del receptor se obtiene colocando una solicitud ARP (Protocolo de resolución de direcciones) en el cable preguntando "¿Quién tiene esa dirección IP?" y el host de destino responderá con su dirección MAC.

Icon

Description automatically generated

Las funciones de la capa de enlace de datos son:

1. **Encuadre:** el encuadre es una función de la capa de enlace de datos. Proporciona una forma para que un remitente transmita un conjunto de bits que son significativos para el receptor. Esto se puede lograr adjuntando patrones de bits especiales al principio y al final del cuadro.
2. **Direccionamiento físico:** Después de crear tramas, la capa de enlace de datos agrega direcciones físicas (dirección MAC) del remitente y/o receptor en el encabezado de cada trama.
3. **Control de errores:** la capa de enlace de datos proporciona el mecanismo de control de errores en el que detecta y retransmite tramas dañadas o perdidas.
4. **Control de flujo:** la tasa de datos debe ser constante en ambos lados; de lo contrario, los datos pueden corromperse. Por lo tanto, el control de flujo coordina la cantidad de datos que se pueden enviar antes de recibir el reconocimiento.
5. **Control de acceso:** cuando varios dispositivos comparten un solo canal de comunicación, la subcapa MAC de la capa de enlace de datos ayuda a determinar qué dispositivo tiene control sobre el canal en un momento dado.

*\* El paquete en la capa de enlace de datos se conoce como* ***trama.***

\*\* La capa de enlace de datos es manejada por la NIC (tarjeta de interfaz de red) y los controladores de dispositivos de las máquinas host.

\*\*\* Switch y Bridge son dispositivos de capa de enlace de datos.

La capa de enlace de datos establece y finaliza una conexión entre dos nodos conectados físicamente en una red. Divide los paquetes en marcos y los envía desde el origen hasta el destino. Esta capa se compone de dos partes: control de enlace lógico (LLC), que identifica protocolos de red, realiza verificación de errores y sincroniza marcos, y control de acceso a medios (MAC), que usa direcciones MAC para conectar dispositivos y definir permisos para transmitir y recibir datos.

# 4.- ¿Qué es una trama en el modelo OSI y su constitución?

En una [red Ethernet](https://www.ionos.mx/digitalguide/servidores/know-how/ethernet-ieee-8023/) los dispositivos se intercambian paquetes de datos entre sí, los llamados paquetes Ethernet. En su contenido se incluye la trama Ethernet (a menudo denominada también trama de datos), que a su vez se divide en varios conjuntos de datos. Estos registros consisten en código binario que proporciona información importante, incluyendo direcciones, información de control, datos de uso y sumas de comprobación.

Dependiendo del estándar Ethernet, las tramas Ethernet se estructuran de forma diferente y pueden contener más o menos campos de datos, dependiendo del [protocolo de red](https://www.ionos.mx/digitalguide/servidores/know-how/los-protocolos-de-red-en-la-transmision-de-datos/).

En el [modelo OSI](https://www.ionos.mx/digitalguide/servidores/know-how/el-modelo-osi-un-referente-para-normas-y-protocolos/), la trama se encuentra en la capa de enlace responsable de la transmisión y separa el flujo de datos de bits en bloques o tramas. La primera versión de Ethernet (Ethernet I) todavía estaba basada en campos de datos de 16 bits sin bytes definidos. Las tramas Ethernet modernas se utilizaron por primera vez en la llamada estructura Ethernet II antes de que el IEEE (Institute of Electrical and Electronics Engineers) desarrollara Ethernet en 1983 en el protocolo estándar IEEE 802.3 (primer IEEE 802.3raw).

En el transcurso del progreso técnico, la estructura se adaptó varias veces para que las tramas pudieran transportar datos informativos más definidos. En formato IEEE 802.3, se crearon la trama MAC básica y la trama SNAP para el proceso de multiplexación y para los datos de identificación relacionados con el fabricante. Para el desarrollo de la VLAN, la trama Ethernet II y la trama Ethernet IEEE 802.3 se desarrollaron como variantes etiquetadas y contienen datos de control que pueden asignar la trama a una VLAN específica.

Una trama Ethernet debe tener **al menos 64 bytes** para que funcione la detección de colisiones y puede tener **un máximo de 1518 bytes**. El paquete comienza con un preámbulo, que controla la sincronización entre el emisor y el receptor, y un **SFD** (Start Frame Delimiter), que define la trama. Ambos valores son secuencias de bits en el formato "10101010 ....". La trama en sí contiene información sobre las **direcciones de origen y destino** (formato MAC), información de control (en el caso de Ethernet II el campo de tipo, una especificación de longitud), seguida por el registro de datos que se envía (Data). Una secuencia de comprobación de trama (**FCS**) es un código de detección de errores que cierra la trama (si no se cuenta al preámbulo y al SFD). El paquete se completa con una Inter Frame Gap, que define una pausa de transmisión de 9.6 μs.

Ethernet II utiliza la estructura de trama clásica con un **campo de tipo** (Type) que define varios protocolos en la capa de red. En el modelo OSI, la capa de red es importante para conectar y proporcionar direcciones de red. El campo tipo fue reemplazado por una especificación de longitud en formatos de trama posteriores.

# 5.- Proceso de serialización y deserialización de cadena

**Text

Description automatically generated**

**Definiciones básicas.**

* Serialización: Consiste en convertir un objeto de Python (normalmente una lista o diccionario) en un string.
* Deserialización: Consiste en convertir un string en un objeto de Python (normalmente una lista o diccionario).

La serialización y deserialización de objetos Python es un aspecto importante de cualquier programa no trivial. Si guardas algo en un archivo en Python, si lees un archivo de configuración o si respondes a una solicitud HTTP, puedes serializar y deserializar objetos.

En cierto sentido, la serialización y la deserialización son las cosas más aburridas del mundo. ¿A quién le importan todos los formatos y protocolos? Solo deseas conservar o transmitir algunos objetos de Python y recuperarlos intactos más tarde.

Esta es una forma muy saludable de ver el mundo a nivel conceptual. Pero, a nivel pragmático, el esquema, formato o protocolo de serialización que elijas puede determinar qué tan rápido se ejecuta tu programa, qué tan seguro es, cuánta libertad tienes para mantener tu estado y qué tan bien vas a interoperar con otros sistemas.

La razón por la que hay tantas opciones es que diferentes circunstancias requieren diferentes soluciones. No hay "un tamaño que se adapte a todos". En este tutorial de dos partes, repasaré los pros y los contras de los esquemas de serialización y deserialización más exitosos, mostraré cómo usarlos y proporcionaré pautas para elegir entre ellos cuando se enfrente a un caso de uso específico.

**Ejemplo de ejecución**

En las siguientes secciones, serializaré y deserializaré los mismos gráficos de objetos de Python usando diferentes serializadores. Para evitar la repetición, definiré estos gráficos de objetos aquí.

**Gráfico de objetos simple**

El gráfico de objetos simple es un diccionario que contiene una lista de números enteros, una cadena, un float, un booleano y un None.

Graphical user interface, text, application

Description automatically generated

**Gráfico de objetos complejo**

El gráfico de objetos complejo también es un diccionario, pero contiene un objeto datetime y una instancia de clase definida por el usuario que tiene un atributo self.simple, que se configura en el gráfico de objetos simple.

Text

Description automatically generated

**JSON**

JSON (JavaScript Object Notation) forma parte de la biblioteca estándar de Python desde Python 2.5. Lo consideraré un formato nativo en este momento. Es un formato basado en texto y es el rey no oficial de la web en lo que respecta a la serialización de objetos. Su sistema de tipos presenta naturalmente JavaScript, por lo que es bastante limitado.

Serialicemos y deserialicemos los gráficos de objetos simples y complejos y veamos qué sucede. La interfaz es casi idéntica a la interfaz pickle. Tienes funciones dump(), dumps(), load() y loads().

Graphical user interface, text

Description automatically generated

# 6.- Multiplicación de matrices

Una matriz es un arreglo bidimensional. Cada elemento de la matriz tiene dos índices. Veamos un ejemplo en NumPy:

Graphical user interface, text, application

Description automatically generated

En el código anterior, tenemos la matriz [[6, 7], [8, 9]]A. Pedimos el elemento dado en (0,0), y nuestra salida regresa 6. Cuando queremos definir la forma de nuestra matriz, usamos el número de filas por el número de columnas. Eso significa que la matriz A tiene una forma de 2x2.

Ahora, echemos un vistazo a algunos métodos diferentes de multiplicación de matrices NumPy.

**Métodos de multiplicación de matrices NumPy**

Hay **tres formas principales** de realizar la multiplicación de matrices NumPy:

* np.dot(array a, array b): devuelve el producto escalar o escalar de dos matrices
* np.matmul(array a, array b): devuelve el producto matricial de dos matrices
* np.multiply(array a, array b): devuelve la multiplicación de matrices por elementos de dos matrices

Echemos un vistazo más de cerca a cada uno de los tres métodos:

**Multiplicación escalar o producto escalar con numpy.dot**

La multiplicación escalar es una forma simple de multiplicación de matrices. Un escalar es solo un número, como 1, 2o 3. En la multiplicación escalar, **multiplicamos un escalar por una matriz** . Cada elemento de la matriz se multiplica por el escalar, lo que hace que la salida tenga la misma forma que la matriz original.

Con la multiplicación escalar, el **orden no importa** . Obtendremos el mismo resultado si multiplicamos el escalar por la matriz o la matriz por el escalar.

Echemos un vistazo a un ejemplo:

Text

Description automatically generated with medium confidence

Ahora, multipliquemos una matriz bidimensional por otra matriz bidimensional. Al multiplicar dos matrices, el **orden importa** . Eso significa que la matriz A multiplicada por la matriz B no es lo mismo que la matriz B multiplicada por la matriz A.

Antes de comenzar, veamos una imagen de cómo se hace la multiplicación.

Table

Description automatically generated

Text

Description automatically generated

**Producto matriz con numpy.matmul**

La matmul()función nos da el producto matricial de dos matrices bidimensionales. Con este método, no podemos usar valores escalares para nuestra entrada. Si uno de nuestros argumentos es un arreglo unidimensional, la función lo convierte en una matriz agregando un 1 a su dimensión. Esto se elimina después de que se realiza la multiplicación.

Si uno de nuestros argumentos es mayor que 2-d, la función lo trata como una pila de matrices en los dos últimos índices. El matmul()método es excelente para momentos en los que no estamos seguros de cuáles serán las dimensiones de nuestras matrices.

Veamos algunos ejemplos:

**Multiplicar una matriz bidimensional por otra matriz bidimensional**

Text

Description automatically generated

**Multiplicar una matriz de 2 dimensiones por una matriz de 1 dimensión**

A picture containing text

Description automatically generated

**Una matriz con dimensiones superiores a 2-d**

Graphical user interface, text

Description automatically generated with medium confidence

**Multiplicación de matrices por elementos con numpy.multiply**

El numpy.multiply()método toma dos matrices como entradas y realiza una multiplicación por elementos en ellas. La multiplicación por elementos, o producto de Hadamard, multiplica cada elemento de la primera matriz por el elemento equivalente en la segunda matriz. Al usar este método, ambas matrices deben tener las mismas dimensiones.

Veamos un ejemplo:

Graphical user interface

Description automatically generated with low confidence

Podemos pasar ciertas filas, columnas o submatrices al numpy.multiply()método. Los tamaños de las filas, columnas o submatrices que pasamos como nuestros operandos deben ser los mismos. Veamos un ejemplo:

Text

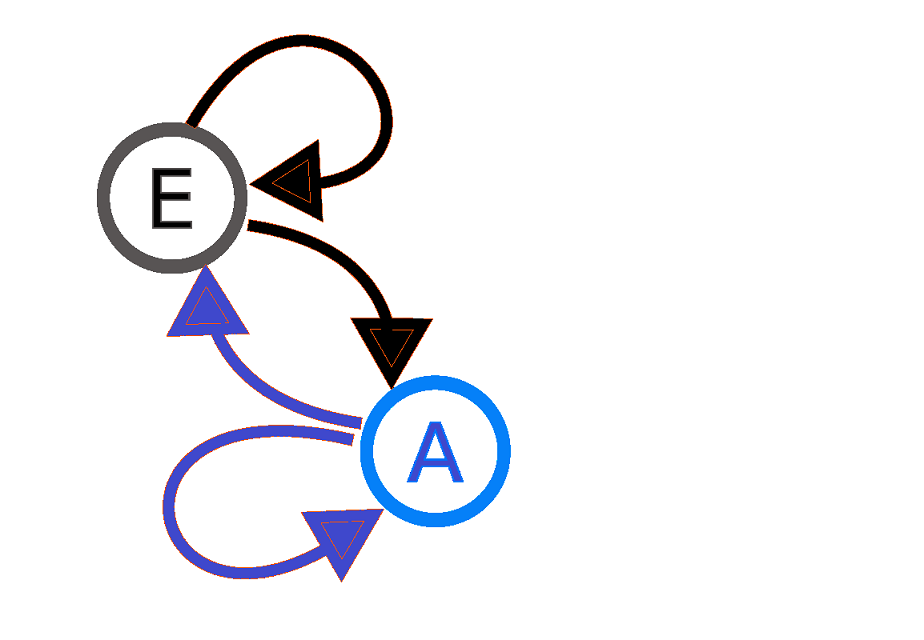
Description automatically generated with medium confidence

# 7.- Instancia de datos

Una instancia de datos es una pieza particular de información digital. Esto es comparable a cómo un organismo es una planta o animal particular de una especie. Podría haber numerosos datos que comparten un valor común, pero una instancia de datos solo se referiría a uno de estos, no al conjunto completo. Por ejemplo, en una variable o una clase que tiene varias instancias, la modificación de una instancia no afectaría necesariamente a ninguna de las otras instancias.

# 8.- Cadenas de markov

Las cadenas de Markov, llamadas así por Andrey Markov, son un modelo estocástico que representa una secuencia de posibles eventos donde las predicciones o probabilidades para el siguiente estado se basan únicamente en su estado de evento anterior, no en los estados anteriores. En palabras sencillas, la probabilidad de que n+1 pasos sean x depende solo del paso n, no de la secuencia completa de pasos anterior a n. Esta propiedad se conoce como propiedad de Markov o falta de memoria.



Un diagrama que representa un proceso de Markov de dos estados (aquí, E y A). Aquí las flechas se originaron en el estado actual y apuntan al estado futuro y el número asociado con las flechas indica la probabilidad de que el proceso de Markov cambie de un estado a otro. Por ejemplo, si el proceso de Markov está en el estado E, entonces la probabilidad de que cambie al estado A es 0,7, mientras que la probabilidad de que permanezca en el mismo estado es 0,3. De manera similar, para cualquier proceso en el estado A, la probabilidad de cambiar a Estado es 0,4 y la probabilidad de permanecer en el mismo estado es 0,6.

**¿Cómo representar la cadena de Markov?**

Del diagrama del proceso de Markov de dos estados, podemos entender que la cadena de Markov es un gráfico dirigido. Entonces podemos representarlo con la ayuda de una matriz de adyacencia.

                +——+——+

                 | un | mi | — Cada elemento denota el peso de probabilidad del borde

     +——+——+——+ conectando los dos vértices correspondientes

      | un | 0.6 | 0.4 | — 0,4 es la probabilidad de que el estado A pase al estado E y 0,6 es la probabilidad

     +——+——+——+ para permanecer en el mismo estado

      | mi | 0.7 | 0.3 | — 0,7 es la probabilidad de que el estado E pase al estado A y 0,3 es la probabilidad

     +——+——+——+ para permanecer en el mismo estado

Esta matriz también se llama **Matriz de Transición** . Si la cadena de Markov tiene N estados posibles, la matriz será una matriz NxN. Cada fila de esta matriz debe sumar 1. Además de esto, una cadena de Markov también tiene un **vector de estado inicial** de orden Nx1. Estas dos entidades son imprescindibles para representar una cadena de Markov.

***Matriz de transición de N pasos:*** ahora aprendamos matrices de transición de orden superior. Nos ayuda a encontrar la posibilidad de que esa transición ocurra en varios pasos. En pocas palabras, ¿cuál será la probabilidad de pasar del estado A al estado E en el paso N? En realidad, hay una manera muy simple de calcularlo. Esto se puede determinar calculando el valor de entrada (A, ) de la matriz obtenida al elevar la matriz de transición a la potencia de N.

**Tipos de cadena de Markov:**

***Cadenas de Markov de tiempo discreto*** : Esto implica que el conjunto de índices T (estado del proceso en el momento t) es un conjunto contable aquí o podemos decir que los cambios ocurren en estados específicos. Generalmente, el término "cadena de Markov" se usa para DTMC.

***Cadenas de Markov de tiempo continuo*** : aquí el conjunto de índices T (estado del proceso en el tiempo t) es un continuo, lo que significa que los cambios son continuos en CTMC.

**Propiedades de la cadena de Markov:**

1. Se dice que una cadena de Markov es **irreducible** si podemos pasar de un estado a otro en un solo paso o en más de uno.
2. Se dice que un estado en una cadena de Markov es **periódico** si volver a él requiere un múltiplo de algún número entero mayor que 1, el máximo común divisor de todas las posibles longitudes de la ruta de retorno será el período de ese estado.
3. Se dice que un estado en una cadena de Markov es **Transitorio** si existe una probabilidad distinta de cero de que la cadena nunca vuelva al mismo estado; de lo contrario, es Recurrente.
4. Un estado en una cadena de Markov se llama **Absorbente** si no hay forma posible de salir de ese estado. Los estados absorbentes no tienen transiciones salientes.

**Aplicación de la cadena de Markov:**

Las cadenas de Markov hacen que el estudio de muchos procesos del mundo real sea mucho más simple y fácil de entender. Usando la cadena de Markov podemos obtener algunos resultados útiles, como la distribución estacionaria y muchos más.

1. MCMC (Cadena de Markov Monte Carlo), que da una solución a los problemas que surgen del factor de normalización, se basa en la Cadena de Markov.
2. Las cadenas de Markov se utilizan en teoría de la información, motores de búsqueda, reconocimiento de voz, etc.
3. La cadena de Markov tiene enormes posibilidades, futuro e importancia en el campo de la ciencia de datos y se solicita a los lectores interesados ​​que aprendan estas cosas correctamente para ser una persona competente en el campo de la ciencia de datos.

**Supuestos para la cadena de Markov:**

1. El sistema estadístico contiene un número finito de estados.
2. Los estados son mutuamente excluyentes y colectivamente exhaustivos.
3. La probabilidad de transición de un estado a otro estado es constante en el tiempo.

Los procesos de Markov son bastante comunes en problemas de la vida real y las cadenas de Markov se pueden implementar fácilmente debido a su propiedad de falta de memoria. El uso de la cadena de Markov puede simplificar el problema sin afectar su precisión.

Tomemos un ejemplo para comprender la ventaja de esta herramienta, supongamos que mi amigo sugiere comer. Puedo decir que no quiero una pizza porque la tengo hace una hora. Pero, ¿es apropiado si digo que no quiero una pizza porque la tengo hace dos meses? Eso significa que, en este caso, mi probabilidad de elegir una comida depende completamente de mi comida inmediatamente anterior. Aquí está la efectividad de la Cadena de Markov.

**Aplicaciones de la vida real**

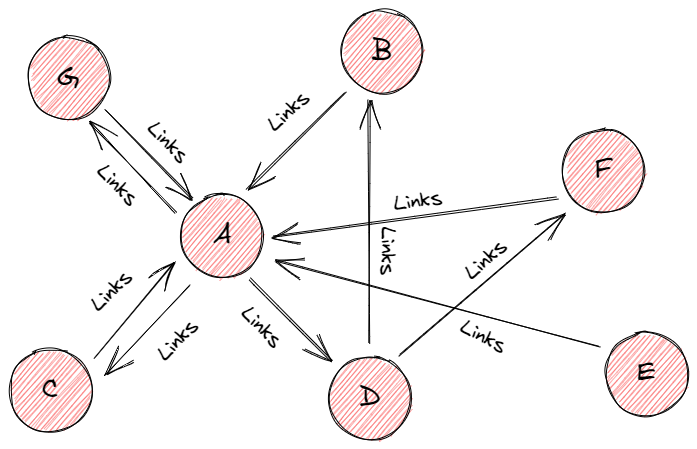
Después de la explicación, examinemos algunas de las aplicaciones reales donde son útiles. Se sorprenderá de cuánto tiempo ha estado usando cadenas de Markov sin su conocimiento.

**“Pagerank” utilizado por Google**

Page y Brin crearon el algoritmo, que se denominó "PageRank" en honor a Larry Page. PageRank es una de las estrategias que utiliza Google para evaluar la relevancia o el valor de una página. Para usar el algoritmo de PageRank, asumimos que la web es un gráfico dirigido, con páginas web que actúan como nodos e hipervínculos que actúan como bordes. PageRank asigna un valor a una página según la cantidad de backlinks que hacen referencia a ella. Una página que está conectada a muchas otras páginas obtiene una clasificación alta.

Para calcular la puntuación de la página, tenga en cuenta que el navegante puede elegir cualquier página. Sin embargo, no siempre eligen las páginas en el mismo orden. La mayoría de las veces, un navegante seguirá los enlaces de una página secuencialmente, por ejemplo, desde la página "A", el navegante seguirá las conexiones salientes y luego irá a uno de los vecinos de la página "A". Sin embargo, este no es siempre el caso.

Una proporción menor pero significativa del tiempo, el navegante abandonará la página actual y seleccionará una página aleatoria de la web para "teletransportarse". Para dar cuenta de tal escenario, Page y Brin idearon el factor de amortiguamiento, que cuantifica la probabilidad de que el navegante abandone la página actual y se "teletransporte" a una nueva. Debido a que el usuario puede teletransportarse a cualquier página web, cada página tiene la posibilidad de ser elegida por la enésima página.



**Predecir las tendencias del mercado**

El mercado de valores es un sistema volátil con un alto grado de imprevisibilidad. Las cadenas de Markov y sus diagramas asociados pueden usarse para estimar la probabilidad de varios climas del mercado financiero y así pronosticar la probabilidad de circunstancias futuras del mercado. Podemos ver que este sistema cambia entre un cierto número de estados al azar. El espacio de estados se refiere a todas las combinaciones concebibles de estos estados. En nuestra situación, podemos ver que un movimiento del mercado de valores solo puede tomar tres formas.

* **Los mercados alcistas** son momentos en los que los precios normalmente suben como resultado de la perspectiva positiva de los jugadores para el futuro.
* **Los mercados bajistas** son momentos en los que los precios suelen caer como resultado de la perspectiva negativa de los jugadores sobre el futuro.
* **Los mercados estancados** son aquellos en los que no hay ni una baja ni un aumento en los precios generales.

Los mercados justos creen que la información del mercado se distribuye uniformemente entre sus participantes y que los precios varían aleatoriamente. Esto indica que todos los actores tienen el mismo acceso a la información, por lo que ningún actor tiene una ventaja debido a la información privilegiada. Ciertos patrones, así como su probabilidad estimada, pueden descubrirse mediante el examen técnico de datos históricos.

Considere los siguientes patrones de datos históricos en un mercado hipotético con propiedades de Markov. Existe un 90% de posibilidades de que otra semana alcista siga a una semana definida por una tendencia alcista del mercado. Además, existe un 7,5% de posibilidades de que a la semana alcista le siga una negativa y un 2,5% de que se mantenga estático. Después de una semana bajista, existe un 80% de probabilidad de que la semana siguiente también sea bajista, y así sucesivamente.

Diagram

Description automatically generated

**Simulación de subreddit**

Hay un bot en Reddit que genera mensajes de texto significativos y aleatorios. Utiliza GTP3 y Markov Chain para generar texto y aleatorizar el texto, pero aún tiende a ser significativo.

En pocas palabras, Subreddit Simulator extrae una parte significativa de TODOS los comentarios y títulos publicados en las muchas comunidades de Reddit, luego analiza la estructura palabra por palabra de cada declaración. Usando estos datos, produce probabilidades palabra por palabra y luego utiliza esas probabilidades para crear títulos y comentarios desde cero.

Graphical user interface, text, application, email

Description automatically generated

[Fuente de imagen](https://pbs.twimg.com/media/D8UU5xNU0AAVIdj?format=jpg&name=900x900)

En el ejemplo anterior, diferentes bots de Reddit se comunican entre sí mediante la cadena GPT3 y Markov.

**Predicción de la siguiente palabra**

La cadena de Markov ayuda a construir un sistema que cuando se le da una oración incompleta, el sistema trata de predecir la siguiente palabra en la oración. Dado que cada palabra tiene un estado y predice la siguiente palabra en función del estado anterior.

Para entender eso, tomemos un ejemplo simple. Considere tres oraciones simples.

* me gusta la fisica
* Me encanta el ciclismo
* Me gustan los libros

Todas las palabras únicas de las declaraciones anteriores, a saber, 'yo', 'me gusta', 'amor', 'física', 'ciclismo' y 'libros', podrían construir los diversos estados. La distribución de probabilidad se ocupa de evaluar la posibilidad de pasar de un estado a otro, en nuestro caso de una palabra a otra. Los ejemplos anteriores muestran que la primera palabra en nuestra situación siempre comienza con la palabra 'yo'.

Como resultado, existe un 100% de probabilidad de que la primera palabra de la frase sea 'Yo'. Debemos seleccionar entre los términos 'me gusta' y 'amor' para el segundo estado. La distribución de probabilidad ahora se trata de calcular la probabilidad de que la siguiente palabra sea 'me gusta' o 'amor' si la palabra anterior es 'yo'.

En nuestro ejemplo, la palabra 'me gusta' aparece en dos de las tres frases después de 'yo', pero la palabra 'amor' aparece solo una vez. Como resultado, hay un 67 % de probabilidad de que 'me gusta' prevalezca después de 'yo' y un 33 % (1/3) de probabilidad de que 'amor' triunfe después de 'yo'. Del mismo modo, hay un 50% de probabilidad de que 'Física' y 'libros' tengan éxito 'me gusta'. Y la palabra 'amor' siempre va seguida de la palabra 'ciclismo'.

Diagram

Description automatically generated