

Facultad: Ingeniería
Escuela: Computación
Asignatura: Aprendizaje Automático y
Ciencia de Datos

Tema: Numpy

Contenido

El propósito de esta practica es aprender los fundamentos de la librería Numpy, para ser aplicado posteriormente en la solución de problemas de aprendizaje automático y ciencias de datos.

Objetivo Especifico

- a) Instalar Numpy y utilizar entornos de desarrollo IDE como Pycharm o Geany.
- b) Aprender la sintaxis de Numpy.

Material y Equipo

- a) Virtual Box
- b) Linux Mint 21.3

Introduccion Teorica

NumPy es la librería para procesamiento numérico de Python. Proporciona funcionalidades para el manejo eficiente de vectores, y es la base de otras librerías de Python.

Según la documentación oficial, NumPy es el paquete fundamental para la computación científica en Python.

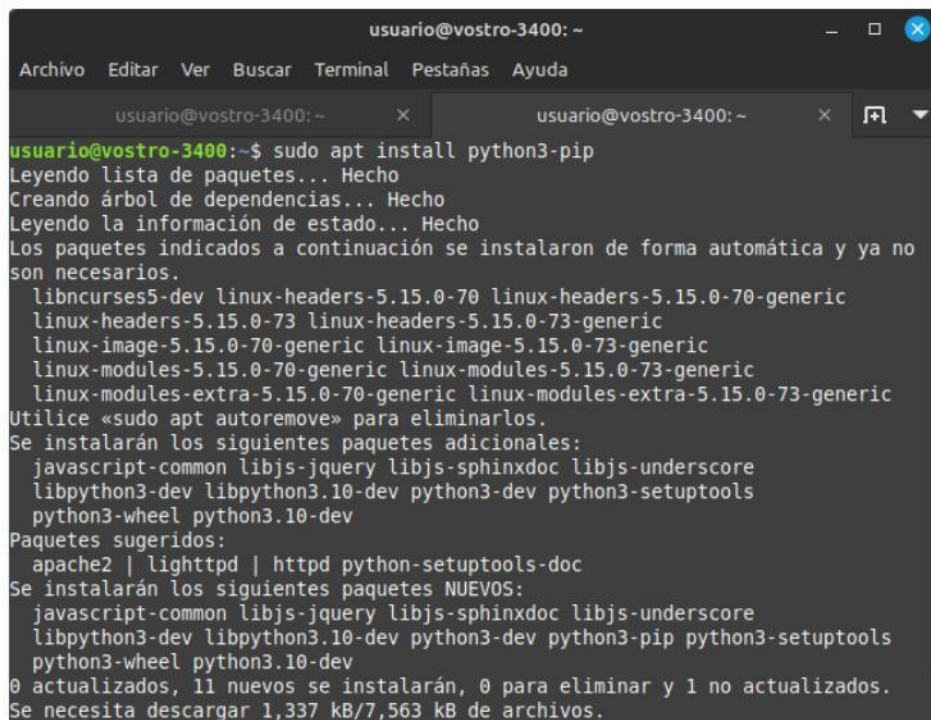
Es una librería o API de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados como matrices y matrices enmascaradas y una variedad de rutinas para operaciones rápidas en matrices, incluidas operaciones matemáticas, lógicas, manipulación de formas, ordenamiento, selección, E/S, transformadas de Fourier discretas, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

Procedimiento

Instalación de Pip

Instalar Pip, en caso de no estar instalado.

```
$ sudo apt install python3-pip
```

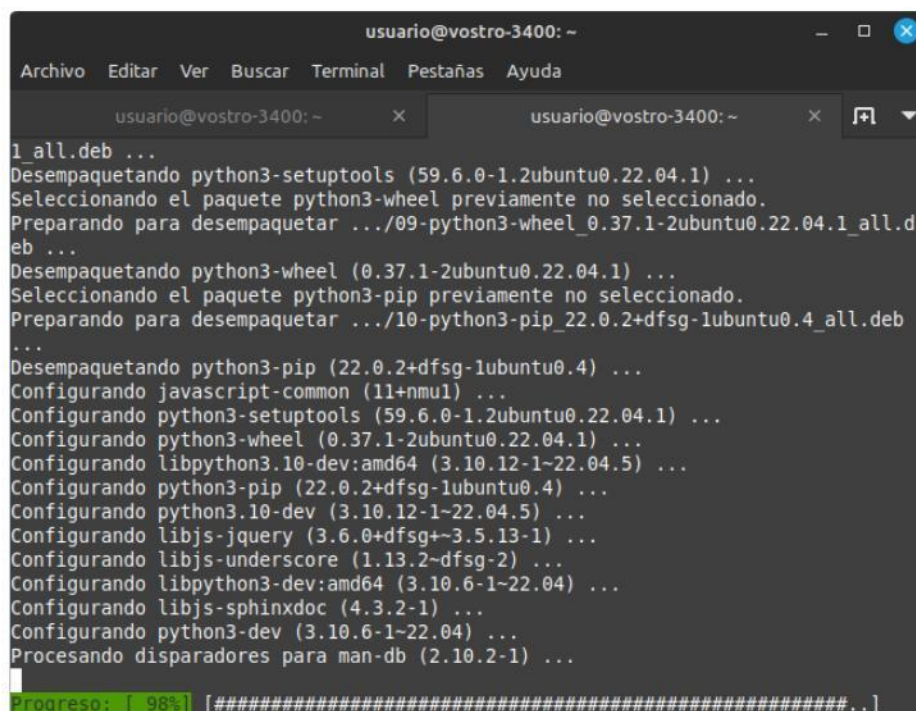


```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
usuario@vostro-3400:~$ sudo apt install python3-pip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  libncurses5-dev linux-headers-5.15.0-70 linux-headers-5.15.0-70-generic
  linux-headers-5.15.0-73 linux-headers-5.15.0-73-generic
  linux-image-5.15.0-70-generic linux-image-5.15.0-73-generic
  linux-modules-5.15.0-70-generic linux-modules-5.15.0-73-generic
  linux-modules-extra-5.15.0-70-generic linux-modules-extra-5.15.0-73-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-setuptools
  python3-wheel python3.10-dev
Paquetes sugeridos:
  apache2 | lighttpd | httpd python3-setuptools-doc
Se instalarán los siguientes paquetes NUEVOS:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-pip python3-setuptools
  python3-wheel python3.10-dev
0 actualizados, 11 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 1,337 kB/7,563 kB de archivos.

```



```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
1 all.deb ...
Desempaquetando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-wheel previamente no seleccionado.
Preparando para desempaquetar .../09-python3-wheel_0.37.1-2ubuntu0.22.04.1_all.d
eb ...
Desempaquetando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-pip previamente no seleccionado.
Preparando para desempaquetar .../10-python3-pip_22.0.2+dfsg-1ubuntu0.4_all.deb
...
Desempaquetando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando javascript-common (11+nmu1) ...
Configurando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Configurando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Configurando libpython3.10-dev:amd64 (3.10.12-1~22.04.5) ...
Configurando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando python3.10-dev (3.10.12-1~22.04.5) ...
Configurando libjs-jquery (3.6.0+dfsg+~3.5.13-1) ...
Configurando libjs-underscore (1.13.2~dfsg-2) ...
Configurando libpython3-dev:amd64 (3.10.6-1~22.04) ...
Configurando libjs-sphinxdoc (4.3.2-1) ...
Configurando python3-dev (3.10.6-1~22.04) ...
Procesando disparadores para man-db (2.10.2-1) ...
Progreso: [ 98%] [#####..]

```

Verificar que Pip esta correctamente instalado

3 ASI104 - Guía 5

\$ pip3 --version

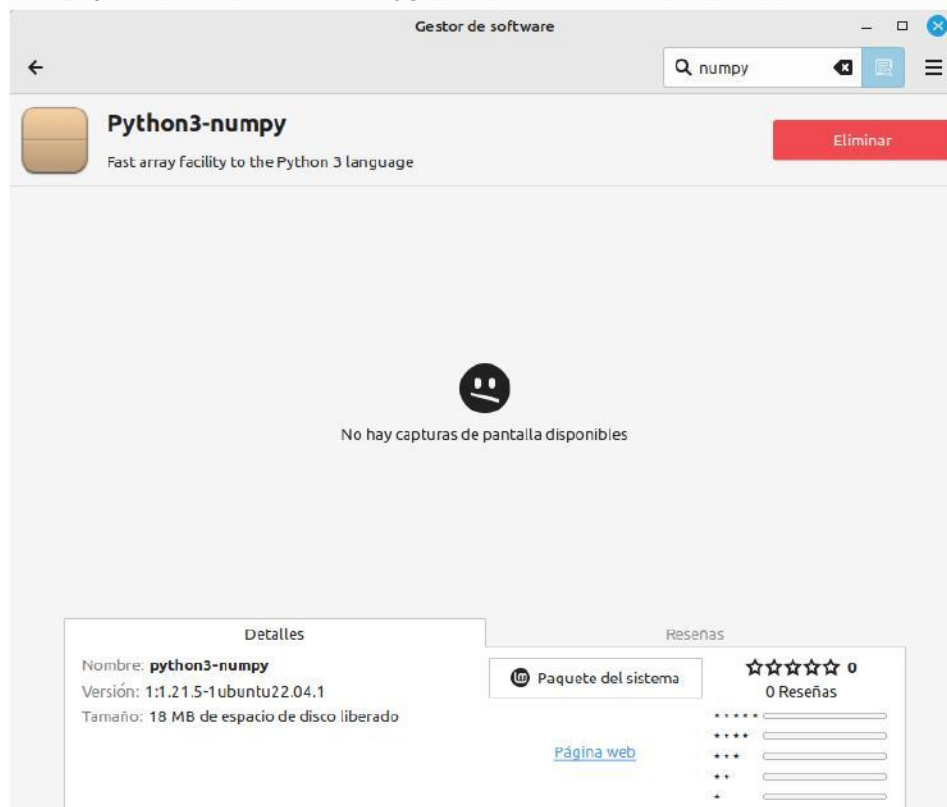
```
usuario@vostro-3400: ~  
Archivo  Editar  Ver  Buscar  Terminal  Pestañas  Ayuda  
usuario@vostro-3400: ~ x usuario@vostro-3400: ~ x [+]  
usuario@vostro-3400:~$ pip3 --version  
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)  
usuario@vostro-3400:~$
```

Instalación de Numpy

Proceder a la instalación de Numpy

```
usuario@vostro-3400: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
usuario@vostro-3400:~$ sudo pip install numpy  
Collecting numpy  
  Downloading numpy-2.0.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64  
  .whl (19.5 MB)  
  3.0/19.5 MB 1.1 MB/s eta 0:00:16
```

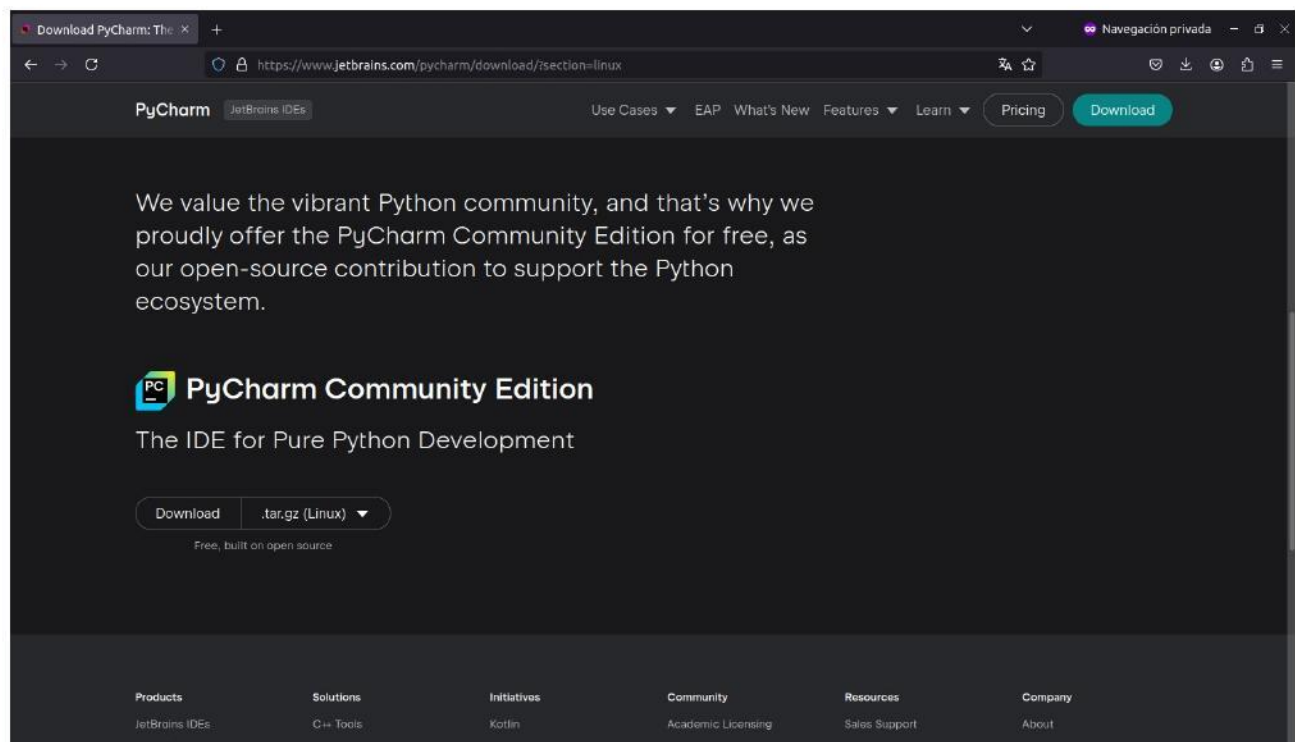
Otra opción es instalar Numpy desde el Gestor de Software



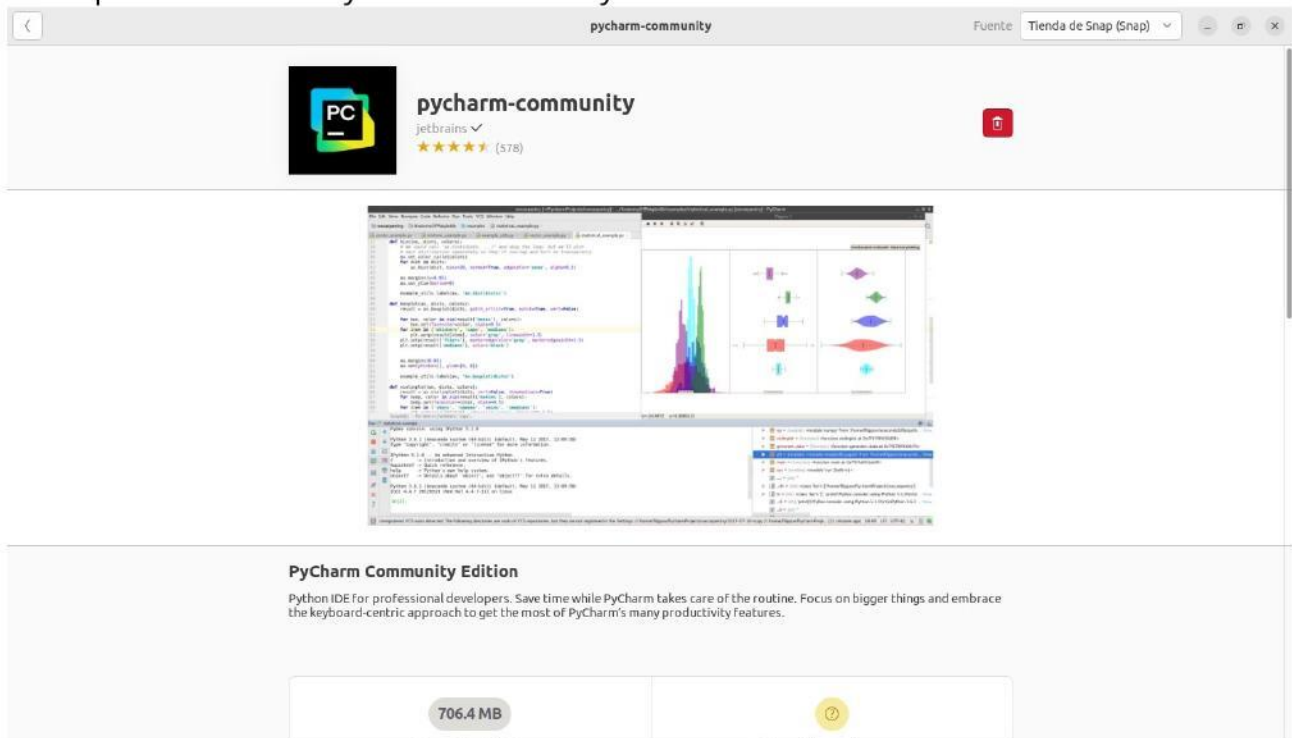
Instalación de Pycharm o Geany como IDE

Descargar Pycharm Community Edition:

<https://www.jetbrains.com/pycharm/download/?section=linux>



Otra opción es instalar Pycharm Community Edition desde el Gestor de Software de Linux:

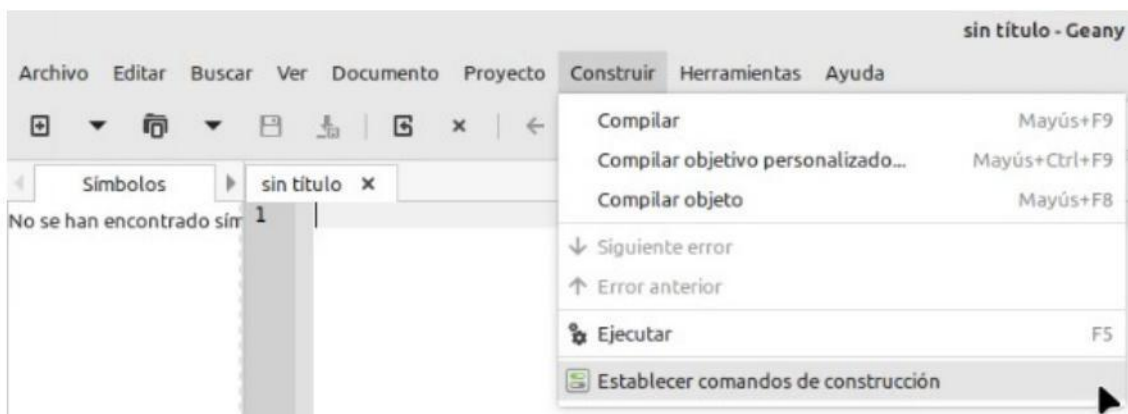


Otra opción más ligera es instalar Geany como IDE:

```
#sudo apt-get install geany
```

Instalado geany se debe configurar las opciones de compilación para que Geany pueda interpretar y ejecutar un programa en Python.

Desde el menú principal de Geany seleccionar la opción “Construir”, después seleccionar “Establecer comandos de construcción”



En la ventana de “Establecer los comandos de construcción”, asegurarse que las opciones “Comandos de Python” y “Comandos de ejecución” estén configurados con la cadena que se indica en el campo “comando”.

Establecer los comandos de construcción

#	Etiqueta	Comando	Directorio de trabajo	Reiniciar
Comandos de Python				
1.	Compile	python3 -m py_compile "%f"		
2.				
3.	Lint	pep8 --max-line-length=80 '		
Expresión regular de error:		(.+) : ([0-9]+) : ([0-9]+)		
Comandos independientes				
1.	Compilar	make		
2.	Compilar objetivo personalizado...	make		
3.	Compilar objeto	make %e.o		
4.				
Expresión regular de error:				
<i>Nota: El elemento 2 abre un diálogo y añade la respuesta al comando.</i>				
Comandos de ejecución				
1.	Execute	python3 "%F"		
2.				
<i>%d, %e, %f, %p y %l se sustituirán en los campos de comandos y directorios, consulte el manual para más información.</i>				
			Cancelar	Aceptar

Fundamentos de Numpy

En NumPy, un array (un objeto de la clase ndarray) es una secuencia multidimensional de valores del mismo tipo. Aunque se pueden especificar tipos complejos, todos los elementos de un array deben ser del mismo tipo.

Vamos a ver cuatro formas diferentes de crear un numpy array:

1. A partir de un iterable de python
2. Usando generadores de secuencias, como arange y linspace
3. Usando funciones específicas de numpy para construir estructuras con un patrón .

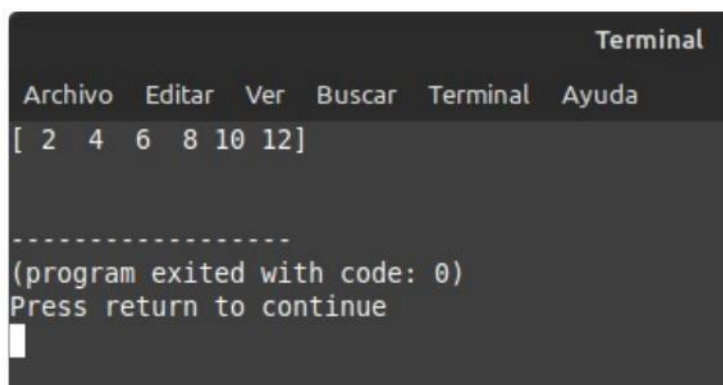
A partir de un iterable de python

Digitar el siguiente script Python en Geany, y desde Geany seleccionar en el menú principal la opción Construir → Ejecutar [F5]



```
ejemplo01.py - /home/usuario - Geany
Archivo  Editar  Buscar  Ver  Documento  Proyecto  Construir  Herramientas  Ayuda
[+] [v] [f] [g] [h] [x] [←] [→] [↺] [⚙] [⚙] [💧] [🔍] [🔍]
Simbolos ejemplo01.py x
  Variables
    v [3]
  Imports
    np [1]
1 import numpy as np
2
3 v = np.array([2, 4, 6, 8, 10, 12]) # Crea un vector de 6 elementos a partir de una lista
4 print(v)
5
```

Salida:



```
Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[ 2  4  6  8 10 12]
-----
(program exited with code: 0)
Press return to continue
```

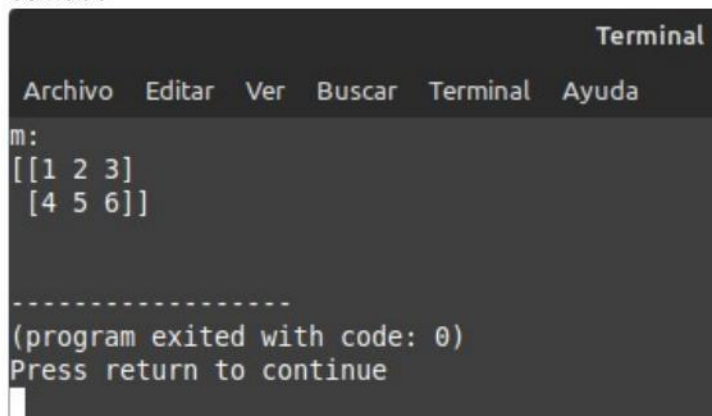
Podemos construir arrays de varias dimensiones, como una matriz:



The screenshot shows a code editor window titled 'ejemplo01.py - /home/usu'. The menu bar includes 'Archivo', 'Editar', 'Buscar', 'Ver', 'Documento', 'Proyecto', 'Construir', 'Herramientas', and 'Ayuda'. The left sidebar shows a 'Símbolos' (Symbols) pane with a tree view containing 'Variables' (with 'm [3]' listed) and 'Imports' (with 'np [1]' listed). The main editor area contains the following Python code:

```
1 import numpy as np
2
3 m = np.array([[1,2,3], [4,5,6]]) # Crea una matriz bidimensional
4 print("m:")
5 print(m)
6
```

Salida:



The screenshot shows a terminal window titled 'Terminal'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal output is as follows:

```
m:
[[1 2 3]
 [4 5 6]]

-----
(program exited with code: 0)
Press return to continue
```


Usando generadores de secuencias, como arange y linspace

arange. Es equivalente a la función built-in de python, salvo que permite números flotantes.

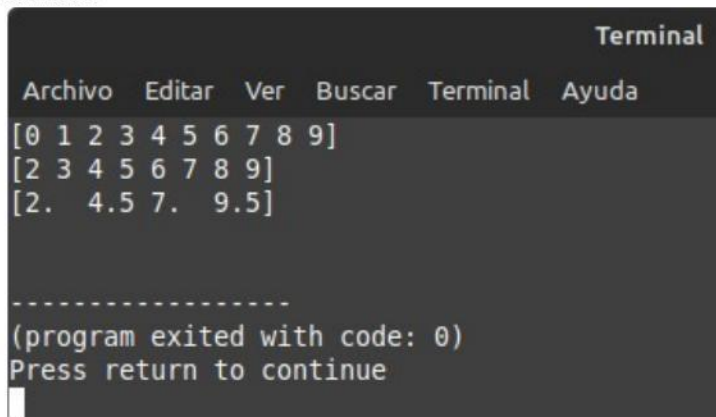


The screenshot shows the Geany IDE with a file named 'ejemplo01.py'. The code in the editor is as follows:

```
1 import numpy as np
2
3 x = np.arange(10) # Crea un array con valores del 0 al 9
4 print(x)
5
6 x = np.arange(2, 10) # Crea un array con valores del 2 al 9
7 print(x)
8
9 x = np.arange(2, 10, 2.5) # Crea un array con valores del 2 al 9 separados por intervalos de 2.5
10 print(x)
11
12
```

The left sidebar shows a 'Símbolos' (Symbols) panel with a tree structure: Variables (x [3], x [6], x [9]), Imports (np [1]).

Salida:



The screenshot shows a terminal window titled 'Terminal' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The output of the script is displayed as follows:

```
[0 1 2 3 4 5 6 7 8 9]
[2 3 4 5 6 7 8 9]
[2. 4.5 7. 9.5]

-----
(program exited with code: 0)
Press return to continue
```

linspace. La función `np.linspace()` es muy parecida a la anterior, pero en lugar de especificar la distancia entre valores, permite especificar el número de valores dentro del intervalo (por defecto, ambos extremos inclusive).

```
ejemplo01.py - /home/usuario - Geany
ar Ver Documento Proyecto Construir Herramientas Ayuda
ejemplo01.py x
1 import numpy as np
2
3 x = np.linspace(10, 20, 5) # Crea un vector con 5 valores igualmente espaciados que van del 10 al 20
4 print(x)
5
6 x = np.linspace(10, 20, 5, endpoint=False) # Se puede excluir el punto final.
7 print(x)
8
9 x = np.linspace(10, 20, 5, retstep=True) # Fijando retstep a True puede devolver también el tamaño del intervalo
10 print(x)
11
12
13
```

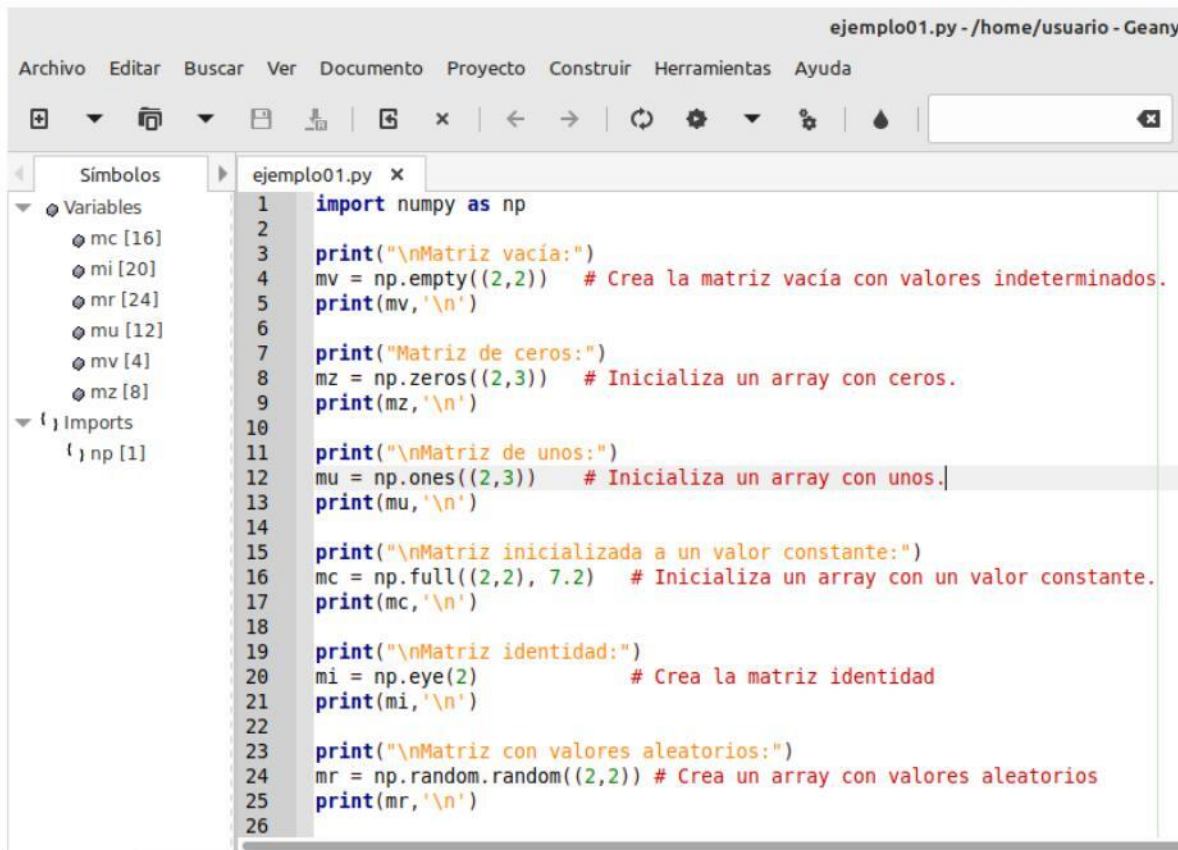
Salida:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[10.  12.5 15.  17.5 20. ]
[10. 12. 14. 16. 18.]
(array([10. , 12.5, 15. , 17.5, 20. ]), 2.5)

-----
(program exited with code: 0)
Press return to continue
```

Funciones específicas de numpy

NumPy proporciona funciones para llevar a cabo distintas inicializaciones de un array n-dimensional sin necesidad de especificar los elementos. En todos los casos, el primer argumento es una tupla, indicando la dimensión.



The screenshot shows a code editor window titled 'ejemplo01.py - /home/usuario - Geany'. The editor contains a Python script that demonstrates various NumPy array initialization functions. The script is as follows:

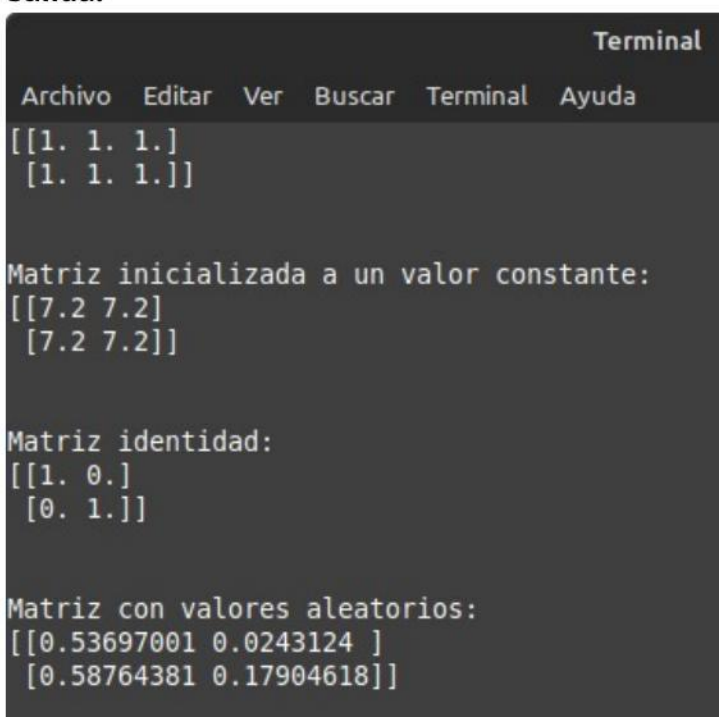
```

1 import numpy as np
2
3 print("\nMatriz vacía:")
4 mv = np.empty((2,2)) # Crea la matriz vacía con valores indeterminados.
5 print(mv, '\n')
6
7 print("Matriz de ceros:")
8 mz = np.zeros((2,3)) # Inicializa un array con ceros.
9 print(mz, '\n')
10
11 print("\nMatriz de unos:")
12 mu = np.ones((2,3)) # Inicializa un array con unos.
13 print(mu, '\n')
14
15 print("\nMatriz inicializada a un valor constante:")
16 mc = np.full((2,2), 7.2) # Inicializa un array con un valor constante.
17 print(mc, '\n')
18
19 print("\nMatriz identidad:")
20 mi = np.eye(2) # Crea la matriz identidad
21 print(mi, '\n')
22
23 print("\nMatriz con valores aleatorios:")
24 mr = np.random.random((2,2)) # Crea un array con valores aleatorios
25 print(mr, '\n')
26

```

The left sidebar shows a 'Simbolos' (Symbols) panel with a tree view of variables and imports. The 'Imports' section shows 'np' with a count of 1.

Salida:



The screenshot shows a terminal window titled 'Terminal' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The terminal displays the output of the Python script, showing the initialization of various NumPy arrays:

```

[[1. 1. 1.]
 [1. 1. 1.]]

Matriz inicializada a un valor constante:
[[7.2 7.2]
 [7.2 7.2]]

Matriz identidad:
[[1. 0.]
 [0. 1.]]

Matriz con valores aleatorios:
[[0.53697001 0.0243124 ]
 [0.58764381 0.17904618]]

```


Propiedades de los arrays

Los numpy arrays tienen propiedades muy utilizadas. Entre las más destacadas, exponemos las tres principales:

La propiedad `ndim` contiene el número de dimensiones del array.

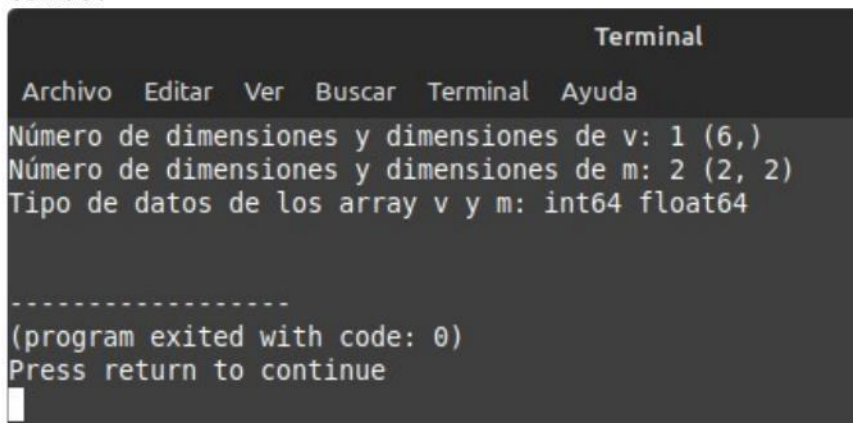
La propiedad denominada `shape` (una tupla) contiene el tamaño del array en cada dimensión.

La propiedad `dtype` indica el tipo de datos que contiene el array.



```
ejemplo01.py - /home/usuario - Geany
ar Ver Documento Proyecto Construir Herramientas Ayuda
ejemplo01.py x
1 import numpy as np
2
3 v = np.array([2, 4, 6, 8, 10, 12])
4 m = np.random.random((2,2))
5
6 print("Número de dimensiones y dimensiones de v:",v.ndim, v.shape) # Array de seis elementos
7 print("Número de dimensiones y dimensiones de m:",m.ndim, m.shape) # Matriz de 2x3
8 print("Tipo de datos de los array v y m:", v.dtype, m.dtype) # Matriz de 2x3
9
10
11
12
13
```

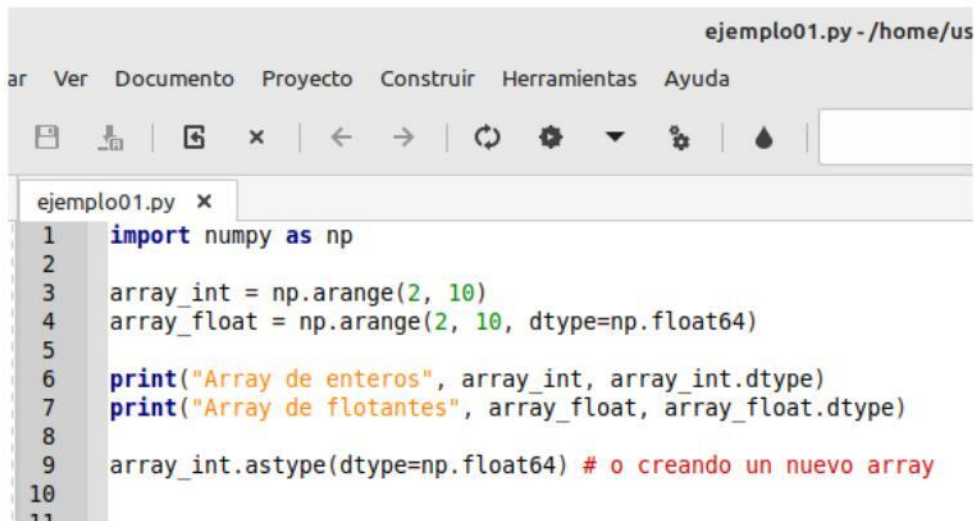
Salida:



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
Número de dimensiones y dimensiones de v: 1 (6,)
Número de dimensiones y dimensiones de m: 2 (2, 2)
Tipo de datos de los array v y m: int64 float64
-----
(program exited with code: 0)
Press return to continue
```

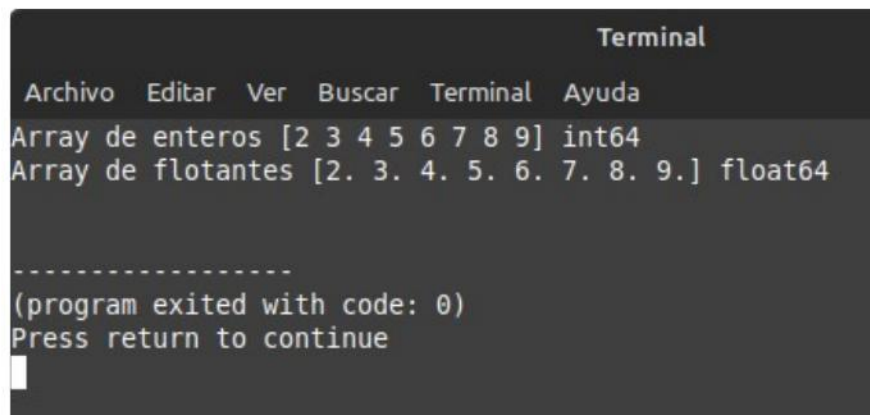
13 ASI104 - Guía 5

Las funciones constructoras tienen generalmente un argumento llamado `dtype` que permite indicar el tipo asociado a los datos. También disponemos de la función `astype` para construir un nuevo array a partir de otro especificando un tipo de datos diferente, para mayor detalla consultar la documentación oficial de Numpy.



```
ejemplo01.py - /home/us
ar Ver Documento Proyecto Construir Herramientas Ayuda
ejemplo01.py x
1 import numpy as np
2
3 array_int = np.arange(2, 10)
4 array_float = np.arange(2, 10, dtype=np.float64)
5
6 print("Array de enteros", array_int, array_int.dtype)
7 print("Array de flotantes", array_float, array_float.dtype)
8
9 array_int.astype(dtype=np.float64) # o creando un nuevo array
10
11
```

Salida:

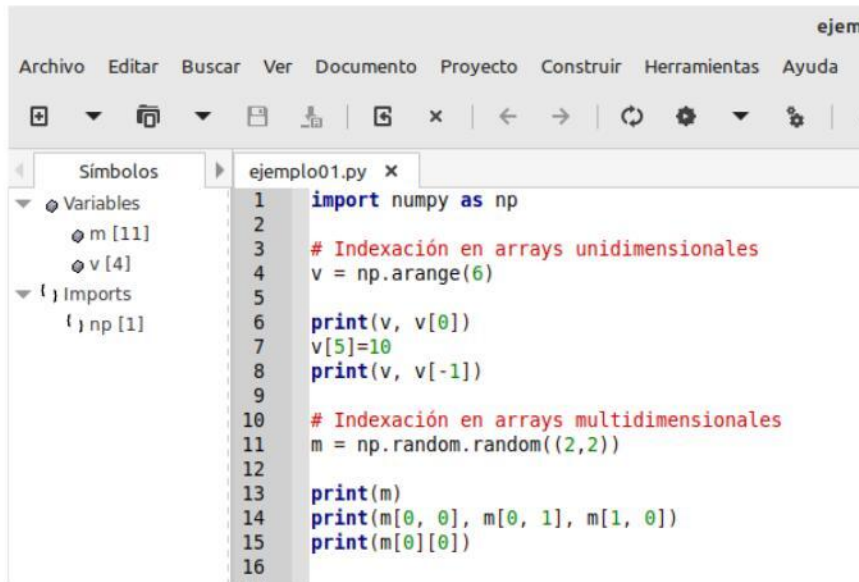


```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
Array de enteros [2 3 4 5 6 7 8 9] int64
Array de flotantes [2. 3. 4. 5. 6. 7. 8. 9.] float64

-----
(program exited with code: 0)
Press return to continue
```


Indexación de arrays

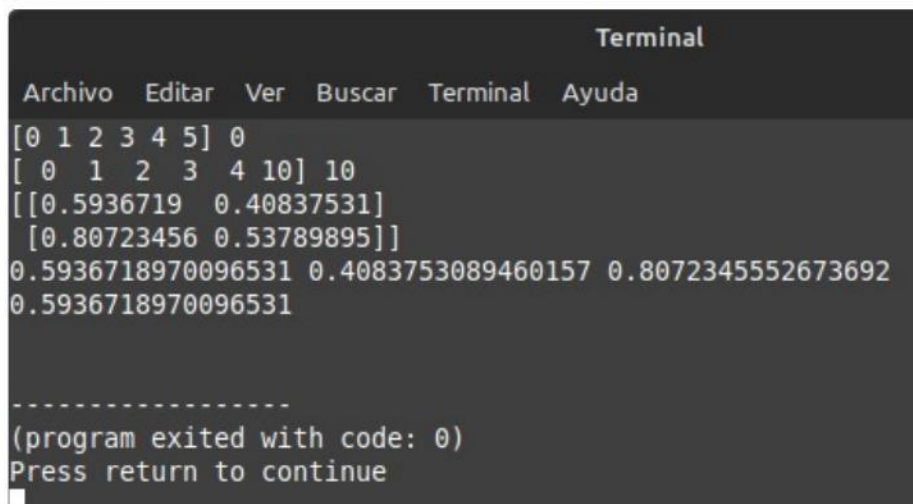
Al igual que en las secuencias de Python los elementos pueden ser accedidos mediante índices especificados entre corchetes. También se puede especificar un subconjunto de índices mediante el slicing.



The screenshot shows an IDE window titled 'ejem' with a menu bar (Archivo, Editar, Buscar, Ver, Documento, Proyecto, Construir, Herramientas, Ayuda) and a toolbar. The left sidebar shows a 'Símbolos' (Symbols) pane with a tree view containing 'Variables' (m [11], v [4]) and 'Imports' (np [1]). The main editor displays a Python script named 'ejemplo01.py' with the following code:

```
1 import numpy as np
2
3 # Indexación en arrays unidimensionales
4 v = np.arange(6)
5
6 print(v, v[0])
7 v[5]=10
8 print(v, v[-1])
9
10 # Indexación en arrays multidimensionales
11 m = np.random.random((2,2))
12
13 print(m)
14 print(m[0, 0], m[0, 1], m[1, 0])
15 print(m[0][0])
16
```

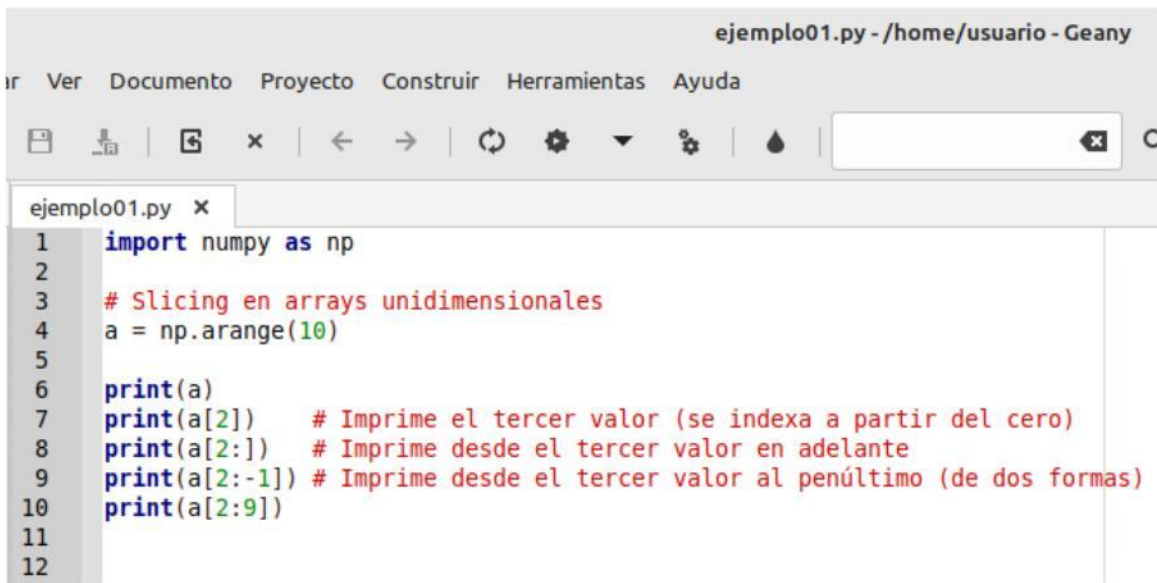
Salida:



The screenshot shows a terminal window titled 'Terminal' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The terminal displays the output of the Python script:

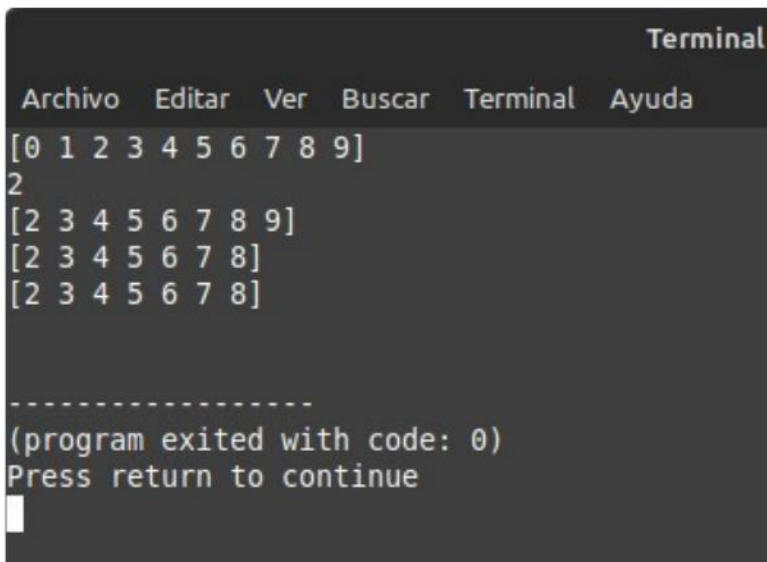
```
[0 1 2 3 4 5] 0
[ 0 1 2 3 4 10] 10
[[0.5936719 0.40837531]
 [0.80723456 0.53789895]]
0.5936718970096531 0.4083753089460157 0.8072345552673692
0.5936718970096531

-----
(program exited with code: 0)
Press return to continue
```



```
ejemplo01.py - /home/usuario - Geany
Ver Documento Proyecto Construir Herramientas Ayuda
ejemplo01.py x
1 import numpy as np
2
3 # Slicing en arrays unidimensionales
4 a = np.arange(10)
5
6 print(a)
7 print(a[2]) # Imprime el tercer valor (se indexa a partir del cero)
8 print(a[2:]) # Imprime desde el tercer valor en adelante
9 print(a[2:-1]) # Imprime desde el tercer valor al penúltimo (de dos formas)
10 print(a[2:9])
11
12
```

Salida:



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[0 1 2 3 4 5 6 7 8 9]
2
[2 3 4 5 6 7 8 9]
[2 3 4 5 6 7 8]
[2 3 4 5 6 7 8]

-----
(program exited with code: 0)
Press return to continue
```

```
ejemplo01.py - /home/usuario - Geany
Ar Ver Documento Proyecto Construir Herramientas Ayuda
[Icons] [Search] [Close]

ejemplo01.py x
1 import numpy as np
2
3 # Slicing en arrays multidimensionales
4 a = np.arange(20) # Crea un array con 20 elementos (4 * 5)
5 m = a.reshape(4,5) # La función reshape permite modificar el shape de un array,
6 # siempre y cuando el número de elementos lo permita
7 print(m)
8 print()
9 print(m[1,2], m[2,4]) # Imprime las posiciones (1,2) y (2,4)
10
11 print("\nImprime la segunda fila de dos modos distintos")
12 print(m[1])
13 print(m[1,:])
14
15 print("\nImprime las dos primeras filas")
16 print(m[:2])
17
18 print("\nImprime desde la primera a la tercera columna")
19 print(m[:, 1:4])
20
21 print("\nImprime desde la segunda fila en adelante, y de la primera a la tercera columna")
22 print(m[1:, 1:4])
23
```

Salida:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
Imprime la segunda fila de dos modos distintos
[5 6 7 8 9]
[5 6 7 8 9]

Imprime las dos primeras filas
[[0 1 2 3 4]
 [5 6 7 8 9]]

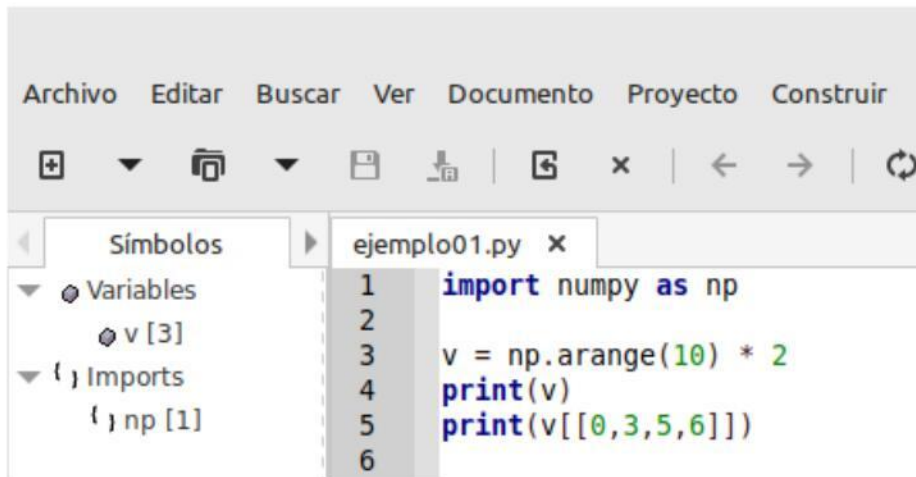
Imprime desde la primera a la tercera columna
[[ 1  2  3]
 [ 6  7  8]
 [11 12 13]
 [16 17 18]]

Imprime desde la segunda fila en adelante, y de la
[[ 6  7  8]
 [11 12 13]
 [16 17 18]]

-----
(program exited with code: 0)
Press return to continue
```

Indexación mediante arrays de enteros

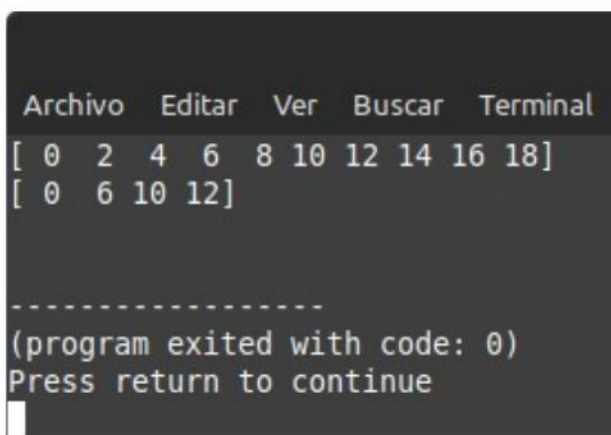
La indexación mediante slices permite acceder a los elementos del array siguiendo un patrón. En el caso de que este patrón no exista, es decir, queramos acceder a una serie de índices arbitrario, se puede conseguir usando un array o lista con los índices que nos interese seleccionar.



The screenshot shows an IDE window titled 'ejemplo01.py'. The menu bar includes 'Archivo', 'Editar', 'Buscar', 'Ver', 'Documento', 'Proyecto', and 'Construir'. The toolbar contains icons for file operations and navigation. On the left, a 'Símbolos' (Symbols) pane shows a tree structure with 'Variables' containing 'v [3]' and 'Imports' containing 'np [1]'. The main editor area contains the following Python code:

```
1 import numpy as np
2
3 v = np.arange(10) * 2
4 print(v)
5 print(v[[0,3,5,6]])
6
```

Salida:

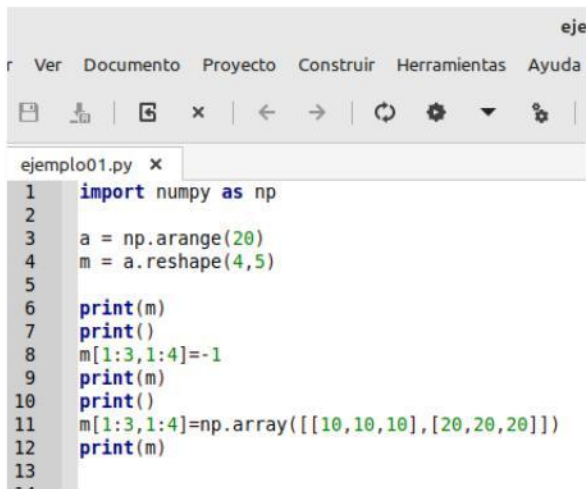


The screenshot shows a terminal window with a menu bar containing 'Archivo', 'Editar', 'Ver', 'Buscar', and 'Terminal'. The terminal output is as follows:

```
[ 0  2  4  6  8 10 12 14 16 18]
[ 0  6 10 12]

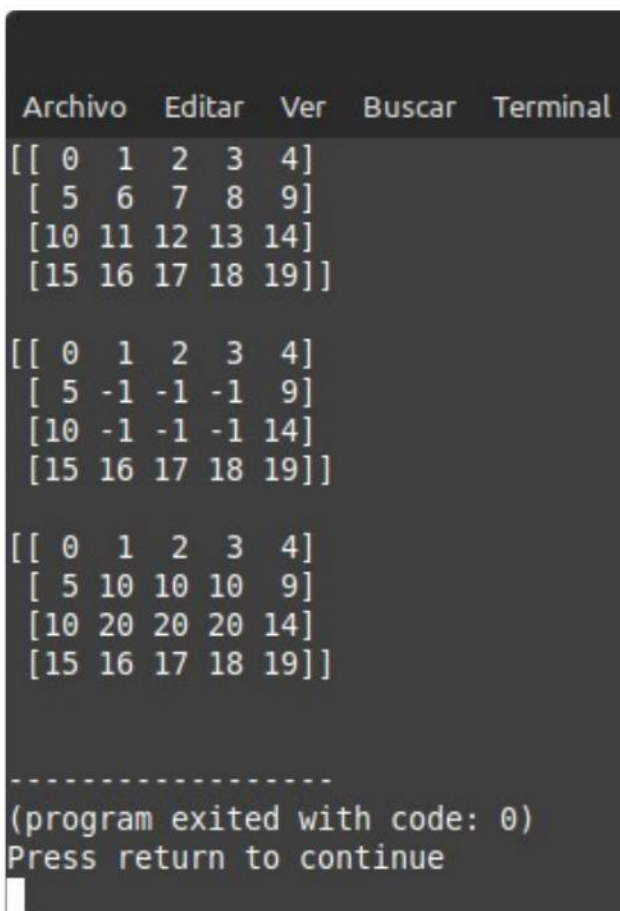
-----
(program exited with code: 0)
Press return to continue
```

La indexación permite también escribir en varias posiciones del array a la vez.



```
1 import numpy as np
2
3 a = np.arange(20)
4 m = a.reshape(4,5)
5
6 print(m)
7 print()
8 m[1:3,1:4]=-1
9 print(m)
10 print()
11 m[1:3,1:4]=np.array([[10,10,10],[20,20,20]])
12 print(m)
13
--
```

Salida:



```
Archivo  Editar  Ver  Buscar  Terminal

[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]

[[ 0  1  2  3  4]
 [ 5 -1 -1 -1  9]
 [10 -1 -1 -1 14]
 [15 16 17 18 19]]

[[ 0  1  2  3  4]
 [ 5 10 10 10  9]
 [10 20 20 20 14]
 [15 16 17 18 19]]

-----
(program exited with code: 0)
Press return to continue
```


Indexación mediante arrays de booleanos

Al igual que con arrays de enteros permite acceder a elementos de un array arbitrariamente. Sin embargo, en este caso el array funciona como una máscara, por lo que el tamaño del array de booleanos debe coincidir con el del array al que se está indexando. Se suele usar para seleccionar elementos que satisfacen alguna condición.

En el caso de usarse sobre arrays multidimensionales, éstos se aplanan (transforman a array unidimensional).

```

ejemplo01.py - /home/usuario - Geany
Ver Documento Proyecto Construir Herramientas Ayuda

ejemplo01.py x
1 import numpy as np
2
3 v = np.array([0,1,2,3,4,5])
4 b = np.array([True, False, False, False, False, True])
5 print(v)
6 v2 = v[b]; # v2 es un array que contiene el primer y último valor de v.
7 print(v2) # Ya que son los dos que se pasan con valor a True.
8 print()
9
10 a = np.arange(20) # Crea una matriz de tamaño 4*5
11 m = a.reshape(4,5)
12 print(m)
13 print()
14
15 print(m % 2==0) # Imprime un array bidimensional de valores booleanos.
16 print() # True si el valor de m es par y False si es impar
17
18
19 m_par = m[m % 2 == 0] # Crea un vector con los elementos pares de m.
20 print(m_par)
21

```

Salida:

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda

[0 1 2 3 4 5]
[0 5]

[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]

[[ True False  True False  True]
 [False  True False  True False]
 [ True False  True False  True]
 [False  True False  True False]]

[ 0  2  4  6  8 10 12 14 16 18]

-----
(program exited with code: 0)
Press return to continue

```

Vectorización

Una de las principales ventajas que aporta Numpy es que muchas funciones se implementan internamente de forma vectorizada, lo que supone un aumento de la eficiencia muy importante (varios órdenes de magnitud) con respecto a las operaciones secuenciales.

```

ejemplo01.py
Archivo  Editar  Buscar  Ver  Documento  Proyecto  Construir  Herramientas  Ayuda
+  ▾  📁  ▾  💾  📄  |  📄  ✕  |  ⬅  ➡  |  🔄  ⚙  ▾  ⚙  |  💧  |
◀  Símbolos  ▶  ejemplo01.py  ✕
  Variables
  ▾ final [10]
  ▾ final [15]
  ▾ gran_vector [5]
  ▾ inicio [8]
  ▾ inicio [13]
  ▾ suma [9]
  ▾ suma [14]
  Imports
  { } np [1]
  { } time [2]
1  import numpy as np
2  import time
3
4  # Crea un vector de 100000 números aleatorios.
5  gran_vector = np.random.randint(0,1000,1000000)
6  print(gran_vector.shape)
7
8  inicio = time.time()
9  suma = sum(gran_vector)
10 final = time.time()
11 print("Tiempo de ejecución fue de: ", inicio - final)
12
13 inicio = time.time()
14 suma = np.sum(gran_vector)
15 final = time.time()
16 print("Tiempo de ejecución fue de: ", inicio - final)
17
18

```

Salida:

```

Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
(1000000,)
Tiempo de ejecución fue de: -0.0452275276184082
Tiempo de ejecución fue de: -0.0006010532379150391

-----
(program exited with code: 0)
Press return to continue

```

Operaciones unarias

Algunas de las funciones más utilizadas son:

max(), min(): Elemento máximo o mínimo.

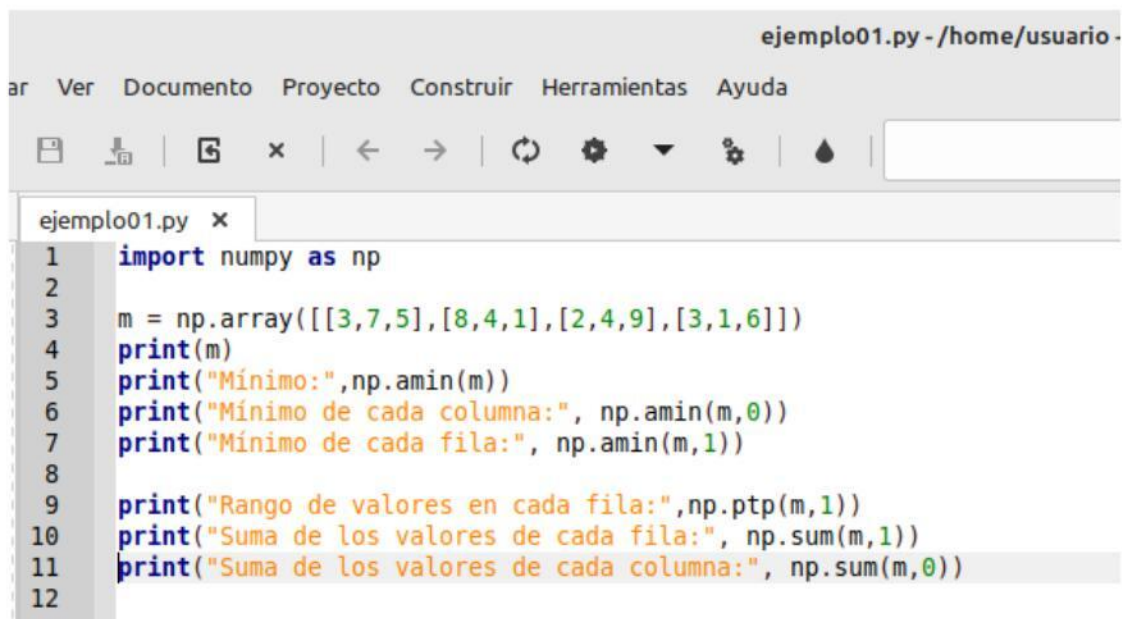
argmin(), argmax(): Índice del elemento máximo o mínimo.

sum(): Suma de los elementos del array.

cumsum(): Suma acumulada de los elementos del array.

prod(): Producto de los elementos del array.

cumprod(): Producto acumulado de los elementos del array.

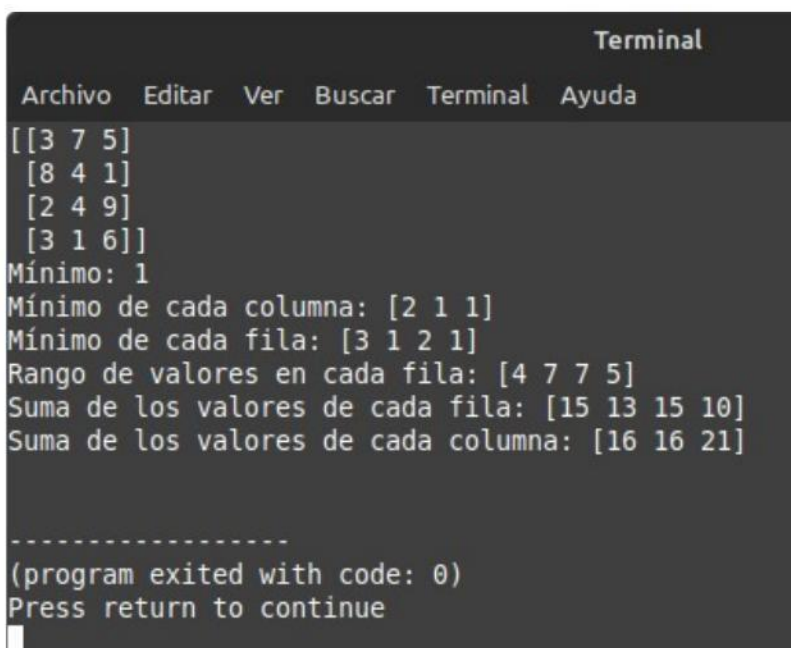


```

ejemplo01.py - /home/usuario
ar Ver Documento Proyecto Construir Herramientas Ayuda
ejemplo01.py x
1 import numpy as np
2
3 m = np.array([[3,7,5],[8,4,1],[2,4,9],[3,1,6]])
4 print(m)
5 print("Mínimo:", np.amin(m))
6 print("Mínimo de cada columna:", np.amin(m,0))
7 print("Mínimo de cada fila:", np.amin(m,1))
8
9 print("Rango de valores en cada fila:", np.ptp(m,1))
10 print("Suma de los valores de cada fila:", np.sum(m,1))
11 print("Suma de los valores de cada columna:", np.sum(m,0))
12

```

Salida:



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[[3 7 5]
 [8 4 1]
 [2 4 9]
 [3 1 6]]
Mínimo: 1
Mínimo de cada columna: [2 1 1]
Mínimo de cada fila: [3 1 2 1]
Rango de valores en cada fila: [4 7 7 5]
Suma de los valores de cada fila: [15 13 15 10]
Suma de los valores de cada columna: [16 16 21]

-----
(program exited with code: 0)
Press return to continue

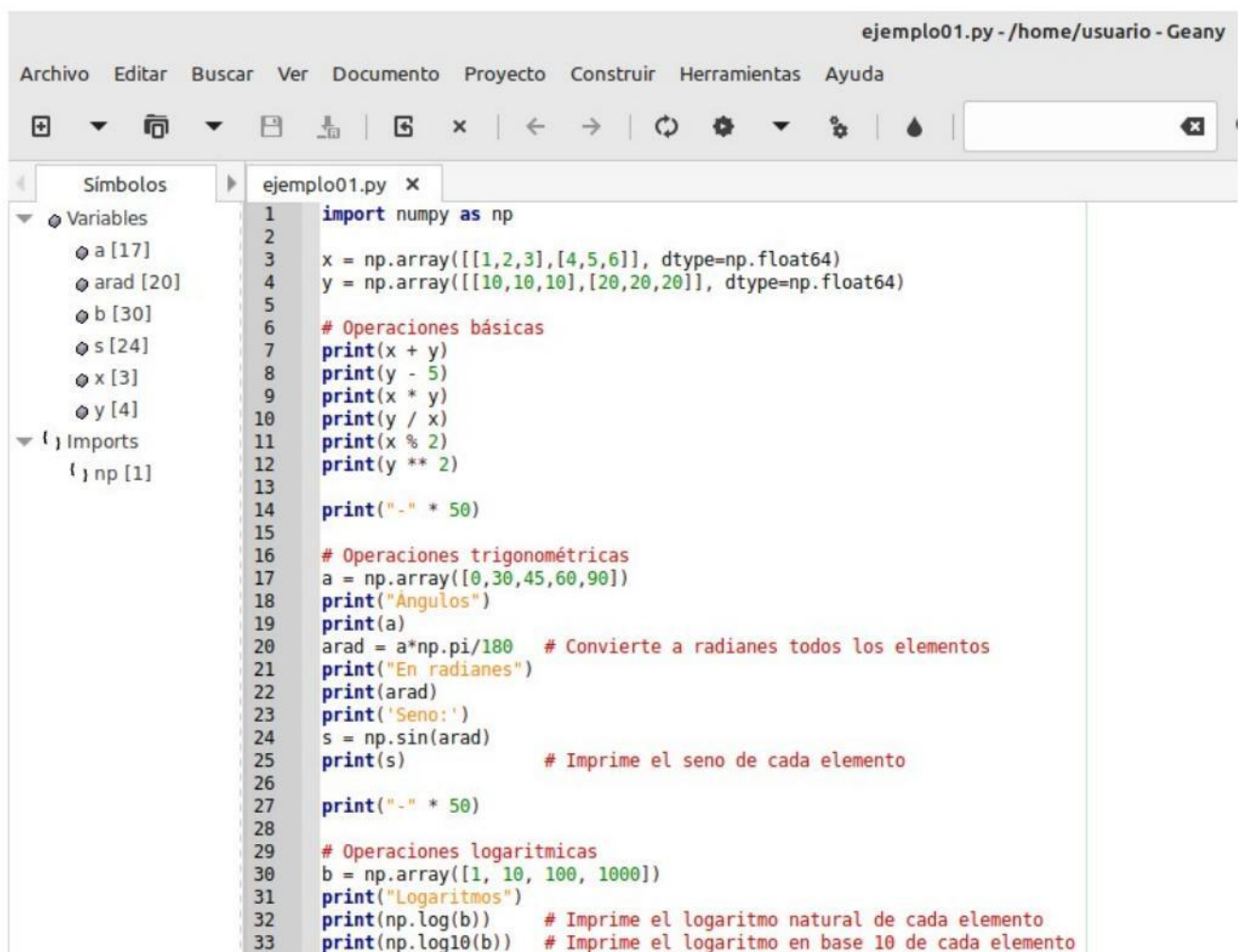
```


Operaciones entre arrays

NumPy implementa numerosas funciones entre arrays. Algunas de ellas, denominadas universales (ufunc), se aplican de manera eficiente (son implementaciones vectorizadas) elemento por elemento, y soportan algunas características como broadcasting (que es un mecanismo que permite más flexibilidad, y se verá más adelante). También se pueden aplicar entre cada elemento de un array y un escalar.

A continuación se describen algunas de las funciones universales de uso más frecuente. El resto de funciones disponibles (también las no universales) pueden consultarse en la referencia de Numpy.

Nota: también permite la operación de arrays con escalares, como multiplicar un array por una constante.



```

ejemplo01.py - /home/usuario - Geany
Archivo  Editar  Buscar  Ver  Documento  Proyecto  Construir  Herramientas  Ayuda

1  import numpy as np
2
3  x = np.array([[1,2,3],[4,5,6]], dtype=np.float64)
4  y = np.array([[10,10,10],[20,20,20]], dtype=np.float64)
5
6  # Operaciones básicas
7  print(x + y)
8  print(y - 5)
9  print(x * y)
10 print(y / x)
11 print(x % 2)
12 print(y ** 2)
13
14 print("-" * 50)
15
16 # Operaciones trigonométricas
17 a = np.array([0,30,45,60,90])
18 print("Ángulos")
19 print(a)
20 arad = a*np.pi/180 # Convierte a radianes todos los elementos
21 print("En radianes")
22 print(arad)
23 print('Seno:')
24 s = np.sin(arad)
25 print(s)          # Imprime el seno de cada elemento
26
27 print("-" * 50)
28
29 # Operaciones logarítmicas
30 b = np.array([1, 10, 100, 1000])
31 print("Logaritmos")
32 print(np.log(b))   # Imprime el logaritmo natural de cada elemento
33 print(np.log10(b)) # Imprime el logaritmo en base 10 de cada elemento
  
```

Salida:

```
Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[[11. 12. 13.]
 [24. 25. 26.]]
[[ 5.  5.  5.]
 [15. 15. 15.]]
[[ 10.  20.  30.]
 [ 80. 100. 120.]]
[[10.      5.      3.33333333]
 [ 5.      4.      3.33333333]]
[[1. 0. 1.]
 [0. 1. 0.]]
[[100. 100. 100.]
 [400. 400. 400.]]
-----
Ángulos
[ 0 30 45 60 90]
En radianes
[0.      0.52359878 0.78539816 1.04719755 1.57079633]
Seno:
[0.      0.5      0.70710678 0.8660254  1.      ]
-----
Logaritmos
[0.      2.30258509 4.60517019 6.90775528]
[0. 1. 2. 3.]
-----
(program exited with code: 0)
Press return to continue
```


Operaciones entre matrices

NumPy implementa funciones específicas muy utilizadas en álgebra lineal (para vectores y matrices). Las de uso más común son el producto vectorial y matricial, la inversa de matrices y la transposición de matrices.

Producto vectorial

La función `np.dot()` implementa el producto vectorial entre dos vectores o dos matrices. En el caso de dos vectores, el producto vectorial se calcula como:

$$[u_1, \dots, u_n] \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = u_1 \cdot v_1 + \dots + u_n \cdot v_n$$

$$[1 \quad 2 \quad 3] \cdot \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix} = 1 \cdot 10 + 2 \cdot 20 + 3 \cdot 30 = 140$$

```
1 import numpy as np
2
3 u = np.array([1,2,3])
4 v = np.array([10,20,30])
5
6 print(np.dot(u,v)) # Se puede llamar la función de las dos maneras.
7 print(u.dot(v))
8
```

Salida:

```

Terminal
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
140
140
-----
(program exited with code: 0)
Press return to continue

```

Producto matricial

El producto vectorial de una matriz de tamaño $(m \times n)$ y otra de tamaño $(n \times o)$, es una nueva matriz, de tamaño $(m \times o)$, en la que el valor de la posición (i,j) es obtenido como el producto vectorial de la fila i de la primera matriz, y la columna j de la segunda matriz.

$$\begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1o} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{no} \end{bmatrix} = \begin{bmatrix} u_{11} \cdot v_{11} + \cdots + u_{1n} \cdot v_{n1} & \cdots & u_{11} \cdot v_{1o} + \cdots + u_{1n} \cdot v_{no} \\ \vdots & \ddots & \vdots \\ u_{m1} \cdot v_{11} + \cdots + u_{mn} \cdot v_{n1} & \cdots & u_{m1} \cdot v_{1o} + \cdots + u_{mn} \cdot v_{no} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 10 & 100 & 200 \\ 2 & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 18 & 112 & 216 \\ 11 & 46 & 324 & 632 \\ 17 & 74 & 536 & 1048 \end{bmatrix}$$

Si el número de columnas de la primera matriz es distinto del número de filas de la segunda, las matrices no se pueden multiplicar.

The screenshot shows a code editor with a menu bar (Archivo, Editar, Buscar, Ver, Documento, Proyecto, Construir, Herramientas) and a toolbar. On the left, a 'Símbolos' (Symbols) pane shows a project tree with 'Variables' (u [3], v [4]) and 'Imports' (np [1]). The main editor window, titled 'ejemplo01.py', contains the following code:

```

1 import numpy as np
2
3 u = np.array([[1,2],[3,4],[5,6]])
4 v = np.array([[1,10,100,200],[2,4,6,8]])
5
6 print(u @ v)
7

```

Salida:

The screenshot shows a terminal window titled 'Terminal' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The output of the program is displayed as follows:

```

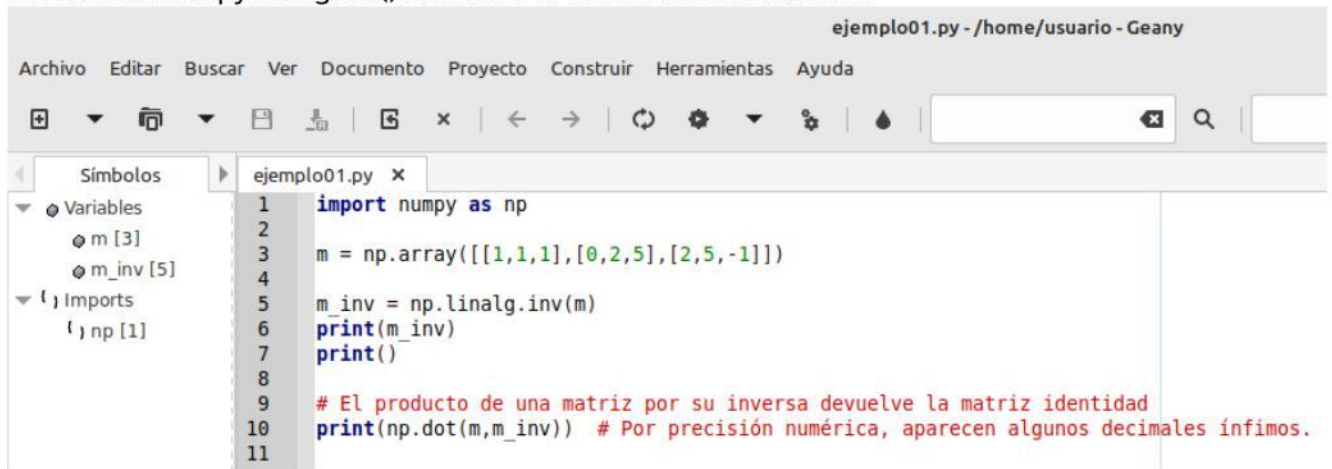
[[ 5  18 112 216]
 [ 11 46 324 632]
 [ 17 74 536 1048]]

-----
(program exited with code: 0)
Press return to continue

```

Inversa de una matriz

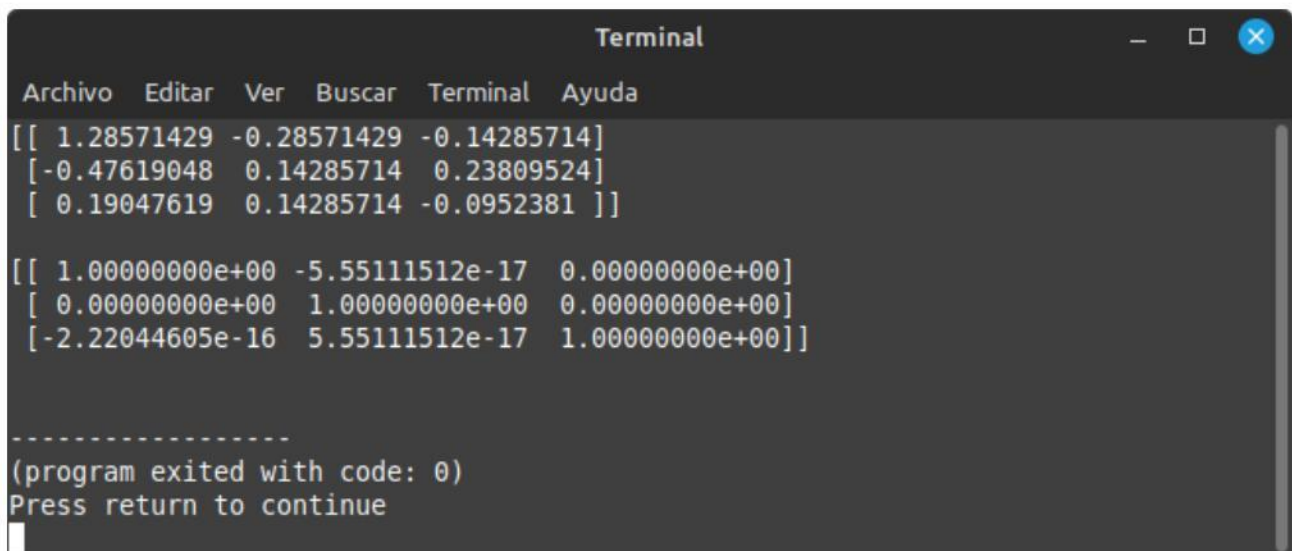
La función `numpy.linalg.inv()` devuelve la inversa de una matriz.



The screenshot shows a Geany IDE window titled "ejemplo01.py - /home/usuario - Geany". The menu bar includes Archivo, Editar, Buscar, Ver, Documento, Proyecto, Construir, Herramientas, and Ayuda. The left sidebar shows a "Símbolos" (Symbols) pane with a tree view containing "Variables" (m [3], m_inv [5]) and "Imports" (np [1]). The main editor displays the following Python code:

```
1 import numpy as np
2
3 m = np.array([[1,1,1],[0,2,5],[2,5,-1]])
4
5 m_inv = np.linalg.inv(m)
6 print(m_inv)
7 print()
8
9 # El producto de una matriz por su inversa devuelve la matriz identidad
10 print(np.dot(m,m_inv)) # Por precisión numérica, aparecen algunos decimales ínfimos.
11
```

Salida:



The screenshot shows a terminal window titled "Terminal" with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The output of the script is displayed as follows:

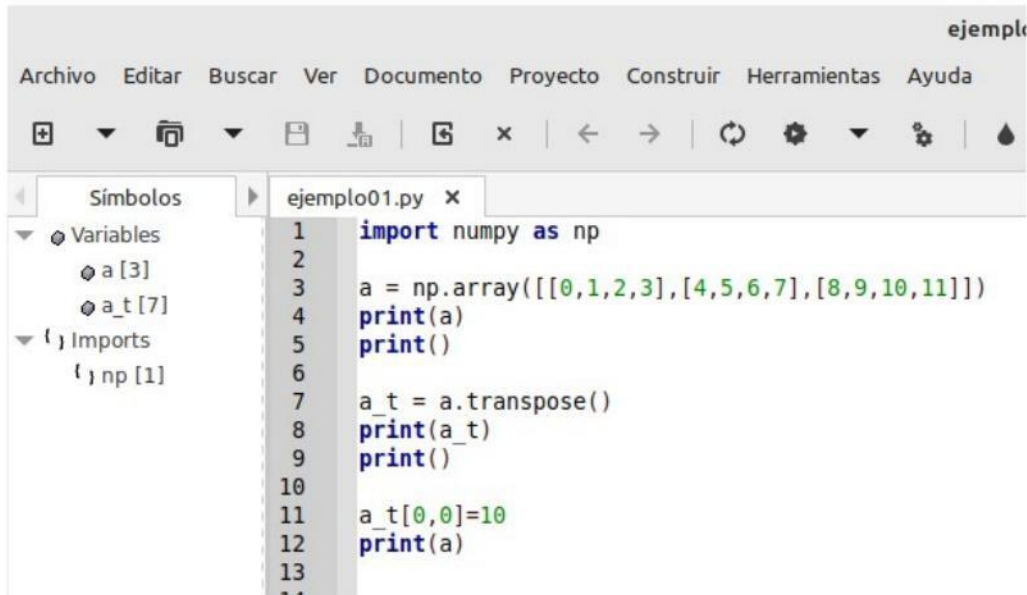
```
[[ 1.28571429 -0.28571429 -0.14285714]
 [-0.47619048  0.14285714  0.23809524]
 [ 0.19047619  0.14285714 -0.0952381 ]]

[[ 1.00000000e+00 -5.55111512e-17  0.00000000e+00]
 [ 0.00000000e+00  1.00000000e+00  0.00000000e+00]
 [-2.22044605e-16  5.55111512e-17  1.00000000e+00]]

-----
(program exited with code: 0)
Press return to continue
```

Transposición de matrices

Aunque existen algunos otros métodos, `transpose` y `ndarray.T` son los más importantes. Son equivalentes, y devuelven una vista. Por lo tanto, los elementos que se cambien en la vista, se cambiarán en la matriz original.



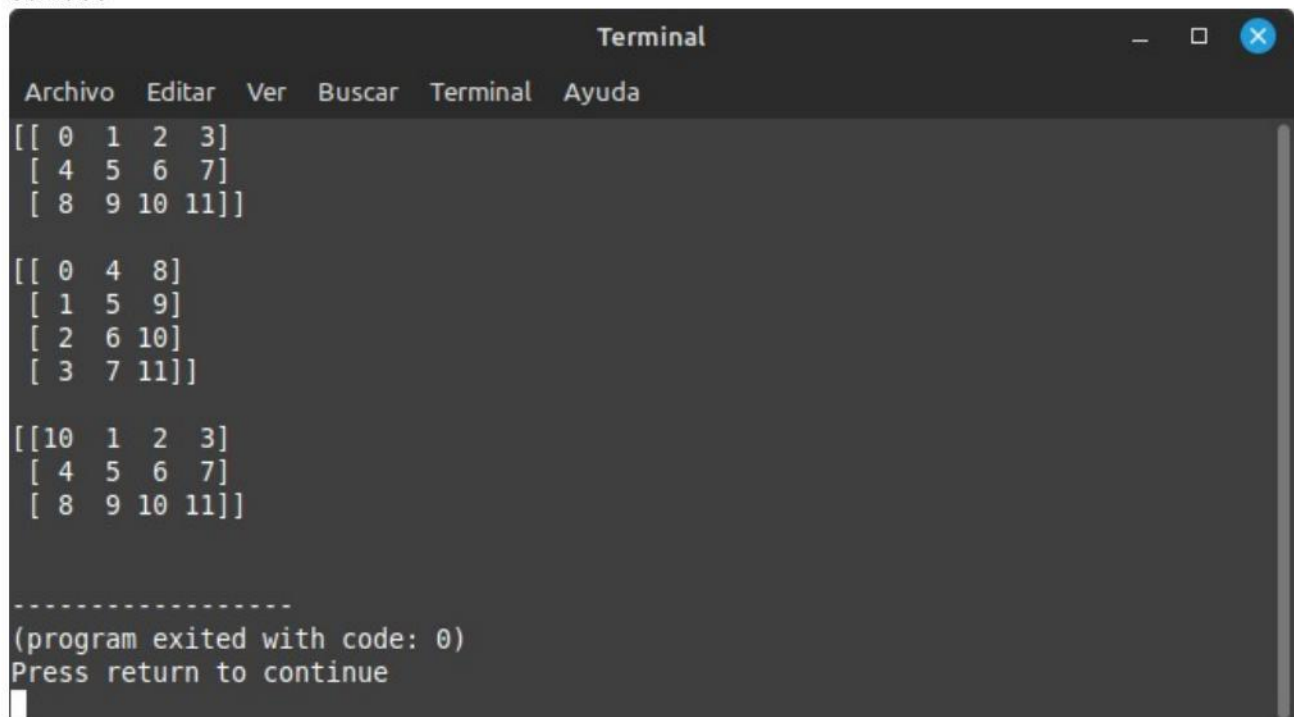
The screenshot shows a code editor window titled 'ejemplo01.py'. The menu bar includes 'Archivo', 'Editar', 'Buscar', 'Ver', 'Documento', 'Proyecto', 'Construir', 'Herramientas', and 'Ayuda'. The left sidebar shows a 'Símbolos' (Symbols) pane with a tree view containing 'Variables' (a [3], a_t [7]) and 'Imports' (np [1]). The main editor area contains the following Python code:

```

1 import numpy as np
2
3 a = np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
4 print(a)
5 print()
6
7 a_t = a.transpose()
8 print(a_t)
9 print()
10
11 a_t[0,0]=10
12 print(a)
13
14

```

Salida:



The screenshot shows a terminal window titled 'Terminal'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal output displays the results of the Python script:

```

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]

[[10  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

-----
(program exited with code: 0)
Press return to continue

```