In [10]:

```
# NumPy Exercises
```

In [11]:

```
# 1. Import the numpy package under the name np
import numpy as np
```

In [12]:

```
# 2. Print the numpy version and the configuration
print(np.__version__)
np.show_config()
```

```
2.1.0
Build Dependencies:
  blas:
    detection method: pkgconfig
    found: true
    include directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-run-ofdn91k7/cp310-wi
n_amd64/build/venv/Lib/site-packages/scipy_openblas64/include
    lib directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-run-ofdn91k7/cp310-win_am
d64/build/venv/Lib/site-packages/scipy_openblas64/lib
    name: scipy-openblas
    openblas configuration: OpenBLAS 0.3.27  USE64BITINT DYNAMIC_ARCH NO_AFFINITY
      Haswell MAX_THREADS=24
    pc file directory: D:/a/numpy/numpy/.openblas
    version: 0.3.27
  lapack:
    detection method: pkgconfig
    found: true
    include directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-run-ofdn91k7/cp310-wi
n_amd64/build/venv/Lib/site-packages/scipy_openblas64/include
    lib directory: C:/Users/runneradmin/AppData/Local/Temp/cibw-run-ofdn91k7/cp310-win_am
d64/build/venv/Lib/site-packages/scipy_openblas64/lib
    name: scipy-openblas
    openblas configuration: OpenBLAS 0.3.27  USE64BITINT DYNAMIC_ARCH NO_AFFINITY
      Haswell MAX_THREADS=24
    pc file directory: D:/a/numpy/numpy/.openblas
    version: 0.3.27
Compilers:
  c:
    commands: cl
    linker: link
    name: msvc
    version: 19.29.30154
  c++:
    commands: cl
    linker: link
    name: msvc
    version: 19.29.30154
  cython:
    commands: cython
    linker: cython
    name: cython
    version: 3.0.11
Machine Information:
  build:
    cpu: x86_64
    endian: little
    family: x86_64
    system: windows
  host:
    cpu: x86_64
    endian: little
    family: x86_64
    system: windows
Python Information:
```

```
  path: C:\Users\runneradmin\AppData\Local\Temp\build-env-_mpwizpm\Scripts\python.exe
  version: '3.10'
SIMD Extensions:
  baseline:
  - SSE
  - SSE2
  - SSE3
  found:
  - SSSE3
  - SSE41
  - POPCNT
  - SSE42
  - AVX
  - F16C
  - FMA3
  - AVX2
  not found:
  - AVX512F
  - AVX512CD
  - AVX512_SKX
  - AVX512_CLX
  - AVX512_CNL
  - AVX512_ICL
```

In [13]:

```
# 3. Create a null vector of size 10
null_vector = np.zeros(10)
print(null_vector)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [14]:

```
# 4. How to find the memory size of any array
print(f"Memory size of null_vector: {null_vector.nbytes} bytes")
```

```
Memory size of null_vector: 80 bytes
```

In [15]:

```
# 5. How to get the documentation of the numpy add function from the command line?
# In the command line, you would use: `numpy.info(numpy.add)`, for example:
np.info(np.add)
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok
=True[, signature])

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.
```

```
Returns
-------
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
--------
>>> import numpy as np
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[  0.,   2.,   4.],
       [  3.,   5.,   7.],
       [  6.,   8.,  10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on ndarrays.

>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> x1 + x2
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
```

In [16]:

```
# 6. Create a null vector of size 10 but the fifth value which is 1
vector = np.zeros(10)
vector[4] = 1
print(vector)
```

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

In [17]:

```
# 7. Create a vector with values ranging from 10 to 49
vector = np.arange(10, 50)
print(vector)
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
```

In [18]:

```
# 8. Reverse a vector (first element becomes last)
reversed_vector = vector[::-1]
print(reversed_vector)
```

```
[49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26
 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]
```

In [19]:

```
# 9. Create a 3x3 matrix with values ranging from 0 to 8
matrix_3x3 = np.arange(9).reshape(3, 3)
print(matrix_3x3)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

In [20]:

```
# 10. Find indices of non-zero elements from [1,2,0,0,4,0]
```

```
non_zero_indices = np.nonzero([1, 2, 0, 0, 4, 0])
print(non_zero_indices)
```

```
(array([0, 1, 4]),)
```

In [21]:

```
# 11. Create a 3x3 identity matrix
identity_matrix = np.eye(3)
print(identity_matrix)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [22]:

```
# 12. Create a 3x3x3 array with random values
random_array_3x3x3 = np.random.random((3, 3, 3))
print(random_array_3x3x3)
```

```
[[[0.13870815 0.19888005 0.60239506]
  [0.67635287 0.23456389 0.33902694]
  [0.0047782  0.66115358 0.1804157 ]]

 [[0.29871072 0.14173643 0.90922764]
  [0.95657383 0.5260002  0.6379815 ]
  [0.67591786 0.54436966 0.90106931]]

 [[0.76552458 0.34168298 0.29936024]
  [0.06410715 0.95767401 0.75171747]
  [0.35162902 0.80809907 0.93285392]]]
```

In [23]:

```
# 13. Create a 10x10 array with random values and find the minimum and maximum values
random_array_10x10 = np.random.random((10, 10))
min_val = random_array_10x10.min()
max_val = random_array_10x10.max()
print(f"Min: {min_val}, Max: {max_val}")
```

```
Min: 0.04937422768400812, Max: 0.9980596161802442
```

In [24]:

```
# 14. Create a random vector of size 30 and find the mean value
random_vector = np.random.random(30)
mean_val = random_vector.mean()
print(f"Mean value: {mean_val}")
```

```
Mean value: 0.521833365850465
```

In [25]:

```
# 15. Create a 2d array with 1 on the border and 0 inside
array_2d = np.ones((5, 5))
array_2d[1:-1, 1:-1] = 0
print(array_2d)
```

```
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

In [26]:

```
# 16. How to add a border (filled with 0's) around an existing array?
array_with_border = np.pad(array_2d, pad_width=1, mode='constant', constant_values=0)
print(array_with_border)
```

```
[[0. 0. 0. 0. 0. 0. 0.]
```

```
 [0. 1. 1. 1. 1. 1. 0.]
 [0. 1. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 1. 0.]
 [0. 1. 1. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]
```

In [27]:

```
# 17. What is the result of the following expression?
print(0 * np.nan)
print(np.nan == np.nan)
print(np.inf > np.nan)
print(np.nan - np.nan)
print(np.nan in set([np.nan]))
print(0.3 == 3 * 0.1)
```

```
nan
False
False
nan
True
False
```

In [28]:

```
# 18. Create a 5x5 matrix with values 1,2,3,4 just below the diagonal
matrix_5x5 = np.diag(1 + np.arange(4), k=-1)
print(matrix_5x5)
```

```
[[0 0 0 0 0]
 [1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]]
```

In [29]:

```
# 19. Create an 8x8 matrix and fill it with a checkerboard pattern
checkerboard = np.zeros((8, 8), dtype=int)
checkerboard[1::2, ::2] = 1
checkerboard[::2, 1::2] = 1
print(checkerboard)
```

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

In [30]:

```
# 20. Consider a (6,7,8) shape array, what is the index (x,y,z) of the 100th element?
array_6x7x8 = np.zeros((6, 7, 8))
index_100th = np.unravel_index(100, array_6x7x8.shape)
print(index_100th)
```

```
(np.int64(1), np.int64(5), np.int64(4))
```

In [31]:

```
# 21. Create a checkerboard 8x8 matrix using the tile function
checkerboard_tile = np.tile(np.array([[0, 1], [1, 0]]), (4, 4))
print(checkerboard_tile)
```

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
```

```
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

In [32]:

```
# 22. Normalize a 5x5 random matrix
random_matrix_5x5 = np.random.random((5, 5))
norm_matrix = (random_matrix_5x5 - np.mean(random_matrix_5x5)) / np.std(random_matrix_5x
5)
print(norm_matrix)
```

```
[[-1.52399729 -0.74304313  0.781897    0.78343468 -1.64750811]
 [ 0.56029137  0.79579412  0.7201107   1.16334161  0.29391682]
 [-1.18004155  1.26806091  0.6519149   0.07350136  1.27912281]
 [-1.19667804 -0.17031096  0.76006821 -0.42507787  0.10970852]
 [-0.76885816 -1.00305842  1.69108873 -0.45126642 -1.82241179]]
```

In [41]:

```
# 23. Create a custom dtype that describes a color as four unsigned bytes (RGBA)
color_dtype = np.dtype([("r", np.ubyte, 1),
                        ("g", np.ubyte, 1),
                        ("b", np.ubyte, 1),
                        ("a", np.ubyte, 1)])
print(color_dtype)
```

```
[('r', 'u1', (1,)), ('g', 'u1', (1,)), ('b', 'u1', (1,)), ('a', 'u1', (1,))]
```

In [34]:

```
# 24. Multiply a 5x3 matrix by a 3x2 matrix (real matrix product)
matrix_5x3 = np.random.random((5, 3))
matrix_3x2 = np.random.random((3, 2))
matrix_product = np.dot(matrix_5x3, matrix_3x2)
print(matrix_product)
```

```
[[0.69001759 0.76682327]
 [1.13859257 1.52836768]
 [1.03134599 0.91678998]
 [0.38541333 0.62570974]
 [0.82212226 1.0218611 ]]
```

In [35]:

```
# 25. Given a 1D array, negate all elements which are between 3 and 8, in place.
Z = np.arange(11)
Z[(3 < Z) & (Z < 8)] *= -1
print(Z)
```

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

In [36]:

```
# 26. What is the output of the following script?
# Author: Jake VanderPlas

print(sum(range(5), -1))
from numpy import *
print(sum(range(5), -1))
```

```
10
10
```

In [37]:

```
# 27. Consider an integer vector Z, which of these expressions are legal?
Z = np.arange(5)
print(Z**Z)
print(2 << Z >> 2)
```

```
print(Z < -Z)
print(1j * Z)
print(Z / 1 / 1)
print(Z < Z > Z)  # This is an illegal expression, the truth value of an array with more than one element is ambiguous.
```

```
[  1   1   4  27 256]
[0 1 2 4 8]
[False False False False False]
[0.+0.j 0.+1.j 0.+2.j 0.+3.j 0.+4.j]
[0. 1. 2. 3. 4.]

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[37], line 8
      6 print(1j * Z)
      7 print(Z / 1 / 1)
----> 8 print(Z < Z > Z)  # This is an illegal expression, the truth value of an array wi
th more than one element is ambiguous.

ValueError: The truth value of an array with more than one element is ambiguous. Use a.an
y() or a.all()
```

In [ ]:

```
# 28. What are the result of the following expressions?
print(np.array(0) / np.array(0))
print(np.array(0) // np.array(0))
print(np.array([np.nan]).astype(int).astype(float))
```

In [ ]:

```
# 29. How to round away from zero a float array?
float_array = np.random.uniform(-10, 10, 10)
rounded_away_from_zero = np.copysign(np.ceil(np.abs(float_array)), float_array)
print(rounded_away_from_zero)
```

In [ ]:

```
# 30. How to find common values between two arrays?
array1 = np.random.randint(0, 10, 10)
array2 = np.random.randint(0, 10, 10)
common_values = np.intersect1d(array1, array2)
print(common_values)
```

In [ ]:

```
# 31. How to ignore all numpy warnings (not recommended)?
np.seterr(all="ignore")
# Ignored warnings, not recommended
```

Out[ ]:

```
{'divide': 'warn', 'over': 'warn', 'under': 'ignore', 'invalid': 'warn'}
```

In [42]:

```
# 32. Is the following expression true?
print(np.sqrt(-1) == np.emath.sqrt(-1))
```

```
False
```

In [43]:

```
# 33. How to get the dates of yesterday, today and tomorrow?
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
today = np.datetime64('today', 'D')
tomorrow = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
print(yesterday, today, tomorrow)
```

```
2024-08-18 2024-08-19 2024-08-20
```