Nome: João Emanuel - Matrícula: 162080263 - Data 01/Setembro/2020

In [1]:

```
import numpy as np
```

In [2]:

```
np.__version__
```

Out[2]:

```
'1.18.5'
```

In [3]:

```
help(np.array)
```

```
Help on built-in function array in module numpy:

array(...)
    array(object, dtype=None, copy=True, order='K', subok=False, ndm
in=0)

    Create an array.

    Parameters
    ----------
    object : array_like
        An array, any object exposing the array interface, an object
whose
        __array__ method returns an array, or any (nested) sequence.
    dtype : data-type, optional
        The desired data-type for the array.  If not given, then the
type will
        be determined as the minimum type required to hold the objec
ts in the
        sequence.
    copy : bool, optional
        If true (default), then the object is copied.  Otherwise, a
copy will
        only be made if __array__ returns a copy, if obj is a nested
sequence,
        or if a copy is needed to satisfy any of the other requireme
nts
        (`dtype`, `order`, etc.).
    order : {'K', 'A', 'C', 'F'}, optional
        Specify the memory layout of the array. If object is not an
array, the
        newly created array will be in C order (row major) unless
'F' is
        specified, in which case it will be in Fortran order (column
major).
        If object is an array the following holds.

        ===== ========= ===========================================
=======
        order  no copy                     copy=True
        ===== ========= ===========================================
=======
        'K'   unchanged F & C order preserved, otherwise most simila
r order
        'A'   unchanged F order if input is F and not C, otherwise C
order
        'C'   C order   C order
        'F'   F order   F order
        ===== ========= ===========================================
=======

        When ``copy=False`` and a copy is made for other reasons, th
e result is
        the same as if ``copy=True``, with some exceptions for `A`,
see the
        Notes section. The default order is 'K'.
    subok : bool, optional
        If True, then sub-classes will be passed-through, otherwise
        the returned array will be forced to be a base-class array
(default).
    ndmin : int, optional
```

```
        Specifies the minimum number of dimensions that the resultin
g
        array should have.  Ones will be pre-pended to the shape as
        needed to meet this requirement.

    Returns
    -------
    out : ndarray
        An array object satisfying the specified requirements.

    See Also
    --------
    empty_like : Return an empty array with shape and type of input.
    ones_like : Return an array of ones with shape and type of inpu
t.
    zeros_like : Return an array of zeros with shape and type of inp
ut.
    full_like : Return a new array with shape of input filled with v
alue.
    empty : Return a new uninitialized array.
    ones : Return a new array setting values to one.
    zeros : Return a new array setting values to zero.
    full : Return a new array of given shape filled with value.


    Notes
    -----
    When order is 'A' and `object` is an array in neither 'C' nor
'F' order,
    and a copy is forced by a change in dtype, then the order of the
result is
    not necessarily 'C' as expected. This is likely a bug.

    Examples
    --------
    >>> np.array([1, 2, 3])
    array([1, 2, 3])

    Upcasting:

    >>> np.array([1, 2, 3.0])
    array([ 1.,  2.,  3.])

    More than one dimension:

    >>> np.array([[1, 2], [3, 4]])
    array([[1, 2],
           [3, 4]])

    Minimum dimensions 2:

    >>> np.array([1, 2, 3], ndmin=2)
    array([[1, 2, 3]])

    Type provided:

    >>> np.array([1, 2, 3], dtype=complex)
    array([ 1.+0.j,  2.+0.j,  3.+0.j])

    Data-type consisting of more than one element:
```

```
>>> x = np.array([(1,2),(3,4)],dtype=[('a','<i4'),('b','<i4')])
>>> x['a']
array([1, 3])

Creating an array from sub-classes:

>>> np.array(np.mat('1 2; 3 4'))
array([[1, 2],
       [3, 4]])

>>> np.array(np.mat('1 2; 3 4'), subok=True)
matrix([[1, 2],
        [3, 4]])
```

In [4]:

```
v1 = np.array( [0, 1, 2, 3, 4, 5, 6, 7, 8 ] )
v1
```

Out[4]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [5]:

```
type(v1)
```

Out[5]:

```
numpy.ndarray
```

In [6]:

```
v1.cumsum()
```

Out[6]:

```
array([ 0,  1,  3,  6, 10, 15, 21, 28, 36])
```

In [7]:

```
print(v1)
```

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

In [8]:

```
v1[0] = 100
v1
```

Out[8]:

```
array([100,   1,   2,   3,   4,   5,   6,   7,   8])
```

In [10]:

```
v1[1] = 'cct'
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-10-458db5d6f861> in <module>
----> 1 v1[1] = 'cct'

ValueError: invalid literal for int() with base 10: 'cct'
```

In [ ]:

```
v1.shape
```

In [ ]:

```
v1.dtype
```

Funções

In [ ]:

```
# Intervalo - Inicio, Fim (Exclusive), Passo ou Incremento
v2 = np.arange(0., 4.5, .5)
v2
```

In [ ]:

```
type(v2)
```

In [ ]:

```
np.shape(v2)
```

In [ ]:

```
v2.dtype
```

In [ ]:

```
x = np.arange(1, 10, 0.25)
x
```

In [11]:

```
b = np.array([True, False, True, True])
b
```

Out[11]:

```
array([ True, False,  True,  True])
```

In [12]:

```
b.dtype
```

Out[12]:

```
dtype('bool')
```

In [13]:

```
s = np.array( ['python', 'R', 'java', 'érika'] )
s
```

Out[13]:

```
array(['python', 'R', 'java', 'érika'], dtype='<U6')
```

Matrizes

In [14]:

```
m = np.array( [ [1,2,3], [4,5,6]  ] )
m
```

Out[14]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [15]:

```
m.shape
```

Out[15]:

```
(2, 3)
```

In [16]:

```
m1 = np.ones((2,3))
m1
```

Out[16]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [17]:

```
lista = [ [13, 24, 56], [23, 76, 12], [68, 61, 8]  ]
lista
```

Out[17]:

```
[[13, 24, 56], [23, 76, 12], [68, 61, 8]]
```

In [18]:

```
type(lista)
```

Out[18]:

```
list
```

In [19]:

```
m2 = np.matrix(lista)
m2
```

Out[19]:

```
matrix([[13, 24, 56],
        [23, 76, 12],
        [68, 61,  8]])
```

In [20]:

```
m2.shape
```

Out[20]:

```
(3, 3)
```

In [21]:

```
type(m2)
```

Out[21]:

```
numpy.matrix
```

In [22]:

```
m2.size # numero de elementos
```

Out[22]:

```
9
```

In [23]:

```
m2[2,1]
```

Out[23]:

```
61
```

In [24]:

```
m2[1,0] = 100
m2
```

Out[24]:

```
matrix([[ 13,  24,  56],
        [100,  76,  12],
        [ 68,  61,   8]])
```

In [25]:

```
z = np.array([1,2])
z.dtype
```

Out[25]:

```
dtype('int64')
```

In [26]:

```
z = np.array([1.0,2.0])
z.dtype
```

Out[26]:

```
dtype('float64')
```

In [27]:

```
z = np.array([1,2], dtype=np.float32)
z.dtype
```

Out[27]:

```
dtype('float32')
```

Método Random()

In [28]:

```
# 10 números aleatórios
x = np.random.rand(10)
x
```

Out[28]:

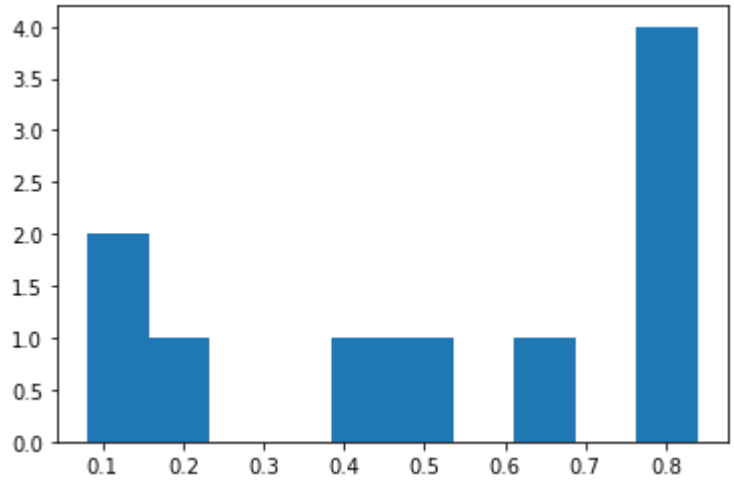
```
array([0.62813991, 0.45657807, 0.46938097, 0.83931671, 0.08174096,
       0.14671445, 0.80637197, 0.18992328, 0.83743415, 0.80319935])
```

In [29]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [30]:

```
plt.show(plt.hist(x))
```

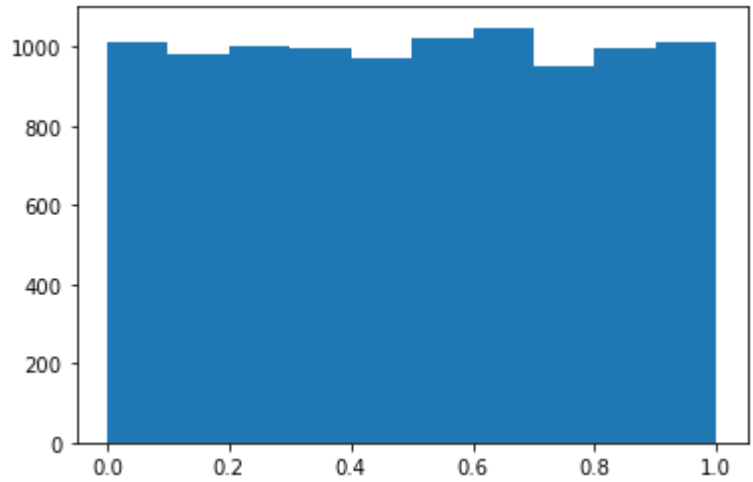


In [31]:

```
x = np.random.rand(10000)
```

In [32]:

```
plt.show(plt.hist(x))
```



In [33]:

```
x[:20]
```

Out[33]:

```
array([0.36786047, 0.08144455, 0.33337257, 0.0412542 , 0.30509273,
       0.6014967 , 0.65102582, 0.83026596, 0.39598027, 0.54571664,
       0.27863985, 0.98809189, 0.46138319, 0.39234486, 0.68987368,
       0.48497615, 0.57619497, 0.73071466, 0.80267159, 0.70641612])
```

In [34]:

```
x = np.random.randn(100000) # distribuicao normal
```

In [35]:

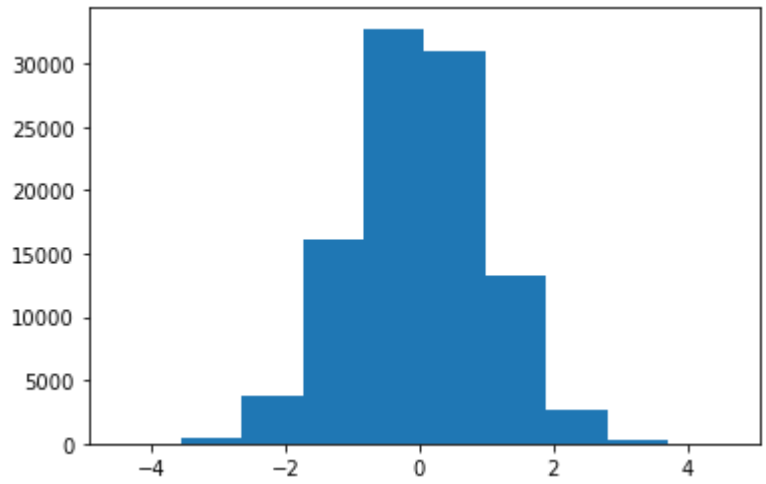
```
x[:20]
```

Out[35]:

```
array([ 0.49341166,  1.15434072,  0.96920172,  0.04547627,  1.130559
09,
        0.60967428,  1.18651882, -0.08299358, -1.5051436 ,  2.708959
06,
       -2.31236856,  0.54667956,  1.58559245,  1.35578449, -0.589041
76,
        0.21509599, -1.95724915,  0.7812032 ,  1.40260506, -0.521945
8 ])
```

In [36]:

```
plt.show(plt.hist(x))
```



In [37]:

```
url = 'https://raw.githubusercontent.com/vladimiralencar/Alunos-UEPB-TopicosEspeciaisEmBancoDeDados/master/Python-Para-Analise-de-Dados/iris.csv'
```

In [38]:

```
f = np.loadtxt(url, delimiter=',', usecols=(0,1,2,3), skiprows=1)
print(f)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.1 1.5 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
```

```
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
```

```
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
```

In [39]:

```
f.shape
```

Out[39]:

```
(150, 4)
```

In [40]:

```
type(f)
```

Out[40]:

```
numpy.ndarray
```

In [44]:

```
var1, var2 = np.loadtxt(url, delimiter=',', usecols=(0,1), skiprows=1, unpack=True)
print(var1[:10])
```

```
[5.1 4.9 4.7 4.6 5.  5.4 4.6 5.  4.4 4.9]
```

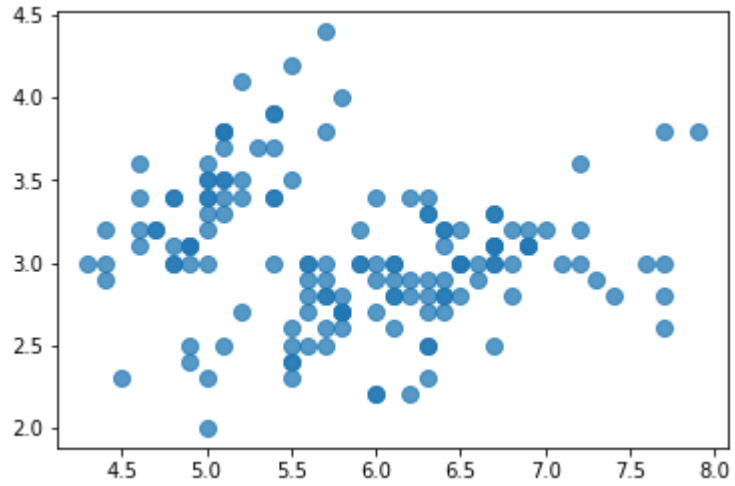
In [45]:

```
print(var2[:10])
```

```
[3.5 3.  3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1]
```

In [57]:

```
plt.show(plt.plot(var1, var2, 'o', markersize=8, alpha=0.75))
```



Estatística - Funções

In [70]:

```
a = np.array([15, 34, -10, 100, 50, 70, 30, 10, 10, 10, 20, 20, 20])
```

In [71]:

```
a
```

Out[71]:

```
array([ 15,  34, -10, 100,  50,  70,  30,  10,  10,  10,  20,  20,
20])
```

In [72]:

```
a.dtype
```

Out[72]:

```
dtype('int64')
```

In [73]:

```
np.mean(a)
```

Out[73]:

```
29.153846153846153
```

In [74]:

```
media = np.mean(a)
media
```

Out[74]:

```
29.153846153846153
```

In [75]:

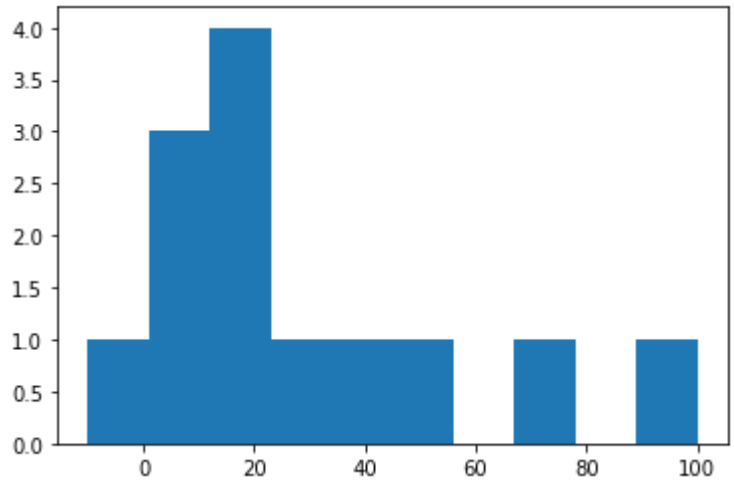
```
# desvio padrão
np.std(a)
```

Out[75]:

```
28.054469420002754
```

In [76]:

```
plt.show(plt.hist(a))
```



In [77]:

```
# variancia
np.var(a)
```

Out[77]:

```
787.0532544378696
```

In [78]:

```
d = np.arange(1,10) # exclusive 10 - 1
d
```

Out[78]:

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [79]:

```
np.sum(d)
```

Out[79]:

```
45
```

In [80]:

```
np.prod(d)
```

Out[80]:

```
362880
```

In [81]:

```
# soma acumulada
np.cumsum(d)
```

Out[81]:

```
array([ 1,  3,  6, 10, 15, 21, 28, 36, 45])
```

In [82]:

```
a = np.random.randn(400, 2)
a.shape
```

Out[82]:

```
(400, 2)
```

In [83]:

```
a[:2]
```

Out[83]:

```
array([[-0.08896297,  1.2855831 ],
       [-0.99628991,  1.21655525]])
```

In [84]:

```
m = np.mean(a)
m
```

Out[84]:

```
-0.042545512981045024
```

In [85]:

```
m0 = a.mean(0)
m0
```

Out[85]:

```
array([-0.01524503, -0.069846  ])
```

In [89]:

```
a[:, 0] # todas a linhas da coluna 0
```

Out[89]:

```
array([-0.08896297, -0.99628991, -0.03442457, -0.71752453, -0.498621
43,
        0.40981241,  0.48650734, -0.46942676,  0.27730296,  0.800987
16,
       -0.47342642, -1.42349899,  1.34806072,  0.81256941, -0.862071
3 ,
        0.20420516, -0.47119176, -0.35680977,  0.55928944, -0.482308
98,
        0.28746628,  1.41434547, -1.85735815,  0.36335563, -0.080982
97,
        1.82292957, -0.80330311, -0.79074767, -1.26935369, -0.520120
53,
        0.2257878 ,  0.02583964,  0.11730608,  0.43906555, -0.233370
36,
       -0.69119394, -0.34690569,  1.38214369, -0.17557417, -1.185558
83,
       -1.91996682,  1.60219474,  0.52833372,  0.54299085, -0.385529
14,
       -2.75845001, -1.17713382,  1.07559986, -2.32405285,  1.765777
26,
       -0.55289749, -0.32926781, -0.66448778,  1.20968307,  0.798199
74,
       -1.90323581, -1.78552237, -0.90054357, -0.34023549,  0.967845
72,
        0.18415518, -1.71908345,  0.75980014, -0.3143959 , -0.082025
65,
       -0.04216491, -0.97966231,  0.38336049,  0.12223894,  0.962863
68,
       -1.27842221,  0.39629752, -1.78173792, -0.58591618,  0.952279
84,
       -0.56456123,  3.31894431, -0.2327632 ,  0.65542765, -1.076331
33,
        0.38862499,  0.82620214,  0.47062181,  0.53377822,  0.005022
25,
        1.37086598, -0.19036965, -0.63531557, -0.92672333,  0.404963
25,
       -0.08346725,  1.4003971 , -0.88069367, -1.30880106,  0.608423
82,
       -1.13076713,  0.1826006 , -0.65578418, -2.05613491, -0.451327
57,
        0.95741848,  0.89113194,  0.36893033, -0.27857811, -0.016540
9 ,
        1.38898112,  0.7593468 ,  1.40554648,  0.20596416,  0.592920
29,
        2.04109018,  0.39017339,  0.30157292,  0.1488681 , -0.670947
04,
       -1.24530588, -0.09183606,  2.07654014,  0.73212552, -0.007654
31,
        0.04493141, -0.31209694,  0.36473542, -1.24122361,  0.029884
57,
       -3.39665508, -2.2641536 , -0.92944785,  1.83651088,  0.496565
35,
       -0.53601453,  1.14753233, -0.68491051, -0.17396628,  0.418255
15,
        0.21252997,  0.26108618, -1.02585486,  2.53849299,  0.582026
96,
       -0.05165561, -0.45174709, -0.30949401, -0.57811698,  0.180396
94,
        0.49180664, -0.381137  ,  1.26844531, -0.12556808,  1.270403
```

```
28,
        1.37422827, -0.7478236 ,  0.21240183,  0.05082903,  0.212306
62,
       -0.17658372,  0.2413321 , -1.4512537 , -1.33017574, -1.347370
99,
       -1.12746425,  1.41817774,  1.49030725,  1.06631803,  1.216072
29,
        1.4885019 ,  0.37110263,  0.55160444, -1.91676295, -0.896398
58,
        0.51419037,  0.5738549 , -0.61175248, -0.12773118, -0.970345
83,
       -0.11518977, -0.07704008, -0.20823596,  0.89008646, -0.113241
56,
        0.77894665,  0.93881405,  0.32904321,  1.19693699, -0.689187
95,
        0.05736148, -0.976513  , -0.1140565 ,  1.08775946, -0.590053
23,
       -2.16431493,  1.18168587, -0.79226024,  1.11595221, -1.120342
57,
        2.41069957, -1.69453769, -0.15392141, -1.19529872,  1.055264
83,
       -0.63981145,  1.39536062, -0.69168936,  0.06832387,  1.266999
65,
       -0.92291076,  0.9839584 , -0.90095652, -0.53368681,  0.806864
37,
       -0.65331697,  0.36509829, -0.39641029, -1.37638555,  0.414107
4 ,
       -0.57976013, -1.13157183, -1.2163263 ,  0.7553763 , -0.687559
19,
        1.06975471,  0.40217524, -0.23370566, -0.19337451, -0.797517
47,
       -1.01533368, -0.74067565, -0.70575521,  0.43841783,  0.525963
79,
       -0.77172742, -2.04448161,  0.46193405, -2.24750972, -1.436182
13,
       -0.47421208,  0.22073216,  0.98525773,  0.13725243, -1.793957
56,
       -1.90160869,  0.87461783,  0.31156614,  1.07029222, -0.663189
4 ,
       -1.1140649 ,  0.5209052 , -1.50936943, -0.01386233, -0.448351
23,
        0.42340409, -1.36140501,  1.7322169 ,  1.85342192, -1.198725
89,
        0.36682074,  0.69144319, -0.15501374, -0.61724403, -0.246600
85,
       -0.19590041, -0.87413294,  0.97744922, -0.76742194,  0.205217
95,
       -0.96835695,  2.081304  , -0.03532837, -2.02539223, -1.531107
27,
        0.49822217, -0.42293933,  0.46323893,  1.52040264,  0.682222
37,
       -0.01709417,  1.50789459,  0.15274143, -0.38633795, -0.434426
59,
        0.35977974,  0.10994393,  0.14643599,  1.07783976, -1.078023
78,
        0.71974486, -0.033623  ,  0.80733156, -0.06581134,  0.046288
13,
       -0.39104057,  0.71350532,  0.12215295,  0.03023434,  0.505534
66,
       -0.84311353,  0.03607727,  0.89929888,  1.1488878 , -0.337522
64,
```

```
       -1.04855686,  0.47900072, -0.88404105, -1.49742756,  0.576570
15,
       -0.77682728,  0.43424601, -0.34209781,  1.57731845,  2.590991
86,
        0.61843805, -1.87139853, -1.14737307,  0.11826155, -2.008558
75,
       -1.31532117, -0.89508956, -1.41657898,  0.22234693, -0.408087
33,
       -0.77232336, -0.43137565, -0.09335466, -0.61951439, -0.574349
02,
       -0.86195049,  2.5824003 ,  2.29856769,  1.38228161, -0.120355
57,
        0.0163426 ,  1.37301397, -2.41954967,  2.21898328,  0.206999
36,
       -0.44426707, -0.28300455, -0.65090364,  0.56035625, -0.287316
44,
        0.21891034,  1.33869336, -0.14964721, -0.90614327, -2.242522
65,
       -0.20738837, -1.06225013,  0.12713545,  0.30906346,  1.432362
65,
       -0.72362812, -0.07515363,  0.89529865,  0.2722238 ,  0.553063
42,
       -0.74905904,  0.38863431, -1.0298591 ,  0.2133172 ,  1.684151
99,
       -0.44589025,  1.81781619,  0.51918159,  0.07778738,  1.145851
9 ,
       -1.03534791,  1.16038312,  0.56175461, -1.23505705,  0.579345
97,
        0.30825931, -1.12645226, -0.06608052,  0.22322695,  1.472233
96,
       -0.75841857, -0.41820033, -0.352581  ,  1.21246646,  1.630469
74,
        0.73486422,  0.65345383, -0.52145511,  0.97282551, -1.636688
,
        0.22328364, -1.54413184,  1.20611709, -0.70539712,  0.202031
5 ,
        1.45243195,  1.0273714 , -0.63884557, -0.19580227,  0.128399
04,
       -0.63746679,  0.08576558,  0.84450437,  2.23815724,  0.941033
6 ])
```

In [90]:

```
a[0:400, 0] # todas a linhas da coluna 0
```

Out[90]:

```
array([-0.08896297, -0.99628991, -0.03442457, -0.71752453, -0.498621
43,
        0.40981241,  0.48650734, -0.46942676,  0.27730296,  0.800987
16,
       -0.47342642, -1.42349899,  1.34806072,  0.81256941, -0.862071
3 ,
        0.20420516, -0.47119176, -0.35680977,  0.55928944, -0.482308
98,
        0.28746628,  1.41434547, -1.85735815,  0.36335563, -0.080982
97,
        1.82292957, -0.80330311, -0.79074767, -1.26935369, -0.520120
53,
        0.2257878 ,  0.02583964,  0.11730608,  0.43906555, -0.233370
36,
       -0.69119394, -0.34690569,  1.38214369, -0.17557417, -1.185558
83,
       -1.91996682,  1.60219474,  0.52833372,  0.54299085, -0.385529
14,
       -2.75845001, -1.17713382,  1.07559986, -2.32405285,  1.765777
26,
       -0.55289749, -0.32926781, -0.66448778,  1.20968307,  0.798199
74,
       -1.90323581, -1.78552237, -0.90054357, -0.34023549,  0.967845
72,
        0.18415518, -1.71908345,  0.75980014, -0.3143959 , -0.082025
65,
       -0.04216491, -0.97966231,  0.38336049,  0.12223894,  0.962863
68,
       -1.27842221,  0.39629752, -1.78173792, -0.58591618,  0.952279
84,
       -0.56456123,  3.31894431, -0.2327632 ,  0.65542765, -1.076331
33,
        0.38862499,  0.82620214,  0.47062181,  0.53377822,  0.005022
25,
        1.37086598, -0.19036965, -0.63531557, -0.92672333,  0.404963
25,
       -0.08346725,  1.4003971 , -0.88069367, -1.30880106,  0.608423
82,
       -1.13076713,  0.1826006 , -0.65578418, -2.05613491, -0.451327
57,
        0.95741848,  0.89113194,  0.36893033, -0.27857811, -0.016540
9 ,
        1.38898112,  0.7593468 ,  1.40554648,  0.20596416,  0.592920
29,
        2.04109018,  0.39017339,  0.30157292,  0.1488681 , -0.670947
04,
       -1.24530588, -0.09183606,  2.07654014,  0.73212552, -0.007654
31,
        0.04493141, -0.31209694,  0.36473542, -1.24122361,  0.029884
57,
       -3.39665508, -2.2641536 , -0.92944785,  1.83651088,  0.496565
35,
       -0.53601453,  1.14753233, -0.68491051, -0.17396628,  0.418255
15,
        0.21252997,  0.26108618, -1.02585486,  2.53849299,  0.582026
96,
       -0.05165561, -0.45174709, -0.30949401, -0.57811698,  0.180396
94,
        0.49180664, -0.381137  ,  1.26844531, -0.12556808,  1.270403
```

```
  28,
         1.37422827, -0.7478236 ,  0.21240183,  0.05082903,  0.212306
  62,
        -0.17658372,  0.2413321 , -1.4512537 , -1.33017574, -1.347370
  99,
        -1.12746425,  1.41817774,  1.49030725,  1.06631803,  1.216072
  29,
         1.4885019 ,  0.37110263,  0.55160444, -1.91676295, -0.896398
  58,
         0.51419037,  0.5738549 , -0.61175248, -0.12773118, -0.970345
  83,
        -0.11518977, -0.07704008, -0.20823596,  0.89008646, -0.113241
  56,
         0.77894665,  0.93881405,  0.32904321,  1.19693699, -0.689187
  95,
         0.05736148, -0.976513  , -0.1140565 ,  1.08775946, -0.590053
  23,
        -2.16431493,  1.18168587, -0.79226024,  1.11595221, -1.120342
  57,
         2.41069957, -1.69453769, -0.15392141, -1.19529872,  1.055264
  83,
        -0.63981145,  1.39536062, -0.69168936,  0.06832387,  1.266999
  65,
        -0.92291076,  0.9839584 , -0.90095652, -0.53368681,  0.806864
  37,
        -0.65331697,  0.36509829, -0.39641029, -1.37638555,  0.414107
  4 ,
        -0.57976013, -1.13157183, -1.2163263 ,  0.7553763 , -0.687559
  19,
         1.06975471,  0.40217524, -0.23370566, -0.19337451, -0.797517
  47,
        -1.01533368, -0.74067565, -0.70575521,  0.43841783,  0.525963
  79,
        -0.77172742, -2.04448161,  0.46193405, -2.24750972, -1.436182
  13,
        -0.47421208,  0.22073216,  0.98525773,  0.13725243, -1.793957
  56,
        -1.90160869,  0.87461783,  0.31156614,  1.07029222, -0.663189
  4 ,
        -1.1140649 ,  0.5209052 , -1.50936943, -0.01386233, -0.448351
  23,
         0.42340409, -1.36140501,  1.7322169 ,  1.85342192, -1.198725
  89,
         0.36682074,  0.69144319, -0.15501374, -0.61724403, -0.246600
  85,
        -0.19590041, -0.87413294,  0.97744922, -0.76742194,  0.205217
  95,
        -0.96835695,  2.081304  , -0.03532837, -2.02539223, -1.531107
  27,
         0.49822217, -0.42293933,  0.46323893,  1.52040264,  0.682222
  37,
        -0.01709417,  1.50789459,  0.15274143, -0.38633795, -0.434426
  59,
         0.35977974,  0.10994393,  0.14643599,  1.07783976, -1.078023
  78,
         0.71974486, -0.033623  ,  0.80733156, -0.06581134,  0.046288
  13,
        -0.39104057,  0.71350532,  0.12215295,  0.03023434,  0.505534
  66,
        -0.84311353,  0.03607727,  0.89929888,  1.1488878 , -0.337522
  64,
```

```
       -1.04855686,  0.47900072, -0.88404105, -1.49742756,  0.576570
15,
       -0.77682728,  0.43424601, -0.34209781,  1.57731845,  2.590991
86,
        0.61843805, -1.87139853, -1.14737307,  0.11826155, -2.008558
75,
       -1.31532117, -0.89508956, -1.41657898,  0.22234693, -0.408087
33,
       -0.77232336, -0.43137565, -0.09335466, -0.61951439, -0.574349
02,
       -0.86195049,  2.5824003 ,  2.29856769,  1.38228161, -0.120355
57,
        0.0163426 ,  1.37301397, -2.41954967,  2.21898328,  0.206999
36,
       -0.44426707, -0.28300455, -0.65090364,  0.56035625, -0.287316
44,
        0.21891034,  1.33869336, -0.14964721, -0.90614327, -2.242522
65,
       -0.20738837, -1.06225013,  0.12713545,  0.30906346,  1.432362
65,
       -0.72362812, -0.07515363,  0.89529865,  0.2722238 ,  0.553063
42,
       -0.74905904,  0.38863431, -1.0298591 ,  0.2133172 ,  1.684151
99,
       -0.44589025,  1.81781619,  0.51918159,  0.07778738,  1.145851
9 ,
       -1.03534791,  1.16038312,  0.56175461, -1.23505705,  0.579345
97,
        0.30825931, -1.12645226, -0.06608052,  0.22322695,  1.472233
96,
       -0.75841857, -0.41820033, -0.352581  ,  1.21246646,  1.630469
74,
        0.73486422,  0.65345383, -0.52145511,  0.97282551, -1.636688
,
        0.22328364, -1.54413184,  1.20611709, -0.70539712,  0.202031
5 ,
        1.45243195,  1.0273714 , -0.63884557, -0.19580227,  0.128399
04,
       -0.63746679,  0.08576558,  0.84450437,  2.23815724,  0.941033
6 ])
```

In [91]:

```
a.shape
```

Out[91]:

```
(400, 2)
```

In [94]:

```
linhas = a.shape[0]
linhas
```

Out[94]:

```
400
```

In [95]:

```
colunas = a.shape[1]
colunas
```

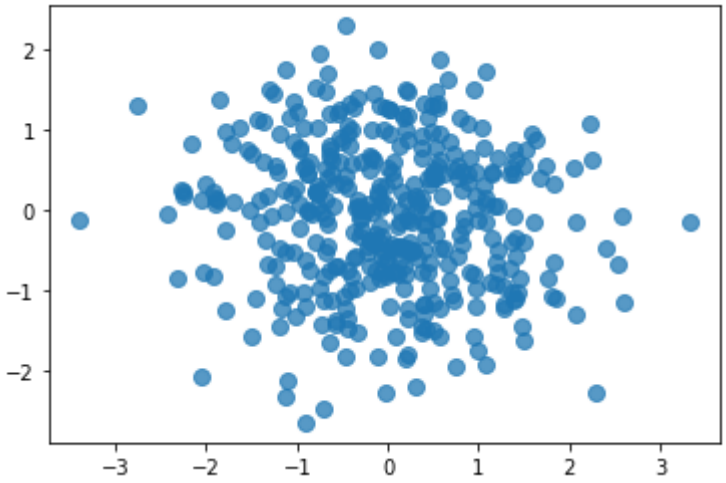
Out[95]:

2

In [96]:

```
import matplotlib.pyplot as plt

plt.plot(a[:,0], a[:,1], 'o', markersize=8, alpha=0.75)
plt.show()
```



In [97]:

```
media0 = np.mean(a[:,0])
media0
```

Out[97]:

-0.015245030821984308

In [98]:

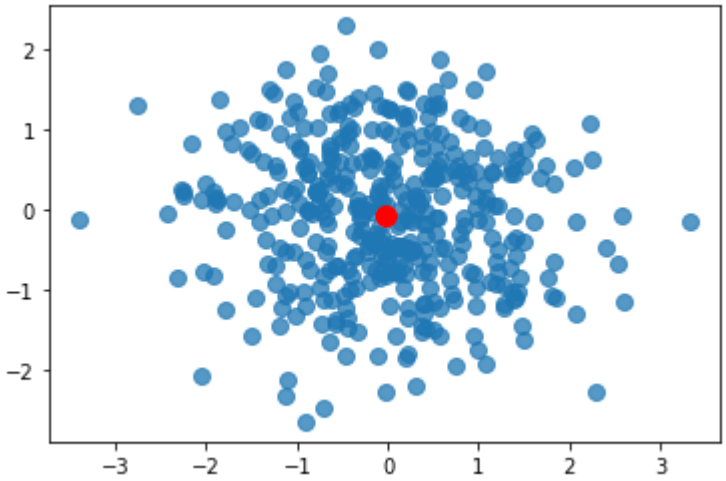
```
media1 = np.mean(a[:,1])
media1
```

Out[98]:

-0.06984599514010575

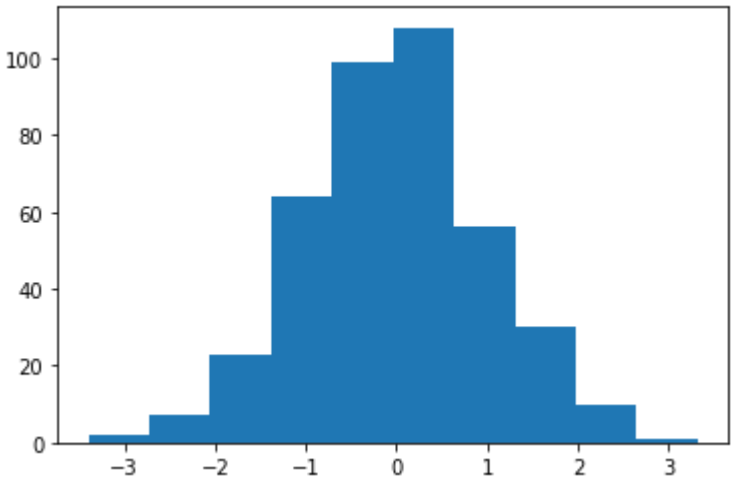
In [99]:

```
plt.plot(a[:,0], a[:,1], 'o', markersize=8, alpha=0.75)
plt.plot(media0, media1, 'ro', markersize=10)
plt.show()
```



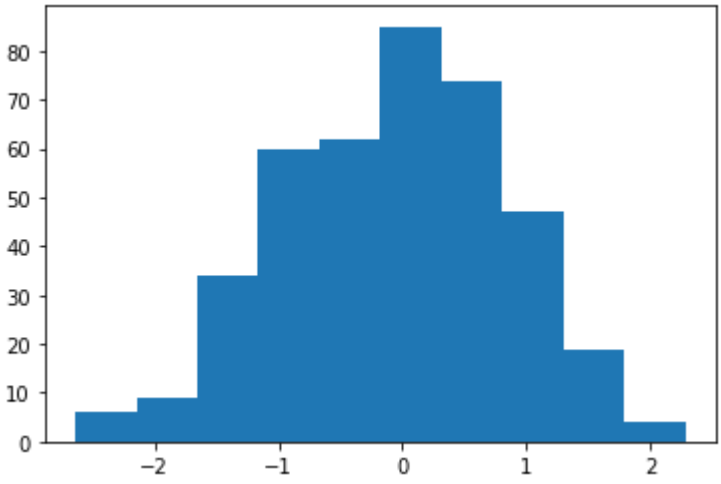
In [100]:

```
plt.show(plt.hist(a[:,0]))
```



In [101]:

```
plt.show(plt.hist(a[:,1]))
```



In [102]:

```
m0 = a[:, 0]
m0.shape
```

Out[102]:

```
(400,)
```

In [103]:

```
m0
```

Out[103]:

```
array([-0.08896297, -0.99628991, -0.03442457, -0.71752453, -0.498621
43,
        0.40981241,  0.48650734, -0.46942676,  0.27730296,  0.800987
16,
       -0.47342642, -1.42349899,  1.34806072,  0.81256941, -0.862071
3 ,
        0.20420516, -0.47119176, -0.35680977,  0.55928944, -0.482308
98,
        0.28746628,  1.41434547, -1.85735815,  0.36335563, -0.080982
97,
        1.82292957, -0.80330311, -0.79074767, -1.26935369, -0.520120
53,
        0.2257878 ,  0.02583964,  0.11730608,  0.43906555, -0.233370
36,
       -0.69119394, -0.34690569,  1.38214369, -0.17557417, -1.185558
83,
       -1.91996682,  1.60219474,  0.52833372,  0.54299085, -0.385529
14,
       -2.75845001, -1.17713382,  1.07559986, -2.32405285,  1.765777
26,
       -0.55289749, -0.32926781, -0.66448778,  1.20968307,  0.798199
74,
       -1.90323581, -1.78552237, -0.90054357, -0.34023549,  0.967845
72,
        0.18415518, -1.71908345,  0.75980014, -0.3143959 , -0.082025
65,
       -0.04216491, -0.97966231,  0.38336049,  0.12223894,  0.962863
68,
       -1.27842221,  0.39629752, -1.78173792, -0.58591618,  0.952279
84,
       -0.56456123,  3.31894431, -0.2327632 ,  0.65542765, -1.076331
33,
        0.38862499,  0.82620214,  0.47062181,  0.53377822,  0.005022
25,
        1.37086598, -0.19036965, -0.63531557, -0.92672333,  0.404963
25,
       -0.08346725,  1.4003971 , -0.88069367, -1.30880106,  0.608423
82,
       -1.13076713,  0.1826006 , -0.65578418, -2.05613491, -0.451327
57,
        0.95741848,  0.89113194,  0.36893033, -0.27857811, -0.016540
9 ,
        1.38898112,  0.7593468 ,  1.40554648,  0.20596416,  0.592920
29,
        2.04109018,  0.39017339,  0.30157292,  0.1488681 , -0.670947
04,
       -1.24530588, -0.09183606,  2.07654014,  0.73212552, -0.007654
31,
        0.04493141, -0.31209694,  0.36473542, -1.24122361,  0.029884
57,
       -3.39665508, -2.2641536 , -0.92944785,  1.83651088,  0.496565
35,
       -0.53601453,  1.14753233, -0.68491051, -0.17396628,  0.418255
15,
        0.21252997,  0.26108618, -1.02585486,  2.53849299,  0.582026
96,
       -0.05165561, -0.45174709, -0.30949401, -0.57811698,  0.180396
94,
        0.49180664, -0.381137  ,  1.26844531, -0.12556808,  1.270403
```

```
   28,
          1.37422827, -0.7478236 ,  0.21240183,  0.05082903,  0.212306
   62,
         -0.17658372,  0.2413321 , -1.4512537 , -1.33017574, -1.347370
   99,
         -1.12746425,  1.41817774,  1.49030725,  1.06631803,  1.216072
   29,
          1.4885019 ,  0.37110263,  0.55160444, -1.91676295, -0.896398
   58,
          0.51419037,  0.5738549 , -0.61175248, -0.12773118, -0.970345
   83,
         -0.11518977, -0.07704008, -0.20823596,  0.89008646, -0.113241
   56,
          0.77894665,  0.93881405,  0.32904321,  1.19693699, -0.689187
   95,
          0.05736148, -0.976513  , -0.1140565 ,  1.08775946, -0.590053
   23,
         -2.16431493,  1.18168587, -0.79226024,  1.11595221, -1.120342
   57,
          2.41069957, -1.69453769, -0.15392141, -1.19529872,  1.055264
   83,
         -0.63981145,  1.39536062, -0.69168936,  0.06832387,  1.266999
   65,
         -0.92291076,  0.9839584 , -0.90095652, -0.53368681,  0.806864
   37,
         -0.65331697,  0.36509829, -0.39641029, -1.37638555,  0.414107
    4 ,
         -0.57976013, -1.13157183, -1.2163263 ,  0.7553763 , -0.687559
   19,
          1.06975471,  0.40217524, -0.23370566, -0.19337451, -0.797517
   47,
         -1.01533368, -0.74067565, -0.70575521,  0.43841783,  0.525963
   79,
         -0.77172742, -2.04448161,  0.46193405, -2.24750972, -1.436182
   13,
         -0.47421208,  0.22073216,  0.98525773,  0.13725243, -1.793957
   56,
         -1.90160869,  0.87461783,  0.31156614,  1.07029222, -0.663189
    4 ,
         -1.1140649 ,  0.5209052 , -1.50936943, -0.01386233, -0.448351
   23,
          0.42340409, -1.36140501,  1.7322169 ,  1.85342192, -1.198725
   89,
          0.36682074,  0.69144319, -0.15501374, -0.61724403, -0.246600
   85,
         -0.19590041, -0.87413294,  0.97744922, -0.76742194,  0.205217
   95,
         -0.96835695,  2.081304  , -0.03532837, -2.02539223, -1.531107
   27,
          0.49822217, -0.42293933,  0.46323893,  1.52040264,  0.682222
   37,
         -0.01709417,  1.50789459,  0.15274143, -0.38633795, -0.434426
   59,
          0.35977974,  0.10994393,  0.14643599,  1.07783976, -1.078023
   78,
          0.71974486, -0.033623  ,  0.80733156, -0.06581134,  0.046288
   13,
         -0.39104057,  0.71350532,  0.12215295,  0.03023434,  0.505534
   66,
         -0.84311353,  0.03607727,  0.89929888,  1.1488878 , -0.337522
   64,
```

```
       -1.04855686,  0.47900072, -0.88404105, -1.49742756,  0.576570
15,
       -0.77682728,  0.43424601, -0.34209781,  1.57731845,  2.590991
86,
        0.61843805, -1.87139853, -1.14737307,  0.11826155, -2.008558
75,
       -1.31532117, -0.89508956, -1.41657898,  0.22234693, -0.408087
33,
       -0.77232336, -0.43137565, -0.09335466, -0.61951439, -0.574349
02,
       -0.86195049,  2.5824003 ,  2.29856769,  1.38228161, -0.120355
57,
        0.0163426 ,  1.37301397, -2.41954967,  2.21898328,  0.206999
36,
       -0.44426707, -0.28300455, -0.65090364,  0.56035625, -0.287316
44,
        0.21891034,  1.33869336, -0.14964721, -0.90614327, -2.242522
65,
       -0.20738837, -1.06225013,  0.12713545,  0.30906346,  1.432362
65,
       -0.72362812, -0.07515363,  0.89529865,  0.2722238 ,  0.553063
42,
       -0.74905904,  0.38863431, -1.0298591 ,  0.2133172 ,  1.684151
99,
       -0.44589025,  1.81781619,  0.51918159,  0.07778738,  1.145851
9 ,
       -1.03534791,  1.16038312,  0.56175461, -1.23505705,  0.579345
97,
        0.30825931, -1.12645226, -0.06608052,  0.22322695,  1.472233
96,
       -0.75841857, -0.41820033, -0.352581  ,  1.21246646,  1.630469
74,
        0.73486422,  0.65345383, -0.52145511,  0.97282551, -1.636688
,
        0.22328364, -1.54413184,  1.20611709, -0.70539712,  0.202031
5 ,
        1.45243195,  1.0273714 , -0.63884557, -0.19580227,  0.128399
04,
       -0.63746679,  0.08576558,  0.84450437,  2.23815724,  0.941033
6 ])
```

In [104]:

```
m0[:10]
```

Out[104]:

```
array([-0.08896297, -0.99628991, -0.03442457, -0.71752453, -0.498621
43,
        0.40981241,  0.48650734, -0.46942676,  0.27730296,  0.800987
16])
```

In [105]:

```
m0[-10:]
```

Out[105]:

```
array([ 1.45243195,  1.0273714 , -0.63884557, -0.19580227,  0.128399
04,
       -0.63746679,  0.08576558,  0.84450437,  2.23815724,  0.941033
6 ])
```

In [106]:

```
m0.min()
```

Out[106]:

```
-3.3966550790909436
```

In [107]:

```
m0.max()
```

Out[107]:

```
3.318944307848348
```

In [108]:

```
m0.mean()
```

Out[108]:

```
-0.015245030821984308
```

In [109]:

```
np.array([ 1.45243195,  1.0273714 , -0.63884557]) + 10
```

Out[109]:

```
array([11.45243195, 11.0273714 ,  9.36115443])
```

In [110]:

```
np.around([ 1.45243195,  1.0273714 , -0.63884557])
```

Out[110]:

```
array([ 1.,  1., -1.])
```

In [111]:

```
c = np.array([ 1.45243195,  1.0273714 , -0.63884557])
c
```

Out[111]:

```
array([ 1.45243195,  1.0273714 , -0.63884557])
```

In [112]:

```
d = c.flatten()
d
```

Out[112]:

```
array([ 1.45243195,  1.0273714 , -0.63884557])
```

In [113]:

```
w = np.array([5,6])
w
```

Out[113]:

```
array([5, 6])
```

In [114]:

```
v = np.array([1,2,3,4])
v
```

Out[114]:

```
array([1, 2, 3, 4])
```

In [116]:

```
z = np.concatenate((v,w), axis=0)
z
```

Out[116]:

```
array([1, 2, 3, 4, 5, 6])
```

In [117]:

```
# Reshape
a = np.arange(9)
a
```

Out[117]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [118]:

```
a2d = a.reshape(3,3)
a2d
```

Out[118]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [119]:

```
a = np.arange(10)
a
```

Out[119]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [120]:

```
a2d = a.reshape(5,2)
a2d
```

Out[120]:

```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

In [121]:

```
a2d = a.reshape(2,5)
a2d
```

Out[121]:

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [122]:

```
a2d.shape
```

Out[122]:

```
(2, 5)
```

In [ ]:

```

```