> ## Assignment 2 - Numpy Array Operations
>
> This assignment is part of the course "Data Analysis with Python: Zero to Pandas". The objective of this assignment is to develop a solid understanding of Numpy array operations. In this assignment you will:
>
> 1. Pick 5 interesting Numpy array functions by going through the documentation: https://numpy.org/doc/stable/reference/routines.html
> 2. Run and modify this Jupyter notebook to illustrate their usage (some explanation and 3 examples for each function). Use your imagination to come up with interesting and unique examples.
> 3. Upload this notebook to your Jovian profile using `jovian.commit` and make a submission here: https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/assignment-2-numpy-array-operations
> 4. (Optional) Share your notebook online (on Twitter, LinkedIn, Facebook) and on the community forum thread: https://jovian.ml/forum/t/assignment-2-numpy-array-operations-share-your-work/10575 .
> 5. (Optional) Check out the notebooks shared by other participants and give feedback & appreciation.
>
> The recommended way to run this notebook is to click the "Run" button at the top of this page, and select "Run on Binder". This will run the notebook on mybinder.org, a free online service for running Jupyter notebooks.
>
> Try to give your notebook a catchy title & subtitle e.g. "All about Numpy array operations", "5 Numpy functions you didn't know you needed", "A beginner's guide to broadcasting in Numpy", "Interesting ways to create Numpy arrays", "Trigonometic functions in Numpy", "How to use Python for Linear Algebra" etc.
>
> **NOTE**: Remove this block of explanation text before submitting or sharing your notebook online - to make it more presentable.

Data Analysis With Python: Zero To Pandas : Assignment 2

In this Assigment, we are going to learn numpy functions and its use cases with examples.

NumPy is one of the packages that everyone must be able to use and know if we want to do data science or data analysis with Python. It offers a great alternative to Python lists, as NumPy arrays are more compact, allow faster access in reading and writing items, and are more convenient and more efficient overall.

- Numpy zeros function
- Numpy add function
- Numpy concatenate function
- Numpy repeat function

- Numpy Statistical function like mean, median, mode

The recommended way to run this notebook is to click the "Run" button at the top of this page, and select "Run on Binder". This will run the notebook on mybinder.org, a free online service for running Jupyter notebooks.

In [1]:
```
!pip install jovian --upgrade -q
```

In [2]:
```
import jovian
```

In [3]:
```
jovian.commit(project='numpy-array-operations')
```

```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
```
Out[3]:
```
'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'
```

Let's begin by importing Numpy and listing out the functions covered in this notebook.

In [4]:
```
import numpy as np
```

List of functions explained

function1 = np.zeros() # (change this)

function2 = np.add()

function3 = np.concatenate()

function4 = np.repeat()

function5 = np.mean()

Function 1 - np.zeros (change this)

It will create an array with a given shape and filled it with zeroes

In [9]:
```
# Example 1 - working (change this)
arr1 = np.zeros(5)
arr1
```

Out[9]:
```
array([0., 0., 0., 0., 0.])
```

Here we see the example of how to create a numpy array by using `np.zeros` function

In [13]:
```
# Example 2 - working
arr2 = np.zeros((3,4))
arr2
```

Out[13]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Here we see how to create a zeros array by giving the shape of rows and columns

In [14]:
```
# Example 3 - breaking (to illustrate when it breaks)
np.zeros(3,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_37/327418347.py in <module>
      1 # Example 3 - breaking (to illustrate when it breaks)
----> 2 np.zeros(3,2)

TypeError: Cannot interpret '2' as a data type
```

In this example, we caught an error because it can only interpret 3 and not 2 so this code doesn't know how to create zeros array of 2x3 matrix so in order to fix this we have to put both the values or the dimensions inside a parenthesis or ().

In [15]:
```
#fixing issue
np.zeros((3,2))
```

Out[15]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

If we want to create an array and filled with zeros we can use numpy zeros functions.

In [16]:
```
jovian.commit()
```

```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
```

Out[16]:
```
'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'
```

Function 2 - np.add( )

This function is used when we want to compute the addition of two array.

In [27]:
```
# Example 1 - working
input1 = 23
input2 = 45
```

In [28]:
```
np.add(input1, input2)
```

Out[28]:
```
68
```

Here we have taken two values and add them by using `np.add` function

In [29]:
```
# Example 2 - working
arr2 = np.array([2,3,6])
arr3 = np.array([4,5,6])
```

In [30]:
```
np.add(arr2, arr3)
```

Out[30]:
```
array([ 6,  8, 12])
```

Here we have created two arrays and added them

In [35]:
```
# Example 3 - breaking (to illustrate when it breaks)
x1 = np.arange(9.0)
x2 = np.arange(3.0)

np.add(x1, x2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_37/3401778716.py in <module>
      3 x2 = np.arange(3.0)
      4
----> 5 np.add(x1, x2)

ValueError: operands could not be broadcast together with shapes (9,) (3,)
```

Here we can't add the arrays because of mismatch of shapes so in order to fix it we have to give an equal dimension shape to add the arrays.

In [36]:
```
#fixing issue
x1 = np.arange(9.0).reshape(3,3)
x2 = np.arange(3.0)

np.add(x1, x2)
```

Out[36]:
```
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
```

We can use this function if we want to perform addition between two values or two arrays

In [37]:
```
jovian.commit()
```

```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
```

Out[37]:
```
'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'
```

Function 3 - np.concatenate()

We can concatenate or add a sequence of array to an existing array

In [40]:
```
# Example 1 - working
arr1 = np.array([[2, 4], [6, 8]])
arr2 = np.array([[3, 5], [7, 9]])

arr3 = np.concatenate((arr1, arr2))
arr3
```

Out[40]:
```
array([[2, 4],
```

```
       [6, 8],
       [3, 5],
       [7, 9]])
```

Here we have created two array and then we concatenate two arrays and store the value in a new array `arr3`

In [41]:
```
# Example 2 - working
arr1 = np.array([[2, 4], [6, 8]])
arr2 = np.array([[3, 5], [7, 9]])

arr3 = np.concatenate((arr1, arr2), axis=1)
arr3
```

Out[41]:
```
array([[2, 4, 3, 5],
       [6, 8, 7, 9]])
```

Here we have concatenated two array and also we have given an `axis` which is so that the new array will give the results in a column format

In [42]:
```
# Example 3 - breaking (to illustrate when it breaks)
arr1 = np.array([[2, 4], [6, 8]])
arr2 = np.array([[3, 5], [7, 9]])

arr3 = np.concatenate(arr1, arr2)
arr3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_37/3912692189.py in <module>
      3 arr2 = np.array([[3, 5], [7, 9]])
      4
----> 5 arr3 = np.concatenate(arr1, arr2)
      6 arr3

<__array_function__ internals> in concatenate(*args, **kwargs)

TypeError: only integer scalar arrays can be converted to a scalar index
```

In this error we have to give the arguments inside a parenthesis

We can use this function when we want to concatenate an array with an existing array

In [43]:
```
jovian.commit()
```

Out[43]:
```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'
```

Function 4 - np.repeat()

This function repeats elements of the array

In [53]:
```
# Example 1 - working
arr1 = np.arange(9)
np.repeat(arr1, 1)
```

Out[53]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In this example we have repeated the elements in an array only one time

In [54]:
```
# Example 2 - working
arr1 = np.arange(9)
np.repeat(arr1, 2)
```

Out[54]:
```
array([0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8])
```

In this example we have repeated the elements in an array two times.

In [55]:
```
# Example 3 - breaking (to illustrate when it breaks)
arr1 = np.arange(9)
np.repeat(arr1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_37/663837236.py in <module>
      1 # Example 3 - breaking (to illustrate when it breaks)
      2 arr1 = np.arange(9)
----> 3 np.repeat(arr1)

<__array_function__ internals> in repeat(*args, **kwargs)

TypeError: _repeat_dispatcher() missing 1 required positional argument: 'repeats'
```

In this error what we can see that in order to repeat the elements of an array we have to pass two arguments or parameters one is the array name and other is the numerical value or an integer like how many times this element have to repeat.

Some closing comments about when to use this function.

In [56]:
```
jovian.commit()
```

Out[56]:
```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'
```

Function 5 - np.mean()

It's a statistical function to calculate the mean of an array

In [44]:
```
# Example 1 - working
arr1 = np.array([2,4,5,8,10,11,14,16])
np.mean(arr1)
```

Out[44]:
```
8.75
```

In this example we have created this array and we calculated the mean of its array by using `np.mean` function

In [47]:
```
# Example 2 - working
```

```
arr2 = [[14, 17, 12, 33, 44],
       [15, 6, 27, 8, 19],
       [23, 2, 54, 1, 4, ]]

np.mean((arr2), axis=1)
```

Out[47]:
```
array([24. , 15. , 16.8])
```

Here we have created a 2D array and then we have taken out the mean of axis 1 of the given array

In [49]:
```
# Example 3 - breaking (to illustrate when it breaks)
np.mean((arr1),axis=1)
```

```
---------------------------------------------------------------------------
AxisError                                 Traceback (most recent call last)
/tmp/ipykernel_37/2225889945.py in <module>
      1 # Example 3 - breaking (to illustrate when it breaks)
----> 2 np.mean((arr1),axis=1)

<__array_function__ internals> in mean(*args, **kwargs)

/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py in mean(a, axis, dt
ype, out, keepdims, where)
   3417             return mean(axis=axis, dtype=dtype, out=out, **kwargs)
   3418
-> 3419     return _methods._mean(a, axis=axis, dtype=dtype,
   3420                           out=out, **kwargs)
   3421

/opt/conda/lib/python3.9/site-packages/numpy/core/_methods.py in _mean(a, axis, dtyp
e, out, keepdims, where)
    164     is_float16_result = False
    165
--> 166     rcount = _count_reduce_items(arr, axis, keepdims=keepdims, where=where)
    167     if rcount == 0 if where is True else umr_any(rcount == 0, axis=None):
    168         warnings.warn("Mean of empty slice.", RuntimeWarning, stacklevel=2)

/opt/conda/lib/python3.9/site-packages/numpy/core/_methods.py in _count_reduce_items
(arr, axis, keepdims, where)
     73         items = nt.intp(1)
     74         for ax in axis:
---> 75             items *= arr.shape[mu.normalize_axis_index(ax, arr.ndim)]
     76     else:
     77         # TODO: Optimize case when `where` is broadcast along a non-reductio
n

AxisError: axis 1 is out of bounds for array of dimension 1
```

In this error we can see that we can't calculate the mean of one dimensional array or a vector by mentioning the axis=1, what we can do to calculate the mean of a vector is to ignore the axis or can write axis=None to resolve the issue.

If we want to calculate the mean or perform statistical operation then we can use this function.

In [50]:
```
jovian.commit()
```

```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
```

```
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
```

Out[50]: 'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'

Conclusion

In this notebook we have covered numpy function by creating some examples and practing the numpy functions to perform numerical operation is python without writing long codes or complex functions.

Reference Links

Provide links to your references and other interesting articles about Numpy arrays:

- Numpy official tutorial : https://numpy.org/doc/stable/user/quickstart.html
- Numpy video tutorial : https://youtu.be/d0E0_87CrFA

In [57]:
```
jovian.commit()
```

```
[jovian] Updating notebook "shahnawazmohammad446/numpy-array-operations" on https://
jovian.ai
[jovian] Committed successfully! https://jovian.ai/shahnawazmohammad446/numpy-array-
operations
```

Out[57]: 'https://jovian.ai/shahnawazmohammad446/numpy-array-operations'

In [ ]:
```

```