NumPy Package

Contents

- Array Basics
- List vs Array
- N- Dimensional Arrays
- Arrays from data frames
- Indexing & Slicing
- Boolean Index
- Initial Placeholders
- Random number generation
- Reshape
- Max and Min
- argmax()

NumPy

- Stands for Numerical Python
- NumPy helps us to create and work with arrays in python
- NumPy is for fast operations on vectors and matrices, including mathematical, logical, shape manipulation, sorting, selecting.
- It is the foundation on which all higher level tools for scientific python packages are built

NumPy

- How to define arrays?
- What is the function name?

NumPy

```
import numpy as np
income = np.array([1200, 1300, 1400, 1500, 1600, 1700])

type(income)

print(income[0])

expenses=income*0.653
print(expenses)

savings=income-expenses
print(savings)
```

List vs Array

- What is the difference ?

Array is different from a List

```
list1=[1,2,3]
list2=[4,5,6]

arr1=np.array([1,2,3])
arr2=np.array([4,5,6])
```

What is the expected output of the below code?

```
list1+list2


arr1+arr2
```

Elements are Homogenous inside an array

```
c = np.array([[1, 2,3], [4, 'a', 6]])
print(c)
```

- What type of object is c?

N- Dimensional Arrays

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

b=np.array([[1, 2, 3], [4, 5, 6], [9, 9, 9]])
print(b)
```

Shape of array

- How to get the shape of arrays?
- How to extract only the number of dimensions?
- How to extract the total number of elements across all the dimensions?

Shape, Size and ndim

What is the expected output of the below code?

```
print("Shape", a.shape)
print("Size", a.size)
print("ndim", a.ndim)

print("Shape", b.shape)
print("Size", b.size)
print("ndim", b.ndim)
```

LAB : Basics of Array

- Create 4 numpy arrays with numbers(list values) of your choice
  - arr0 = 0d array (scalar array)
  - arr1 = 1d array
  - arr2 = 2d array
  - arr3 = 3d array
- Get shape, size and dimension of all these arrays, use:
  - .shape()
  - .size()
  - .ndim()

Arrays from data frames

```
import pandas as pd
bank=pd.read_csv("https://raw.githubusercontent.com/venkatareddykonasani/Datasets/master/Bank%20Tele%20Marketing/bank_market.csv")

bank.info()

age_var=np.array(bank["age"])
type(age_var)
age_var.shape
```

Arrays from data frames

```
two_vars=np.array(bank[["age","balance"]])
type(two_vars)

two_vars.shape

two_vars.size
```

Indexing & Slicing

- How to access only first 10 elements of an array ?
- How to access only the last element of an array?
- How to access the elements at indexes 1, 9, 10

Indexing & Slicing

```
age_var=np.array(bank["age"])
age_var
```

- What is the expected output of the below code?

```
age_var[0]
age_var[0:10]
age_var[-1]
age_var[[1,9,10]]
```

Indexing & Slicing

- Take a two dimensional array, how to access the first row, first column?
- What is the output for the index notation `[0:2,1]`

Indexing & Slicing

```
two_vars=np.array(bank[["age","balance"]])
two_vars
```

What is the expected output of the below code?

```
two_vars[0,0]
two_vars[0,1]
two_vars[0:2,1]
two_vars[0:2,0]
two_vars[0:2,2]
two_vars[-1]
two_vars[-1, 0:2]
two_vars[-2, 0:2]
two_vars[:, 0]
two_vars[:, 1]
two_vars[0, :]
```

Boolean index

- How to filter select the values of an array based on a condition?

Boolean index

```
age_var=np.array(bank["age"])
age_var

condition=age_var<50
condition

new_age=age_var[condition]

print(age_var.shape)
print(new_age.shape)

#Mark age_var as 1 if condition is met
age_var[condition]=1
age_var
```

LAB : Accessing Arrays

- Create a 2d array name it ‘a’, with shape (3,4)
- Slice it such as:
  - middle two values of the first two rows are selected.
  - Store this slice as array **‘b’**
- In array ‘b’ change the value of first element
  - Hint: b[0,0]
- Print the Array ‘a’ again. What do you observe?
- Create ‘b’ again, but this time use np.array(a[0:2, 1:3], copy=True) option
- Update ‘b’ and see whether the original array sis updated or not.

Initial Placeholders

- Create a 2dim array with 3rows and 4 columns and fill all the elements with zeros
- Create an array by taking the numbers between 10, 30. Keep the step size as 2.
- Create an array by dividing the space between 10 and 30 into 5 parts.

Initial Placeholders

```
np.zeros((3,4))
np.ones((2,3),dtype=np.int16)

np.arange(10,30,2)
np.arange(10,30,5)
np.arange(10,30,10)

np.linspace(10,30,2)
np.linspace(10,30,10)
np.linspace(10,30,20)
```

LAB: Random number generation

- Generate a random number using NumPy
- Use NumPy and generate 30 random numbers
- Use NumPy and generate 30 uniformly distributed random numbers
- Use NumPy and generate 30 normally distributed random numbers
- Generate a random number matrix with 2 rows and 3cols using NumPy

Random number generation

```
np.random.random(1)
np.random.random(30)
np.random.uniform(size=30)
np.random.normal(size=30)
np.random.random((2,3))
```

numpy.reshape()

```
a=np.random.uniform(size=30)
```

Can you re-shape the above array as a 2D array with 6 rows and five cols?

numpy.reshape()

```
a=np.random.uniform(size=30)
```

Can you re-shape the above array as a 2D array with 6 rows and five cols?

```
a.reshape(6,5)
```

numpy.reshape()

- reshape() doesn’t change the shape - make a note of it.

```
print(a.shape)
print(a.reshape(6,5).shape)
```

numpy.reshape()

- Reshaping as a 3D array

```
a.reshape(3,2,5)
```

numpy.reshape()

- What if, we give wrong dimensions

```
a=np.random.uniform(size=30)
a.reshape(3,1)
```

numpy.reshape()

- What if, we give wrong dimensions

```
a=np.random.uniform(size=30)
a.reshape(3,1)
```

- What if, we want 3 rows and any number of columns.

numpy.reshape()

- You can use negative index for unknown dimension.

```
a.reshape(3,-1)
a.reshape(-1,3)
a.reshape(3,2,-1)
a.reshape(-1,2,3)
a.reshape(-1,2,15)
```

- You can only specify one unknown dimension

```
a.reshape(-1,-1,15)
```

- Flatten the array to one row

```
a.flatten()
```

max and min other functions

```
age_var=np.array(bank["age"])
age_var.max()
age_var.min()
age_var.mean()
age_var.std()
```

Index of max

- Consider this example
- Let this be output probabilities for multiclass classification output for 15 datapoints.
- We need to give only one class as output, the class with max probability.
- How to get the index of the max element?

```
rray([[0.58178245, 0.00234469, 0.97036937, 0.64034516],
      [0.44504406, 0.07178624, 0.50511309, 0.98527334],
      [0.40490749, 0.21520268, 0.66671445, 0.18926015],
      [0.99818906, 0.3702341 , 0.32152925, 0.33452479],
      [0.41693608, 0.99710111, 0.54760253, 0.98896868],
      [0.43080255, 0.6232379 , 0.60616554, 0.41871962],
      [0.57980182, 0.30218979, 0.48831486, 0.17218716],
      [0.38477543, 0.40937626, 0.60831249, 0.23314077],
      [0.24803288, 0.13615116, 0.38076504, 0.80648948],
      [0.64015809, 0.11270068, 0.67419178, 0.63834555],
      [0.0711749 , 0.72234198, 0.83176517, 0.26625898],
      [0.92572131, 0.31060026, 0.39069662, 0.72056121],
      [0.76615175, 0.75503287, 0.57738505, 0.3122232 ],
      [0.67315251, 0.4502434 , 0.64605349, 0.47127994],
      [0.08272156, 0.53467679, 0.29487162, 0.16681734]])
```

numpy.argmax()

- The numpy.argmax() function returns indices of the max element of the array in a particular axis.

```
output_prob.argmax(axis=1)
```

```
rray([[0.58178245, 0.00234469, 0.97036937, 0.64034516],
      [0.44504406, 0.07178624, 0.50511309, 0.98527334],
      [0.40490749, 0.21520268, 0.66671445, 0.18926015],
      [0.99818906, 0.3702341 , 0.32152925, 0.33452479],
      [0.41693608, 0.99710111, 0.54760253, 0.98896868],
      [0.43080255, 0.6232379 , 0.60616554, 0.41871962],
      [0.57980182, 0.30218979, 0.48831486, 0.17218716],
      [0.38477543, 0.40937626, 0.60831249, 0.23314077],
      [0.24803288, 0.13615116, 0.38076504, 0.80648948],
      [0.64015809, 0.11270068, 0.67419178, 0.63834555],
      [0.0711749 , 0.72234198, 0.83176517, 0.26625898],
      [0.92572131, 0.31060026, 0.39069662, 0.72056121],
      [0.76615175, 0.75503287, 0.57738505, 0.3122232 ],
      [0.67315251, 0.4502434 , 0.64605349, 0.47127994],
      [0.08272156, 0.53467679, 0.29487162, 0.16681734]])
```

LAB : Placeholders and Simple Functions

- Create 1d array of 50 values between 0-1, with equidistance.
- Reshape it to (5,10) – name it array ‘a’
- Generate random sample with 50 values and reshape and multiply it with array 'a'.
- What is the location of max argument in each row?
- What is the min value in each column?

Conclusion

- Here we have discussed some of the most widely used functions and commands.
- There are many more functions and operations available in NumPy