numpy

github.com/SagarChhabriya

1. Creating Arrays

- np.array(param)
- np.arange(start, stop, step)
- np.linspace(start, stop, num)
- np.arange(4).reshape(2,2) → 2 x 2
  np.arange(4).reshape(3, 2,2) → 3 x (2 x 2)
  np.arange(4).reshape(2,2,2,2) →2 x (2 x(2 x 2))
- np.ones(scalar/tuple)
  np.zeros(scalar/tuple)
  np.random.random(scalar/tuple)
  np.concatenate(arrays_tuple)
- np.identity( ), np.eye( ), np.diag( )

2. Array Attributes

- shape, size, dtype, ndim, itemsize

3. Array Operations

- Scalar: arr + 2, arr * 3
- Relational: arr > 5
- Vector: operations b/w arrays
  e.g., arr1 + arr2
- Arithmetic: +, -, *, /, //, **

4. Array Functions

- np.random.random((a,b))

  # random values-> 7 item array from 10 to 50
  np.random.randint(10, 50, 7)

  # multi dim array
  np.random.randint(10, 50, size=(2, 3))

  np.round(arr * 100)
- arr.mean( ), arr.median( ), arr.std( ), arr.var( )
- arr.max( ), arr.min( ), arr.sum( ),
  arr.prod(arr, axis=0), axis 0: col, axis 1: row
- trigonometric, log, exponents, dotproduct, round, floor, ceil
- arr.astype(np.int32)
-

5. Indexing and Slicing

Indexing

1D array: array[index]
2D array: array[row_index, col_index]

- N-dimensional array: array[indices]
  (where indices can be tuples of multiple indices

  Boolean Indexing: arr[condition & condition...]
  Integer Array Indexing: arr[int_list]
  Fancy Indexing: arr[[list rows], [list cols]]
- Slicing
  array[start:stop:step]
  Slicing with ellipsis (...)

6. Reshaping

- arr.reshape(x, y)
  np.transpose(arr)
  arr.ravel( ) ---> Flatten
- Stacking
  np.hstack((arr1,arr2))
  np.vstack((arr1,arr2))
- Splitting
  np.vsplit(arr, num)
  np.hsplit(arr, num)
- Adding and Deleting Elements

  np.append(existing_arr, [elements])
  np.insert(existing_arr, index, [element])
  np.delete(existing_arr, [indicies])

MISC

- Brodacasting
- Ufuncs

  np.sqrt(arr)
  np.exp(arr)
- Linear Algebra

  np.dot(arr1, arr2)
  np.matmul(matrix1, matrix2)

  Eigenvalues
  eig_values, eig_vectors = np.linalg.eig(matrix1)

  np.corrcoef(data1, data2)
  np.cov(data1, data2)
- Searching

  np.where(condition, true, false)
  np.unique(arr)
- np.savetxt('array.txt', array1)
- np.expand_dim(arr, axis=0/1)
  np.argmax(array, axis)
  np.cumsum(arr, axis)
  np.percentile(arr, value)
  np.median(arr, axis)
  np.histogram(arr, bins=[0,50,100 etc]
  np.flip(arr)
  np.isin(arr1, arr2) --> Intersection

  Set Functions
  np.union1d(arr1, arr2)
  np.intersect1d(arr1, arr2)
  np.setdiff1d(arr1, arr2)
  np.setxor1d(arr1, arr2)
  np.in1d(arr, targe_element)
- np.clip(arr, a_min = 25, a_max=75)
  values <25 will be replaced with 25
  values >75 will be replaced with 75
  otherwise remain same

  np.swapaxes()
  np.uniform( )
  np.count_nonzero( )
  np.tile( )
  np.allclose( )
  np.repeat( )
  np.equals( )

Advanced Array Manipulation

- Structured Arrays
- Record Arrays
- Performance Optimization

  Memory Layout
  array = np.arange(10)
  print("Memory layout of array:", array.flags)

  # Vectorization for Performance
  array1 = np.random.rand(1000000)
  array2 = np.random.rand(1000000)
  result = np.dot(array1, array2)
  print("Dot product result:", result)
- # Using `numba` with NumPy
  `numba` can be used to compile Python code into fast machine code for performance.

  # Example: Using `numba` (requires installation of numba)

  from numba import jit
  @jit(nopython=True)
  def compute_sum(array):
  total = 0
  for value in array:
  total += value
  return total

  large_array = np.random.rand(1000000)
  print("Sum of large array using numba:", compute_sum(large_array))
- # 3. Integration with Other Libraries
  # Integration with Pandas
  # Integration with SciPy
  # Integration with Matplotlib
- # 4. Handling Missing Data

  # Masked Arrays
  # Handling NaNs
- # 5. Input/Output Operations

  # Reading and Writing Files
  np.savetxt('array.txt', data)
  loaded_array_text = np.loadtxt('array.txt')
  print("Loaded array from text file:", loaded_array_text)

  # Loading Data from Text and CSV Files
  np.save('array_binary.npy', data)
  loaded_array_binary = np.load('array_binary.npy')
  print("Loaded binary array:", loaded_array_binary)