

# Numpy

Yankun (Alex) Meng

# Lecture Outline

- **Introduction and Basics**
- Array Manipulation
- Calculus / Statistical Functions

# What is NumPy?

NumPy is a package for dealing with arrays that's very fast and efficient.

Creating array:

```
1 arr1 = np.array([3, 5, 7, 3]) # convert list to array
2 arr2 = np.zeros(10) # array with 10 zeroes
3 arr3 = np.ones(10) # array with 10 ones
4 arrR = np.random.random(10) # sample 10 numbers from uniform distribution
5 arrG = np.random.randn(10) # sample 10 numbers from normal distribution
```

# Two most common functions

**linspace** – values equally spaced apart given the number of values

```
1 arr = np.linspace(0, 10, 100) # from 0 to 10, 100 values
```

**arrange** – values equally spaced apart given the spacing

```
1 arr = np.arange(0, 10, 0.02) # from 0 to 10, spacing of 0.02
```

# Lecture Outline

- Introduction and Basics
- **Array Manipulation**
- Calculus / Statistical Functions

# Common Array Operation

## Element-wise operation

```
1 arr + 2      # addition
2 arr - 5      # subtraction
3 2*arr        # multiplication
4 1/arr        # division
5 arr % 5      # Modulus
```

Let's try to do the same element-wise operation but using functions:

```
1 def f(x):
2     return x**2 * np.sin(x) / np.exp(-x)
3
4 x = np.linspace(0, 10, 100)
5 y = f(x) # Element-wise application of f
```

# Common Plotting

## Most basic Plotting

```
1 x = np.linspace(0, 1, 100) # x-axis definition
2 y = x**2                  # element-wise x^2
3 plt.plot(x, x**2)         # plot(x, y)
4 plt.show()                # show the output
```

## Histogram

```
1 plt.hist(arr) # plotting the array as histogram
```

# Indexing and Slicing

## Normal Indexing

```
1 arr = np.array([2, 4, 6, 8, 10])
2 arr[2] # return 6
3 arr[2:] # return 6, 8, 10 (including index 2)
4 arr[:-2] # return 2, 4, 6 (excluding index -2)
5 arr[1:2] # return 4
```

## Boolean Indexing

```
1 arr > 5 # return a boolean array with true or false as elements
2 arr[arr>5] # filtering → return the elements that are true
```

## Using Boolean Indexing to filter data

```
1 names = np.array(['Jim', 'Luke', 'Josh', 'Pete'])
2 first_letter_j = np.vectorize(lambda s: s[0])(names) == 'J'
3 print(names[first_letter_j]) # gets the names that have first letter as J
```

Now what is this doing?



# What is that doing?

```
1 first_letter_j = np.vectorize(lambda s: s[0])(names) == 'J'
```

## Lambda Function

```
1 f = lambda s: s[0]
```

This is a function that says give me a string and I'll give you the first character of the string. Note that `String` is not an array in python, but it supports the same indexing operations as arrays.

## What is `np.vectorize`?

It creates a for-loop, applies the function `func` to every element in the array `arr`, and return the result as an array.

```
1 np.vectorize(func)(arr) # this is an array
```

# Lecture Outline

- Introduction and Basics
- Array Manipulation
- **Calculus / Statistical Functions**

# Common Statistics Functions

```
1 arr = 2*np.random.randn(10000) + 10 # 2 times std dev with a shift of 10 to the right\  
2 np.mean(arr) # approx 10  
3 np.std(arr) # approx 2  
4 np.percentile(arr, 80) # 80% numbers less than this number
```

# Integrals and Derivatives

```
1 x = np.linspace(1, 10, 100)
2 y = 1/x**2 + np.sin(x)
3 plt.plot(x, y)
```

## Derivatives

```
1 dydx = np.gradient(y, x)
```

## Integrals

```
1 np.cumsum([1, 2, 3, 4]) # return [1,3,6,10]
2 y_int = np.cumsum(y) * (x[1] - x[0]) # int y dx
```

# Resources Used

[1] NumPy Tutorial [2] 100 numpy practice problems