Numpy Indexing & Slicing

-MUKESH KUMAR

INDEXING

Indexing 1D array

- +ve indices > 0 to length of array
- -ve indices > -1 to beginning of array

```
# Creating a 1D array
array_1d = np.array([10, 20, 30, 40, 50])

# Accessing elements
print(array_1d[0])  # Output: 10
print(array_1d[2])  # Output: 30

# Negative indexing (from the end)
print(array_1d[-1])  # Output: 50
print(array_1d[-3])  # Output: 30
```

Indexing 2D Array

- Accessing rows and individual elements

```
# Creating a 2D array
array_2d = np.array([[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]])

# Accessing elements
print(array_2d[0, 0])  # Output: 1 (first row, first column)
print(array_2d[1, 2])  # Output: 6 (second row, third column)

# Accessing a full row or column
print(array_2d[1])  # Output: [4 5 6] (entire second row)
print(array_2d[2])  # Output: [3 6 9] (entire third row)
```

5x5 Matrix

(row,col)

columns

rows

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

SLICING

Slicing 1D array

- The basic slice syntax is :

i:j:k

  - where *i* is the starting index,
  - *j* is the stopping index,
  - and *k* is the step (k≠0)

```
>>> x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> x[1:7:2]
array([1, 3, 5])
```

i not given

- Defaults to 0

```
>>> x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
>>> x[5:]
array([5, 6, 7, 8, 9])
```

j not given

- Defaults to n, where n is number of elements

```
>>> x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

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

```
arr[:5]
```

```
array([0, 1, 2, 3, 4])
```

2D Slicing

- Mat[row_start : row_end : step , col_start : col_end : step]

- Fetch alternate rows
- Fetch alternate cols
- Fetch corner elements

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

Boolean array indexing

- Boolean array indexing lets you pick out arbitrary elements of an array. Frequently this type of indexing is used to select the elements of an array that satisfy some condition. Here is an example:

```
a = np.arange(1,11)
a
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
a[[True,True,True,False,False,False,False,False,False,False]]
```

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

```
a > 4
```

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

Fancy slicing

- Fancy indexing is conceptually simple: it means passing an array of indices to access multiple array elements at once. For example, consider the following array.