Accessing, Deleting, and Inserting Elements Into ndarrays

We strongly encourage you to type the commands that you have learned in this demo. However, the notebook file demonstrated in the video above is available at the bottom of this page.

Now that you know how to create a variety of ndarrays, we will now see how NumPy allows us to effectively manipulate the data within the ndarrays. NumPy ndarrays are mutable, meaning that the elements in ndarrays can be changed after the ndarray has been created. NumPy ndarrays can also be sliced, which means that ndarrays can be split in many different ways. This allows us, for example, to retrieve any subset of the ndarray that we want. Often in Machine Learning you will use slicing to separate data, as for example when dividing a data set into training, cross validation, and testing sets.

We will start by looking at how the elements of an ndarray can be accessed or modified by indexing. Elements can be accessed using indices inside square brackets, [ ]. NumPy allows you to use both positive and negative indices to access elements in the ndarray. Positive indices are used to access elements from the beginning of the array, while negative indices are used to access elements from the end of the array. Let's see how we can access elements in rank 1 ndarrays:

Example 1. Access individual elements of 1-D array

```
# We create a rank 1 ndarray that contains integers from 1 to 5

x = np.array([1, 2, 3, 4, 5])


# We print x

print()

print('x = ', x)

print()


# Let's access some elements with positive indices

print('This is First Element in x:', x[0])

print('This is Second Element in x:', x[1])

print('This is Fifth (Last) Element in x:', x[4])

print()


# Let's access the same elements with negative indices

print('This is First Element in x:', x[-5])

print('This is Second Element in x:', x[-4])
```

```
print('This is Fifth (Last) Element in x:', x[-1])
```

```
x = [1 2 3 4 5]

This is First Element in x: 1
This is Second Element in x: 2
This is Fifth (Last) Element in x: 5

This is First Element in x: 1
This is Second Element in x: 2
This is Fifth (Last) Element in x: 5
```

Notice that to access the first element in the ndarray we have to use the index 0 not 1. Also notice, that the same element can be accessed using both positive and negative indices. As mentioned earlier, positive indices are used to access elements from the beginning of the array, while negative indices are used to access elements from the end of the array.

Now let's see how we can change the elements in rank 1 ndarrays. We do this by accessing the element we want to change and then using the = sign to assign the new value:

Example 2. Modify an element of 1-D array

```
# We create a rank 1 ndarray that contains integers from 1 to 5

x = np.array([1, 2, 3, 4, 5])


# We print the original x

print()

print('Original:\n x = ', x)

print()


# We change the fourth element in x from 4 to 20

x[3] = 20


# We print x after it was modified

print('Modified:\n x = ', x)
```

```
Original:
x = [1 2 3 4 5]

Modified:
x = [ 1 2 3 20 5]
```

Similarly, we can also access and modify specific elements of rank 2 ndarrays. To access elements in rank 2 ndarrays we need to provide 2 indices in the form [row, column]. Let's see some examples.

Example 3. Access individual elements of 2-D array

```
# We create a 3 x 3 rank 2 ndarray that contains integers from 1 to 9

X = np.array([[1,2,3],[4,5,6],[7,8,9]])


# We print X

print()

print('X = \n', X)

print()


# Let's access some elements in X

print('This is (0,0) Element in X:', X[0,0])

print('This is (0,1) Element in X:', X[0,1])

print('This is (2,2) Element in X:', X[2,2])
```

```
X =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

This is (0,0) Element in X: 1
This is (0,1) Element in X: 2
This is (2,2) Element in X: 9
```

Remember that the index [0, 0] refers to the element in the first row, first column.

Elements in rank 2 ndarrays can be modified in the same way as with rank 1 ndarrays. Let's see an example:

Example 4. Modify an element of 2-D array

```
# We create a 3 x 3 rank 2 ndarray that contains integers from 1 to 9

X = np.array([[1,2,3],[4,5,6],[7,8,9]])


# We print the original x

print()
```

```
print('Original:\n X = \n', X)

print()


# We change the (0,0) element in X from 1 to 20

X[0,0] = 20


# We print X after it was modified

print('Modified:\n X = \n', X)
```

```
Original:
X =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Modified:
X =
[[20 2 3]
 [ 4 5 6]
 [ 7 8 9]]
```

Now, let's take a look at how we can add and delete elements from ndarrays. We can delete elements using the np.delete(ndarray, elements, axis) function. This function deletes the given list of elements from the given ndarray along the specified axis. For rank 1 ndarrays the axis keyword is not required. For rank 2 ndarrays, axis = 0 is used to select *rows*, and axis = 1 is used to select *columns*. Let's see some examples:

Example 5. Delete elements

```
# We create a rank 1 ndarray

x = np.array([1, 2, 3, 4, 5])


# We create a rank 2 ndarray

Y = np.array([[1,2,3],[4,5,6],[7,8,9]])


# We print x

print()

print('Original x = ', x)
```

```
# We delete the first and last element of x
x = np.delete(x, [0,4])

# We print x with the first and last element deleted
print()
print('Modified x = ', x)

# We print Y
print()
print('Original Y = \n', Y)

# We delete the first row of y
w = np.delete(Y, 0, axis=0)

# We delete the first and last column of y
v = np.delete(Y, [0,2], axis=1)

# We print w
print()
print('w = \n', w)

# We print v
print()
print('v = \n', v)
```

```
Original x = [1 2 3 4 5]
Modified x = [2 3 4]
Original Y =
[[1 2 3]
```

```
 [4 5 6]
 [7 8 9]]

w =
[[4 5 6]
 [7 8 9]]

v =
[[2]
 [5]
 [8]]
```

numpy.append

Syntax:

```
numpy.append(array, values, axis=None)
```

It appends values to the end of an array. Refer here for more details about additional arguments.

Now, let's see how we can append values to ndarrays. We can append values to ndarrays using the np.append(ndarray, elements, axis) function. This function appends the given list of elements to ndarray along the specified axis. Let's see some examples:

Example 6. Append elements

```
# We create a rank 1 ndarray

x = np.array([1, 2, 3, 4, 5])


# We create a rank 2 ndarray

Y = np.array([[1,2,3],[4,5,6]])


# We print x

print()

print('Original x = ', x)


# We append the integer 6 to x

x = np.append(x, 6)


# We print x
```

```
print()
print('x = ', x)

# We append the integer 7 and 8 to x
x = np.append(x, [7,8])

# We print x
print()
print('x = ', x)

# We print Y
print()
print('Original Y = \n', Y)

# We append a new row containing 7,8,9 to y
v = np.append(Y, [[7,8,9]], axis=0)

# We append a new column containing 9 and 10 to y
q = np.append(Y,[[9],[10]], axis=1)

# We print v
print()
print('v = \n', v)

# We print q
print()
print('q = \n', q)
```

```
Original x = [1 2 3 4 5]
x = [1 2 3 4 5 6]
```

```
x = [1 2 3 4 5 6 7 8]

Original Y =
[[1 2 3]
 [4 5 6]]

v =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

q =
[[ 1 2 3 9]
 [ 4 5 6 10]]
```

Notice that when appending rows or columns to rank 2 ndarrays the rows or columns must have the correct shape, so as to match the shape of the rank 2 ndarray.

Now let's see now how we can insert values to ndarrays. We can insert values to ndarrays using the np.insert(ndarray, index, elements, axis) function. This function inserts the given list of elements to ndarray right before the given index along the specified axis. Let's see some examples:

Example 7. Insert elements

```
# We create a rank 1 ndarray

x = np.array([1, 2, 5, 6, 7])


# We create a rank 2 ndarray

Y = np.array([[1,2,3],[7,8,9]])


# We print x

print()

print('Original x = ', x)


# We insert the integer 3 and 4 between 2 and 5 in x.

x = np.insert(x,2,[3,4])


# We print x with the inserted elements

print()
```

```
print('x = ', x)

# We print Y
print()
print('Original Y = \n', Y)

# We insert a row between the first and last row of y
w = np.insert(Y,1,[4,5,6],axis=0)

# We insert a column full of 5s between the first and second column of y
v = np.insert(Y,1,5, axis=1)

# We print w
print()
print('w = \n', w)

# We print v
print()
print('v = \n', v)
```

```
Original x = [1 2 5 6 7]
x = [1 2 3 4 5 6 7]
Original Y =
[[1 2 3]
 [7 8 9]]
w =
[[1 2 3]
 [4 5 6]
 [7 8 9]]
v =
[[1 5 2 3]
 [7 5 8 9]]
```

numpy.hstack and numpy.vstack

Syntax:

```
numpy.hstack(sequence_of_ndarray)
```

It returns a stacked array formed by stacking the given arrays in sequence horizontally (column-wise). See the in-depth details here.

```
numpy.vstack(sequence_of_ndarray)
```

It returns a stacked array formed by stacking the given arrays, will be at least 2-D, in sequence vertically (row-wise). See the in-depth details here.

NumPy also allows us to stack ndarrays on top of each other, or to stack them side by side. The stacking is done using either the np.vstack() function for vertical stacking, or the np.hstack() function for horizontal stacking. It is important to note that in order to stack ndarrays, the shape of the ndarrays must match. Let's see some examples:

Example 8. Stack arrays

```
# We create a rank 1 ndarray
x = np.array([1,2])


# We create a rank 2 ndarray
Y = np.array([[3,4],[5,6]])


# We print x
print()
print('x = ', x)


# We print Y
print()
print('Y = \n', Y)


# We stack x on top of Y
z = np.vstack((x,Y))
```

```
# We stack x on the right of Y. We need to reshape x in order to stack it on the right of Y.
w = np.hstack((Y,x.reshape(2,1)))


# We print z
print()
print('z = \n', z)


# We print w
print()
print('w = \n', w)
```

```
x = [1 2]

Y =
[[3 4]
 [5 6]]

z =
[[1 2]
 [3 4]
 [5 6]]

w =
[[3 4 1]
 [5 6 2]]
```

Supporting Materials

- Accessing, Deleting, and Inserting Elements Into ndarrays