Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

1.What is NumPy's array class called? Use an example to explain how to create one.

It's called the ndarray. It's n dimensions and N length.

```
import numpy as np

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

print(type(arr))
```

This outputs `<class 'numpy.ndarray'>`

2. Use an example to explain operations along different axis for a NumPy array.

Axes are the cartesian measurements of an array. Axis 0 traverses the columns in an array and axis 1 traverses the rows in an array.

```
1  import numpy as np
2
3  arr = np.arange(10).reshape([2, 5])
4
5  print(f'Arr[0]: {arr[0]}\nArr[1]: {arr[1]}')
6
7  print('Sum Cols: ', np.sum(arr, axis=0)) #sum columns
8  print('Sum Rows: ', np.sum(arr, axis=1)) #sum rows
```

```
Sum Rows:  [10 35]
```

In this example, performing sum along axis=0 will traverse each column and append the result to a list. This would be equivalent to running a nested for loop:

```
arr2 = [0] * len(arr[0])

for i in arr:

    for j in range(len(i)):

        arr2[j] += i[j]
```

Performing with axis=1 would be equivalent to adding each element for i in arr: arr2.append(sum(i)).

Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

3.Use a 3D NumPy array to explain how indexing, slicing as well as three dots work.

```
import numpy as np
threeDarray = np.arange(45).reshape(3,3,5)
print(threeDarray)
print()
#Indexing works by putting in array[2D array number][row][column]
print(threeDarray[2][1][1])
print()
#Slicing works by array[starting array: ending array, starting row:ending row, starting column: ending column]
print(threeDarray[1:, 0:2, 1:4])
print()
print()
#Three dots work by using all other values in stead of specifying for slicing. Notice that the output is the third column
#on each row of each array.
print (threeDarray[..., 2])
```

```
In [22]: runfile('C:/User
[[[ 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]
  [25 26 27 28 29]]

 [[30 31 32 33 34]
  [35 36 37 38 39]
  [40 41 42 43 44]]]

36

[[[16 17 18]
  [21 22 23]]

 [[31 32 33]
  [36 37 38]]]


[[ 2  7 12]
 [17 22 27]
 [32 37 42]]

In [23]:
```

4. Use an example to explain how to use ravel(), reshape() and resize() on a 3D NumPy array.

Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

```
10
11  print(threeDarray)
12  print()
13  #ravel simply flattens out the array
14  print(threeDarray.ravel())
15  print()
16  #reshape returns the given array resized in a (row number, column number) format
17  print(threeDarray.reshape(9,5))
18  print()
19  #Resize is very similar to reshape, but it actually changes the array to be the new size
20  #It is equivant to array = array.reshape()
21  threeDarray.resize(9,5)
22  print(threeDarray)
```

```
In [32]: runfile('C:/Users/Matt/Documents/Programming 2/Q4/untitled0.py', wo
[[[ 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]
  [25 26 27 28 29]]

 [[30 31 32 33 34]
  [35 36 37 38 39]
  [40 41 42 43 44]]]

[ 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 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44]

[[ 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]
 [25 26 27 28 29]
 [30 31 32 33 34]
 [35 36 37 38 39]
 [40 41 42 43 44]]

[[ 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]
 [25 26 27 28 29]
 [30 31 32 33 34]
 [35 36 37 38 39]
 [40 41 42 43 44]]

In [33]:
```

5. Use an example to explain how vstack(), hstack(), column_stack()and row_stack() work on a NumPy array.

Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

vstack(): vstack stands for vertical stack for example having 2 arrays a and b

a = ([[9., 7.], [5., 2.]])

b = ([[1., 9.], [5., 1.]])

np.vstack((a, b)

Would stack it like this

([[9., 7.],

[5., 2.],

[1., 9.],

[5., 1.]])

hstack(): hstack stands for horizontal stack and having 2 arrays a and b as before and hstacking would make it like this

([[9., 7., 1., 9.],

[5., 2., 5., 1.]])

column_stack(): column stack is just like hstack, but for 2D arrays and example

```
np.column_stack((a, b))  # returns a 2D array
array([[4., 3.],
       [2., 8.]])
```

row_stack(): row_stack is equivalent to vstack for any input arrays. In fact, row_stack is an alias for vstack

## 6. Use an example to explain how to use fancy indexing and indexing tricks to extract a region from a NumPy array, rearrange rows or columns in an extracted region as well as how to expand the dimension using original array's rows or columns.

Fancy indexing is simply extracting certain elements from an array. With an array a = [14, 2, 5, 22, 16, 7] we can use a[1] to get 2 from the array.

Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

We can also do x = [2,3] and use a[x] to get [2,5].

Or x = np.array([1,2],[5,6]) and for a[x] we would get array([14,2],[16,7]) which is not a 2D array.

## 7. Use an example to explain how to use NumPy's argmax().

```
import numpy as geek

array = geek.arange(15).reshape(3, 5)

print("ARRAY: \n", array)

print("Max element = ", geek.argmax(array))

print("Max element indicies = ", geek.argmax(array, axis=0))

print("Max element indicies = ", geek.argmax(array, axis=1))
```

```
ARRAY:
 [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
Max element =  14
Max element indicies =  [2 2 2 2 2]
Max element indicies =  [4 4 4]
```

## 8. Use an example to explain how to do indexing with Boolean array in NumPy.

```
import numpy as np

#boolean array

a1 = np.array([5, 50, 500, 55, 555])
```

Ryan Custard
Jon Harsy
Sean Poston
Timothy Schlottman

#print the indexes of the array greater than 55

print(a1[a1>55])

```
[500 555]
```