Arrays

- 1-dimensional vs. multidimensional
- Commands
- Some applications

Arrays

An array is an arrangement of objects, which could be numbers, letters, ordered pairs, etc.

The array could be in row or column or matrix form.

Arrays

The ordered pairs (3, 5) (2, -4) and (-3, 0) could be entered as…

…a list, called a 1-dimensional array:

```
S = {(3, 5), (2, -4), (-3, 0)}
```

or a multidimensional array (2D in this case):

```
A = [3 5; 2 -4; -3 0]
```

Note Julia's different feedback for each one. Although they look similar, they are treated very differently by computers.

Commands with 1D Arrays

`sort(S)`	puts the elements of S in order
`S[2]`	returns the 2nd element of S
`findfirst(S,(-3,0))`	tells which element is (-3,0)
`S[end]`	returns last element
`push!(S, (2, -5))`	adds (2, -5) at the end of S
`pop!(S)`	removes the last element of S
`shift!(S)`	removes the first element of S
`(a, b) = S[3]`	assigns a, b to the values of S[3]
`length(S)`	gives the number of elements in S
`sum(S)`	finds the sum of single terms in S

Commands with MultiD Arrays

`sortrows(A)`	puts rows in order
`hcat(A, B)`	Augments B at the end of A
`vcat(A, B)`	Augments B below A
`A[5]`	gives 5th element of A
`A[3, 2]`	gives the 3rd row, 2nd column of A
`A[1, : ]`	gives first row of A
`(a, b) = A[2, : ]`	assigns 2nd row values to a, b
`length(A)`	gives the length of A
`sum(A)`	adds all the elements of A

Basic mathematical operations (+, -, ·c) work on multidimensional arrays but not on 1D arrays.

Practice Problem 1

1. Create a program that lists the first *n* Fibonacci numbers as a 1D array. To do this, start with the 1D array [1, 1], then push! new elements on the end.

Practice Problem 2

2. Recall that a useful application of vectors is the equation new = old + scalar · vector.

Write a function Vector(S, T) where S and T are points in array form. This function must…

a. calculate the vector from S to T
b. calculate another point V along the same vector, but 1.5 times farther from S than T
c. report the vector and the point V in user-friendly language.
d. Test your code!

Practice Problem 3

3. Sometimes when working with vectors it's important to find a *unit vector* for a given vector v. This unit vector goes the same direction as v but has a magnitude of 1. To find it, you first find the magnitude of v using Pythagorean Theorem, then divide each term in v by that magnitude.

Write a program that finds the *unit vector* of a given vector v. Don't forget to test your code!

Practice Problem 4

4. A common calculation with vectors is the “dot product.” Assuming the two vectors are the same length, you find the product of both first terms, then both second terms, then both third terms, and so on; then you add the results.

Write a function Dot(A, B) that finds the dot product of two vectors A and B. Test your code.