

# Lecture 10: Arrays and lists

# Tips for learning a new language (e.g. Python)

- Start with something (small) you know how to do in R
- Figure out the translation to Python
  - Gives you some concrete examples to further explore
  - Some questions to ask:
    - What kinds of objects are available?
    - How is data stored?
    - How does iteration work? etc.
- Investigate similarities and differences

# Recap: vectors in R

```
1 x <- c(1, 2, 3)
2
3 sqrt(x)
```

```
[1] 1.000000 1.414214 1.732051
```

```
1 x + 1
```

```
[1] 2 3 4
```

```
1 x + c(2, 3, 4)
```

```
[1] 3 5 7
```

- Vectors only contain one type
- Many functions are (or can be) vectorized
- Math often works element-wise

# NumPy arrays

```
1 import numpy as np
2
3 x = np.array([1, 2, 3])
4
5 np.sqrt(x)
```

```
array([1.          , 1.41421356, 1.73205081])
```

```
1 x + 1
```

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

```
1 x + np.array([2, 3, 4])
```

```
array([3, 5, 7])
```

1-dimensional arrays work like R vectors:

- Only store one type
- Many functions and math can be applied element-wise

# Indexing vectors and arrays

```
1 x <- c(1, 2, 3)
2 y <- c(2, 4, 8)
3 x[1:3]
```

```
[1] 1 2 3
```

```
1 x = np.array([1, 2, 3])
2 y = np.array([2, 4, 8])
3 x[0:2]
```

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

- Similarity: Square brackets [ ] used for both R and Python
- Difference: R is 1-indexed, Python is 0-indexed
- Similarity: Indexing can be used to select multiple entries

# Indexing vectors and arrays

```
1 x <- sample(1:100, 10)
2 x
```

```
[1] 67 92 10 5 80 17 40 61 76 78
```

**Question:** How would I select the entries in `x` which are `< 50`?

# Indexing vectors and arrays

```
1 x <- sample(1:100, 10)
2 x
```

**Question:** How would I select the entries in `x` which are `< 50`?

```
1 x[x < 50]
```

```
[1] 10  5 17 40
```

# Indexing vectors and arrays

```
1 x <- sample(1:100, 10)
```

**Question:** How would I write this code in Python?



# Indexing vectors and arrays

```
1 x = np.random.choice(np.arange(1, 101), 10)
```

```
2 x
```

```
array([74, 87, 32, 59, 91, 69, 5, 6, 79, 70])
```

```
1 x[x < 50]
```

```
array([32, 5, 6])
```

- Similarity: Using booleans to index works similarly in R and Python
- Difference: `np.arange` includes the start, but *not* the end

# Indexing vectors and arrays

Indexing doesn't *always* behave the same:

```
1 x <- c(1, 2, 3)
2 x[-1]
```

```
[1] 2 3
```

```
1 x = np.array([1, 2, 3])
2 x[-1]
```

```
3
```

## Recap: lists in R

Question: How are *lists* different from *vectors* in R?

# Recap: lists in R

```
1 x <- list(c("a", "b"), list(1, 2, c(4, 5)))
```

**Question:** How would I select just the vector `c(4, 5)`?

# Recap: lists in R

```
1 x <- list(c("a", "b"), list(1, 2, c(4, 5)))
```

**Question:** How would I select just the vector `c(4, 5)`?

```
1 x[[2]][[3]]
```

```
[1] 4 5
```

# Lists in Python

```
1 x = np.array(["a", 0, 1])
```

- Like vectors in R, arrays can only store one type

# Lists in Python

In R:

```
1 x = list("a", 0, 1)
2 x[[1]]
```

```
[1] "a"
```

In Python:

```
1 x = ["a", 0, 1]
2 x[0]
```

```
'a'
```

# Lists in Python

In R:

```
1 x <- list(c("a", "b"), list(1, 2, c(4, 5)))  
2 x[[2]][[3]]
```

```
[1] 4 5
```

In Python:

```
1 x = [np.array(["a", "b"]), [1, 2, np.array([4, 5])]]  
2 x[1]
```

```
[1, 2, array([4, 5])]
```

```
1 x[1][2]
```

```
array([4, 5])
```



# Lists in Python

What will happen if I run the following R code?

```
1 x <- list(0, 1, 2)
2 x + 1
3 x * 2
```

# Lists in Python

What will happen if I run the following R code?

```
1 x <- list(0, 1, 2)
2 x + 1
```

Error in x + 1: non-numeric argument to binary operator

```
1 x * 2
```

Error in x \* 2: non-numeric argument to binary operator

# Lists in Python

What if I run the code in Python?

```
1 x = [0, 1, 2]  
2 x + [1]  
3 x * 2
```

# Lists in Python

What if I run the code in Python?

```
1 x = [0, 1, 2]
2 x + [1]
```

[0, 1, 2, 1]

```
1 x * 2
```

[0, 1, 2, 0, 1, 2]

- R vectors, and NumPy arrays, are built for math and data
- Python lists are a much more general tool

# Class activity

[https://sta279-f23.github.io/class\\_activities/ca\\_lecture\\_10.html](https://sta279-f23.github.io/class_activities/ca_lecture_10.html)

