Programming Principles in Python (CSCI 503/490)

Review

Dr. David Koop





Tasks Machine Learning can Help With

Identifying the zip code from handwritten digits on an envelope



- Detecting fraudulent activity in credit card transactions
- Identifying topics in a set of blog posts
- Grouping customers with similar preferences

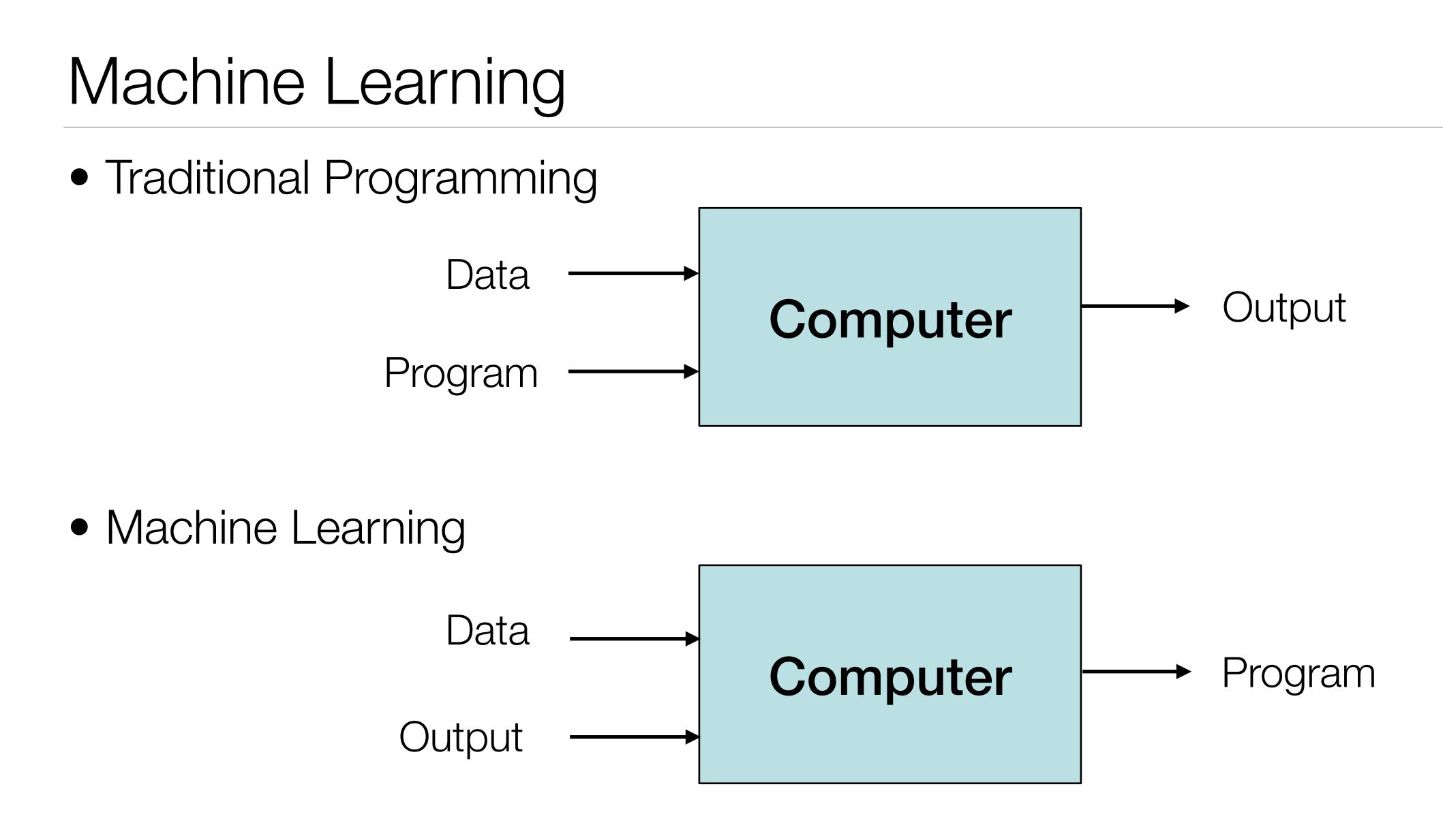
[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]

D. Koop, CSCI 503/490, Fall 2022





2









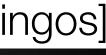


Types of Learning

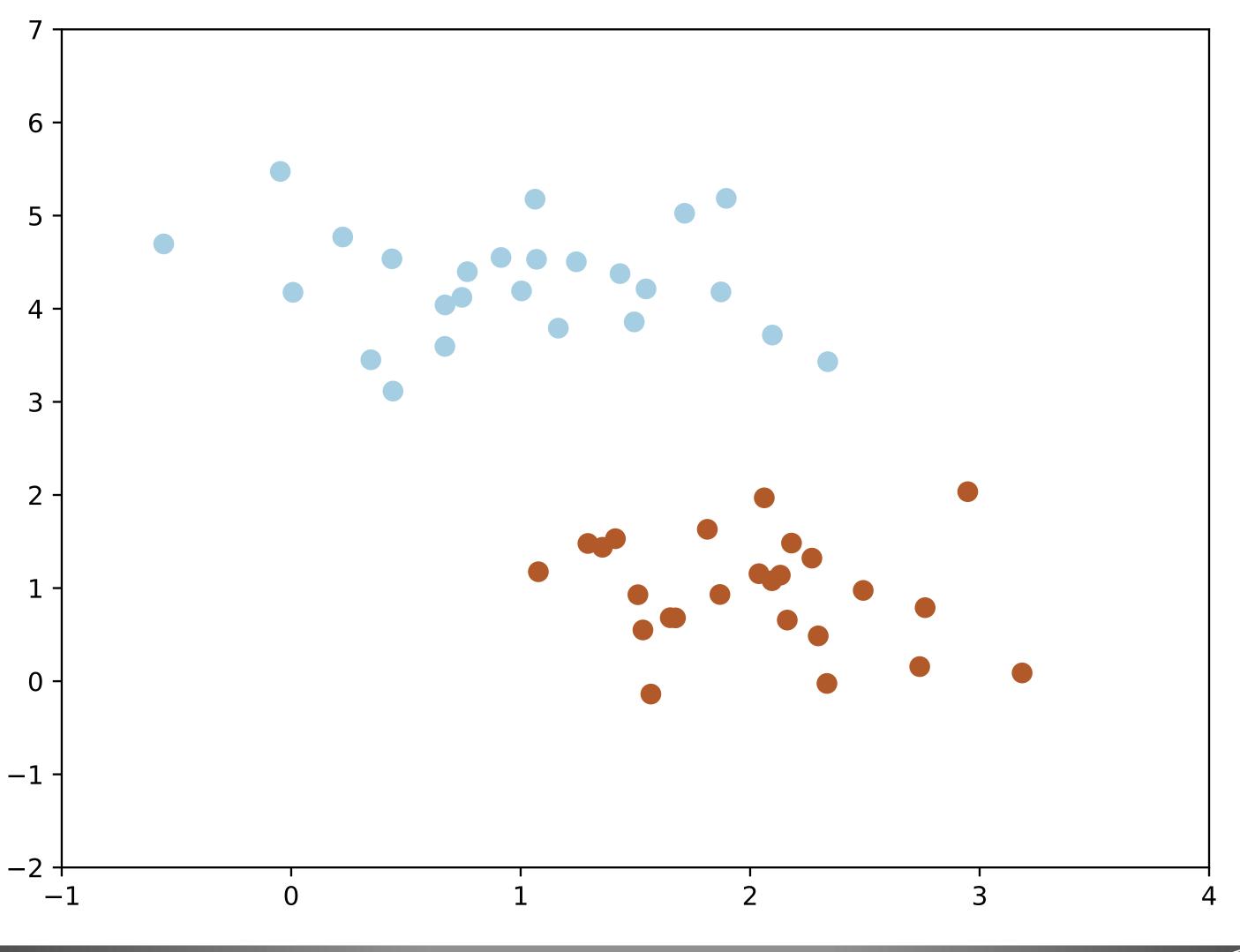
- Supervised (inductive) learning - Training data includes desired outputs
- Unsupervised learning
 - Training data does not include desired outputs
- Semi-supervised learning
 - Training data includes a few desired outputs
- Reinforcement learning
 - Rewards from sequence of actions







Supervised Learning





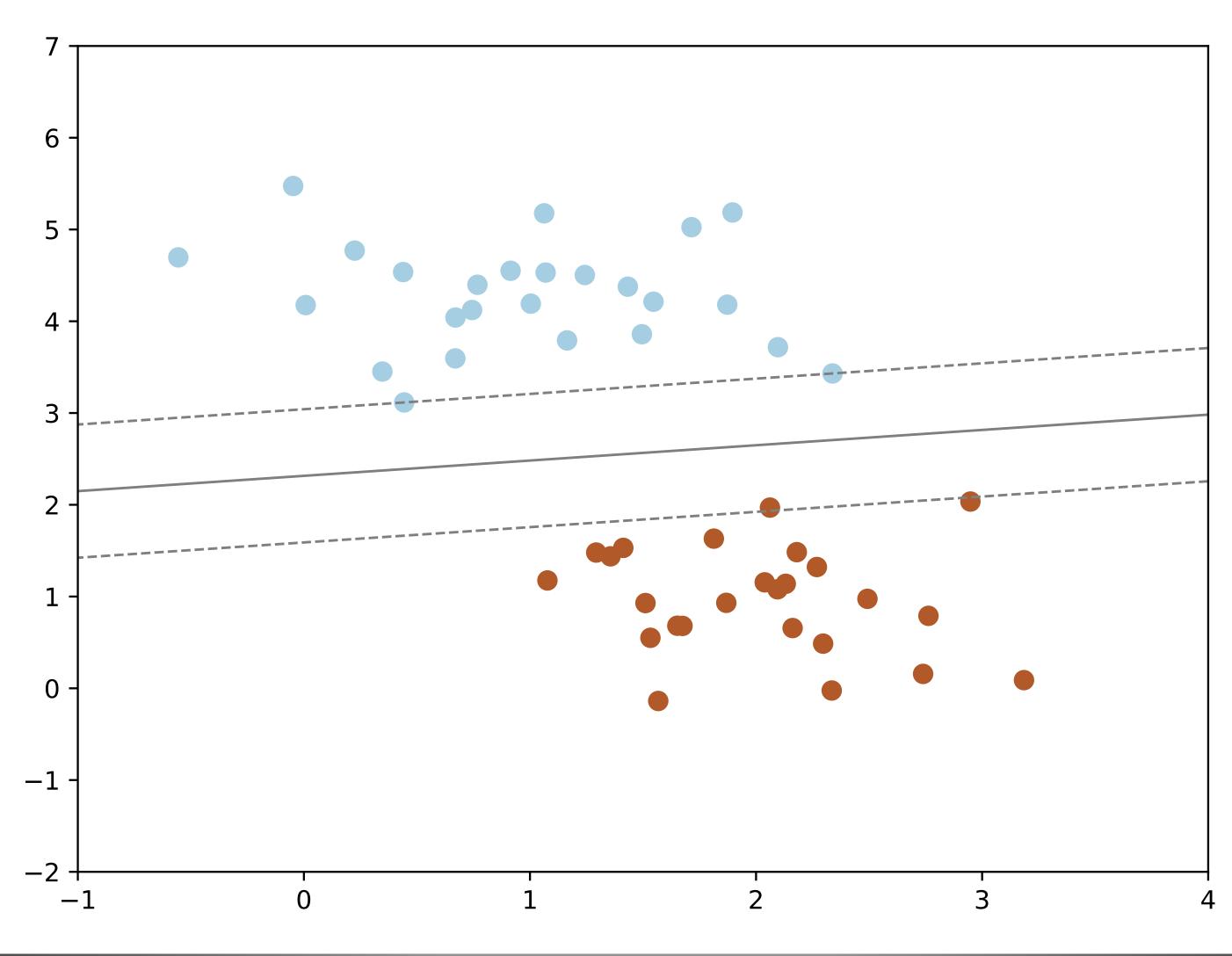








Supervised Learning: Learned Algorithm (Fit)





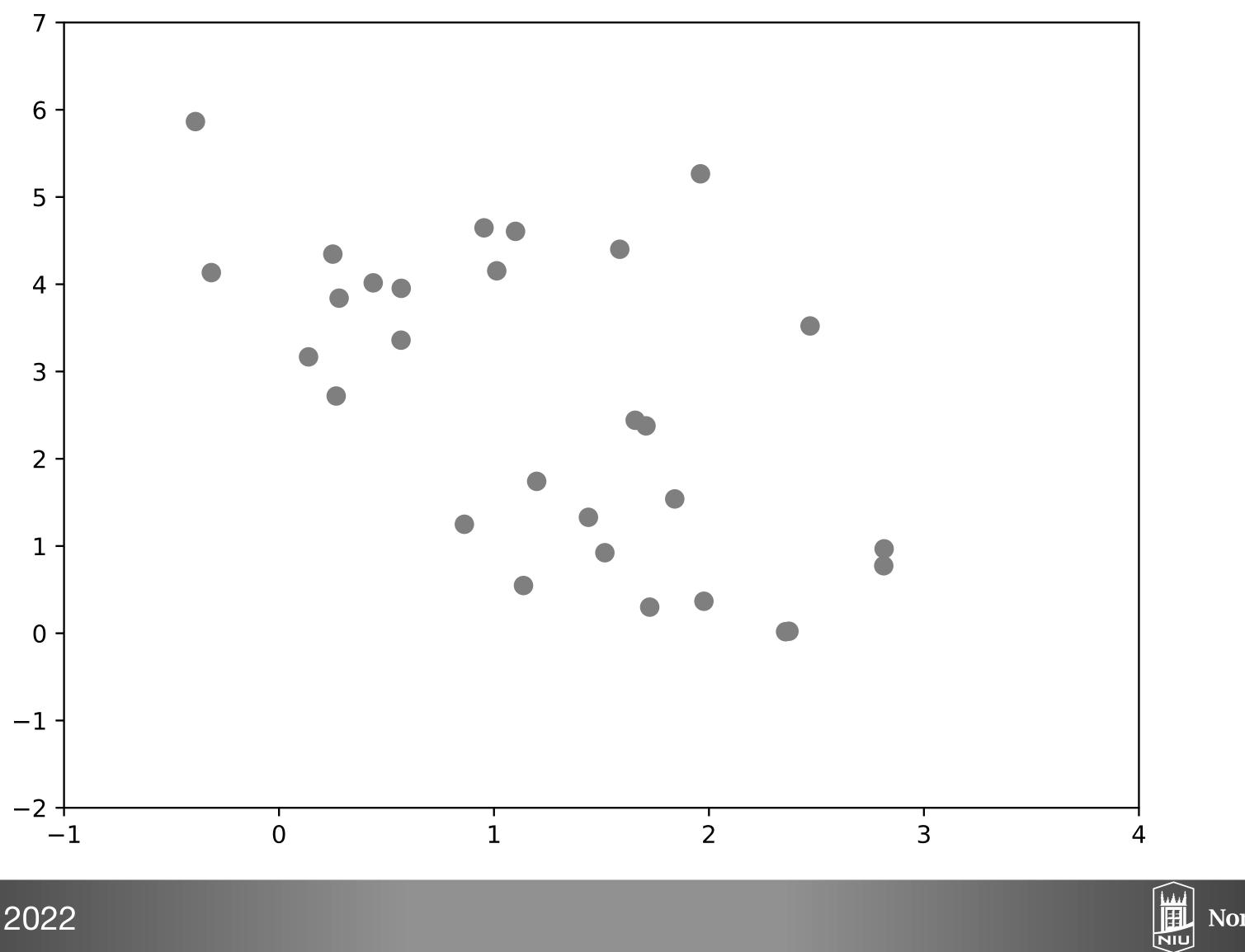








Supervised Learning: Prediction



D. Koop, CSCI 503/490, Fall 2022

[J. VanderPlas]

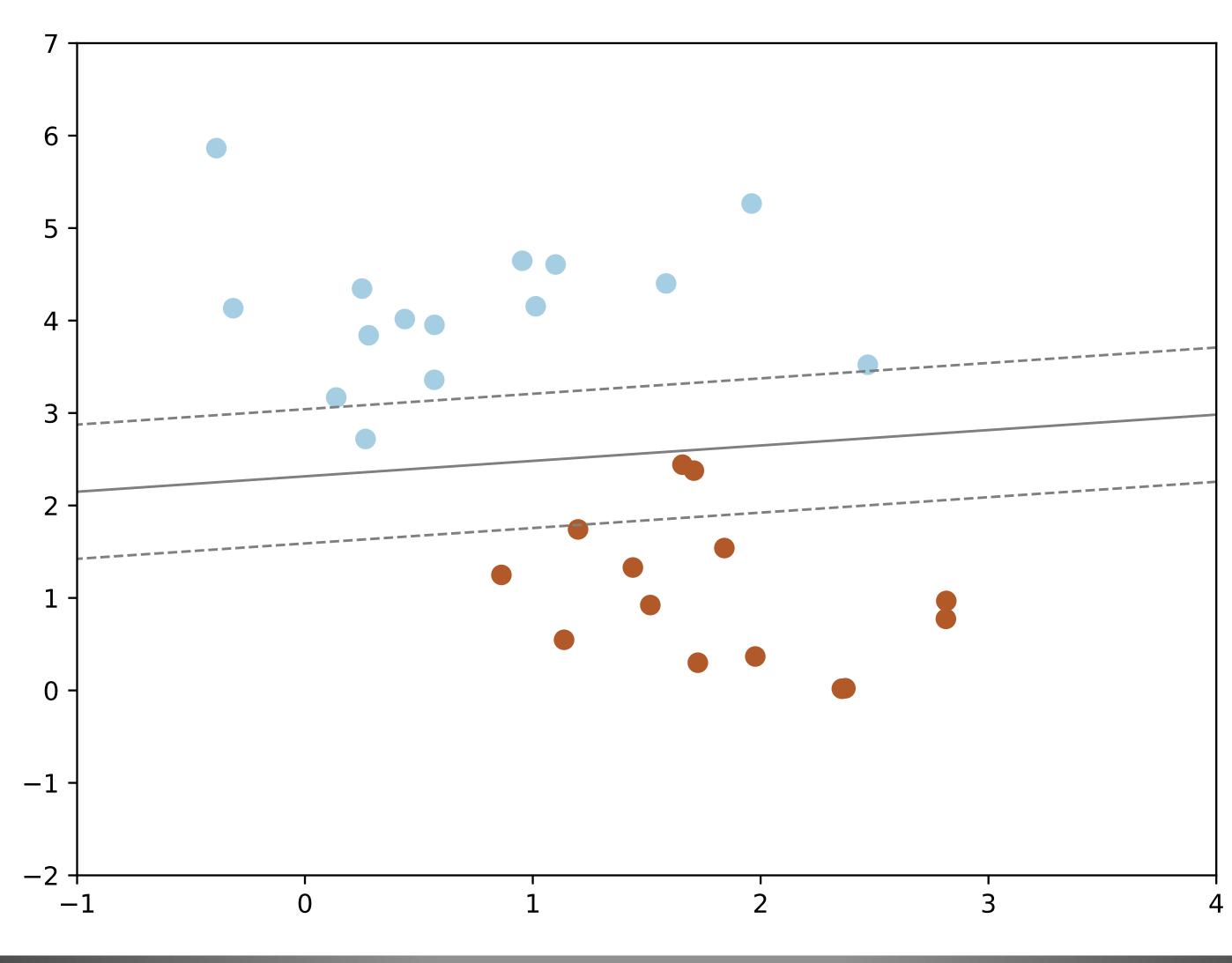
Northern Illinois University







Supervised Learning: Prediction



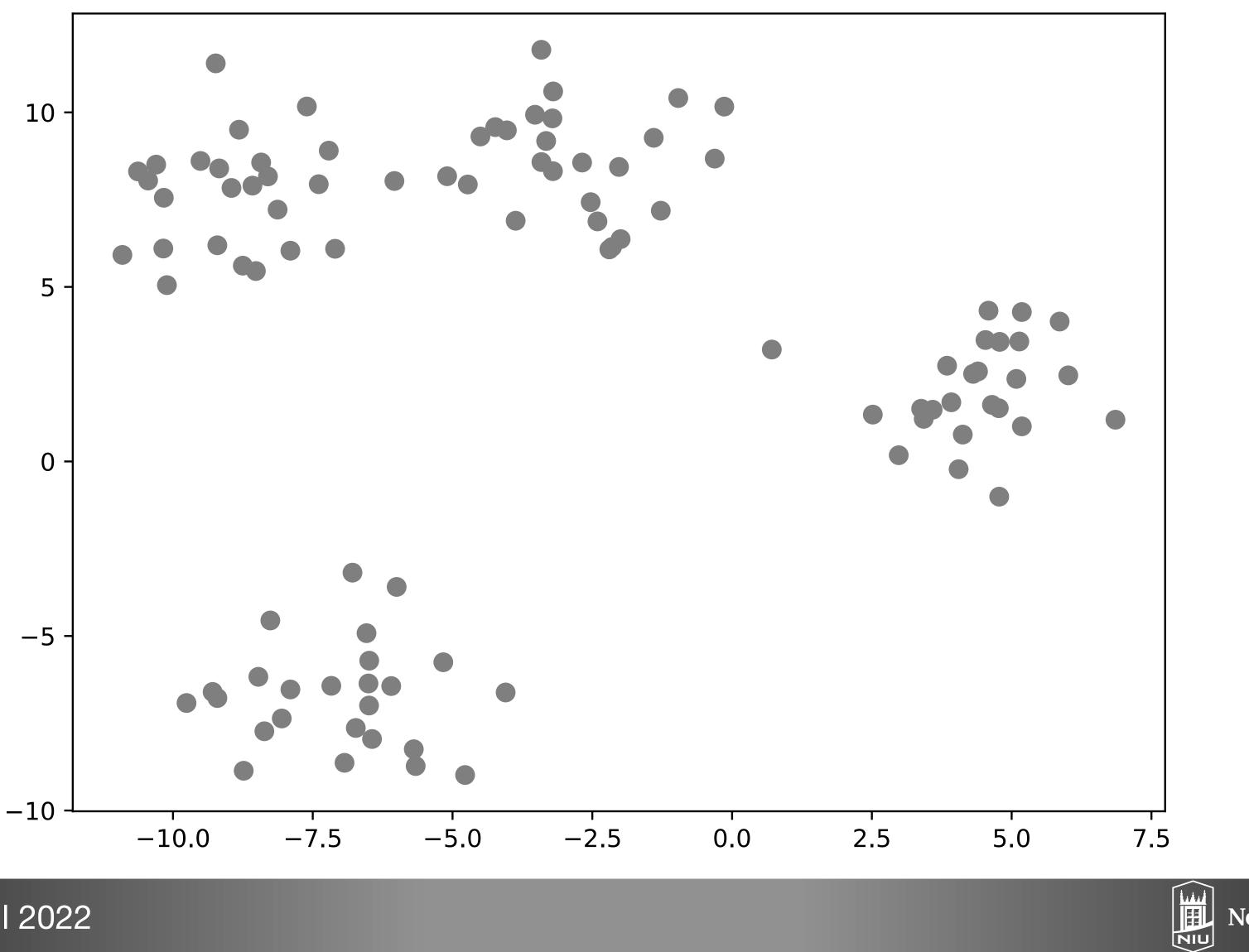








Unsupervised Learning: Input



D. Koop, CSCI 503/490, Fall 2022



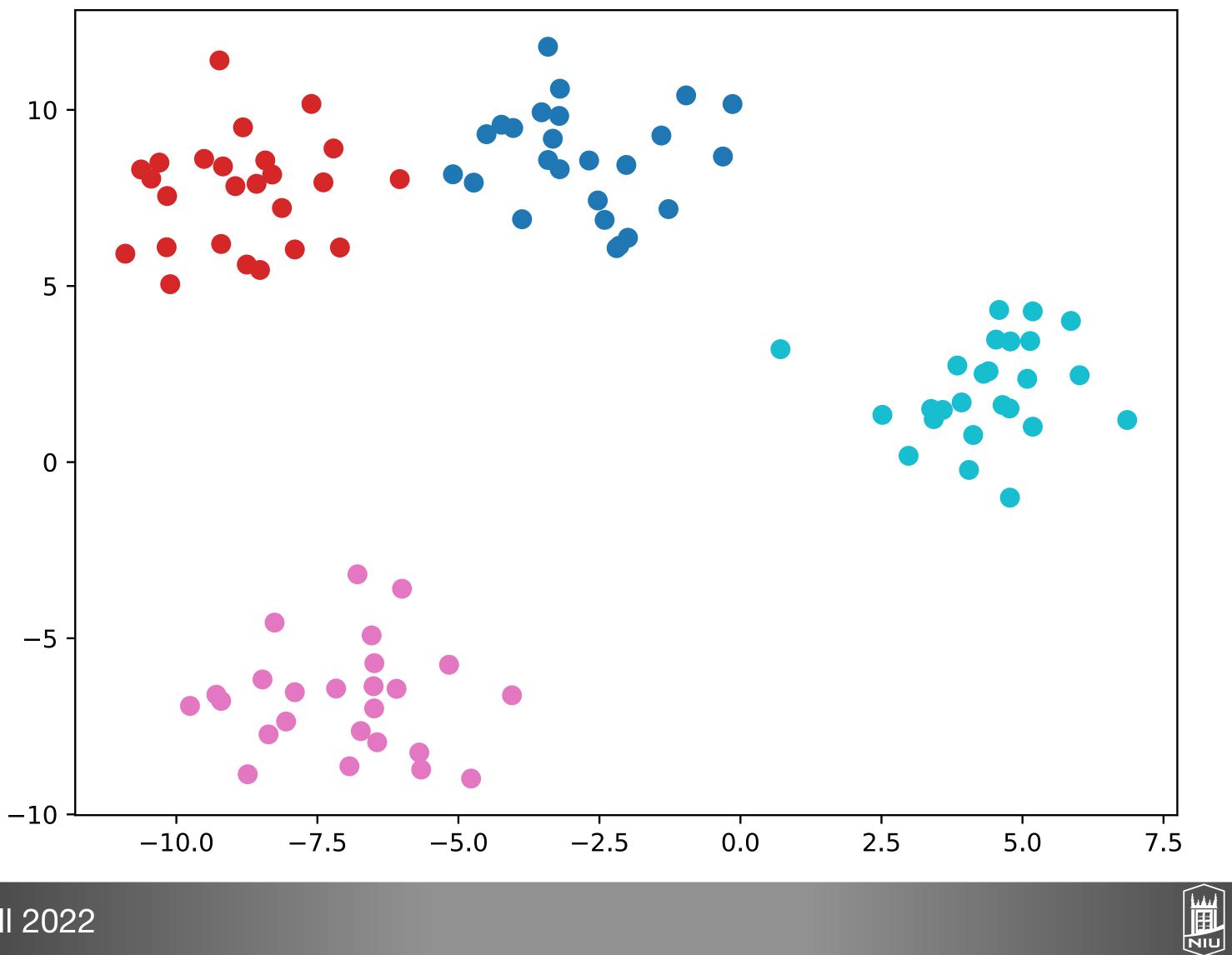
Northern Illinois University







Unsupervised Learning: Output



D. Koop, CSCI 503/490, Fall 2022

[J. VanderPlas]











scikit-learn entities

- Data: numpy matrices (also pandas series, data frames), process batches • Estimator: all supervised & unsupervised algs implement common interface
- - estimator initialization does not do learning, only attaches parameters
 - fit does the learning, learned parameters exposed with trailing underscore
- Predictor: extends estimator with predict method
 - also provides score method to return value indicating prediction quality
- Transformer: help modify or filter data before learning
 - Preprocessing, feature selection, feature extraction, and dimensionality reduction vis transform method
 - Can combine fit and transform via fit transform







scikit-learn Template

- 1. Choose model class
- 2. Instantiate model
- 3. Fit model to data
- 4. Predict on new data

from sklearn.naive bayes import GaussianNB model = GaussianNB() model.fit(Xtrain, ytrain) y model = model.predict(Xtest)

5. (Check accuracy)

from sklearn.metrics import accuracy score accuracy score (ytest, y model)





Deep Learning

- Deep learning is tied to neural netw neurons work together
- Hierarchical with multiple layers
- Usually takes advantage of GPUs
- Frameworks:
 - pytorch
 - TensorFlow
 - keras
 - theano

D. Koop, CSCI 503/490, Fall 2022

Deep learning is tied to neural networks, attempting to mimic how human





<u>Assignment 8</u>

- Due Friday
- Data and Visualization
- Port of Entry Data
- Part 1a: Month includes Year—the whole Data column (e.g. Jan 2022)





Final Exam

- Tuesday, December 6, **12:00**-1:50pm in PM 253
- More comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last four weeks of class (data, visualization, machine learning)
- Similar format





Questions?





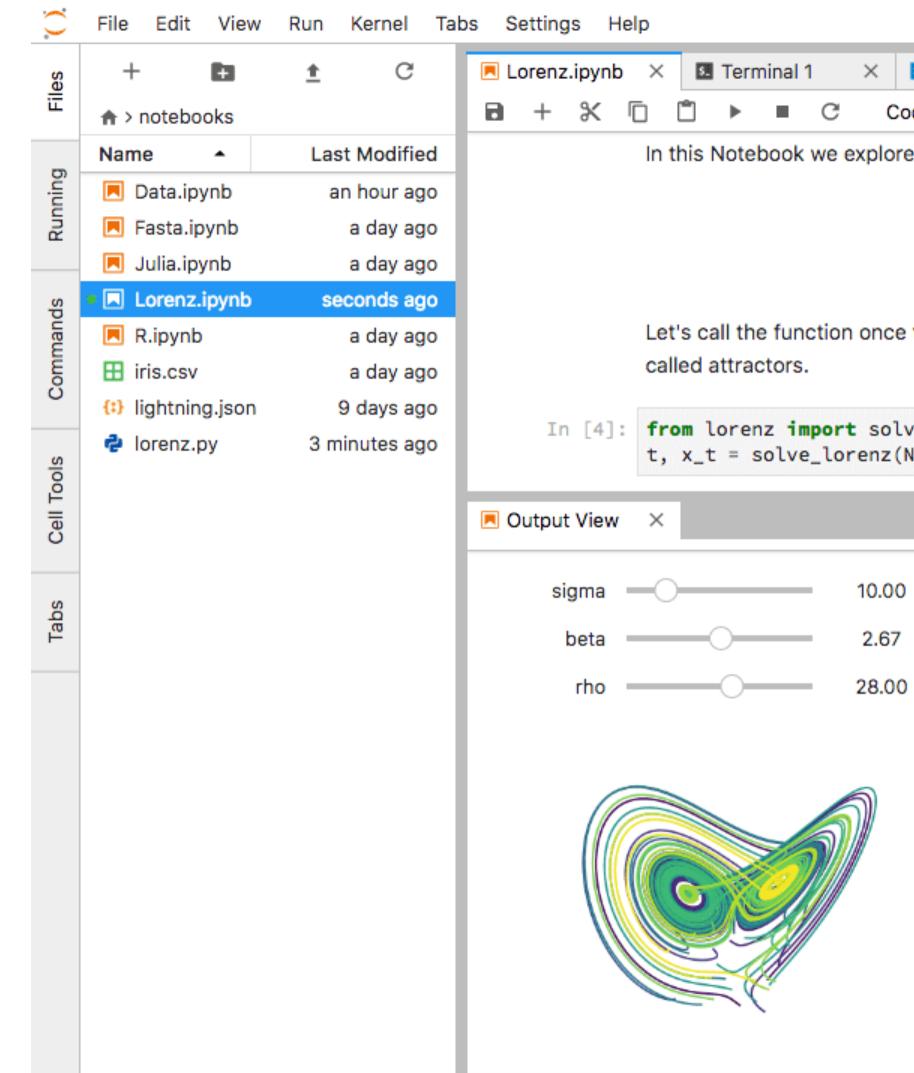
Why Python?

- High-level, readable
- Productivity
- Large standard library
- Libraries, Libraries, Libraries
- What about Speed?
 - What speed are we measuring?
 - Time to code vs. time to execute





JupyterLab and Jupyter Notebooks



Co	nsole 1	×	🗏 Data.ipynt	×	Ŵ	README.r	nd	×		
ode	~								Python 3	0
e the l	Lorenz	system of	of differential	equation	ns:					
				(
				(y-x)						1
				x - y -						
			z = -	$\beta z + xy$,					
to via	w the	colutions	Ear this cat	of porce	oto		ho t	roic	atorias swirling around two points	
to vie	ew the s	solutions	. For this set o	or paran	ietei	rs, we see	the t	raje	ectories swirling around two points,	
ve_lo N=10)	renz									
N-10)										
		renz.py	×							
		enz.py	^							
	9	def sol	lve_lorenz(N	=10, ma	ax_t	ime=4.0,	sign	ma=	10.0, beta=8./3, rho=28.0):	
2	10				b th	ne Lorenz	dif	fer	ential equations."""	
	11	-	g = plt.figu							
	12		= fig.add_a		, 0 ,	1, 1], p	roje	ect	ion='3d')	
)	13	ax.	axis('off')							
	14 15	# ,	prepare the	aves 1.	imit	+c				
	16									
	17 ax.set_ylim((-35, 35))									
	18		.set_zlim((5		·					
	19			,,						
	20	det	f lorenz_der	iv(x_y_	z,	t0, sigma	=sig	gma	, beta=beta, rho=rho):	
	21		"""Compute	the ti	ime-	derivati	e of	fa	Lorenz system."""	
	22		x, y, z =	x_y_z						
	23		return [si	gma \star ((y -	• x), x *	(rho	0 -	z) - y, x * y - beta * z]	
	24									
	25				ting	, points,	uni	for	mly distributed from -15 to 15	
	26		random.seed		_			-		
	27	×О	= -15 + 30	* np.ra	ando	om.random	(N,	3))	
	28								[.luovterl.a	h









Principles: Explicit Code

- Bad:

def make complex(*args): x, y = argsreturn dict(**locals())

• Good

def make complex(x, y): return { 'x': x, 'y': y}

D. Koop, CSCI 503/490, Fall 2022

Complex code isn't necessarily bad, but make sure you can't make it clearer





Northern Illinois University











Principles: Don't Repeat Yourself

- "Two or more, use a for" [Dijkstra]
- Rule of Three: [Roberts]
 - Don't copy-and-paste more than once
 - Refactor into methods
- Repeated code is harder to maintain
- Bad
 - f1 = load file('f1.dat')
 - r1 = get cost(f1)
 - f2 = load file('f2.dat')
 - r2 = get cost(f2)
 - f3 = load file('f3.dat')
 - r3 = get cost(f3)

- Good
 - for i in range(1,4): f = load file(f'f{i}.dat') r = get cost(f)





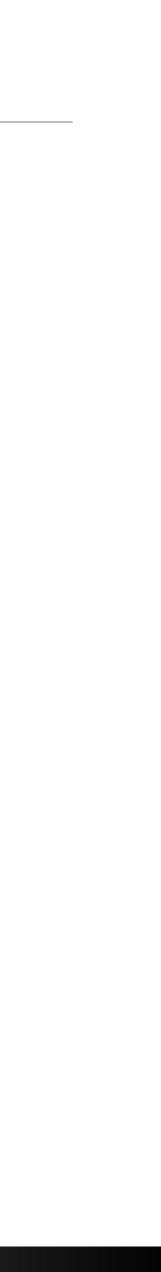




Expression Rules

- Involve
 - Literals (1, "abc"),
 - Variables (a, my height), and
 - Operators (+, -*, /, //, **)
- Spaces are irrelevant within an expression 34 # ok - a +
- Standard precedence rules
 - Parentheses, exponentiation, mult/div, add/sub
 - Left to right at each level
- Also **boolean** expressions









Identifiers

- A sequence of letters, digits, or underscores, but...
- Also includes unicode "letters", spacing marks, and decimals (e.g. Σ) • Must begin with a letter or underscore ()
- Why not a number?
- Case sensitive (a is different from A)
- Conventions:
 - Identifiers beginning with an underscore () are reserved for system use - Use underscores (a long variable), not camel-case (aLongVariable) - Keep identifier names less than 80 characters
- Cannot be reserved words









lypes

- Don't worry about types, but think about types
- Variables can "change types"
 - -a = 0
 - a = "abc"
 - a = 3.14159
- Actually, the name is being moved to a different value
- You can find out the type of the value stored at a variable v using type (v)
- Some literal types are determined by subtle differences
 - 1 vs 1. (integer vs. float)
 - 1.43 vs 1.43 j (float vs. imaginary)
 - '234' vs b'234' (string vs. byte string)





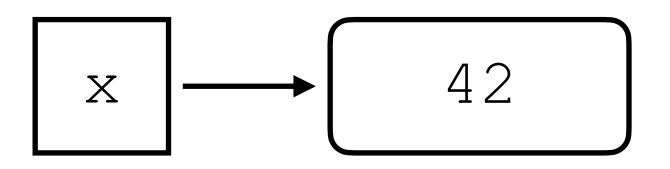




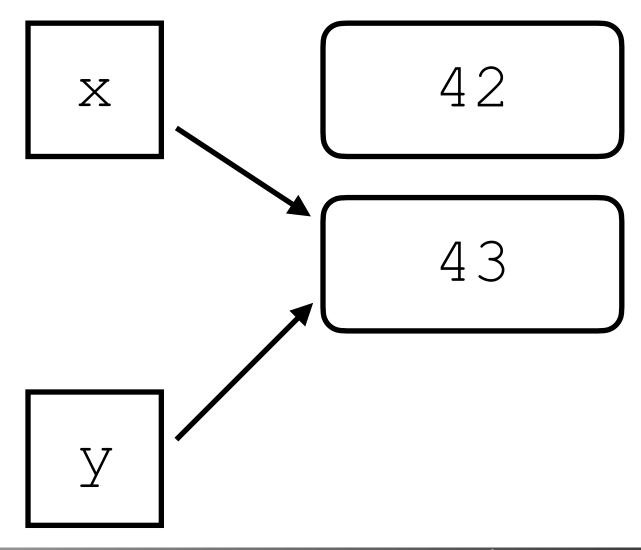
Assignment

- The = operator: a = 34; c = (a + b) ** 2
- Python variables are actually **pointers** to objects
- Also, augmented assignment: +=, -=, *=, /=, //=, **=

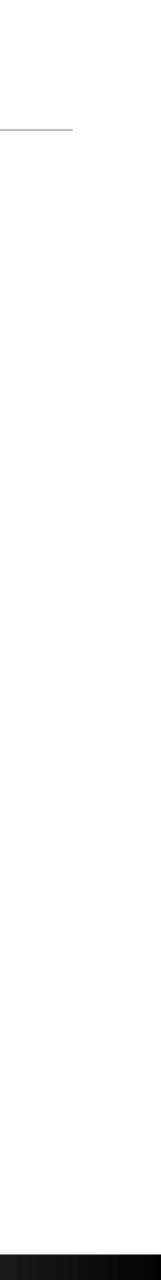
$$x = 42$$



$$\begin{array}{rcl} x &=& x &+& 1 \\ y &=& x \end{array}$$









Boolean Expressions

- Type bool: True Or False
- Note capitalization!
- Comparison Operators: <, <=, >, >=, ==, !=
 - Double equals (==) checks for equal values,
 - Assignment (=) assigns values to variables
- Boolean operators: not, and, or
 - Different from many other languages (!, &&, ||)
- More:
 - is: exact same object (usually a variable is None)
 - in: checks if a value is in a collection (34 in my list)







if, else, elif, pass

```
• if a < 10:
     print("Small")
 else:
     if a < 100:
          print("Medium")
     else:
          if a < 1000:
              print("Large")
          else:
              print("X-Large")
```

- Indentation is critical so else-if branches can become unwieldy (elif helps)
- Remember colons and indentation
- pass can be used for an empty block

D. Koop, CSCI 503/490, Fall 2022



Northern Illinois University

print("Large") else: print("X-Large")

print("Medium") elif a < 1000:

print("Small") elif a < 100:

• if a < 10:







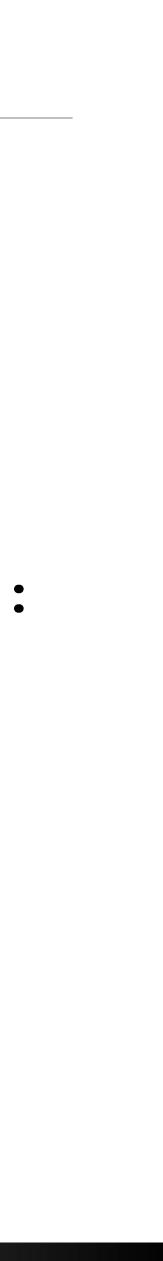
Loop Styles

- Loop-and-a-Half
 d = get_data() # priming rd
 while check(d):
 # do stuff
 d = get data()
- Infinite-Loop-Break
 while True:
 d = get_data()
 if check(d):
 break
 # do stuff

D. Koop, CSCI 503/490, Fall 2022

Assignment Expression (Walrus) while check(d := get_data): # do stuff





27

Functions

- Use return to return a value
- def <function-name>(<parameter-names>): # do stuff return res
- Can return more than one value using commas
- def <function-name>(<parameter-names>): # do stuff return res1, res2
- Use simultaneous assignment when calling:
 - $a_{,}$ b = do something(1,2,5)
- If there is no return value, the function returns None (a special value)









Positional & Keyword Arguments

- Generally, any argument can be passed as a keyword argument
- def f(alpha, beta, gamma=1, delta=7, epsilon=8, zeta=2, # ...
- f(5,6)
- f(alpha=7, beta=12, iota=0.7)

D. Koop, CSCI 503/490, Fall 2022

eta=0.3, theta=0.5, iota=0.24, kappa=0.134):









Pass by object reference

- AKA passing object references by value
- Python doesn't allocate space for a variable, it just links identifier to a value
- Mutability of the object determines whether other references see the change Any immutable object will act like pass by value
- Any mutable object acts like pass by reference unless it is reassigned to a new value









Sequences

- Defining a list: my list = [0, 1, 2, 3, 4]
- But lists can store different types: -my list = [0, "a", 1.34]
- Including other lists:
 - -my list = [0, "a", 1.34, [1, 2, 3]]
- (single or double)

D. Koop, CSCI 503/490, Fall 2022

• Strings "abcde", Lists [1, 2, 3, 4, 5], and Tuples (1, 2, 3, 4, 5)

• Others are similar: tuples use parenthesis, strings are delineated by quotes







Indexing & Slicing

- Positive or negative indices can be used at any step
- my str = "abcde"; my str[1] + my str[-4]# "bb"
- my list = [1,2,3,4,5]; my list[3:-1] # [4]
- Implicit indices
 - my tuple = (1,2,3,4,5); my tuple[-2:] # (4,5)
 - my tuple[:3] # (1,2,3)

$$\begin{bmatrix} 1:3 \\ -4:-2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ a & b & c & d & e \\ -5 & -4 & -3 & -2 & -1 \end{bmatrix}$$







Tuples

- Tuples are immutable sequences
- We've actually seen tuples a few times already
 - Simultaneous Assignment
 - # don't normally do this
- Returning Multiple Values from a Function Python allows us to omit parentheses when it's clear $-b_{,a} = a_{,b} = (a_{,b}) = (a_{,b})$ - t1 = a, b-c, d = f(2, 5, 8) # same as (c, d) = f(2, 5, 8)

- -t2 = f(2, 5, 8) # don't normally do this



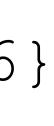




Dictionary

- AKA associative array or map
- Collection of key-value pairs
 - Keys must be unique
 - Values need not be unique
- Syntax:
 - Curly brackets {} delineate start and end
 - Colons separate keys from values, commas separate pairs
 - d = {'DeKalb': 783, 'Kane': 134, 'Cook': 1274, 'Will': 546}
- No type constraints
 - d = {'abc': 25, 12: 'abc', ('Kane', 'IL'): 123.54}









Collections

- A dictionary is **not** a sequence
- Sequences are ordered
- Conceptually, dictionaries need no order
- A dictionary is a **collection**
- Sequences are also collections
- Length for dictionaries counts number of key-value pairs
 - Pass dictionary to the len function
 - d = {'abc': 25, 12: 'abc', ('Kane', 'IL'): 123.54} len(d) # 3

D. Koop, CSCI 503/490, Fall 2022

• All collections have length (len), membership (in), and iteration (loop over values)









List Comprehension

- output = []for d in range (5): output.append(d ** 2 - 1)
- Rewrite as a map:
 - output = [d ** 2 1 for d in range(5)]
- Can also filter:
 - output = [d for d in range(5) if d % 2 == 1]
- Combine map & filter:
 - output = [d ** 2 1 for d in range(5) if d % 2 == 1]







Short-Circuit Evaluation

- Works for and and or
- and:
 - if **any** value is False, stop and return False

$$-a, b = 2, 3$$

 $a > 3$ and $b < 5$

- Or:
 - if **any** value is True, stop and return True
 - a, b, c = 2, 3, 7 a > 3 or b < 5 or c > 8

D. Koop, CSCI 503/490, Fall 2022

• Automatic, works left to right according to order of operations (and before or)









Strings

- Remember strings are sequences of characters
- Strings are collections so have len, in, and iteration
 - s = "Huskies" len(s); "usk" in s; [c for c in s if c == 's']
- Strings are sequences so have
 - indexing and slicing: s[0], s[1:]
 - concatenation and repetition: s + " at NIU"; s * 2
- Single or double quotes 'string1', "string2"
- Triple double-quotes: """A string over many lines"""
- Escaped characters: '\n' (newline) '\t' (tab)







Regular Expressions

- AKA regex
- A syntax to better specify how to decompose strings
- Look for patterns rather than specific characters
- "31" in "The last day of December is 12/31/2016."
- May work for some questions but now suppose I have other lines like: "The last day of September is 9/30/2016."
- ...and I want to find dates that look like:
- {digits}/{digits}/{digits}
- Cannot search for every combination!
- \d+/\d+/\d+ # \d is a character class







Reading & Writing Files

- Can iterate through the file (think of the file as a collection of lines):
 - f = open('huck-finn.txt', 'r') for line in f:
 - if 'Huckleberry' in line: print(line.strip())
- For writing, with statement does "enter" and "exit": don't need to call outf.close()
 - with open ('output.txt', 'w') as outf: for k, v in counts.items(): outf.write(k + ': ' + v + '\n')





Command Line Interfaces (CLIs)

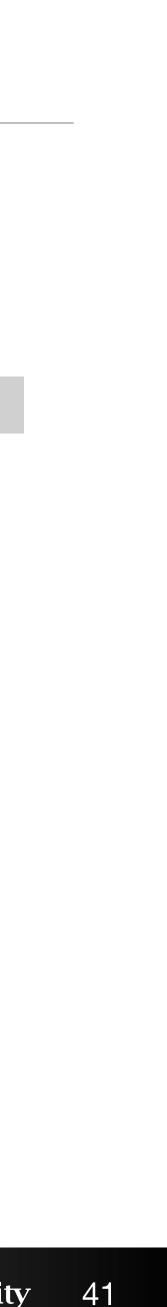
- Prompt:
 - \$
 - V develop > ./setup.py
- Commands
 - \$ cat <filename>
 - \$ git init
- Arguments/Flags: (options)
 - \$ python -h
 - \$ head -n 5 <filename>
 - \$ git branch fix-parsing-bug

D. Koop, CSCI 503/490, Fall 2022

:hon unix







Modules and Packages

- Python allows you to import code from other files, even your own
- A **module** is a collection of definitions
- A **package** is an organized collection of modules
- Modules can be
 - a separate python file
 - a separate C library that is written to be used with Python
 - a built-in module contained in the interpreter
 - a module installed by the user (via conda or pip)
- All types use the same import syntax





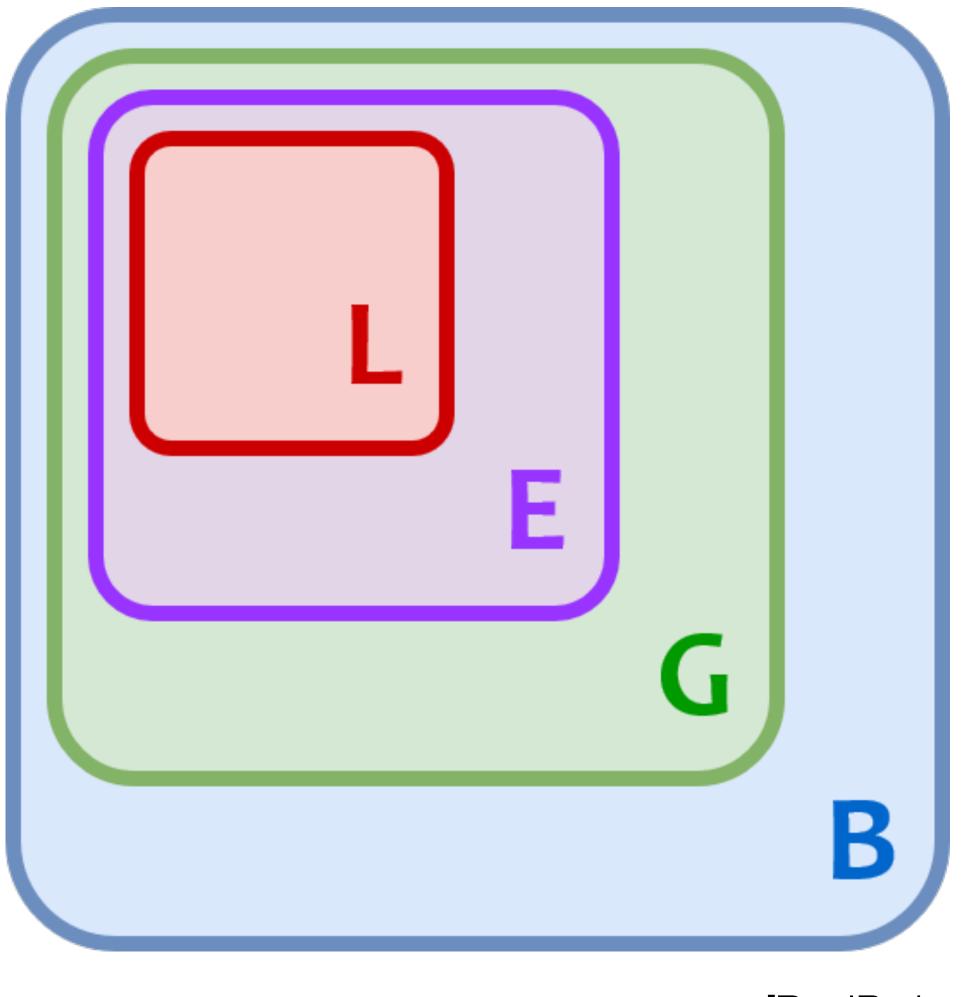


Namespaces

- Namespace is basically a dictionary with names and their values
- Accessing namespaces

builtins , globals(), locals()

- Examine contents of a namespace: dir(<namespace>)
- Python checks for a name in the sequence: local, enclosing, global, builtins
- To access names in outer scopes, use global (global) and nonlocal (enclosing) declarations











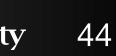


Object-Oriented Programming Concepts

- Abstraction: simplify, hide implementation details, don't repeat yourself
- Encapsulation: represent an entity fully, keep attributes and methods together
- Inheritance: reuse (don't reinvent the wheel), specialization
- Polymorphism: methods are handled by a single interface with different implementations (overriding)







Classes and Instances in Python

• Class Definition: - class Vehicle: self.make = make self.model = model self.year = year self.color = color

> def age(self): return 2021 - self.year

- Instances:
 - car1 = Vehicle('Toyota', 'Camry', 2000, 'red') - car2 = Vehicle('Dodge', 'Caravan', 2015, 'gray')

D. Koop, CSCI 503/490, Fall 2022



def init (self, make, model, year, color):





Subclass

- Just put superclass(-es) in parentheses after the class declaration
- class Car(Vehicle):

. . .

- - self.num doors = num doors
- def open door(self):
- super() is a special method that locates the base class
 - Constructor should call superclass constructor
 - Extra arguments should be initialized and extra instance methods

```
def init (self, make, model, year, color, num doors):
   super(). init (make, model, year, color)
```







lyping

- Dynamic Typing: variable's type can change (what Python does)
- Static Typing: compiler enforces types, variable types generally don't change
- Duck Typing: check method/attribute existence, not type
- Python is a dynamically-typed language (and plans to remain so)
- ...but it has recently added more support for type hinting/annotations that allow static type checking
- Type annotations change **nothing** at runtime!













Dealing with Errors

- Can explicitly check for errors at each step
 - Check for division by zero
 - Check for invalid parameter value (e.g. string instead of int)
- Sometimes all of this gets in the way and can't be addressed succinctly - Too many potential errors to check
- - Cannot handle groups of the same type of errors together
- Allow programmer to determine when and how to handle issues
 - Allow things to go wrong and handle them instead
 - Allow errors to be propagated and addressed once





Try, Except, Else, and Finally

• b = 3a = 0try: c = b / aexcept ZeroDivisionError: print("Division failed") C = 0else: print("Division succeeded", c) finally: print("This always runs")





Debugging

- print statements
- logging library
- pdb
- Extensions for IDEs (e.g. PyCharm)
- JupyterLab Debugger Support









Testing

- If statements
- Assert statements
- Unit Testing
- Integration Testing









Python Modules for Working with the Filesystem

- In general, cross-platform! (Linux, Mac, Windows)
- os: translations of operating system commands
- shutil: better support for file and directory management
- fnmatch, glob: match filenames, paths
- os.path: path manipulations
- some support for matching paths

• pathlib: object-oriented approach to path manipulations, also includes

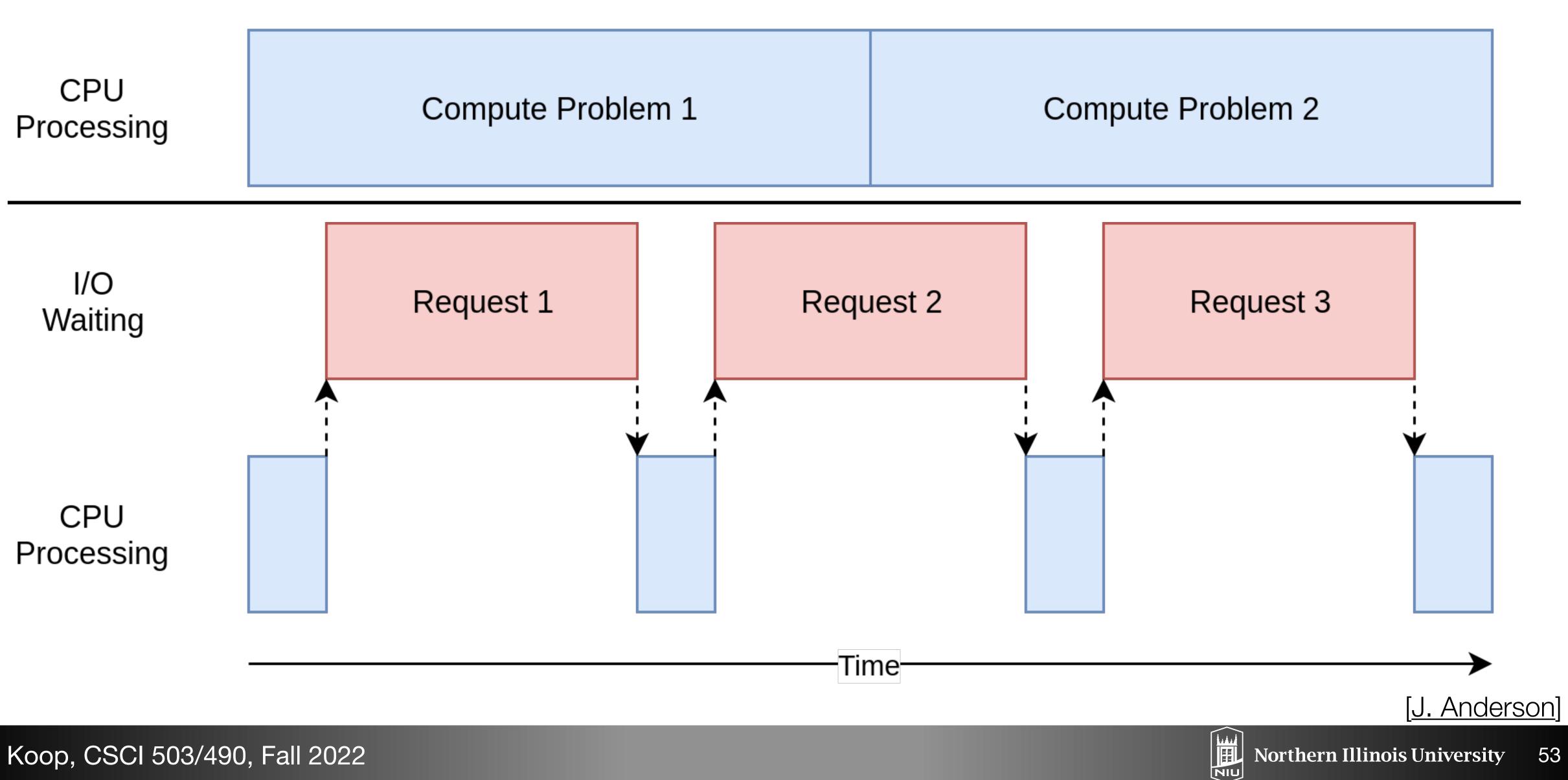








Concurrency: CPU-Bound vs. I/O-Bound



<pre>df = pd.read_csv('penguins_lter.csv')</pre>										
	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns





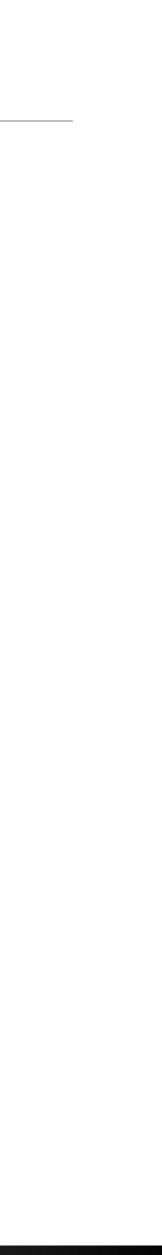




	df =	pd.read_csv	('penguins_l	ter.csv')							
Column Name		studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
	0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
	1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
	2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
	3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
	4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
	339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
	340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
	341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
	342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
	343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns







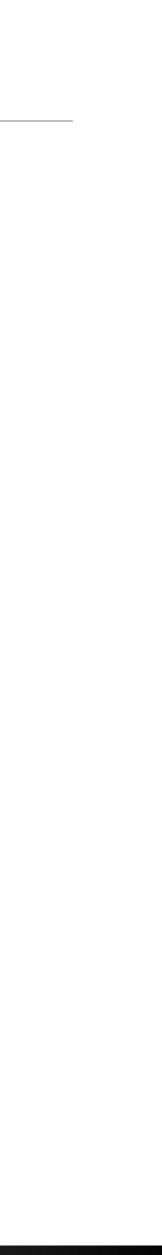


	df =	pd.read_csv	'penguins_lt	ter.csv')							
Column Name	es	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
	0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
	1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
	2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
	3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
	4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
Index											
	339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
	340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
	341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
	342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
	343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

D. Koop, CSCI 503/490, Fall 2022









	df =	pd.read_csv	('penguins_l	ter.csv')							
Column Name	es	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
	0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
	1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
	2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
	3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
	4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
Index											
	339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
	340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
	341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
	342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
	343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

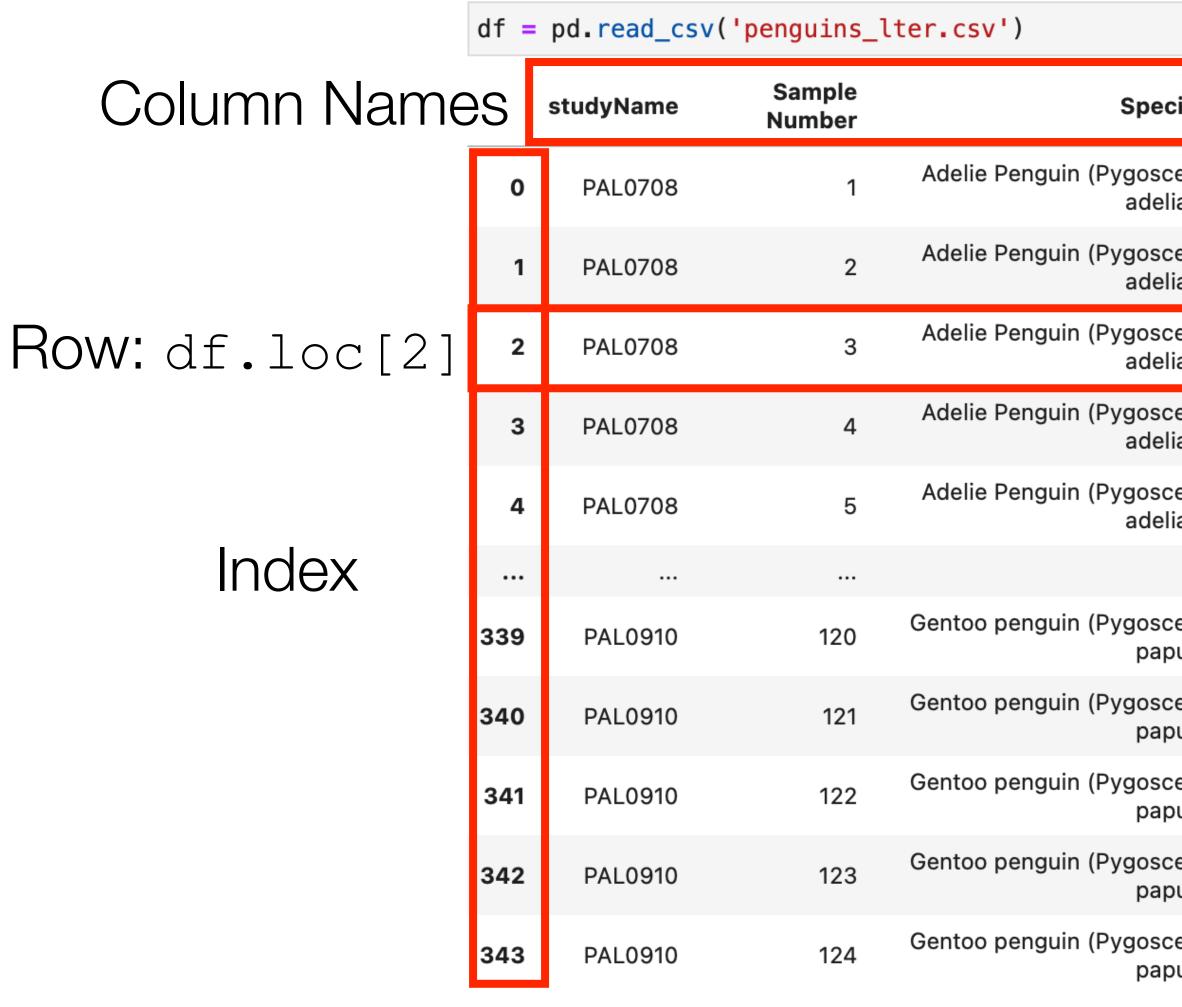
D. Koop, CSCI 503/490, Fall 2022









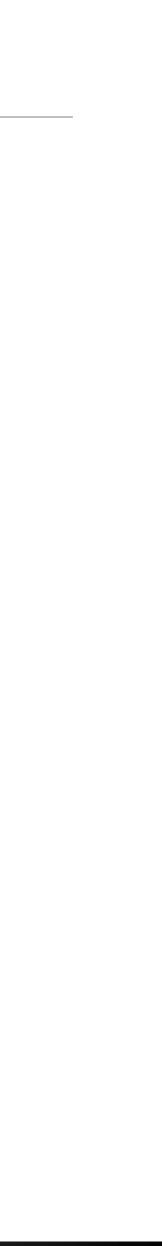


344 rows × 17 columns

D. Koop, CSCI 503/490, Fall 2022

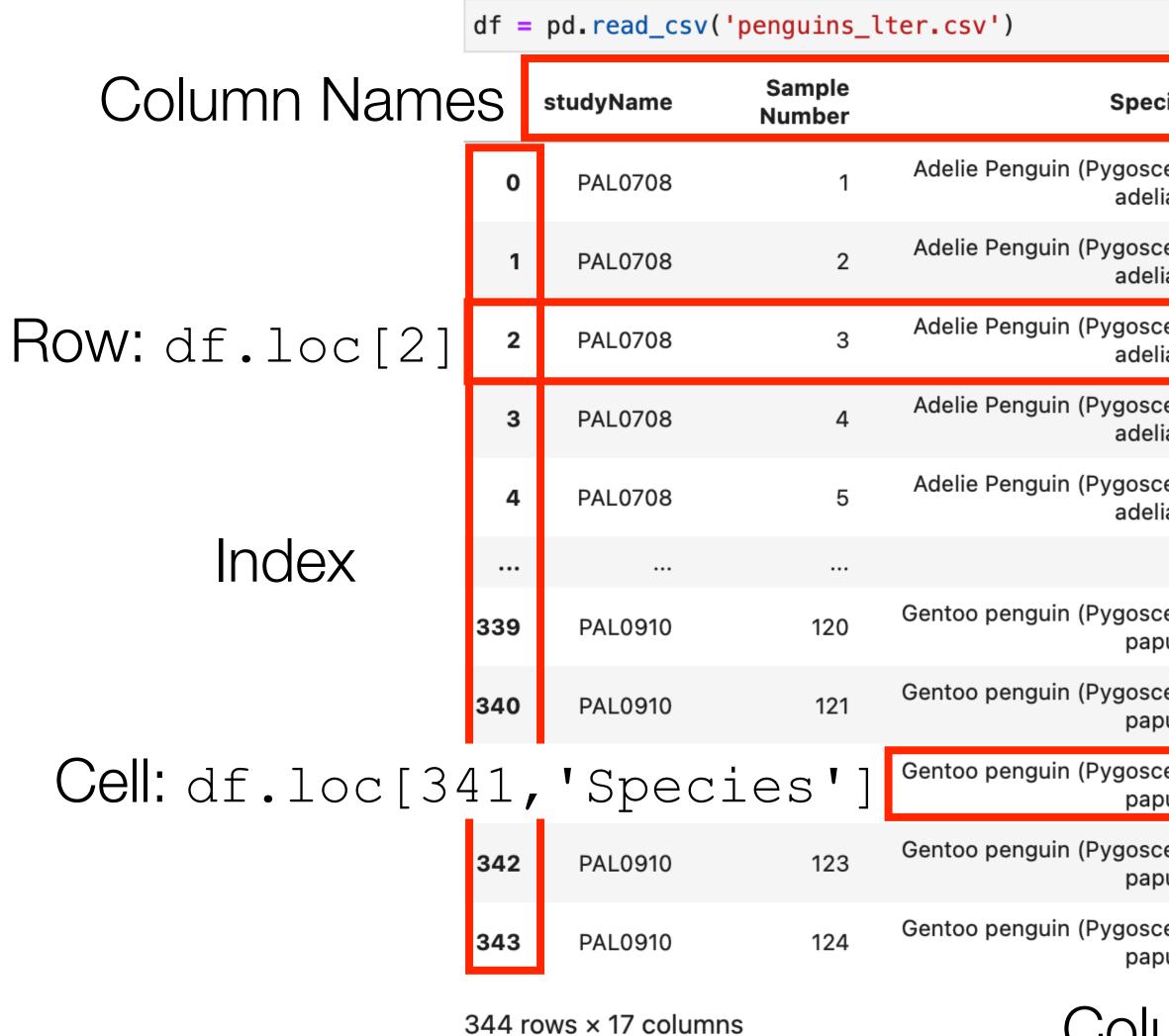
cies	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9











D. Koop, CSCI 503/490, Fall 2022

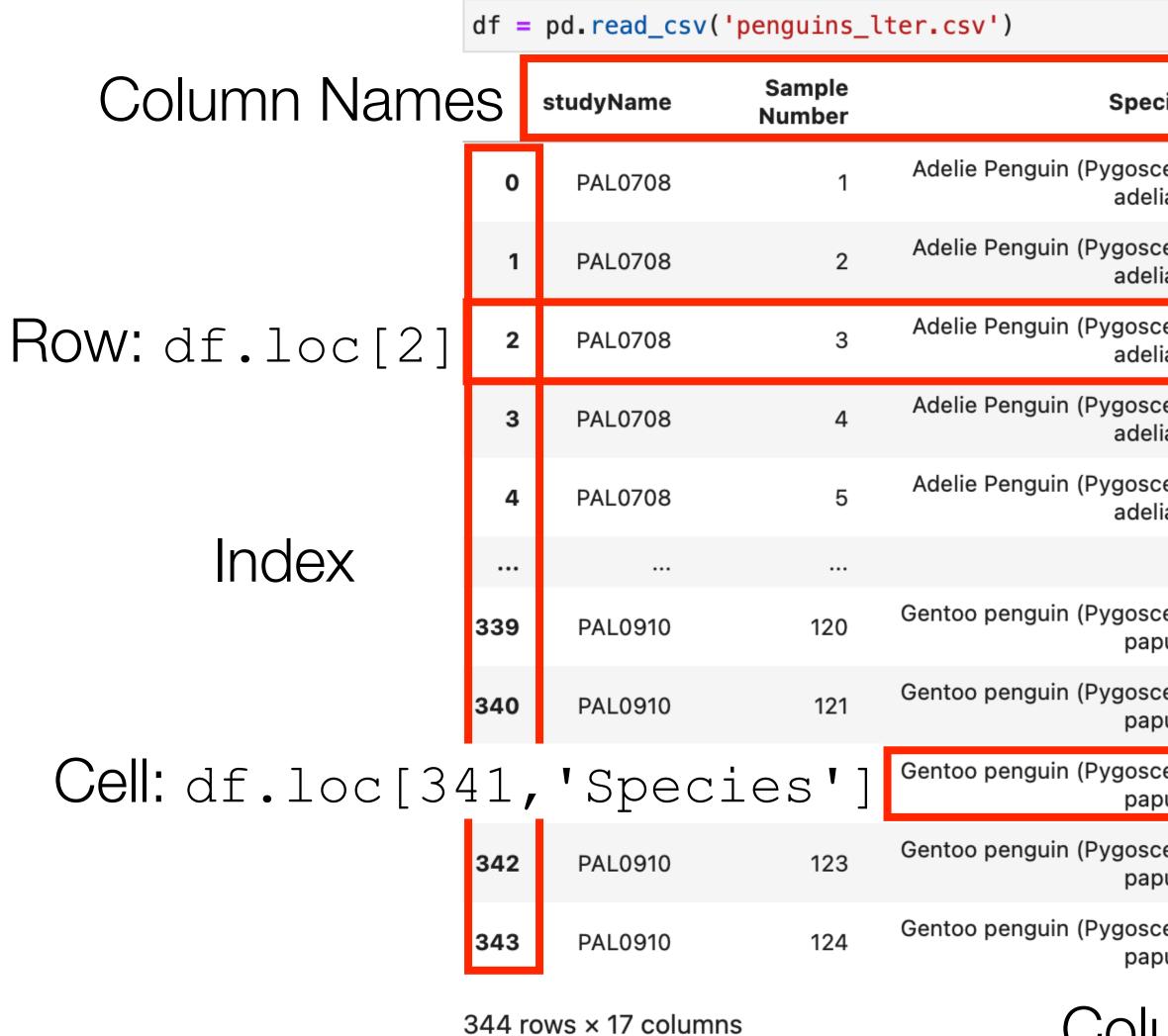
cies	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9











D. Koop, CSCI 503/490, Fall 2022

cies	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
celis liae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	Missing [
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
celis pua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9









Array Operations

- a = np.array([1, 2, 3])b = np.array([6, 4, 3])
- (Array, Array) Operations (**Element-wise**)
 - Addition, Subtraction, Multiplication
 - -a + b # array([7, 6, 6])
- (Scalar, Array) Operations (**Broadcasting**):
 - Addition, Subtraction, Multiplication, Division, Exponentiation
 - a ** 2 # array([1, 4, 9])
 - -b + 3 # array([9, 7, 6])









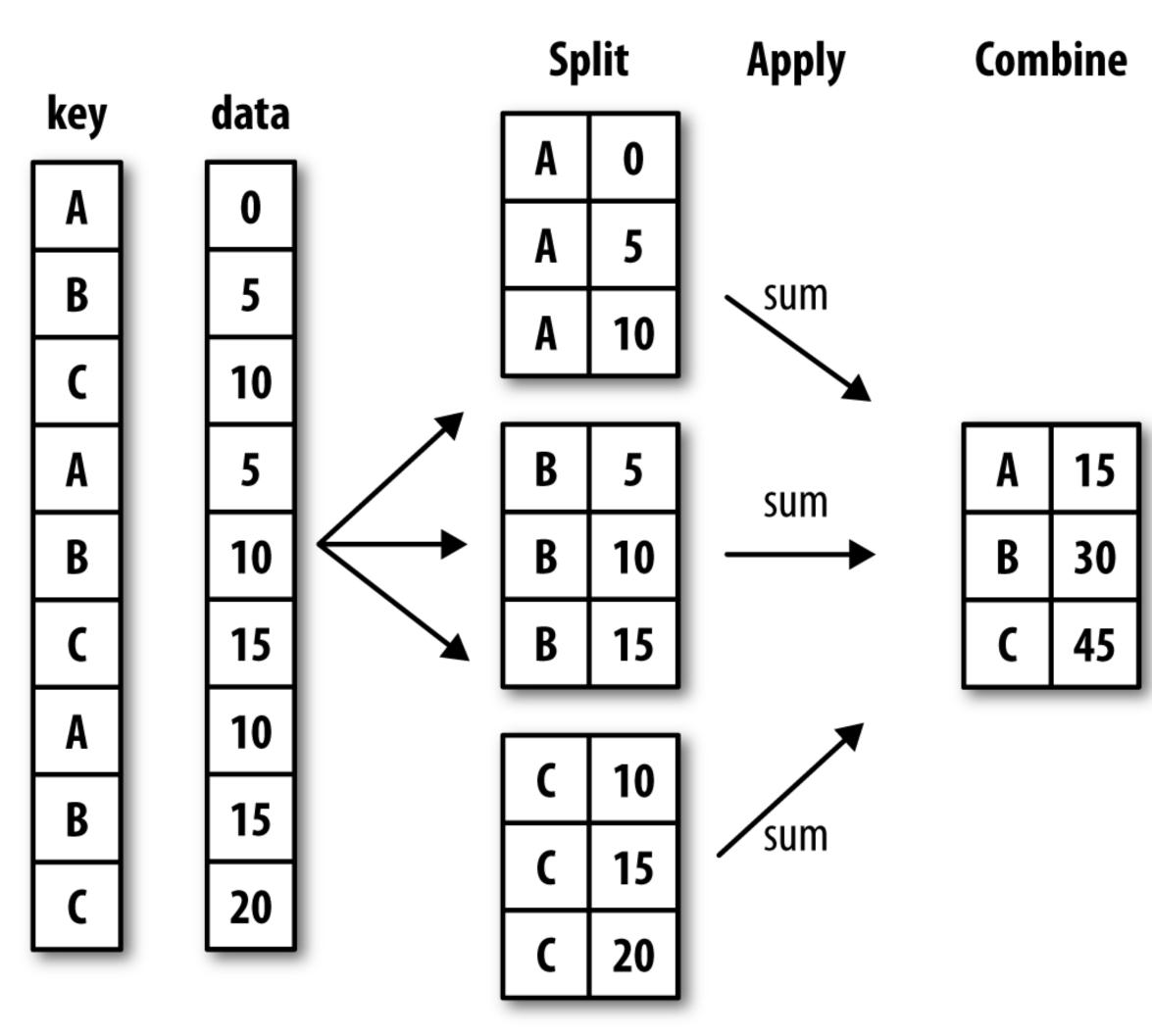
Array Slicing

- 2D+: comma separated indices as shorthand:
 - $\operatorname{arr2} = \operatorname{np.array}([[1.5, 2, 3, 4], [5, 6, 7, 8]])$
 - a[1:2,1:3]
 - a[1:2,:] # works like in single-dimensional lists
- Can combine index and slice in different dimensions
 - a[1,:] # gives a row
 - a[:,1] # gives a column
- Slicing vs. indexing produces different shapes!
 - a[1,:] # 1-dimensional
 - a[1:2,:] # 2-dimensional





Aggregation: Split-Apply-Combine



D. Koop, CSCI 503/490, Fall 2022

[W. McKinney, Python for Data Analysis]









Tidy Data: Melt

Want to keep each observation separate (tidy), aka pivot_longer

	location	Temperature	Jan-2010	Feb-2010	Ма
0	CityA	Predict	30	45	
1	CityB	Actual	32	43	

df.melt(id vars=["location", "Temperature"], var name="Date", value name="Value")

	location	Temperature	Date	Value
0	CityA	Predict	Jan-2010	30
1	CityB	Actual	Jan-2010	32
2	CityA	Predict	Feb-2010	45
3	CityB	Actual	Feb-2010	43
4	CityA	Predict	Mar-2010	24
5	CityB	Actual	Mar-2010	22











Tidy Data: Pivot

- "wide" format (aka pivot_wider)
- Long format: column names are data values...
- Wide format: more like spreadsheet format
- Example:

date	item	value	
0 1959-03-31	realgdp	2710.349	
1 1959-03-31	infl	0.000	
2 1959-03-31	unemp	5.800	
3 1959-06-30	realgdp	2778.801	
4 1959-06-30	infl	2.340	
5 1959-06-30	unemp	5.100	
6 1959-09-30	realgdp	2775.488	
7 1959-09-30	infl	2.740	
8 1959-09-30	unemp	5.300	
9 1959-12-31	realgdp	2785.204	

Sometimes, we have data that is given in "long" format and we would like

```
.pivot('date', 'item', 'value')
```

item	infl	realgdp	unemp
date			
1959-03-31	0.00	2710.349	5.8
1959-06-30	2.34	2778.801	5.1
1959-09-30	2.74	2775.488	5.3
1959-12-31	0.27	2785.204	5.6
1960-03-31	2.31	2847.699	5.2

[W. McKinney, Python for Data Analysis]

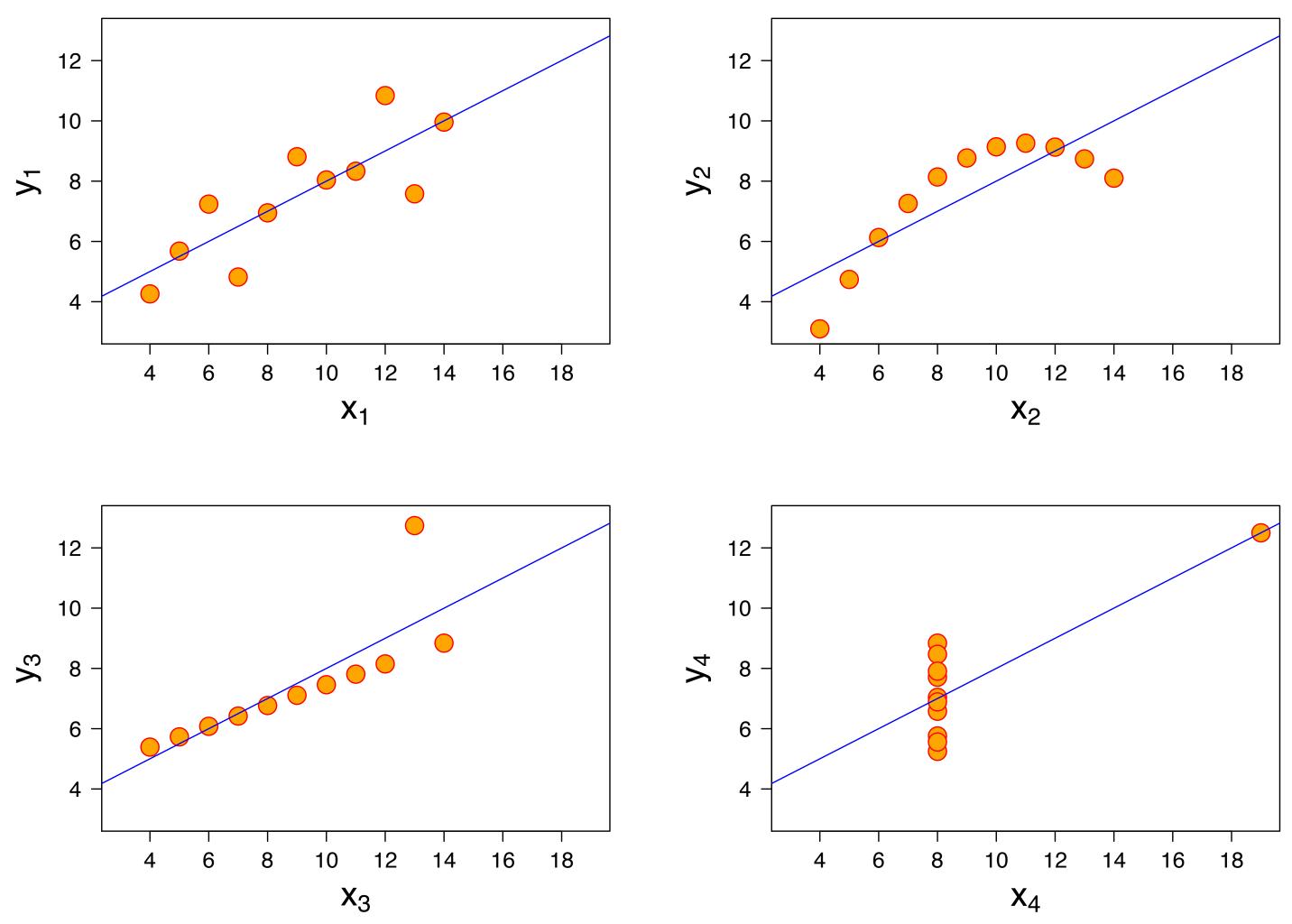








Visualizing Data





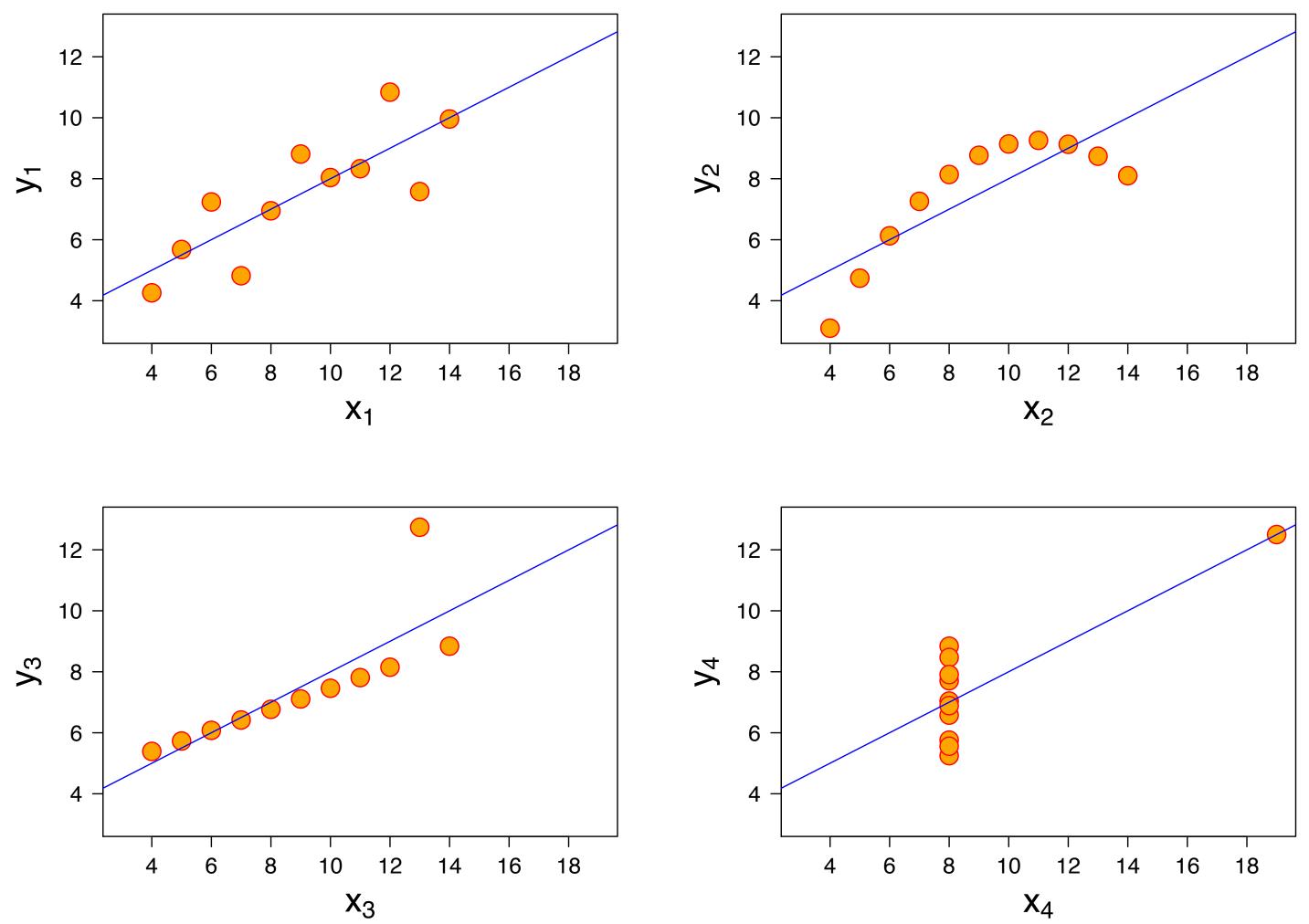








Visualizing Data



D. Koop, CSCI 503/490, Fall 2022

Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816





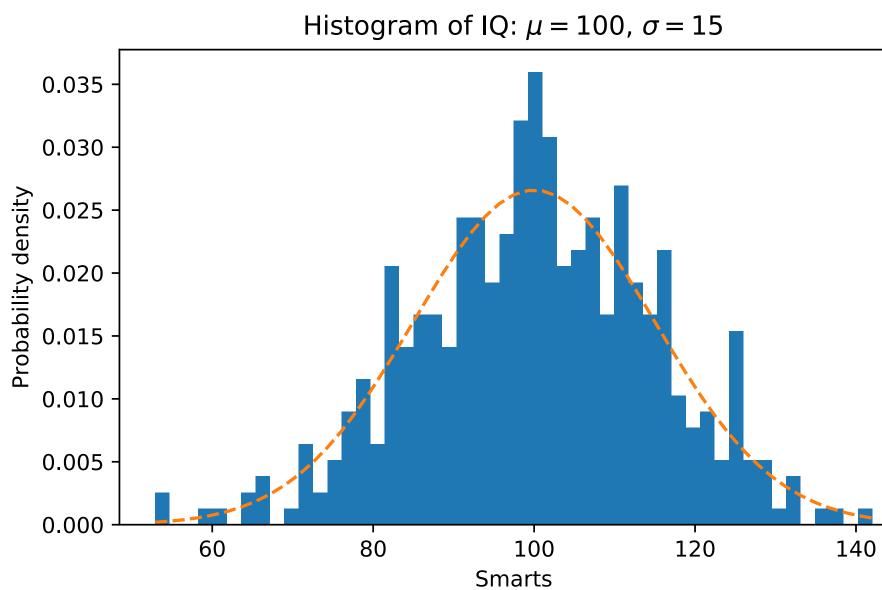


60

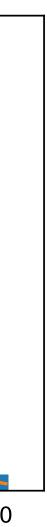


matplotlib

- Strengths:
 - Designed like Matlab
 - Many rendering backends
 - Can reproduce almost any plot
 - Proven, well-tested
- Weaknesses:
 - API is imperative
 - Not originally designed for the web
 - Dated styles





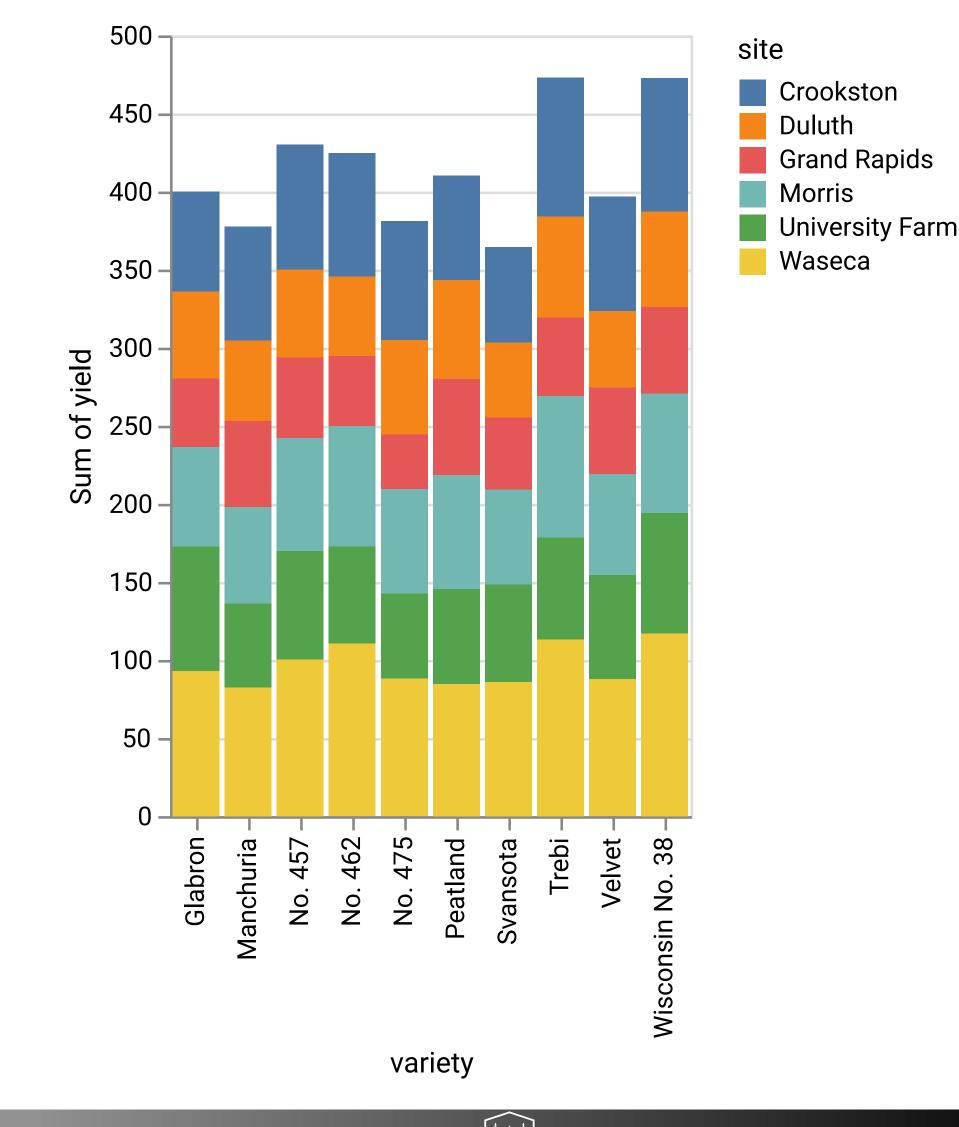




Altair

- Declarative Visualization
 - Specify what instead of how
 - Separate specification from execution
- Based on VegaLite which is browser-based
- Strengths:
 - Declarative visualization
 - Web technologies
- Drawbacks:
 - Moving data between Python and JS
 - Sometimes longer specifications

D. Koop, CSCI 503/490, Fall 2022



Northern Illinois University





Visual Marks

 \rightarrow Points

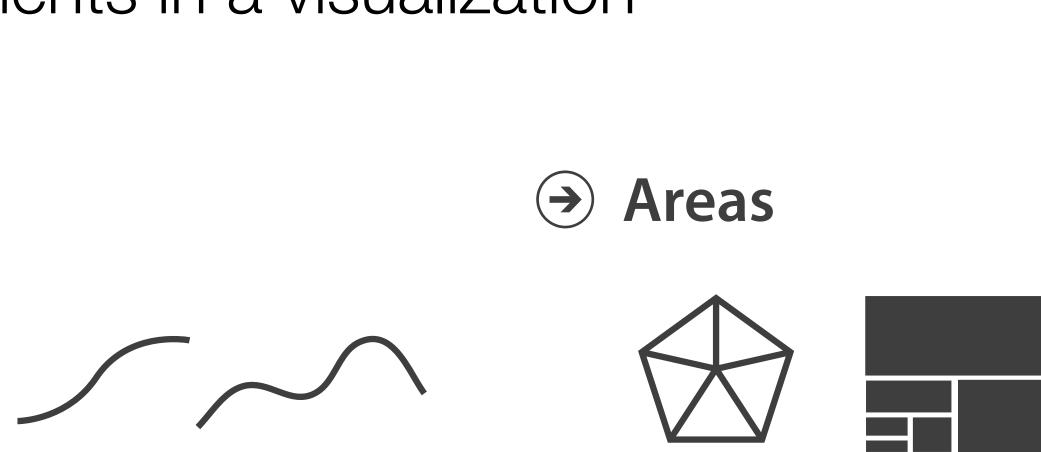
- Marks are the basic graphical elements in a visualization
- Marks classified by dimensionality:

• Also can have surfaces, volumes

 Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions

 \rightarrow Lines

- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
- Also compound marks: boxplot, errorband, errorbar

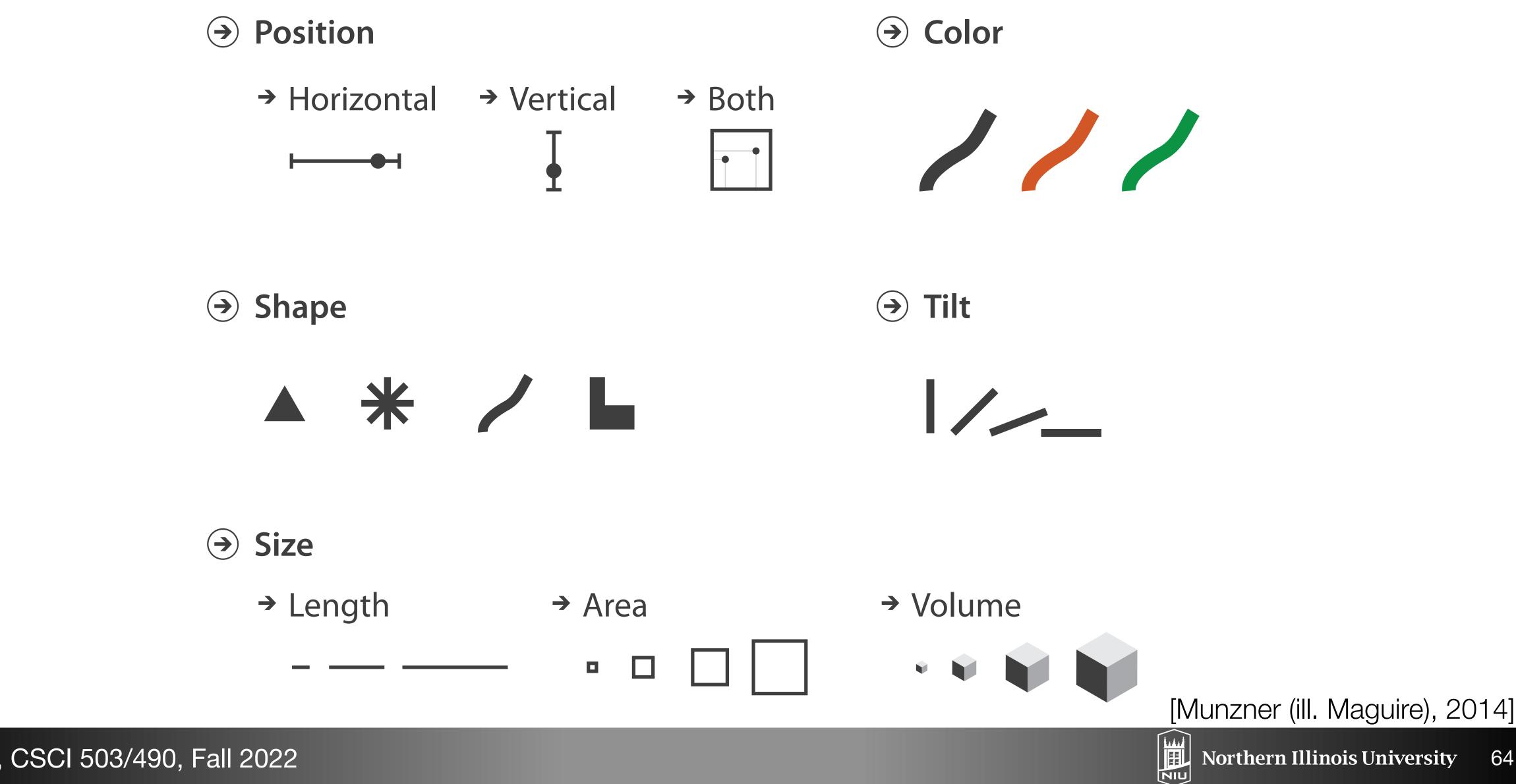






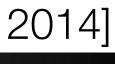


Data is Encoded via Visual Channels



D. Koop, CSCI 503/490, Fall 2022

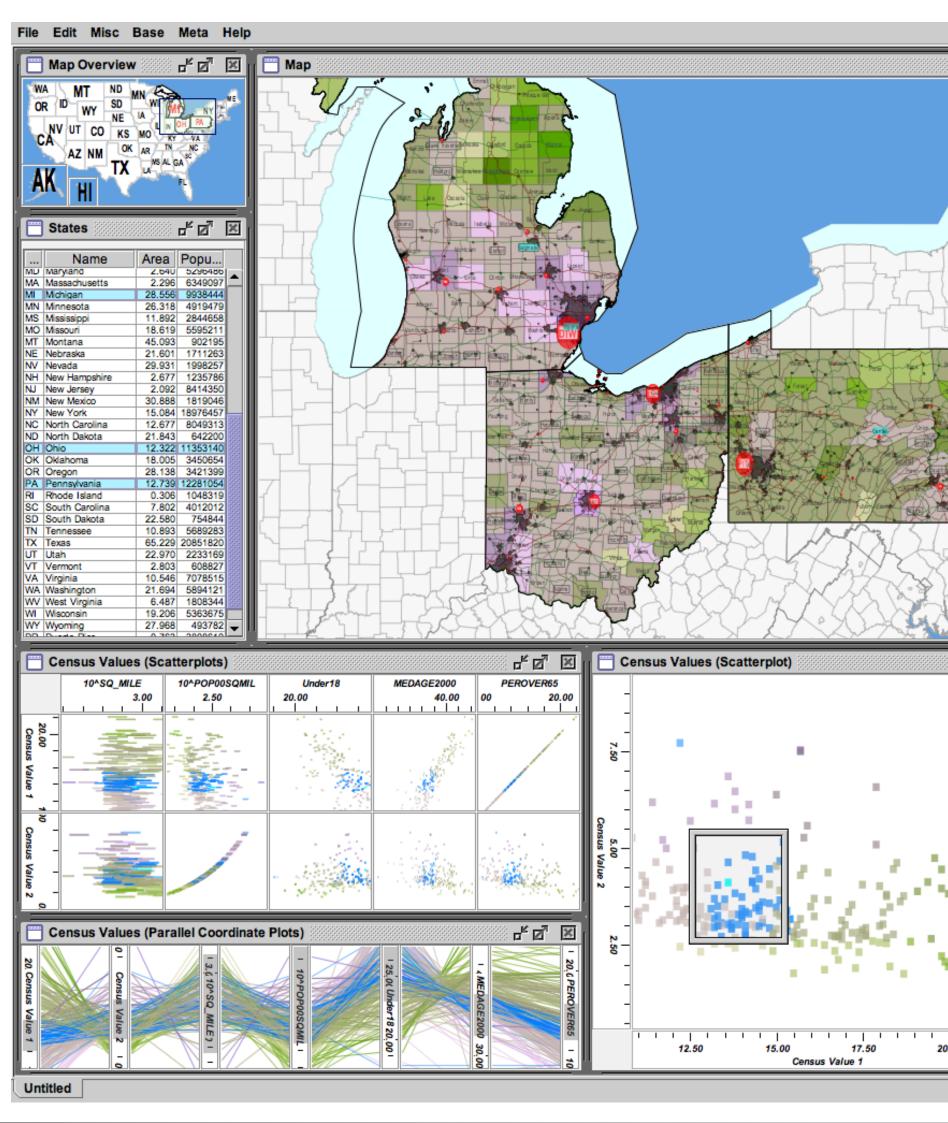
Northern Illinois University







Multiple Views



								_								_
	· 집 🛛 🛛	Counties				80 6	ø I	×	Citie	s (333)				8888 G	٢ø	×
		Name		Area	Popul	Ce	Ce		Na	me			Cou		Pop.	
1000	5 1	Montmorency Muskegon	MI MI	0.167	10315 170200	23.90 12.90	02.08		Allen Park Bellefonte		P	A Ce	ayne Co ntre Co	unty	2937 639	5
		Newaygo	MI	0.248	47874	12.80	02.75		Belleville Birch Run		N		ayne Co Saginaw	unty County	399 165	Constant and a second sec
le la		Oakland Oceana	MI	0.259	1194156 26873	14.00	06.08 02.66		Centre Hall Chesaning		P	A Ce	ntre Co		107	9
1 and		Ogernaw Ontonagon	MI MI	0.168	21645 7818		02.49		Dearborn		N	11 Wa	yne Co	unty	9777	5
13 2 M	2 1	Osceola Oscoda	MI MI	0.167	23197 9418	14.20	02.53		Dearborn He Detroit	lights		1I Wa	syne Co syne Co	unty	5826 95127	
HI	171	Otsego	MI	0.155	23301	13.70	02.59		Ecorse Flat Rock				ayne Co ayne Co		1122	
E. MIL	hi	Ottawa Presque Isle	MI MI	0.164	238314	10.10 22.30	04.53	7773	Frankenmut Garden City	-	N	1I S		County	483 3004	- Callelle
17 m 7 ?		Roscommon Saginaw	MI	0.170	25469 210039		02.64		Gibraltar		N	11 Wa	yne Co	unty	426	4 88
L yhr		Saint Clair Saint Joseph	MI	0.207	164235 62422	12.20	03.88		Grosse Poin Grosse Poin				ayne Co ayne Co		567 976	
$H \square$	276	Sanilac	MI	0.278	44547	15.40	02.61		Grosse Poin Grosse Poin			11 Wa 11 Wa	ayne Co ayne Co	unty	1244 282	
LTL	751	Schoolcraft Shiawassee	MI	0.370	8903 71687	18.60	01.66		Grosse Poin		ds N	11 Wa	syne Co	unty	1708	0
		Tuscola Van Buren	MI MI	0.234	58266 76263	12.80	02.91 03.34		Hamtramck Harper Woo			11 Wa	syne Co syne Co	unty	2297	4
and any and the	2 1 1	Washtenaw	MI	0.204	322895	08.10	04.62		Highland Pa Howard	rk			ayne Co ntre Co		1674	-
and the Dawn	_~~	Wayne Wexford		0.174	2061162 30484	14.00	07.61 02.71		Inkster Lincoln Park		N	1I Wa	ayne Co ayne Co	unty	3011 4000	_
S-C-SQUAT		Adams	OH	0.990	30484 27330	00.00	00.00		Livonia		N	11 Wa	iyne Co	unty	10054	5
and the second second		Allen	OH	0.112	108473 52523	14.20	04.05		Melvindale Merrill		N	1I S	ayne Co Saginaw	unty County	1073	
Contraction of the second		Ashtabula	OH	0.199	102728	14.70	03.48		Milesburg Millheim				ntre Co ntre Co		118	_
	-3-	Athens Auglaize	OH	0.138	62223 46611	09.30	03.33 03.28		Northville				avne Co		645	
	~~	Belmont Brown	OH OH	0.148	70226		03.38		Airp	orts &	Sea	plan	e Ba		ťø	×
STR.		Butler Carroll	OH OH	0.127	332807 28836	10.70	05.17			11.5 0		pium				
	V II	Champaign	OH	0.118	38890	12.60	03.09			Nam			En		ounty	
	1	Clark	OH OH	0.110	144742	14.70 09.40	04.36		M MBS In	Metropo ternatio		wa	1698	MI Wa	yne Co ginaw .	
		Clinton Columbiana	OH OH	0.112	40543		03.15		DET Detroit U Univers	City ity Park		_		MI Wa PA Cer		
		Coshoctor	OH	0.156	36655 46966		02.84		YIP Willow				3046	MI Wa	yne Co	
		Cuyahoga	OH	0.129	1393978	15.60	07.43									
Mar 3		Darke		0.165	53309	15.30	03.07	_								-
් ත් 🗵 🛛 🗂	Color Scheme	e ())))))))))))))))))))))))))))))))))))			- · • 🖽	City-0	City Di	stand	ces						ťø	×
S	equential Sequ	ential Non-Gra	y	•	П				Mi Saginaw County Mi	MI	MI	MI	PA ounty MI	MI	PA	M County
			_	-1					N N N	Jayne Cour	Mayne Count	Mayne County	PA Centre County MI	Wayne County M	PA Centre County	ğ
				. In the second s	ų III.				gina	Way	Way	Way	Cent	Way	Cent	Wayne
				1						8 6 8	8	5		2 8	State College Pop. 38420	e 8
									Zilwaukee Pop. 1799 1 Wyandotte	Woodhaven Pop. 12530	Westland Pop. 86602	Wayne Pop. 19051	Unionville Pop. 313 Trenton	Pop. 19584 Taylor	38.0	301
									Pop VyB	Pop 4	Pop Pop	Wayne Pop. 19	H B F	Pop. 1 Taylor	Stat Pop	Pop
					MI	ne Count		en Park 29376		0.12	0.20	0.18	5.50 0	12 0.0	5.55	0.05
					PA	ne coun	Be	lefonte	0 00 5 5	5.60	5.80	5 77	0 10 5	54 5 6	5 0 15	5 57
					Cent	re Count		p. 6395 elleville			_	-				_
		_			Way	ne Count	y Pop	p. 3997	1.34 0.34	0.25	0.15	0.13	5.76 0	.31 0.2	2 5.80	0.29
	Show	🛞 🛛 🔚 Lat	oel 🕴		MI Sagi	naw Co		rdh Run p. 1653		2 1.24	1.01	1.05	6.37 1	.27 1.14	4 6.42	1.20
	✓ Counties	Co	untie	es	PA	re Count		tre Hall p. 1079		5.70	5.90	5.88	0.20 5	64 5.7	5 0.18	5.67
	Cities	Cit			MI		Ch	esaning	0 35 1 33	1.36	1.12	1.16	6.64 1	40 1.2	6.70	1.34
▝▝	_				Sagi	naw Co		p. 2548 earborn				-	_	_		
	✓ Roads	Ro	ads		Way	ne Count	y Pop.	97775	1.37 0.11	0.19	0.22	0.21	5.49 0	18 0.12	2 5.53	0.11
	Railroads	🗌 🗌 Ra	ilroa	ds	MI Way	ne Count	earborn ly Pop.	Heights 58264		0.20	0.13	0.13	5.58 0	.22 0.10	5.63	0.15
	Airports	🖌 🖌 Air	port	s	MI	ne Count	Dee	Detroit		0.27	0.35	0.34	5.36 0	23 0.2	5.41	0.19
							M PORT	801211								
0.00	Urban Area		tes		MI	ne Count		Ecorse 11229	1 48 0 03	0.14	0.27	0.24	5.44 0	.11 0.12	2 5.48	0.06



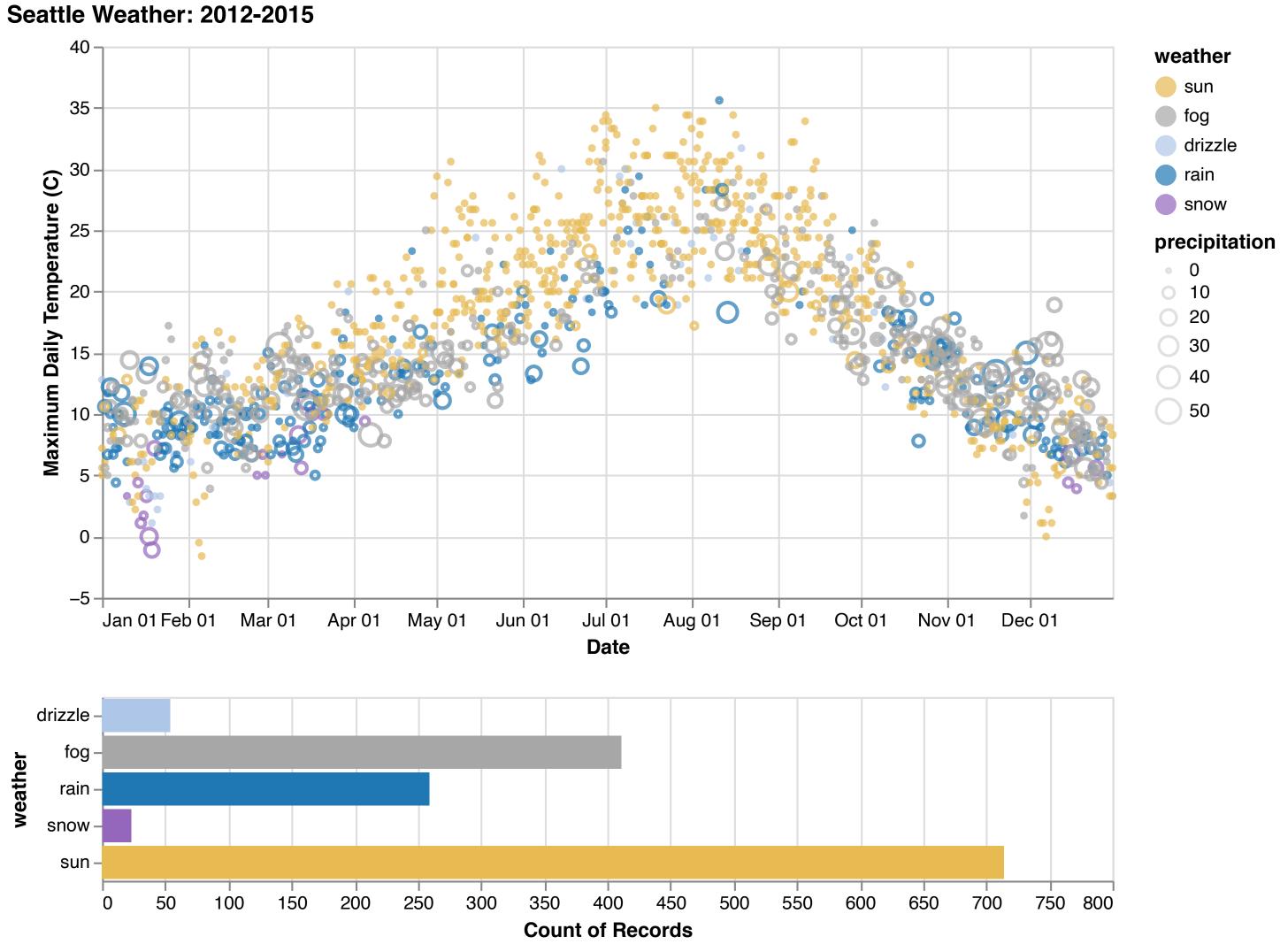








Interaction







Questions?







Final Exam

- Tuesday, December 6, **12:00**-1:50pm in PM 253
- More comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last four weeks of class (data, visualization, machine learning)
- Similar format





