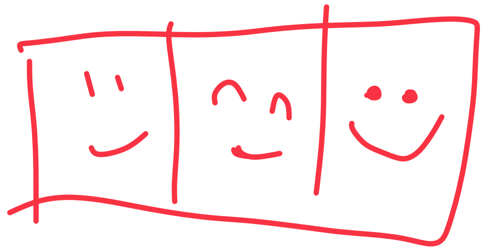
Algorithms and Data Structures for Data Science
Lists and Random Data

CS 277
Brad Solomon

January 29, 2024



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

CS Summer Research Program

About (https://cs.illinois.edu/research/undergraduate-research/srp)
This hybrid program will take place from June to August (exact dates TBD). Students who participate in the summer program engage with a faculty and their research group as well as attend weekly 'Lunch and Learn' sessions which focus on various professional development topics.

This is not a paid program – it serves as a ‘common app’ for matching students to faculty and a support and social structure for undergraduate students over the summer. Actual hiring is still handled directly by faculty individually.

Eligibility [Rolling acceptance – no fixed deadline]
Open to all undergraduates (Illinois CS and external), iCan students, and local high school students.

CRA-WP Distributed Research Experiences for Undergraduates (DREU)

About (https://cra.org/cra-wp/dreu/)

A highly selective program that matches students with a faculty mentor for a summer research experience at the faculty mentor's home institution. DREU interns will receive $700 per week for research (up to 10 weeks), and will be directly involved in a research project and interact with graduate students and professors on a daily basis. This experience is invaluable for those who are considering graduate school; DREU will provide a close-up view of what graduate school is really like and increase interns' competitiveness as an applicant for graduate admissions and fellowships.

Eligibility [Student and mentor application deadline February 15th]

Students who are pursuing an undergraduate degree at an institution in the U.S. or its territories. International Students may apply, however, most of the funds for the DREU program are restricted to US citizens and permanent residents, so the number of non-US student participants will be limited. Priority will be given to persons from populations underrepresented in computing including women, Black/African American, Native American/Alaskan Native/ Pacific Islander, Hispanic/Latinx, LGBTQAI+, Persons with Disabilities, and Veterans.

DaRin Butz Foundation Research Scholars

About (https://isur.engineering.illinois.edu/darin-butz-foundation-research-scholars/)

DaRin Butz Foundation Research Scholars conduct research in the areas of computer science, aerospace, electrical, computer, materials science, nuclear engineering, physics, or astronomy. Scholars will work with faculty mentors who will supervise, guide, and instruct them on their research during the course of the project. They are expected to do research 30–35 hours per week for 10 weeks in summer. Scholars must take ENG 199 UGR in the fall and will present their work in the Fall Engineering Research Fair or the annual ISUR poster expo in the spring semester.

Eligibility [Student application deadline March 31st]

Must be:

- A University of Illinois woman undergraduate in the Grainger College of Engineering
- U.S. citizen or permanent resident
- Rising sophomore, junior, or senior majoring in CS, aerospace, electrical, computer, materials science, nuclear engineering, physics, or astronomy with a GPA of 3.0 or higher

SIAM-Simons Undergraduate Summer Research Program

About (https://www.siam.org/students-education/programs-initiatives/siam-simons-undergraduate-summer-research-program)

Society for Industrial and Applied Mathematics (SIAM) is pleased to announce we are accepting applications for the SIAM-Simons Undergraduate Summer Research Program, which will provide research, networking, and mentorship opportunities to U.S. students from underrepresented groups. Participating students will receive a stipend of $1,000/week and will have their housing, meals, and travel expenses paid. This is an amazing opportunity for students to immerse themselves in applied math, computational science, and/or data science research while simultaneously participating in professional development and community-building activities designed to foster a strong sense of belonging.

Eligibility [Student application deadline February 7th]

Must be a U.S. citizen or permanent resident and an undergraduate student enrolled in a US-based college or university in September 2024. Note that while all projects will have an applied math and/or computational science approach, students do not need to have an applied math background to apply. Projects appropriate for students at all undergraduate levels will be available, and prior research experience is not required.

Reminder: First two labs due today!

If you need an extension, email the request and reason why.

Labs are meant to be completable in the lab section (or shortly after)

Currently labs are due the Monday after they are released

CS 277 Plagiarism Policy

Github Co-pilot and chatGPT can probably pass this class.

Using these tools is against the course policy and will hurt you on exams

It also hurts your peers by giving a false signal on assessments

The labs in CS 277 are designed to give you programming practice

The MPs in CS 277 are designed to be data science explorations

Learning Objectives

Motivate the importance of the list data structure → ADT

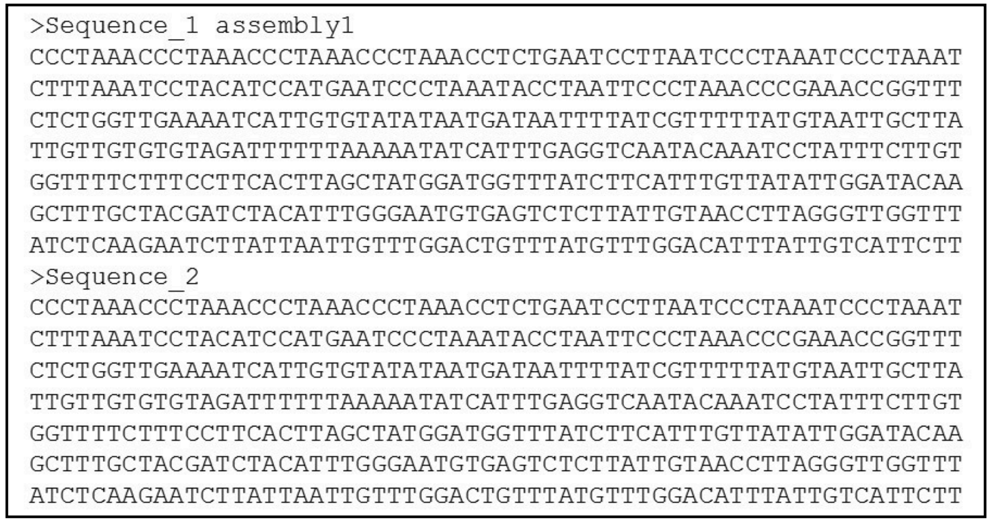
Practice using built-in Python lists

Introduce file I/O in Python

Introduce random() package

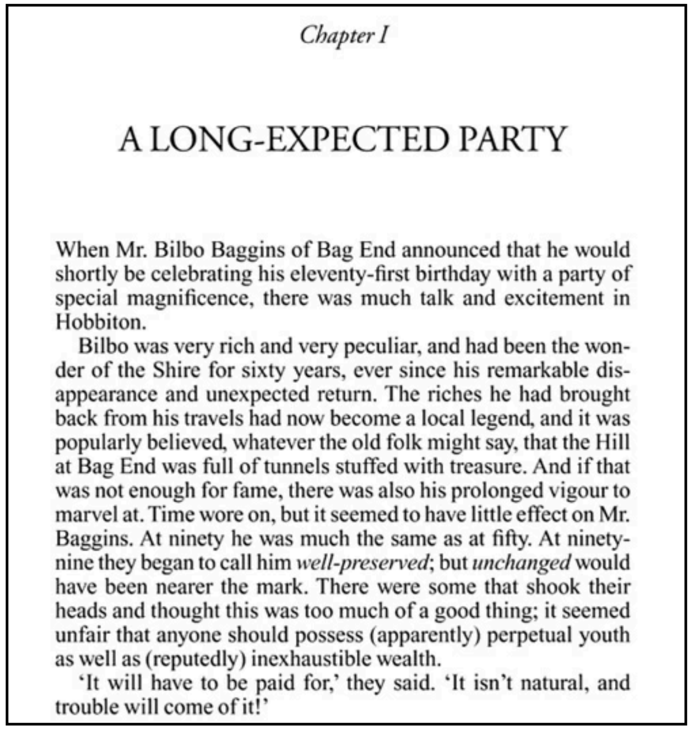
Mini – Project 0

Lists are a great way to store **data**



```
>Sequence_1 assembly1
CCCTAAACCCTAAACCCTAAACCCTAAACCTCTGAATCCTTAATCCCTAAATCCCTAAAT
CTTTAAATCCTACATCCATGAATCCCTAAATACCTAATTCCCTAAACCCGAAACCGGTTT
CTCTGGTTGAAAATCATTGTGTATATAATGATAATTTTATCGTTTTTATGTAATTGCTTA
TTGTTGTGTGTAGATTTTTTAAAAATATCATTTGAGGTCAATACAAATCCTATTTCTTGT
GGTTTTCTTTCCTTCACTTAGCTATGGATGGTTTATCTTCATTTGTTATATTGGATACAA
GCTTTGCTACGATCTACATTTGGGAATGTGAGTCTCTTATTGTAACCTTAGGGTTGGTTT
ATCTCAAGAATCTTATTAATTGTTTGGACTGTTTATGTTTGGACATTTATTGTCATTCTT
>Sequence_2
CCCTAAACCCTAAACCCTAAACCCTAAACCTCTGAATCCTTAATCCCTAAATCCCTAAAT
CTTTAAATCCTACATCCATGAATCCCTAAATACCTAATTCCCTAAACCCGAAACCGGTTT
CTCTGGTTGAAAATCATTGTGTATATAATGATAATTTTATCGTTTTTATGTAATTGCTTA
TTGTTGTGTGTAGATTTTTTAAAAATATCATTTGAGGTCAATACAAATCCTATTTCTTGT
GGTTTTCTTTCCTTCACTTAGCTATGGATGGTTTATCTTCATTTGTTATATTGGATACAA
GCTTTGCTACGATCTACATTTGGGAATGTGAGTCTCTTATTGTAACCTTAGGGTTGGTTT
ATCTCAAGAATCTTATTAATTGTTTGGACTGTTTATGTTTGGACATTTATTGTCATTCTT
```

Fasta → List



Chapter I

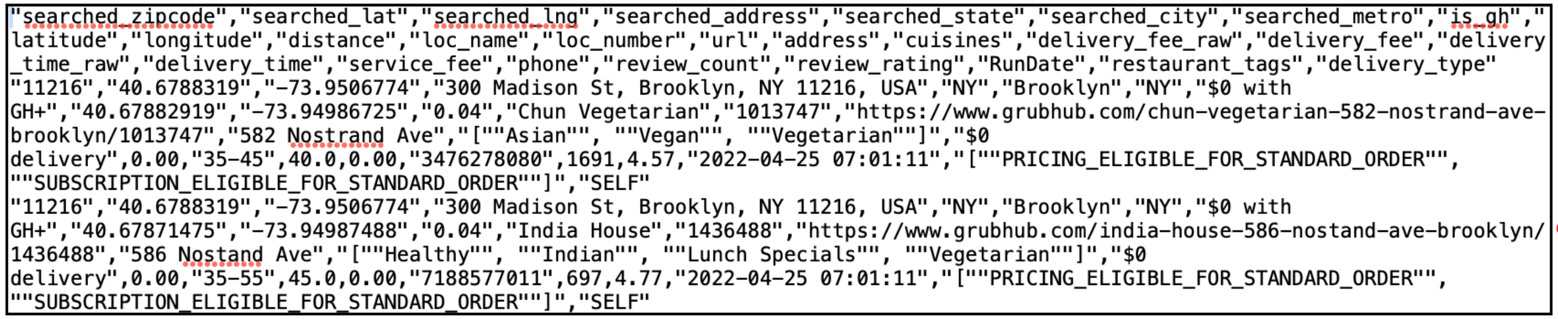
A LONG-EXPECTED PARTY

When Mr. Bilbo Baggins of Bag End announced that he would shortly be celebrating his eleventy-first birthday with a party of special magnificence, there was much talk and excitement in Hobbiton.

Bilbo was very rich and very peculiar, and had been the wonder of the Shire for sixty years, ever since his remarkable disappearance and unexpected return. The riches he had brought back from his travels had now become a local legend, and it was popularly believed, whatever the old folk might say, that the Hill at Bag End was full of tunnels stuffed with treasure. And if that was not enough for fame, there was also his prolonged vigour to marvel at. Time wore on, but it seemed to have little effect on Mr. Baggins. At ninety he was much the same as at fifty. At ninety-nine they began to call him *well-preserved*; but *unchanged* would have been nearer the mark. There were some that shook their heads and thought this was too much of a good thing; it seemed unfair that anyone should possess (apparently) perpetual youth as well as (reputedly) inexhaustible wealth.

'It will have to be paid for,' they said. 'It isn't natural, and trouble will come of it!'

Textbook?

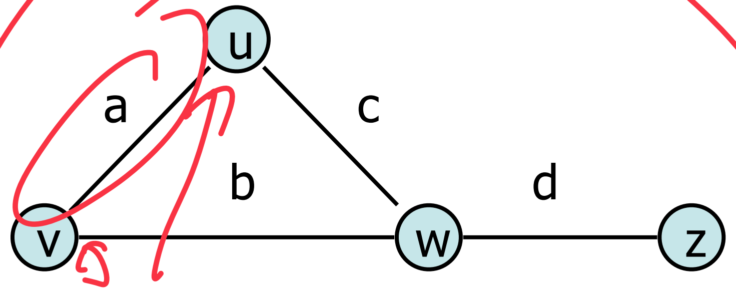


```
"searched_zipcode","searched_lat","searched_lng","searched_address","searched_state","searched_city","searched_metro","is_gh","
latitude","longitude","distance","loc_name","loc_number","url","address","cuisines","delivery_fee_raw","delivery_fee","delivery
_time_raw","delivery_time","service_fee","phone","review_count","review_rating","RunDate","restaurant_tags","delivery_type"
"11216","40.6788319","-73.9506774","300 Madison St, Brooklyn, NY 11216, USA","NY","Brooklyn","NY","$0 with
GH+","40.67882919","-73.94986725","0.04","Chun Vegetarian","1013747","https://www.grubhub.com/chun-vegetarian-582-nostrand-ave-
brooklyn/1013747","582 Nostrand Ave","[""Asian"", ""Vegan"", ""Vegetarian""]","$0
delivery",0.00,"35-45",40.0,0.00,"3476278080",1691,4.57,"2022-04-25 07:01:11","[""PRICING_ELIGIBLE_FOR_STANDARD_ORDER"",
""SUBSCRIPTION_ELIGIBLE_FOR_STANDARD_ORDER""]","SELF"
"11216","40.6788319","-73.9506774","300 Madison St, Brooklyn, NY 11216, USA","NY","Brooklyn","NY","$0 with
GH+","40.67871475","-73.94987488","0.04","India House","1436488","https://www.grubhub.com/india-house-586-nostand-ave-brooklyn/
1436488","586 Nostand Ave","[""Healthy"", ""Indian"", ""Lunch Specials"", ""Vegetarian""]","$0
delivery",0.00,"35-55",45.0,0.00,"7188577011",697,4.77,"2022-04-25 07:01:11","[""PRICING_ELIGIBLE_FOR_STANDARD_ORDER"",
""SUBSCRIPTION_ELIGIBLE_FOR_STANDARD_ORDER""]","SELF"
```

CSV

List of lists

Lists are a great way to store **data structures**



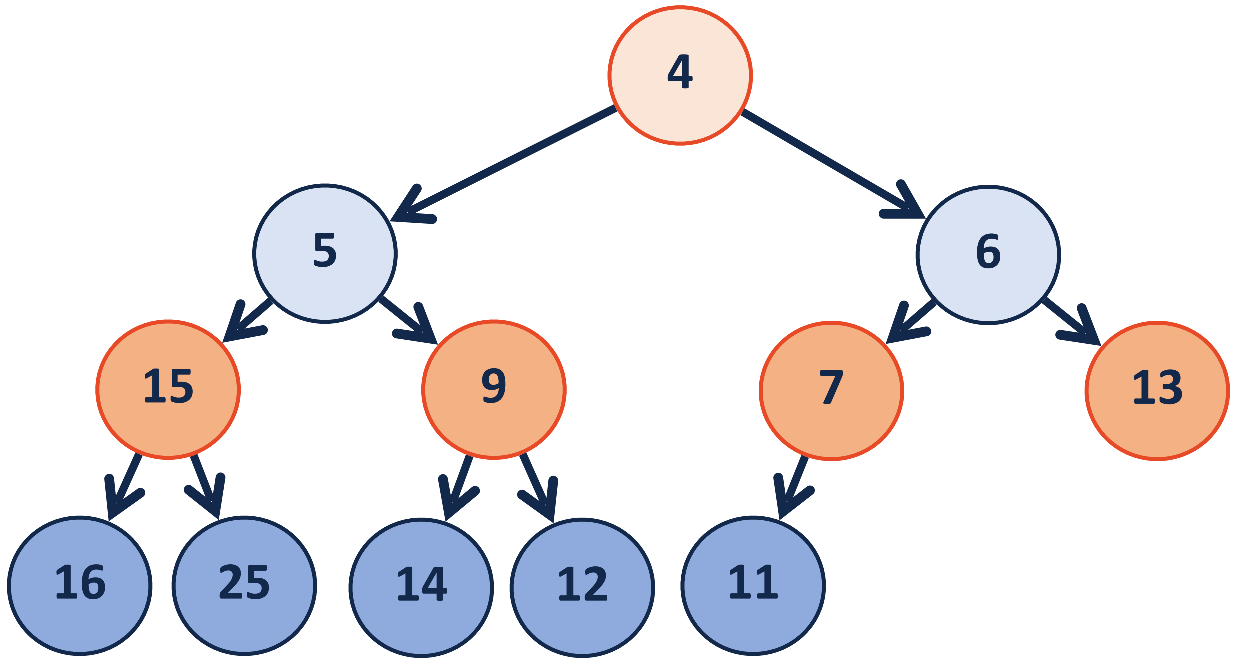
vertices

Edge list

u
v
w
z

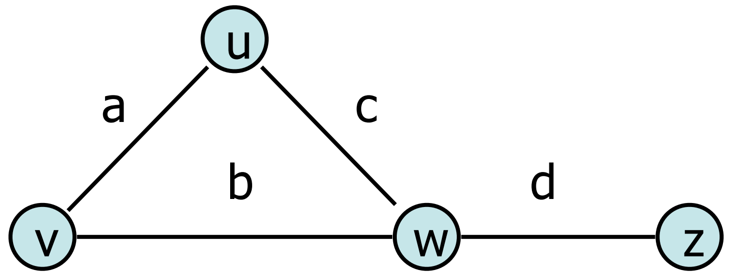
u	v	a
v	w	b
u	w	c
w	z	d

weights



	4	5	6	15	9	7	20	16	25	14	12	11			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

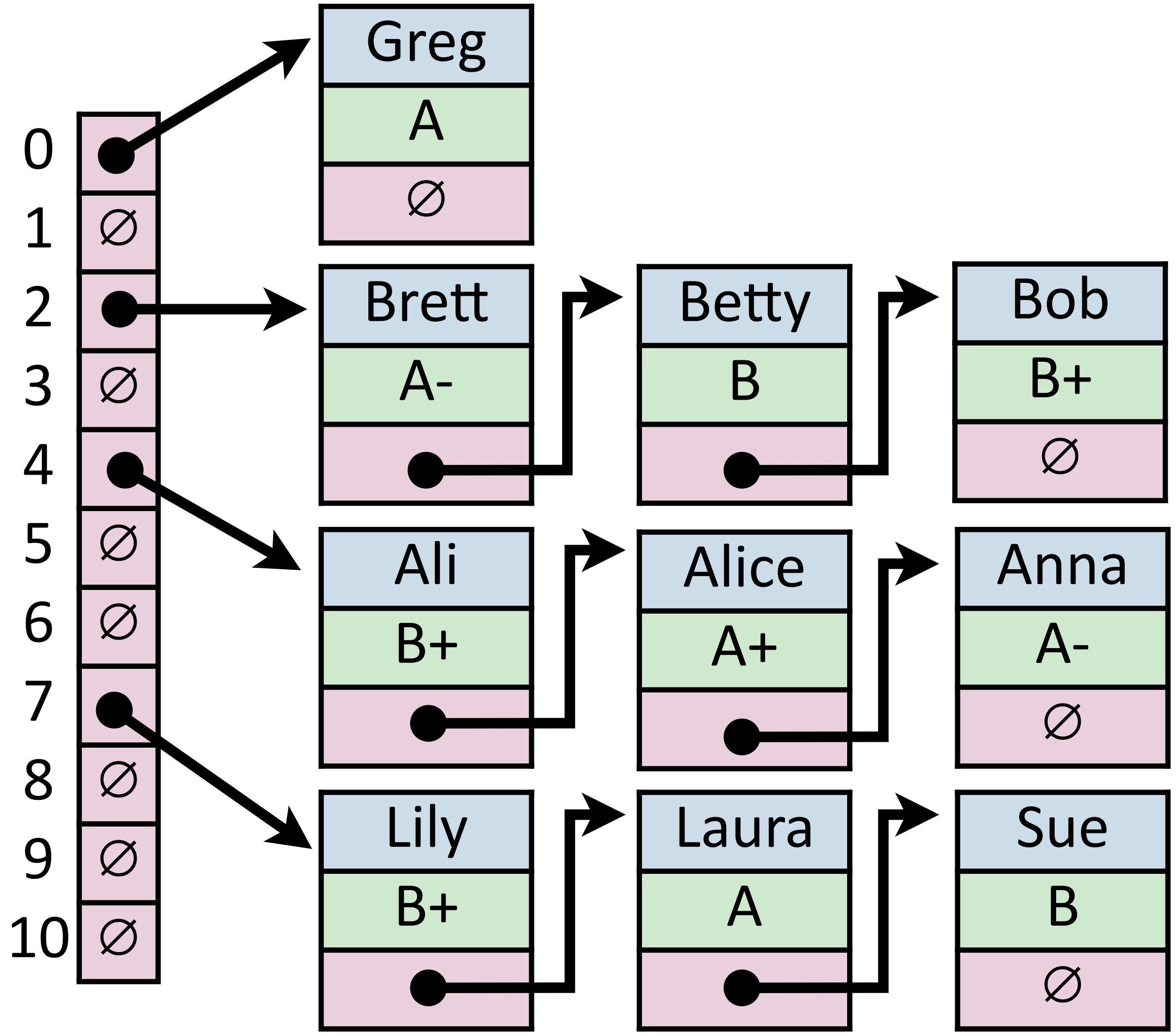
Lists are a great way to store **data structures**



u
v
w
z

u	v	a
v	w	b
u	w	c
w	z	d

Lists are a great way to store **data structures**



$$H = \{h_1, h_2, \ldots, h_k\}$$

0
0
1
0
0
1
0
1
0
0

List Abstract Data Type

Flexible!

A list is an **ordered** collection of items

Items can be either **heterogeneous** or **homogenous** (data)

Objects ↙ store different items

↓ store same object type

The list can be of a **fixed size** or is **resizable**

Mutability (modify size or contents) ↙ efficiency

Can be **localized** or **distributed** in memory

List Abstract Data Type

A minimally functional list must have the following functions:

Constructor: Create a new empty list

Insert: Add an item to a list

Delete: Remove an item from the list

Index: Look up the value at a specific index

Size():** Get the size of the list

Lists in Python: Constructor

There are many ways to construct (or initialize) a Python list.

```
l1 = [1, 2, 3]


l2 = list( (3, 4, 5) )


l3 = l1+l2


l4 = l1.copy()


l5 = l1



```

Lists in Python: Insert

`Insert()` tries to add the object at a particular index in the list

`Append()` adds the item to the end of the list

```python
l1 = []


l1.insert(2, "A")


l1.append("C")


l1.insert(1, "B")


l1.insert(5, "D")
```

why do we have append different from insert?

(Efficiency)

Lists in Python: Remove

`Remove()` removes the first instance of the object

`Remove()` crashes if the object doesn't exist!

`Pop()` removes and returns the object at a specific index (default is last)

```
l1 = [1,2,1,3,1,4]


l1.remove(7)


l1.remove(1)


x = l1.pop()


y = l1.pop(2)



```

Lists in Python: Index

`Index()` returns the index of the first matching item in the list

`__getItem__(), []` returns the index of the first matching item in the list

```
l1 = [1,2,1,3,1,4]

print(l1.index(1))


print(l1.index(5))


print(l1.index(4))


print(l1[0])


print(l1[10])


print(l1[-1])
```

Lists in Python: Size

len() is not a class method but works for most forms of a list in Python

Remember: Python indexing starts at 0

```
l1 = [1,2,1,3,1,4]

print(len(l1))
```

List Abstract Data Type

A minimally functional list must have the following functions:

Constructor:	`__init__()`	
Insert:	`append(x)`	`insert(i, x)`
Delete:	`remove(x)`	`pop()`
Index	`__getitem__() []`	`index(x)`
Size()**	`len(list)`	

Programming Practice: Lists

What is the output of the print statement after this function is run?

```python
def inclass():
    l = [1,2,3,4,5,6]
    x = l.pop()
    l.pop()
    l.pop()
    l.insert(0, x)
    l.append(8)
    l += [1, 2]
    return l

print(inclass())
```

Programming Practice: Index of Every X

Given a list and a character as input, return a list containing the index of every instance of that character.

Programming Practice: Index of Every X

Given a list and a character as input, return a list containing the index of every instance of that character.

Now lets remove every third value from the output list.

Python List

There are many implementations of lists in Python. Here are three*:

```python
myList = [1, 2, 3, 4, 5]

print(myList)

print(len(myList))

print(myList[2])
```

```python
myTuple = (1, 2, 3, 4,
5)

print(myTuple)

print(len(myTuple))

print(myTuple[2])
```

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

print(myNP)

print(len(myNP))

print(myNP[2])
```

Why are there so many different implementations?

Lets find out together in our first class mini-project!

P1) Generate random datasets

P2) Code various analysis functions on datasets using lists

P3) Measure efficiency of **P2** using varying size datasets from **P1**

Mini-Project 0: Random Data Generation

Learning Objectives:

Practice string and list manipulations in the context of file I/O

Introduce seeded random data generation

Introduce how to measure runtime efficiency in Python

Investigate the efficiency of different Python implementations of lists

Programming Toolbox: File I/O

```python
readableFile = open('inputFile.txt', 'r')




writableFile = open('outputFile.txt', 'w')




carefulWriteFile = open('outputFile.txt', 'x')




appendableFile = open('outputFile.txt', 'a')
```

Programming Toolbox: File I/O

Which approach do you prefer?

```
# Approach 1

readableFile = open('inputFile.txt', 'r')


fileData = readableFile.read()


readableFile.close()


# Approach 2

with open('inputFile.txt', 'r') as myFile:
    fileData = myFile.read()
```

Programming Toolbox: File I/O

```python
with open('data/temp1.txt', 'x') as myFile:
    for i in range(10):
        myFile.write(str(i))
    myFile.write("\n")
    myFile.write("Line 2")



myFile = open('data/temp2.txt', 'w')
for i in range(5):
    myFile.write(str(i) + "\n")
myFile.close()



with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")

with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")



```

Programming Toolbox: File I/O

```python
with open('data/temp1.txt', 'x') as myFile:
    for i in range(10):
        myFile.write(str(i))
    myFile.write("\n")
    myFile.write("Line 2")



myFile = open('data/temp2.txt', 'w')
for i in range(5):
    myFile.write(str(i) + "\n")
myFile.close()



with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")

with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")
```

```
0123456789
Line 2
```

temp1.txt

```
0
1
2
3
4

```

temp2.txt

Programming Toolbox: File I/O

```python
with open('data/temp1.txt', 'x') as myFile:
    for i in range(10):
        myFile.write(str(i))
    myFile.write("\n")
    myFile.write("Line 2")



myFile = open('data/temp2.txt', 'w')
for i in range(5):
    myFile.write(str(i) + "\n")
myFile.close()



with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")

with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")
```

```
0123456789
Line 2
```

temp1.txt

```
0
1
2
3
4
Hello World!

```

temp2.txt

Programming Toolbox: File I/O

```
with open('data/temp1.txt', 'x') as myFile:
    for i in range(10):
        myFile.write(str(i))
    myFile.write("\n")
    myFile.write("Line 2")



myFile = open('data/temp2.txt', 'w')
for i in range(5):
    myFile.write(str(i) + "\n")
myFile.close()



with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")

with open('data/temp2.txt', 'a') as myFile:
    myFile.write("Hello World!\n")
```

```
0123456789
Line 2
```

temp1.txt

```
0
1
2
3
4
Hello World!
Hello World!

```

temp2.txt

Programming Toolbox: File I/O

```python
with open('data/temp1.txt') as myFile:
    inList = myFile.readlines()
print(inList)





myFile = open('data/temp2.txt')
for i in range(10):
    print("Line Content: {}".format(myFile.readline()))
myFile.close()





with open('data/temp1.txt') as myFile:
    print(myFile.read())
```

```
0123456789
Line 2
```

temp1.txt

```
0
1
2
3
4
```

temp2.txt

Programming Toolbox: File I/O

```python
with open('data/temp1.txt') as myFile:
    inList = myFile.readlines()
print(inList)





myFile = open('data/temp2.txt')
for i in range(10):
    print("Line Content: {}".format(myFile.readline()))
myFile.close()





with open('data/temp1.txt') as myFile:
    print(myFile.read())
```

```
0123456789
Line 2
```

temp1.txt

```
0
1
2
3
4
```

temp2.txt

```
['0123456789\n', 'Line 2']
Line Content: 0

Line Content: 1

Line Content: 2

Line Content: 3

Line Content: 4

Line Content:
Line Content:
Line Content:
Line Content:
Line Content:
0123456789
Line 2
```

Programming Toolbox: File I/O

`str.strip()` will remove whitespace from the string.

```
0
1
2
3
4

```

temp2.txt

```python
myFile = open('data/temp2.txt')
for i in range(6):
    print("Line Content: {}".format(myFile.readline().strip()))
myFile.close()

with open('data/temp2.txt') as myFile:
    for line in myFile:
        print(line.strip())
```

```
Line Content: 0
Line Content: 1
Line Content: 2
Line Content: 3
Line Content: 4
Line Content: 
0
1
2
3
4
```

Programming Toolbox: File I/O

`str.strip()` will remove whitespace from the string.

```
tmp = "1, 2, 3"
tmp2 = "1,2,3"

x = tmp.split(",")
y = tmp2.split(",")
for i in range(len(x)):
    if x[i]!=y[i]:
        print("No match!", x[i], y[i])
    else:
        print("Match!")


```

Programming Toolbox: File I/O

Four ways to open a file: read, write, carefulWrite, append

Three ways to read a file: all lines, line-by-line, text as string

Writing to file: String formatting is key, file writes are literal copies

Python File I/O

Whats wrong here:

```

with open('dataFile2.txt','w') as myFile:
    data = myFile.readlines()

for line in data:
    print(line)
```

Programming Practice: File I/O

Given an input filename, write a new file that is the same file with reversed lines.

Mini-Project 0: Random Data Generation

Learning Objectives:

Practice string and list manipulations in the context of file I/O

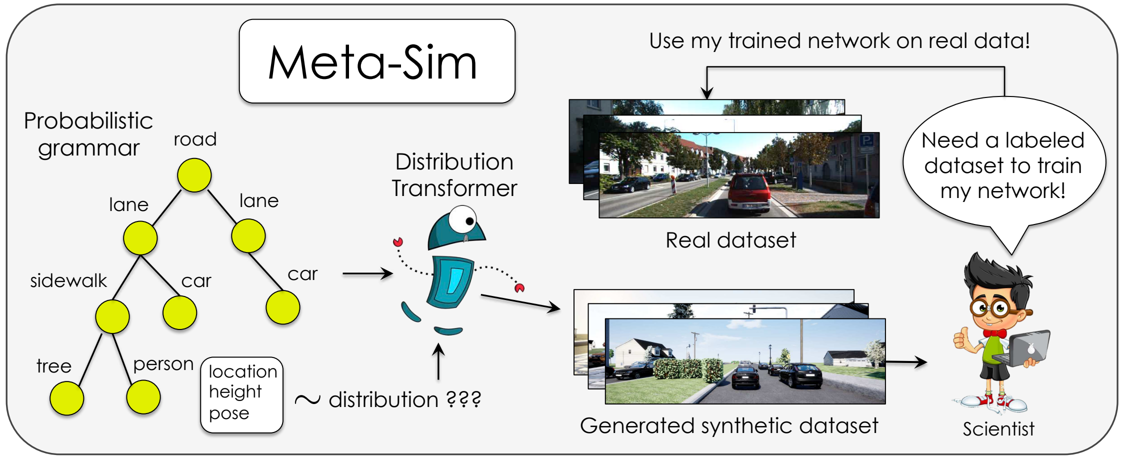
Introduce **seeded random data generation**

Introduce how to measure runtime efficiency in Python

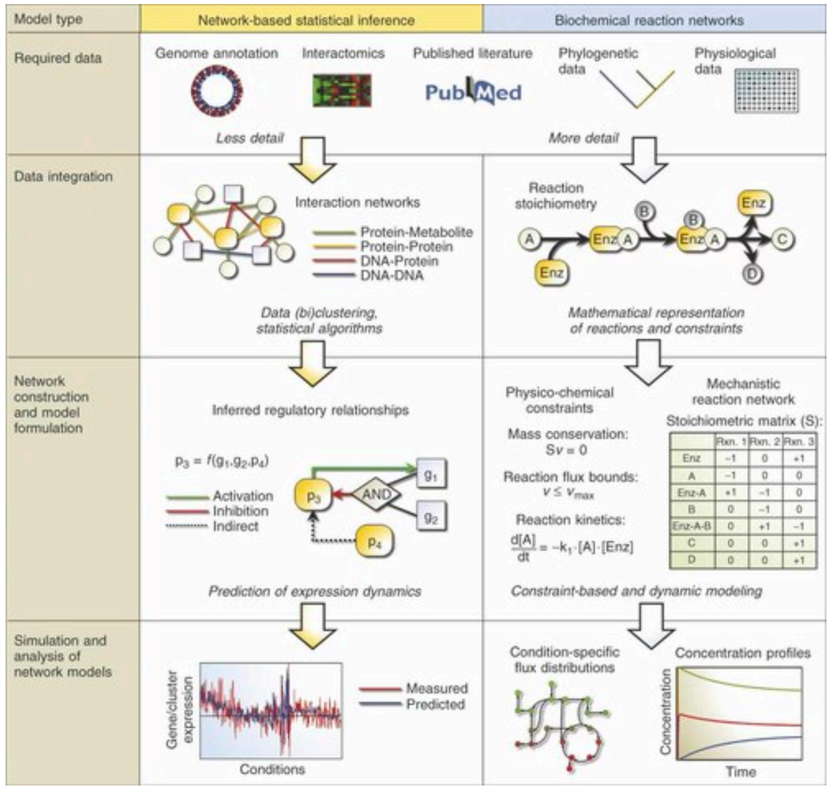
Investigate the efficiency of different Python implementations of lists

Programming Toolbox: Synthetic Datasets

Performance Analysis of K-Means and K-Medoids Clustering Algorithms For A Randomly Generated Data Set. T Velmurugan et al 2008



Meta-Sim: Learning to Generate Synthetic Datasets. A Kar et al. 2019


Model type
Network-based statistical inference
Biochemical reaction networks
Required data
Genome annotation
Interactomics
Published literature
PubMed
Phylogenetic data
Physiological data
Less detail
More detail
Data integration
Interaction networks
Protein-Metabolite
Protein-Protein
DNA-Protein
DNA-DNA
Reaction stoichiometry
Enz
A
B
C
D
Data (bi)clustering, statistical algorithms
Mathematical representation of reactions and constraints
Network construction and model formulation
Inferred regulatory relationships
p3 = f(g1,g2,p4)
Activation
Inhibition
Indirect
AND
Physico-chemical constraints
Mass conservation: Sv = 0
Reaction flux bounds: v ≤ vmax
Reaction kinetics: d[A]/dt = −k1·[A]·[Enz]
Mechanistic reaction network
Stoichiometric matrix (S):
Prediction of expression dynamics
Constraint-based and dynamic modeling
Simulation and analysis of network models
Gene/cluster expression
Conditions
Measured
Predicted
Condition-specific flux distributions
Concentration profiles
Concentration
Time

	Rxn. 1	Rxn. 2	Rxn. 3
Enz	−1	0	+1
A	−1	0	0
Enz-A	+1	−1	0
B	0	−1	0
Enz-A-B	0	+1	−1
C	0	0	+1
D	0	0	+1

In silicon models of cancer. Edelman et al. 2010

"In many applications of computers and other electronic devices, there is a need for a physical source of true random numbers, for example, **in computer simulations of various probabilistic algorithms,** computer games, and, most notably, in possibly randomized cryptographic algorithms and protocols whose security relies on the ability to generate unpredictable secret keys and random numbers."

New Methods for Digital Generation and Postprocessing of Random Data. J.D.J Golic 2006

Programming Toolbox: Random.Random()

Python's random module implements **pseudo-random** number generators

What does pseudo-random mean? Well try running the following:

```
Random.random()
```

$\leftarrow$ Creating $0 \leq x < 1$

```
Random.seed(1)
```

```
Random.random()
```

Programming Toolbox: Random.Seed()

Python's random module implements **pseudo-random** number generators

2^{19937}

The values produced are not actually random!

Given a start value, the output is entirely deterministic!

```
Random.random()

Random.seed(1)

Random.random()                 0.13436424411240122
```

randomly starts at point

but has fixed sequence

Programming Toolbox: Random.Seed()

Python’s random module implements **pseudo-random** number generators

The values produced are not actually random!

Given a start value, the output is entirely deterministic!

This is actually quite useful (for us):

1. Total reproducibility
2. Autogradable randomness
3. It’s built-in

Programming Toolbox: Random.RandInt()

While most uniform random data generation can be done using random(), some other functions are useful for specific tasks.

```
Random.randint(min, max)
```

$\rightarrow$ $min \leq x \leq Max$

$\uparrow$ inclusive

$\hookrightarrow$ integer

Programming Toolbox: Random.Choice()

While most uniform random data generation can be done using random(), some other functions are useful for specific tasks.

```
Random.choice(list)
```

$[1, 2, 3]$

↳ pick one at random

Programming Toolbox: Random.Sample()

While most uniform random data generation can be done using random(), some other functions are useful for specific tasks.

```
Random.sample(list, num)
```

|

Pick num items from list no replacement

↳ num unique items from list

Programming Toolbox: Random.shuffle()

While most uniform random data generation can be done using random(), some other functions are useful for specific tasks.

```
Random.shuffle(list)
```

Programming Toolbox: Lots more to random

https://docs.python.org/3/library/random.html

Programming Practice: Uniform Random

Lets visualize uniform randomness using lists!

Write a function that takes in two numbers, a length and count, and generates count random variables across length. Instead of storing the sequence, we will only store the counts assigned to each value.