The Code-a-thon: Improving Student Engagement through Community Coding

The Code-a-thon: Improving Student Engagement through Community Coding

Charlie Dey, Educator
Training and Professional Development
Texas Advanced Computing Center
Austin, TX
charlie@tacc.utexas.edu

Je'aime Powell, Senior System Administrator
Data Management & Collections
Texas Advanced Computing Center
Austin, TX
jpowell@tacc.utexas.edu

Boyd Wilson, CEO
Omnibond Systems, LLC
Liberty, SC
boyd@omnibond.com

Dr. Linda Hayden, Director
Center of Excellence in Remote Sensing
Education and Research
Elizabeth City State University
Elizabeth City, NC
haydenl@mindspring.com

Learning By Doing

Learning by Doing also known as **Active Learning** is a hands-on, experiential approach to learning that involves actively engaging in tasks or activities to acquire knowledge and skills.

Learning By Doing

Increased Retention and Understanding: By actively participating in tasks or activities, students develop a deeper understanding of concepts and principles. Through hands-on experience, learners can connect theoretical knowledge to real-world applications, making the learning more meaningful and memorable.

Learning By Doing

Skill Development: Our code-a-thons promote the development of practical computing skills that are essential in high-performance computing. By engaging in actual tasks, solving actual computational models, learners have the opportunity to refine their abilities, build competence, and gain confidence in their skills.

Learning By Doing

Critical Thinking and Problem-Solving: When faced with challenges during hands-on activities, learners must think creatively and analytically to find solutions. This process cultivates the ability to think on their feet, adapt to different situations, and develop problem-solving strategies, which are valuable skills in scientific computation

The Role of Active Learning in Computational Science

- Hands on Coding Projects
- Collaborative Learning
- Peer Teaching
- In-Class Discussions
- Flipped Classroom
- Gamification - Coding Challenges
- The Feedback Loop

Rational: Facing the challenges of teaching computational science

Teaching computational science comes with its own set of challenges. There are a slew of topics that need to be covered to **understand and comprehend** and traditional teaching methods may not be enough to fully grasp the topic.

Some of these challenges are the **complexity of concepts**, **learning the syntax and grammar** of modern computer languages, **problem solving**, **abstraction**, **collaborative coding**, and most importantly, **computational thinking and critical thinking**.

The Hackathon Model

Traditional classroom education alone may not suffice to fully prepare students for the challenges they will encounter in their careers.

Enter hackathons, dynamic and immersive events using active learning as an effective tool for teaching and learning computational science. Hackathons contribute to computational science education by **fostering innovation**, **problem-solving skills**, **teamwork**, and **real-world application**.

The Hackathon Model

Hackathons: A Brief Overview

Hackathons are **intensive**, **time-bound** events where teams of participants **come together to collaboratively work on solving real-world problems** or **creating innovative software projects**.

The Hackathon Model

Hackathons as Learning Platforms

- Hands-On Learning
- Problem Solving
- Interdisciplinary Collaboration
- Time Management
- Creativity and Innovation
- Teamwork and Communication
- Networking
- Real world Problems

The Code-a-thon

Code-a-thons borrowed quite a bit from the Hackathon Model. From a learning perspective, they are used to **attain the same goals**. A codeathon has a more narrow **focus primarily** on **iterative coding**, **algorithmic development, over-the-shoulder peer coding**

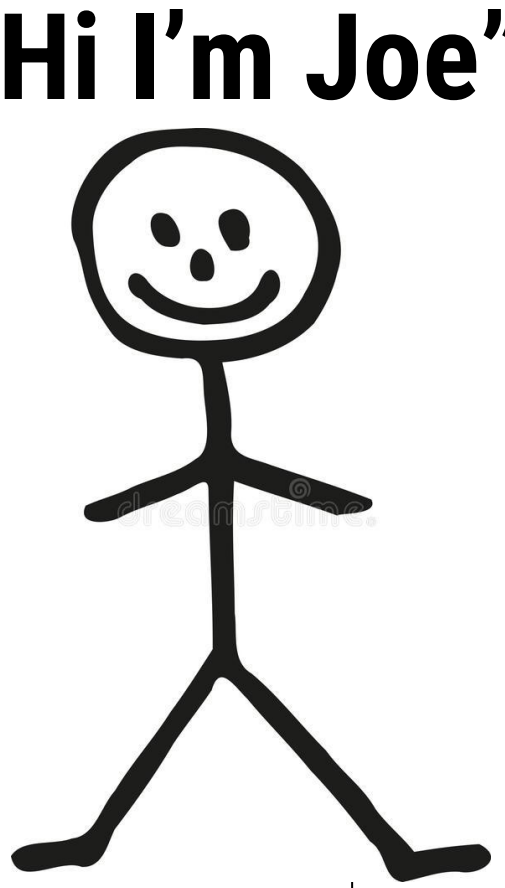
The Code-a-thon

Participants typically **engage** in **coding challenges or competitions**, where each **challenge builds on the previous challenge**. These challenges are algorithmic or data structure-related and **each challenge combines together to become a major project**.

The Code-a-thon

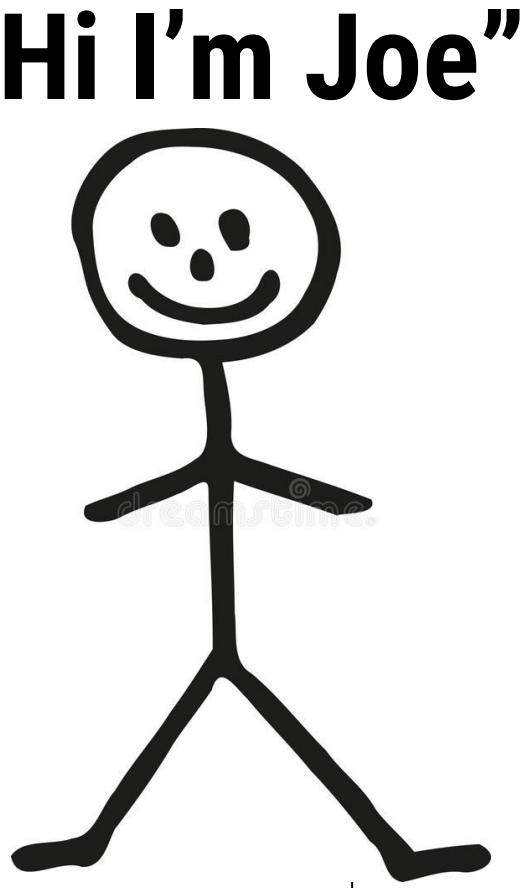
The focus here is for **students to understand** that as a **challenge becomes more complex, the code becomes more complex, sometimes unnecessarily**. Students have to become **creative** in **simplifying their code** in order **to solve the more complex challenge**. In order to simplify the code, they have to **learn** and **implement functions**, **modules**, **object oriented programming**, **data analytics**, **databases** and **dataframes**.

Think about a Scientific Process



Think about a Scientific Process

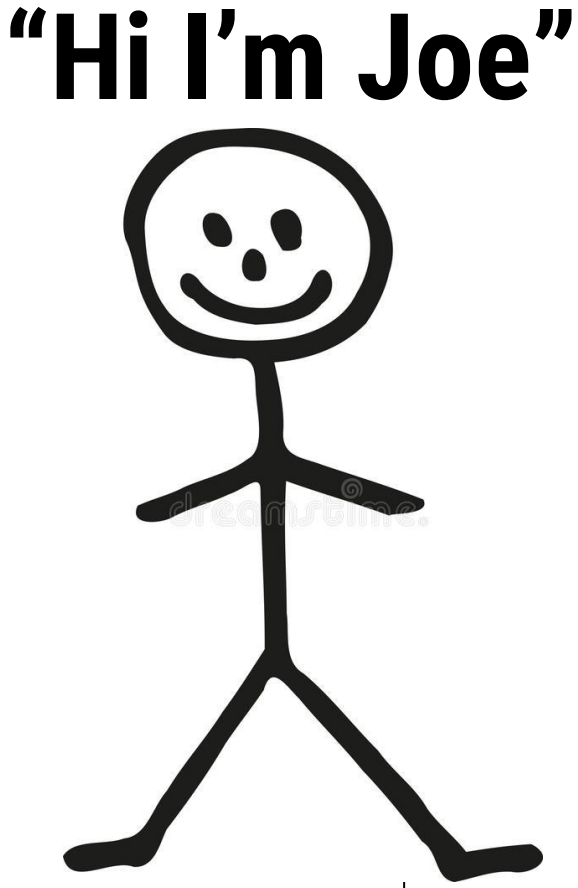
Let's meet Joe.



Think about a Scientific Process

Let's meet Joe.

Joe might get sick.

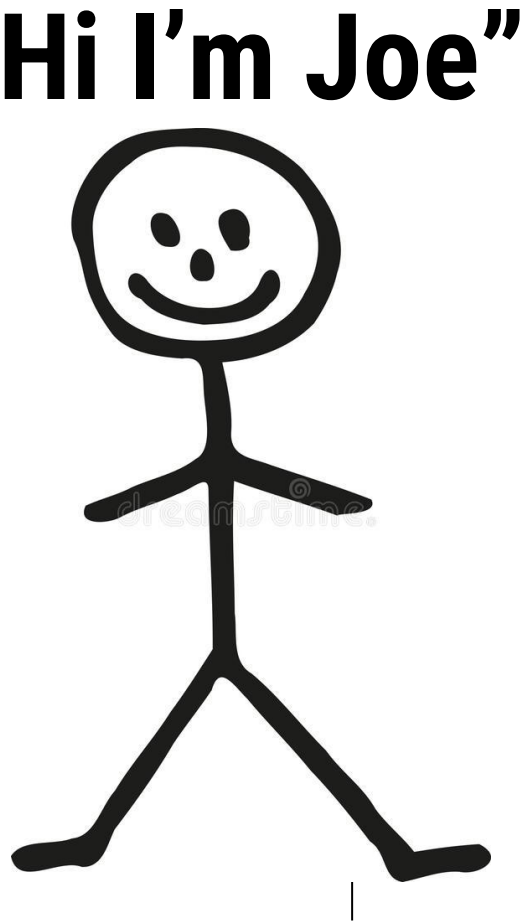


Think about a Scientific Process

Let's meet Joe.

Joe might get sick.

Joe will be sick for 5 days.



Think about a Scientific Process

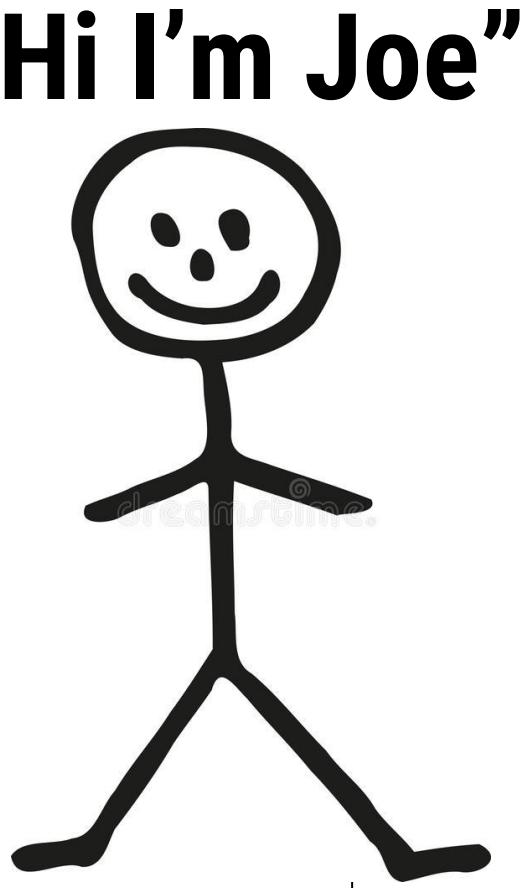
Let's meet Joe.

Joe might get sick.

Joe will be sick for 5 days.

After 5 days, Joe gets better.

Once Joe gets better, Joe can no longer get sick.



Think about a Scientific Process

Let's meet Joe.

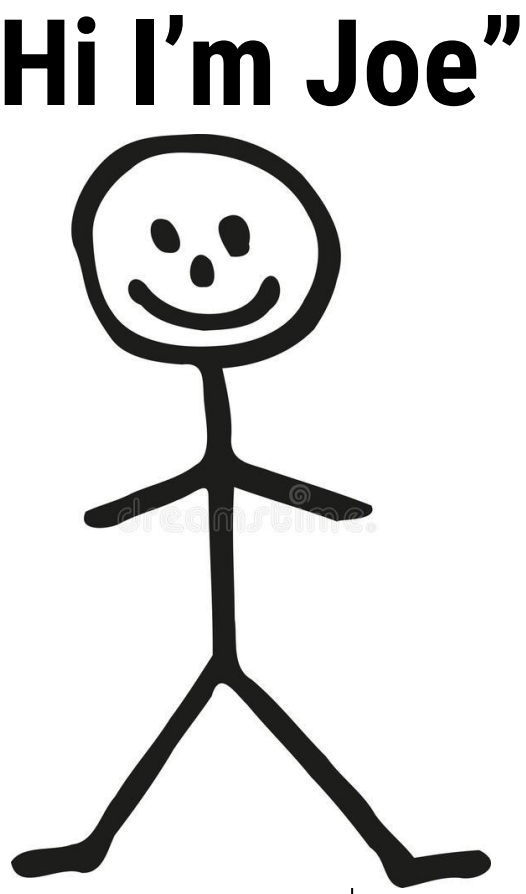
Joe might get sick.

Joe will be sick for 5 days.

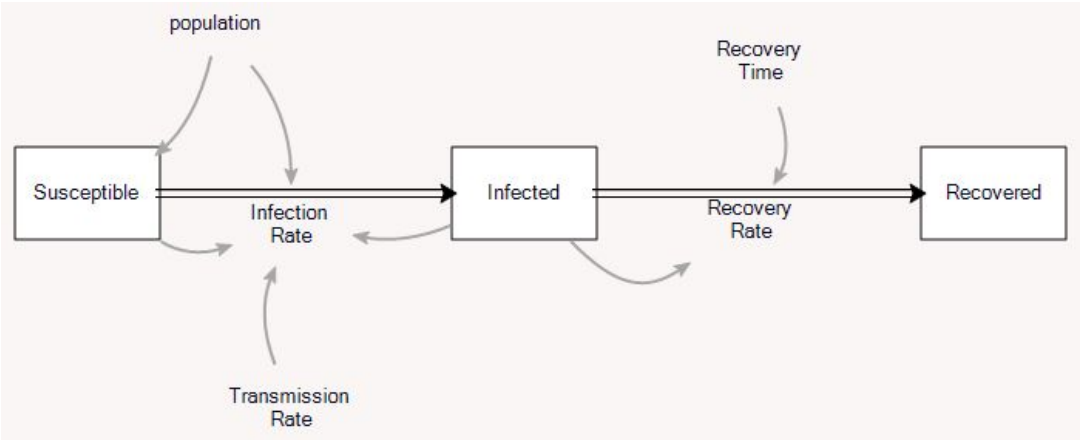
After 5 days, Joe gets better.

Once Joe gets better, Joe can no longer get sick.

How would we "code" Joe?



The SIR Model



Task 1 - Code Joe

Variables to hold data

Mathematical Operations to do math :)

Conditionals to make decisions

Loops to repeat our process

Functions/Subroutines to reuse code

Objects or Classes to define our "things"

Let's meet Joe.

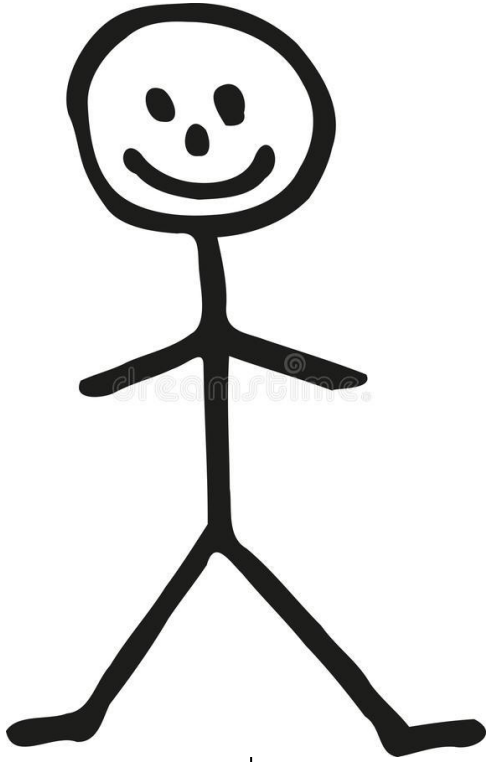
Joe might get sick.

Joe will be sick for 5 days.

After 5 days, Joe gets better.

Once Joe gets better, Joe can no longer get sick.

Let's "code" Joe.



Task 2 Code Joe and Jane

We met Joe.

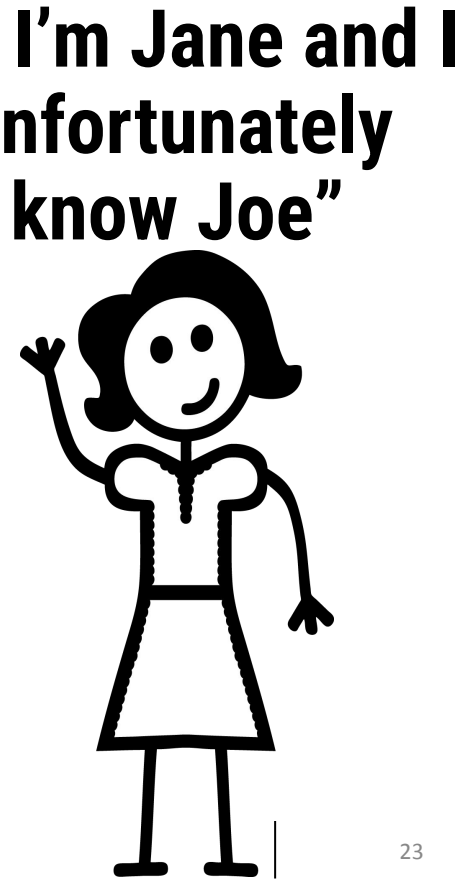
Joe has a friend, Jane

If Joe gets sick, Jane might get sick.

Every day Joe is sick, there's a chance for Jane to get sick.

Modify your code, so when Joe gets sick that triggers Jane to roll a random number to see if Jane gets sick.

Loop through your code until both Joe and Jane get sick and they each get better.

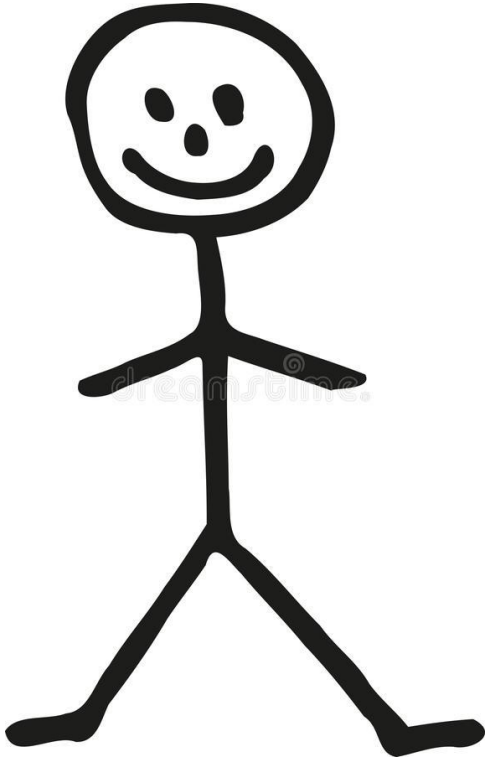


Task 3 Code A Person Object.

Joe, is now of type *Person*. In our "world":

- Joe can get infected
- Joe can get sick for certain number of days
- Joe can get better - once better, let's assume Joe can't ever get sick again
- Joe can get vaccinated - let's assume that the vaccine is awesome, so once again, once vaccinated, Joe can't get sick again.

Code a Person object with the above criteria, and "instantiate" Joe. Now, go back to Task 1, and make Joe the Person Object, sick for 5 days using the same algorithm.



Task 4 Code the Population

We now have a Person object, we now need a Population object

- The Population is basically a Vector of Person objects
- The Population Object will need methods to add a person, return the number of Persons, return the number of sick Persons, well Persons, inoculated Persons
- Your Population will start out at 1000 Persons

Task 5 Code One Day of Interactions

The Person also interacts with a number of other Persons each day

The Person Object will have a "Interactions" Vector
For each Person, randomly fill the Interactions Vector with 10 other Persons.

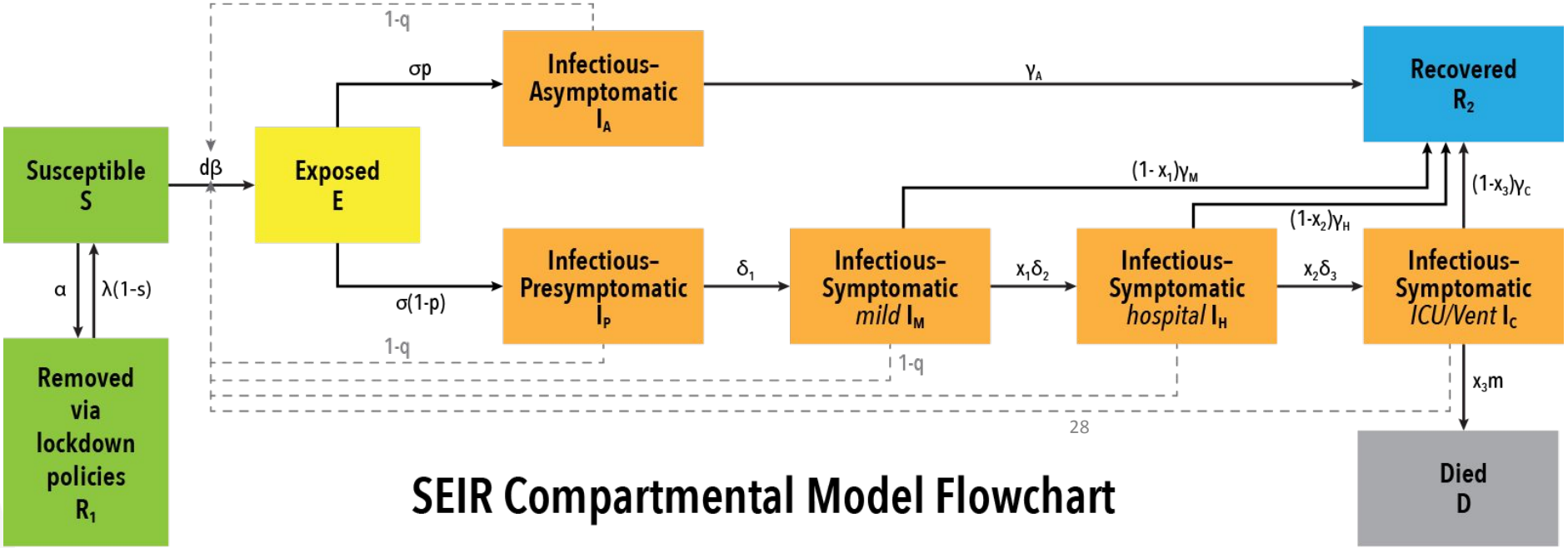
The Interactions do not need to be reciprocal, meaning if Person P2 shows up in Person P1's Interaction Vector, P1 doesn't necessarily have to show up in P2's Interaction Vector (it should! but we should keep our simulation simple)

Task 6 Code the Population
with full Interactions and a propagating Contagion

We now have a Person object and a Population object. Remember, once a person gets better, they can no longer get sick, they are inoculated

- Set a population of 10000 people
- Introduce Patient Zero
- This is your first sick person
  - Day 1. Patient Zero interacts with others, those other may now get sick as well.
- As each Day passes
  - sick people are interacting with well people, causing the virus to spread
  - each day print out the number of sick people, and the number of inoculated people
- The simulation ends, when there are no more sick people.

The Covid SEIR Model



SEIR Compartmental Model Flowchart

Task 7 Code a Mitigation

You see that the Peak in a 10,000 people community is very high

What are the benefits of Masking?

How do Masks affect the infection rate?

Your challenge: Modify your models so that a certain percentage of your Population wears masks, and model the Mask benefits.

Final Thoughts



Active learning is a **powerful educational approach** that aligns with the dynamic and problem-solving nature of Computational Science. By **engaging students** in **hands-on coding projects**, **collaborative learning experiences**, and **interactive discussions**, active learning not only **enhances comprehension** but also **equips students** with the **practical skills** and **critical thinking abilities** necessary for success in the field.

*images were created using Bing Image Creator, using the text on the slide as the image prompt

The Code-a-thon: Improving Student Engagement through Community Coding

Charlie Dey, Educator
Training and Professional Development
Texas Advanced Computing Center
Austin, TX
charlie@tacc.utexas.edu

Je'aime Powell, Senior System Administrator
Data Management & Collections
Texas Advanced Computing Center
Austin, TX
jpowell@tacc.utexas.edu

Boyd Wilson, CEO
Omnibond Systems, LLC
Liberty, SC
boyd@omnibond.com

Dr. Linda Hayden, Director
Center of Excellence in Remote Sensing
Education and Research
Elizabeth City State University
Elizabeth City, NC
haydenl@mindspring.com