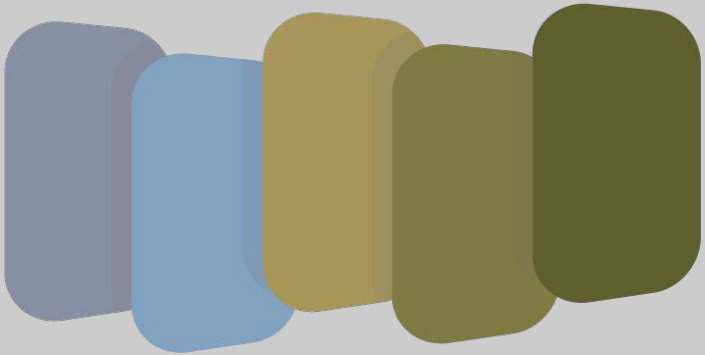
Level Up!
Your Scientific Coding

Level 3: Object-oriented Programming



CSDMS

community surface
dynamics modeling system

Benjamin Campforts

benjamin.campforts@colorado.edu
@bcampforts

Mark Piper

mark.piper@colorado.edu
@mdpiper

CSDMS
community surface dynamics modeling system

Purpose

What's our goal here?

It's worth it for you, a busy geoscientist, to take the time to learn, and use, object-oriented programming.

You'll save time and energy in the long run.

CSDMS
community surface dynamics modeling system

Agenda

Here's what we're showing you in the next ~30 minutes.

- What?
- Why?
  - grad student
  - postdoc
  - researcher
  - professor
- Where?
  - our favorite online resources
- How?
  - a short demo of OOP in Python

What?

What is object-oriented programming (OOP)?

- OOP is a way of modeling a programming problem
- Concepts: class, object
- Terms: attribute, method
- "A bike" (class) vs. "my 1996 Specialized Stumpjumper FS" (object)

Why?

Why should I use OOP if I'm a grad student?

- Organize code used for your thesis
- Job skill

Why?

Why should I use OOP if I'm a postdoc?

- Improve code for journal articles
- Model development
- Productivity tool

CSDMS
community surface dynamics modeling system

Why?

Why should I use OOP if I'm a research scientist?

- Easier to create and maintain complex code
- Easier for others to collaborate and contribute

CSDMS
community surface dynamics modeling system

Why?

Why should I use OOP if I'm a professor?

- Train students and postdocs to be more productive
- Easier to manage and direct software projects

Where?

Here are some of our favorite resources for learning how to use OOP effectively.

- Java tutorials
- Wikipedia
- A Simple Explanation of OOP
- Object-oriented programming in 7 minutes | Mosh

CSDMS
community surface dynamics modeling system

How?

A live demonstration of using OOP.

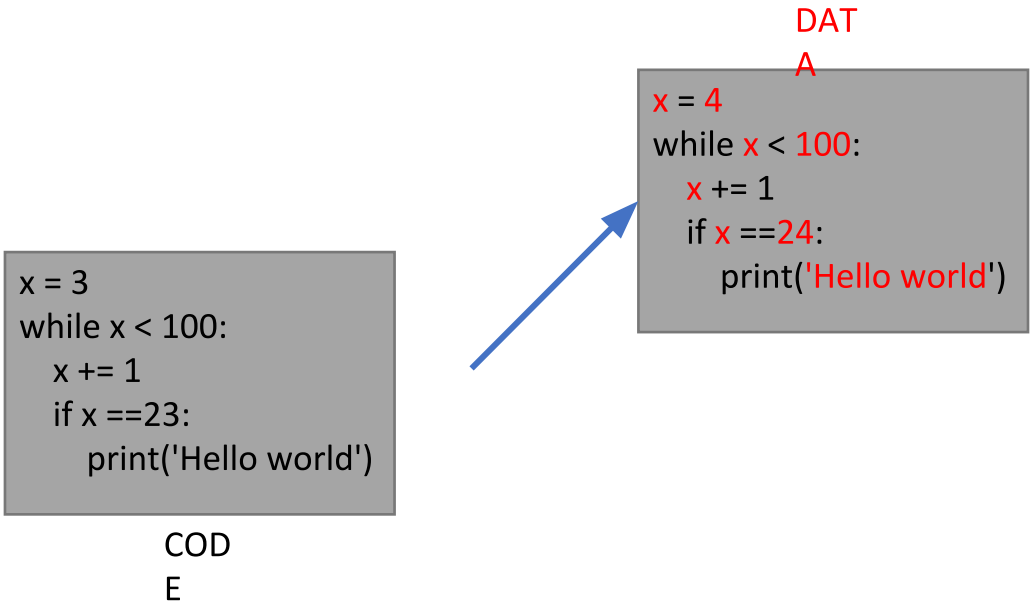
github.com/csdms/level-up

What is a program

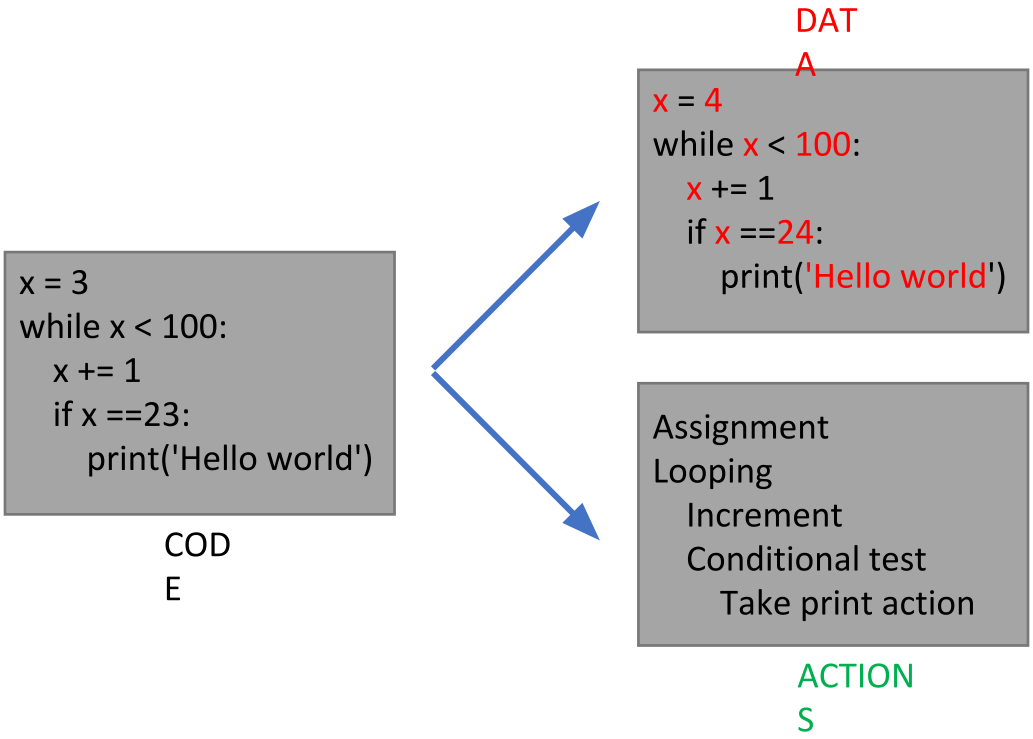
```
x = 3
while x < 100:
    x += 1
    if x ==23:
        print('Hello world')
```

COD
E

What is a program



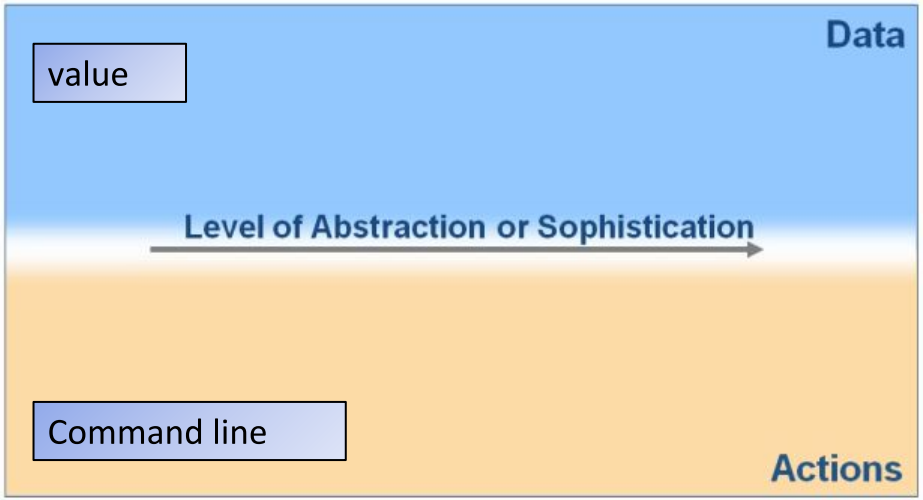
What is a program



Progression of programming techniques

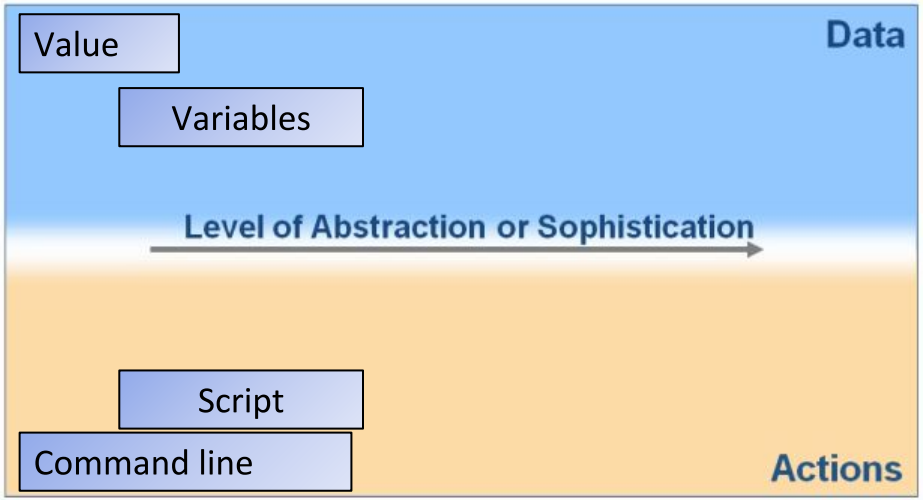
Based on

Progression of programming techniques



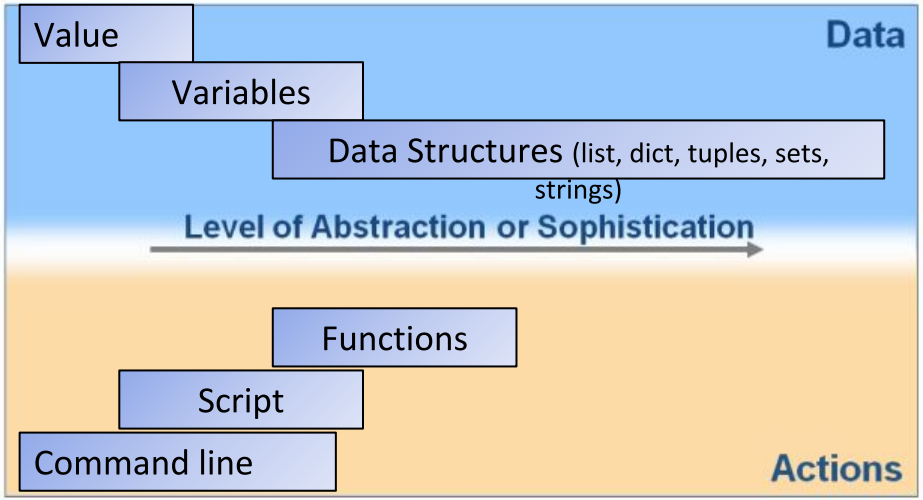
Based on

Progression of programming techniques



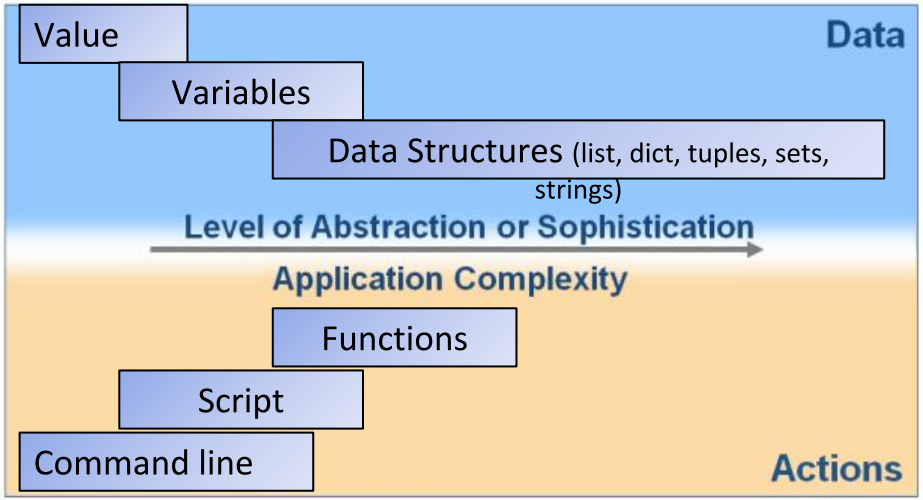
Based on

Progression of programming techniques



Based on

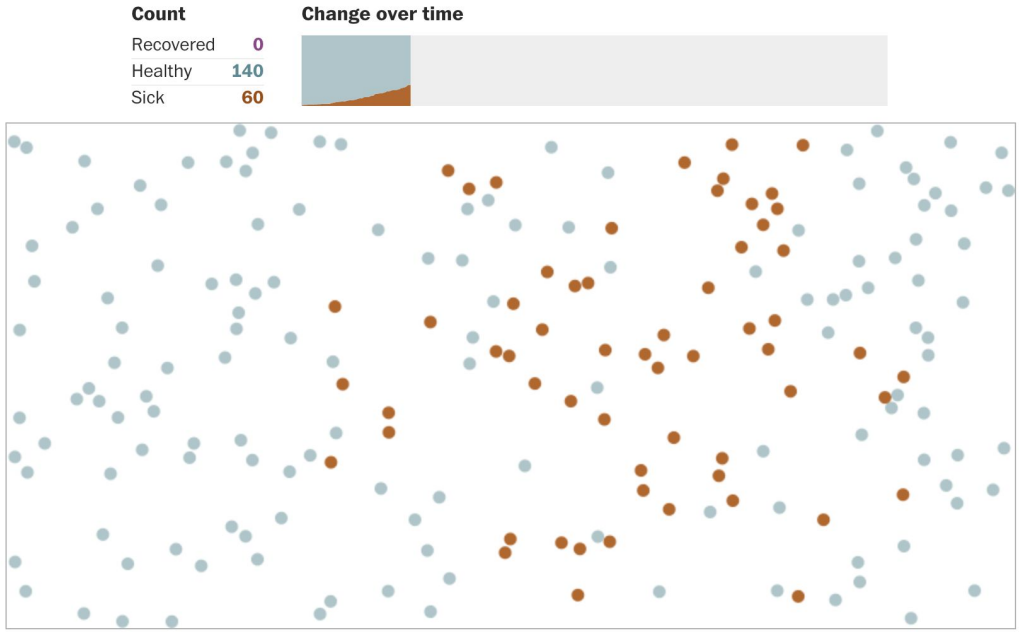
Progression of programming techniques



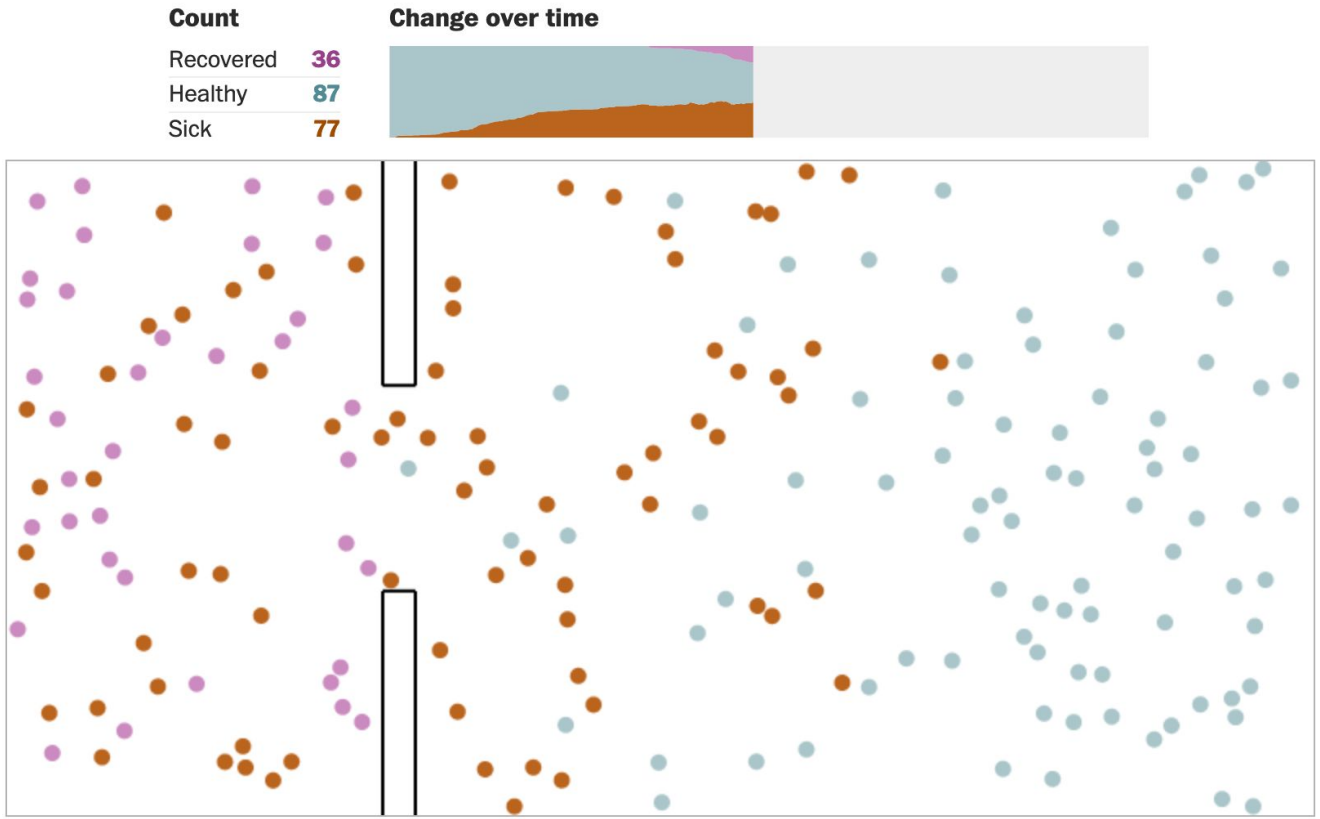
Based on

OO programming in Python: Example

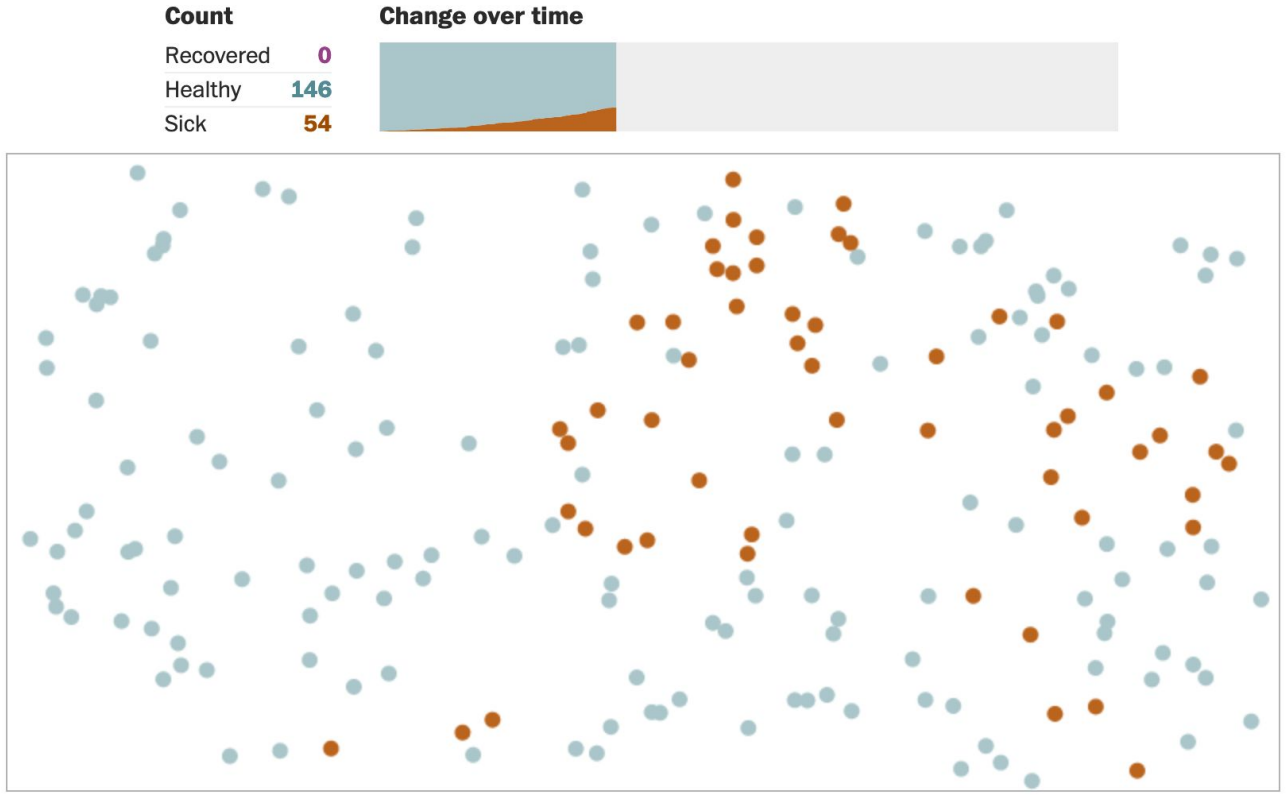
Why outbreaks like coronavirus spread exponentially, and how to “flatten the curve”



https://www.washingtonpost.com/graphics/2020/world/corona-simulator/



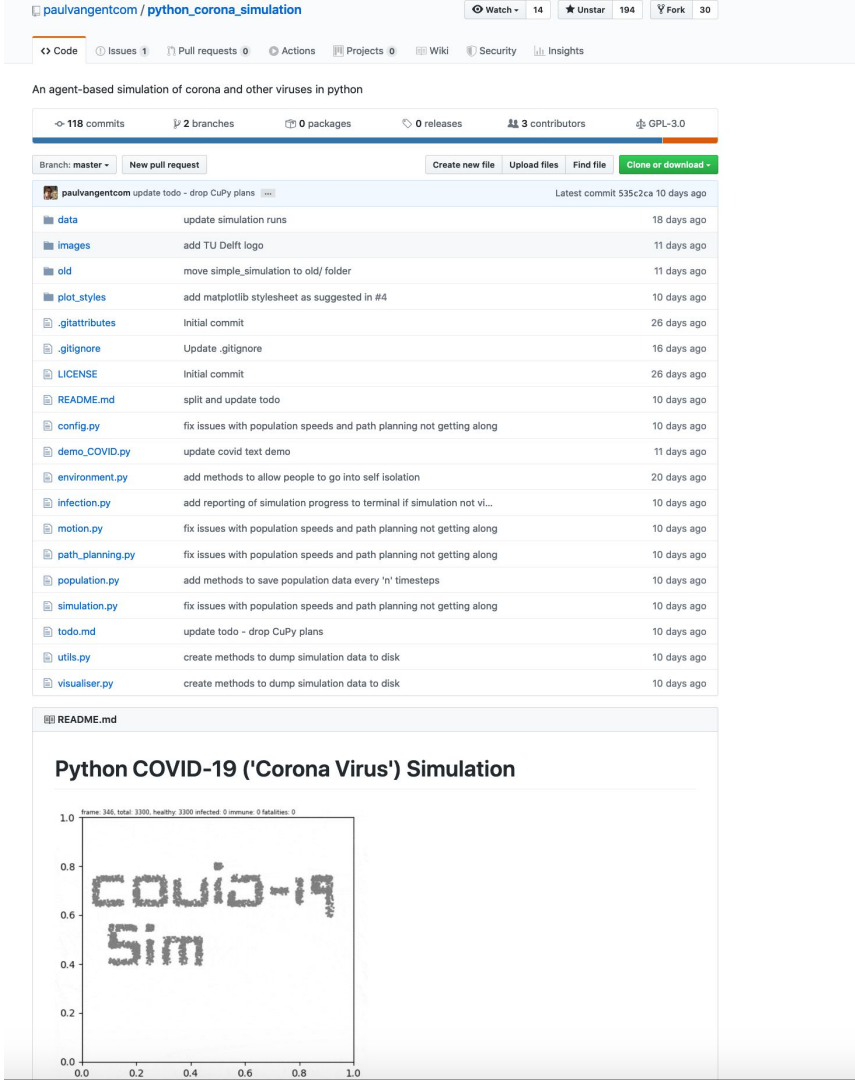
https://www.washingtonpost.com/graphics/2020/world/corona-simulator/

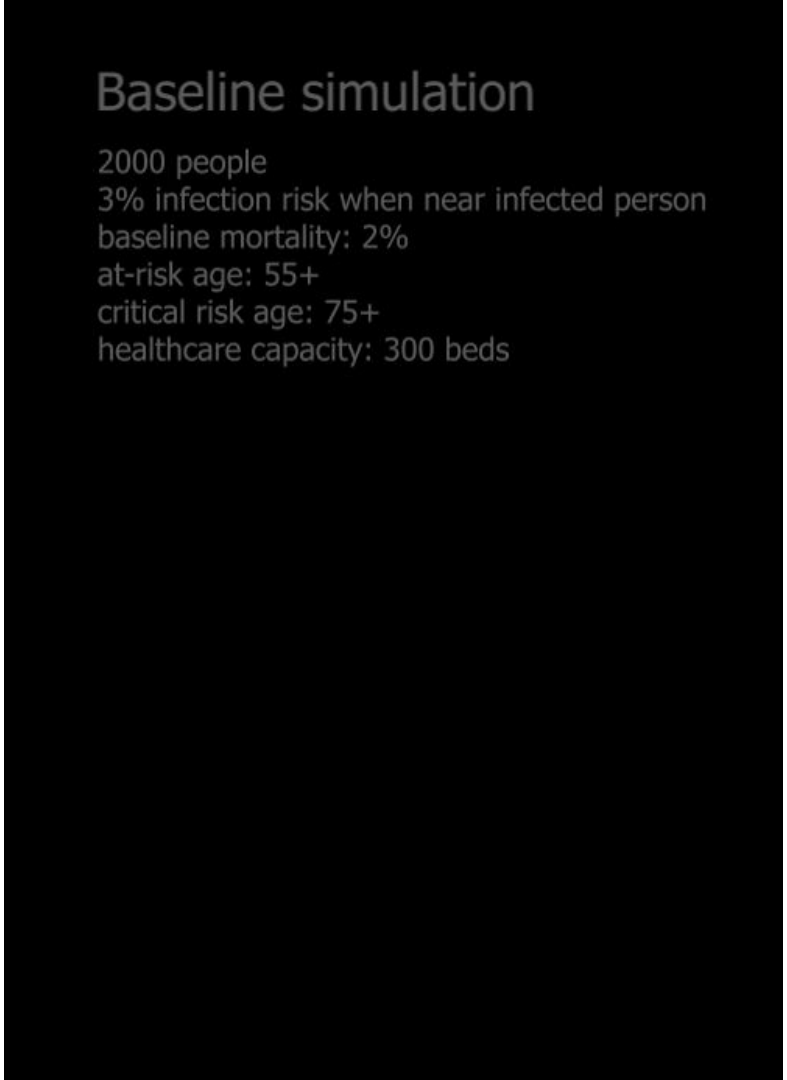


https://www.washingtonpost.com/graphics/2020/world/corona-simulator/

PhD Paul van Gent

- https://github.com/paulvangentcom/python_corona_simulation
- Implemented this idea in python
- Applied it to **Coronavirus (SARS CoV 2)** in a more realistic sense

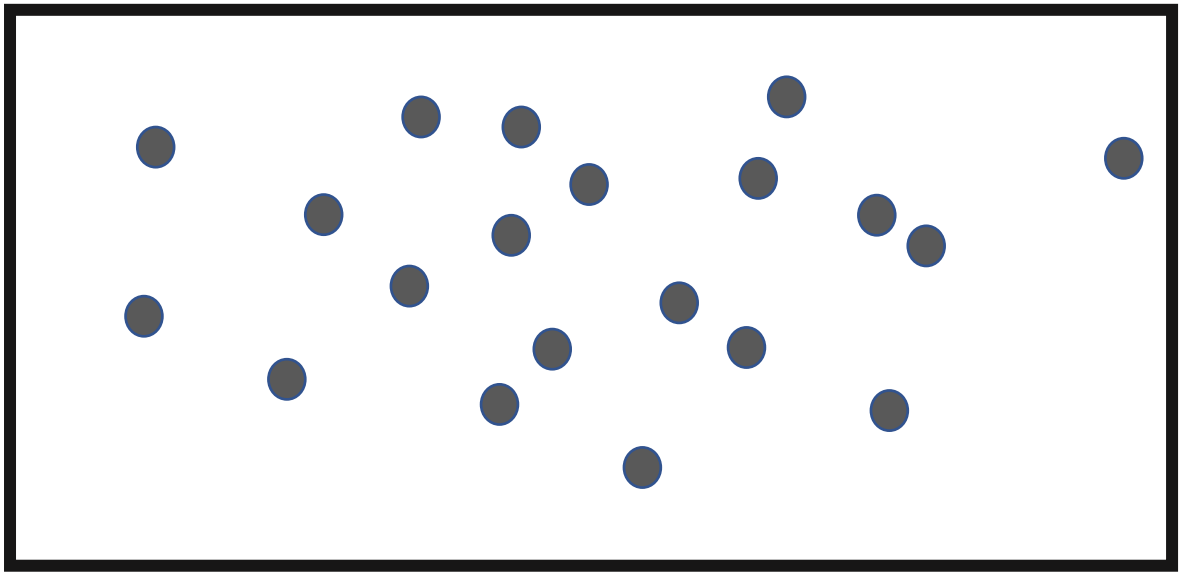




https://github.com/paulvangentcom/python_corona_simulation

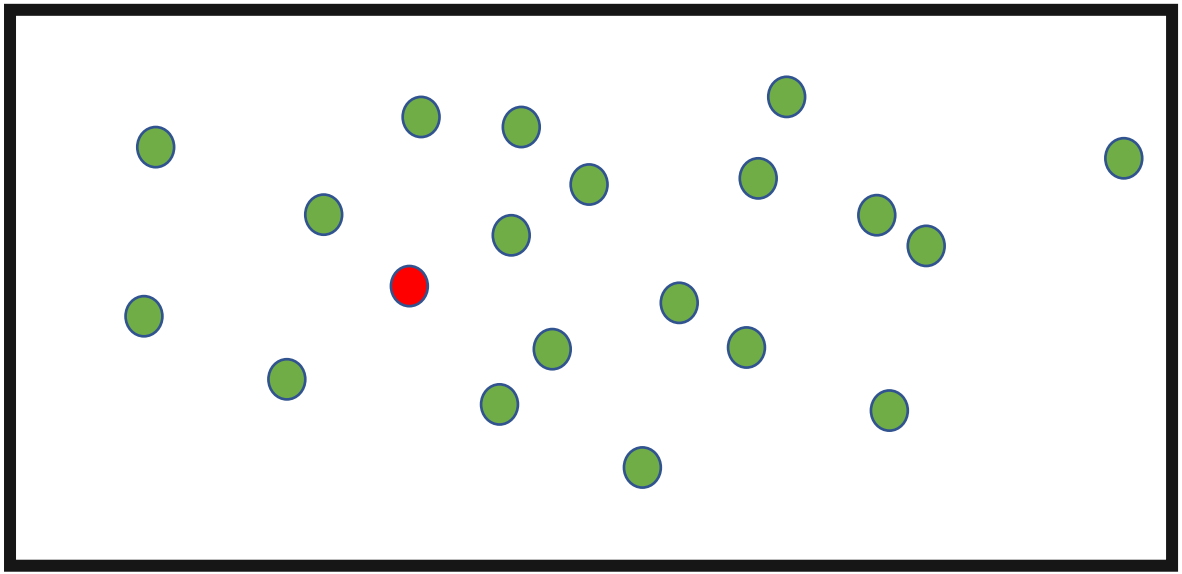
COVID19; the procedural approach

Which variables?



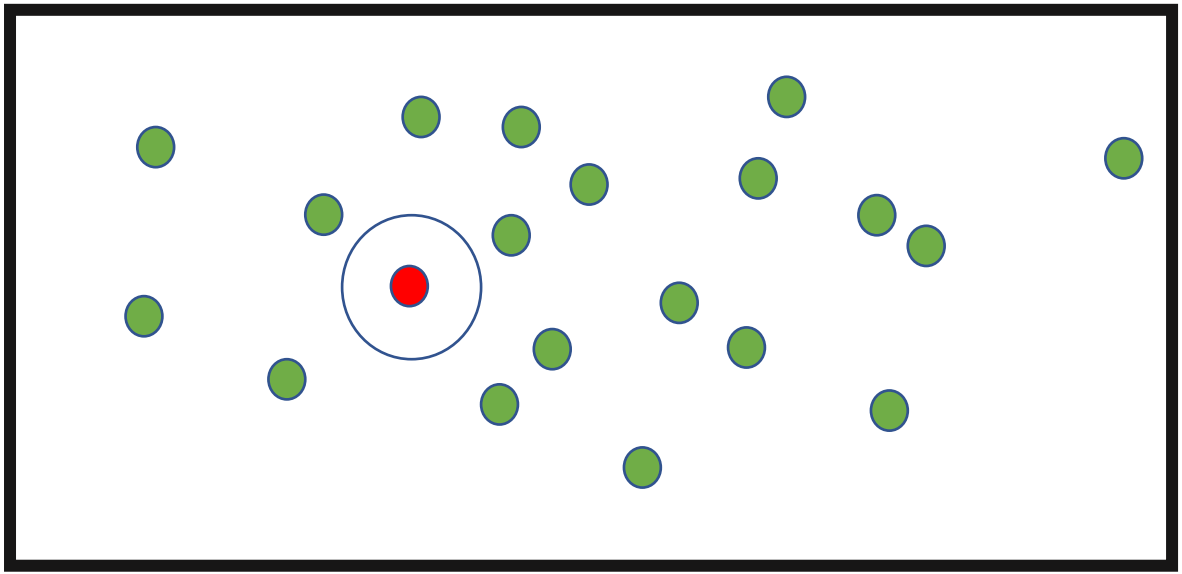
Which variables?

- pop_size



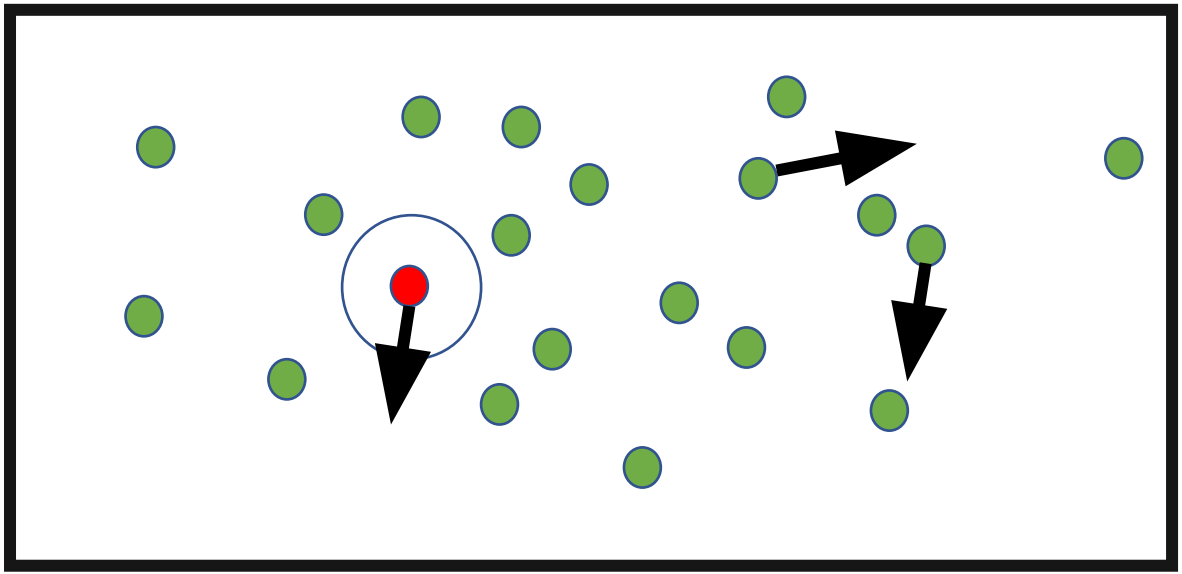
Which variables?

- pop_size
- health_status



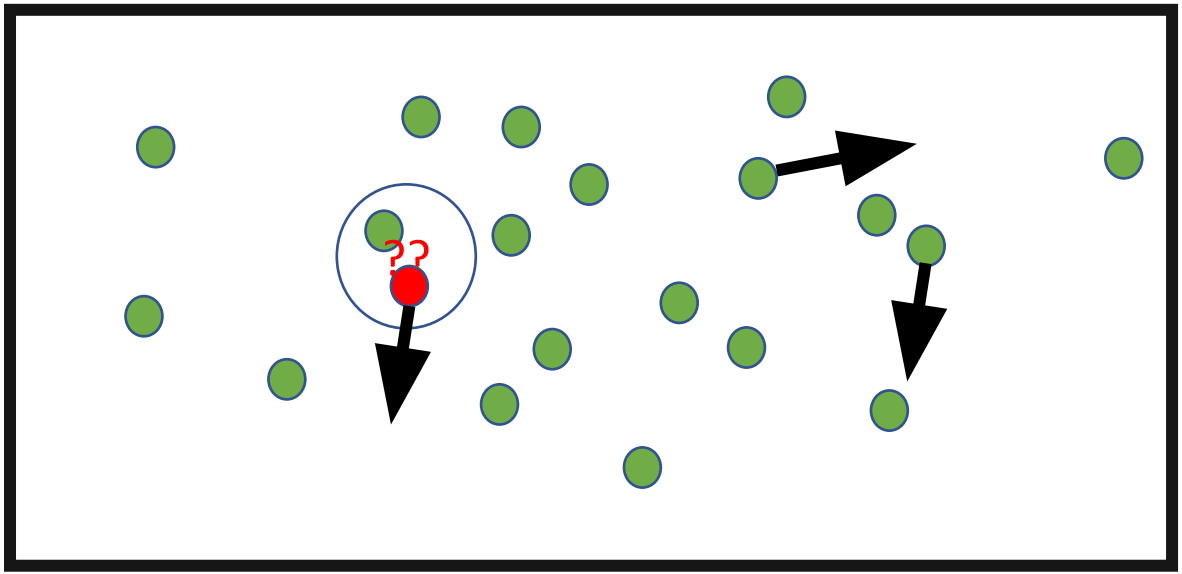
Which variables?

- pop_size
- health_status
- infection_range



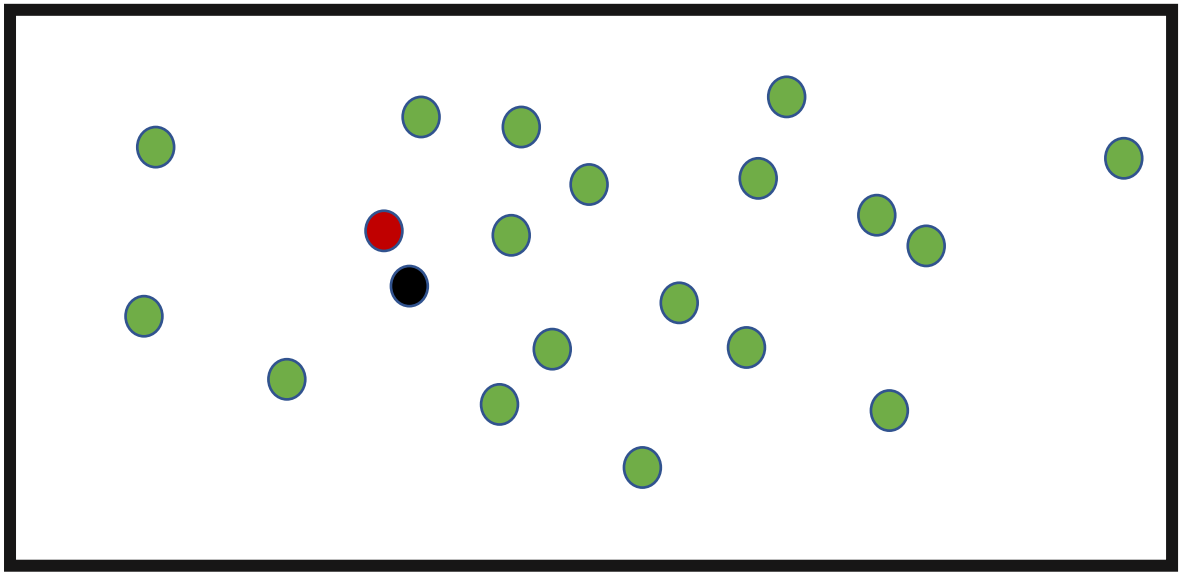
Which variables?

- pop_size
- health_status
- infection_range
- mean_speed



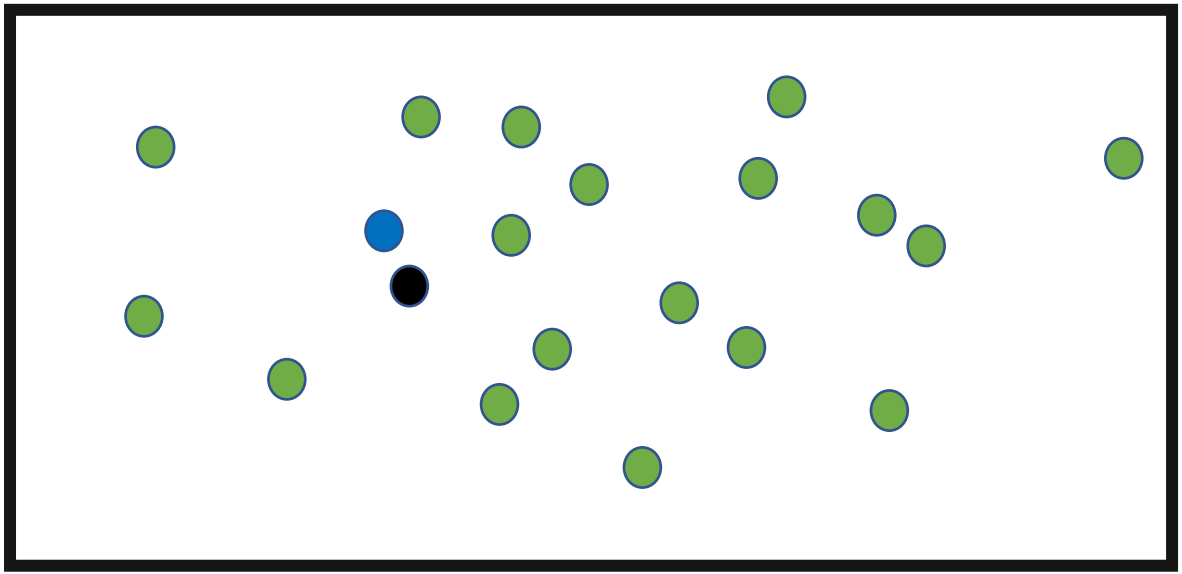
Which variables?

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance



Which variables?

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate



Which variables?

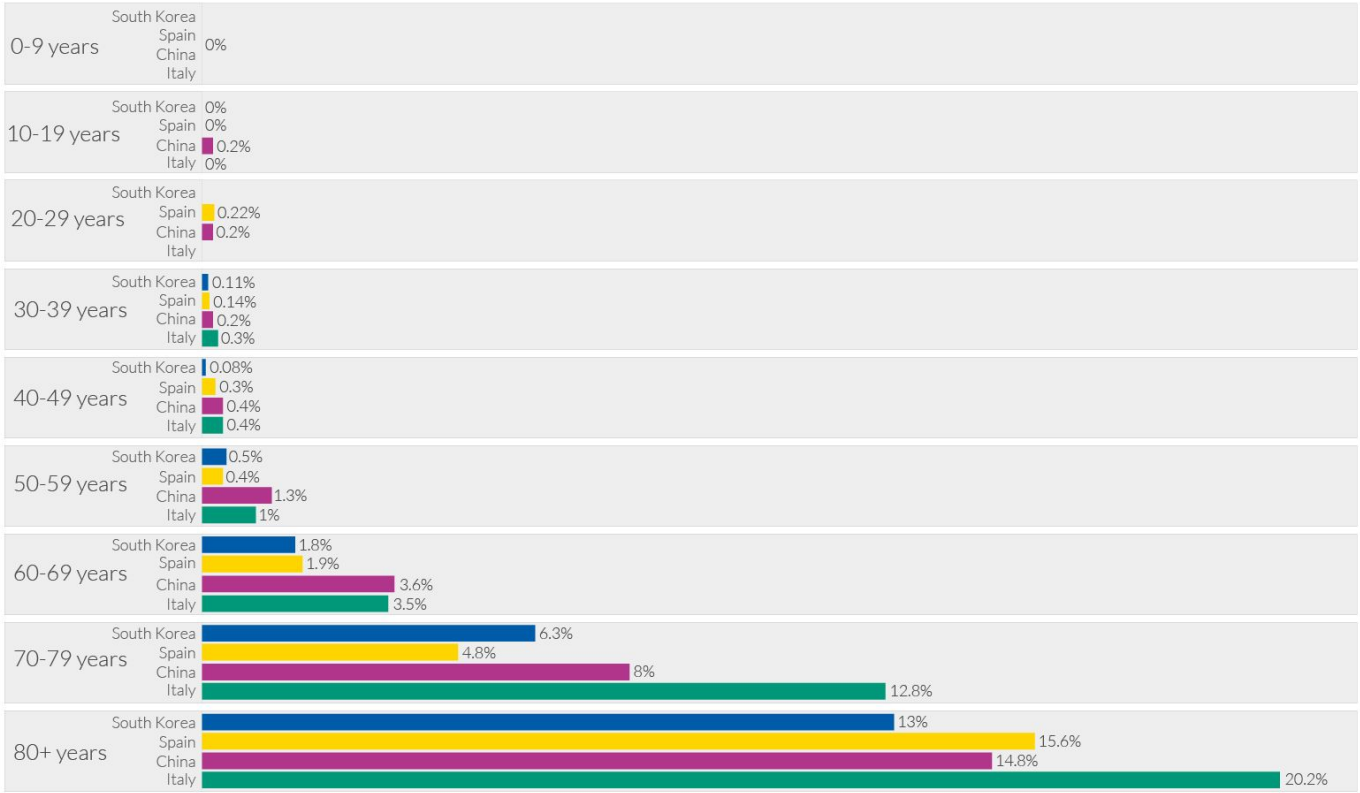
- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate
- recovery_duration

Coronavirus: case fatality rates by age

Our World in Data

Case fatality rate (CFR) is calculated by dividing the total number of confirmed deaths due to COVID-19 by the number of *confirmed* cases.

Two of the main limitations to keep in mind when interpreting the CFR:
(1) many cases within the population are unconfirmed due to a lack of testing.
(2) some individuals who are infected will eventually die from the disease, but are still alive at time of recording.



Note: Case fatality rates are based on confirmed cases and deaths from COVID-19 as of: 17th February (China); 24th March (Spain); 24th March (South Korea); 17th March (Italy).

Data sources: Chinese Center for Disease Control and Prevention (CDC); Spanish Ministry of Health; Korea Centers for Disease Control and Prevention (KCDC).
Onder G, Rezza G, Brusaferro S. Case-Fatality Rate and Characteristics of Patients Dying in Relation to COVID-19 in Italy. *JAMA*.

Which variables?

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate
- recovery_duration
- mean_age

Lockdown

Which variables?

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate
- recovery_duration
- mean_age
- lockdown

Which variables?

- traveling_infects = False
- self_isolate = False
- lockdown = False
- lockdown_percentage = 0.1 #after this proportion is infected, lock-down begins
- lockdown_compliance = 0.95 #fraction of the population that will obey the lockdown
- xbounds = [0.02, 0.98]
- ybounds = [0.02, 0.98]
- visualise = True #whether to visualise the simulation
- plot_mode = 'sir' #default or sir
- x_plot = [0, 1]
- y_plot = [0, 1]
- save_plot = False
- plot_path = 'render/' #folder where plots are saved to
- plot_style = 'default' #can be default, dark, ...
- colorblind_mode = False
- colorblind_type = 'deuteranopia'
- pop_size = 2000
- mean_age = 45
- max_age = 105
- age_dependent_risk = True #whether risk increases with age
- critical_age = 75 #age at and beyond which mortality risk reaches maximum
- critical_mortality_chance = 0.1 #maximum mortality risk for older age
- risk_increase = 'quadratic' #whether risk between risk and critical age increases 'linear' or 'quadratic'
- proportion_distancing = 0
- speed = 0.01 #average speed of population
- wander_range = 0.05
- wander_factor = 1
- wander_factor_dest = 1.5 #area around destination
- infection_range=0.01 #range surrounding sick patient that infections can take place
- infection_chance=0.03 #chance that an infection spreads to nearby healthy people each tick
- recovery_duration=(200, 500) #how many ticks it may take to recover from the illness
- mortality_chance=0.02 #global baseline chance of dying from the disease
- self.healthcare_capacity = 300 #capacity of the healthcare system
- self.treatment_factor = 0.5 #when in treatment, affect risk by this factor
- self.no_treatment_factor = 3 #risk increase factor to use if healthcare system is full
- #risk parameters
- self.treatment_dependent_risk = True #whether risk is affected by treatment
- self.self_isolate_proportion = 0.6
- self.isolation_bounds = [0.02, 0.02, 0.1, 0.98]
- self.lockdown_percentage = 0.1

Which functions?

Data in Variables

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate
- recovery_duration
- mean_age
- lockdown

Actions in Functions

- Create_population(population,…)
- Initialize_status(population,…) OR recover_or_die (population,…)
- Infect(Properties_Virus, population)
- Update_populationLocation(population, speed,…)
- Infect(population, properties,…)
- recover_or_die(population,virus)
- recover_or_die(population, virus,…)
- create_stratified_population(population,…)
- Update_populationLocation(population, speed,…)

Lot’s data and actions to implement

Data and actions to implement

Data in Variables	Actions in Functions
• pop_size	• Create_population(population,…)
• health_status	• Initialize_status(population,…) OR recover_or_die (population,…)
• infection_range	• Infect(Properties_Virus, population)
• mean_speed	• Update_populationLocation(population, speed,…)
• infection_chance	• Infect(population, properties,…)
• mortality_rate	• recover_or_die(population,virus)
• recovery_duration	• recover_or_die(population, virus,…)
• critical_age	• create_stratified_population(population,…)
• lockdown	• activate_lockdown(population, society,…)
• …	• …

Related data

Data in Variables

- pop_size
- health_status
- infection_range
- mean_speed
- infection_chance
- mortality_rate
- recovery_duration
- critical_age
- lockdown
- …

Virus

- infection_range
- infection_chance
- recovery_duration
- *mortality_rate*
- …

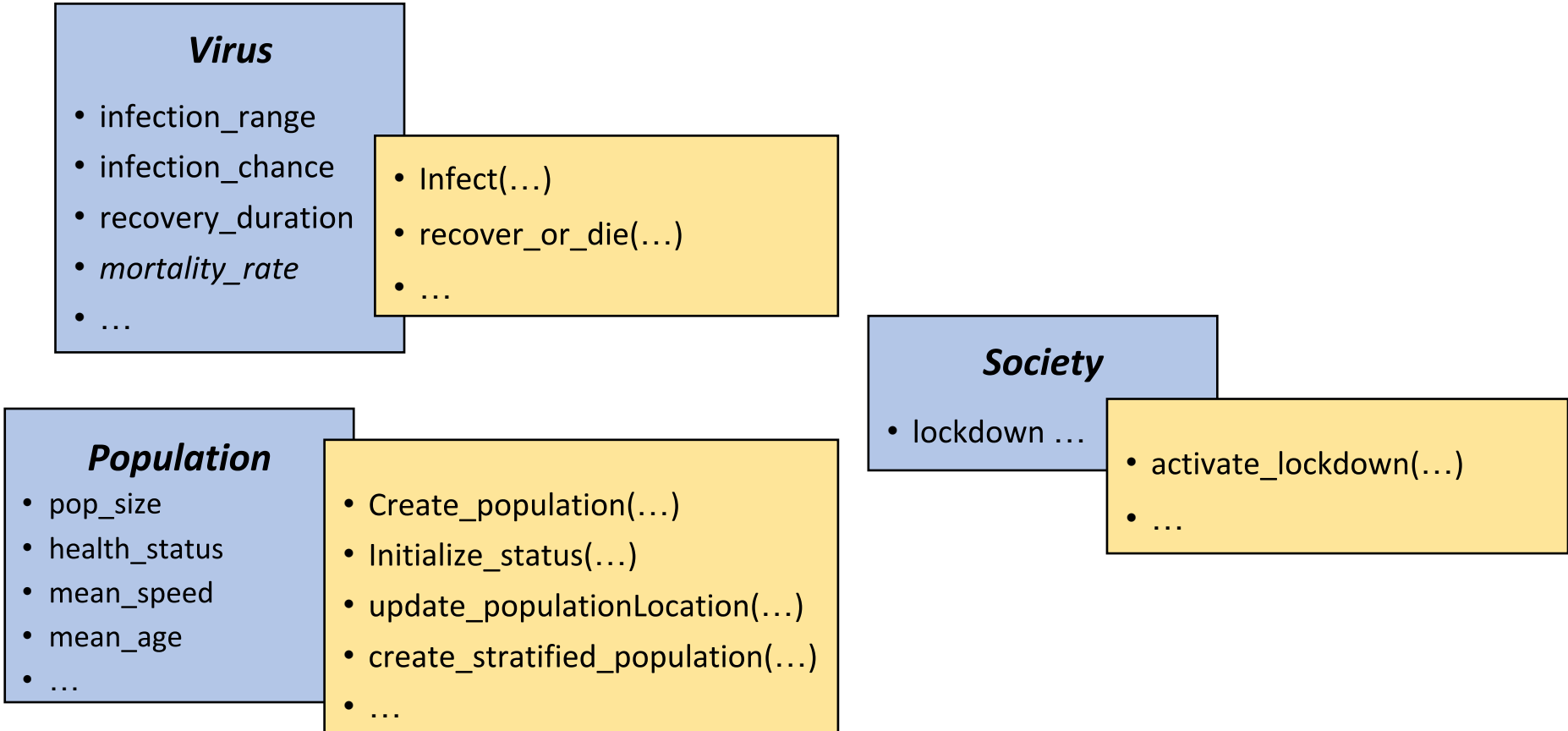
Population

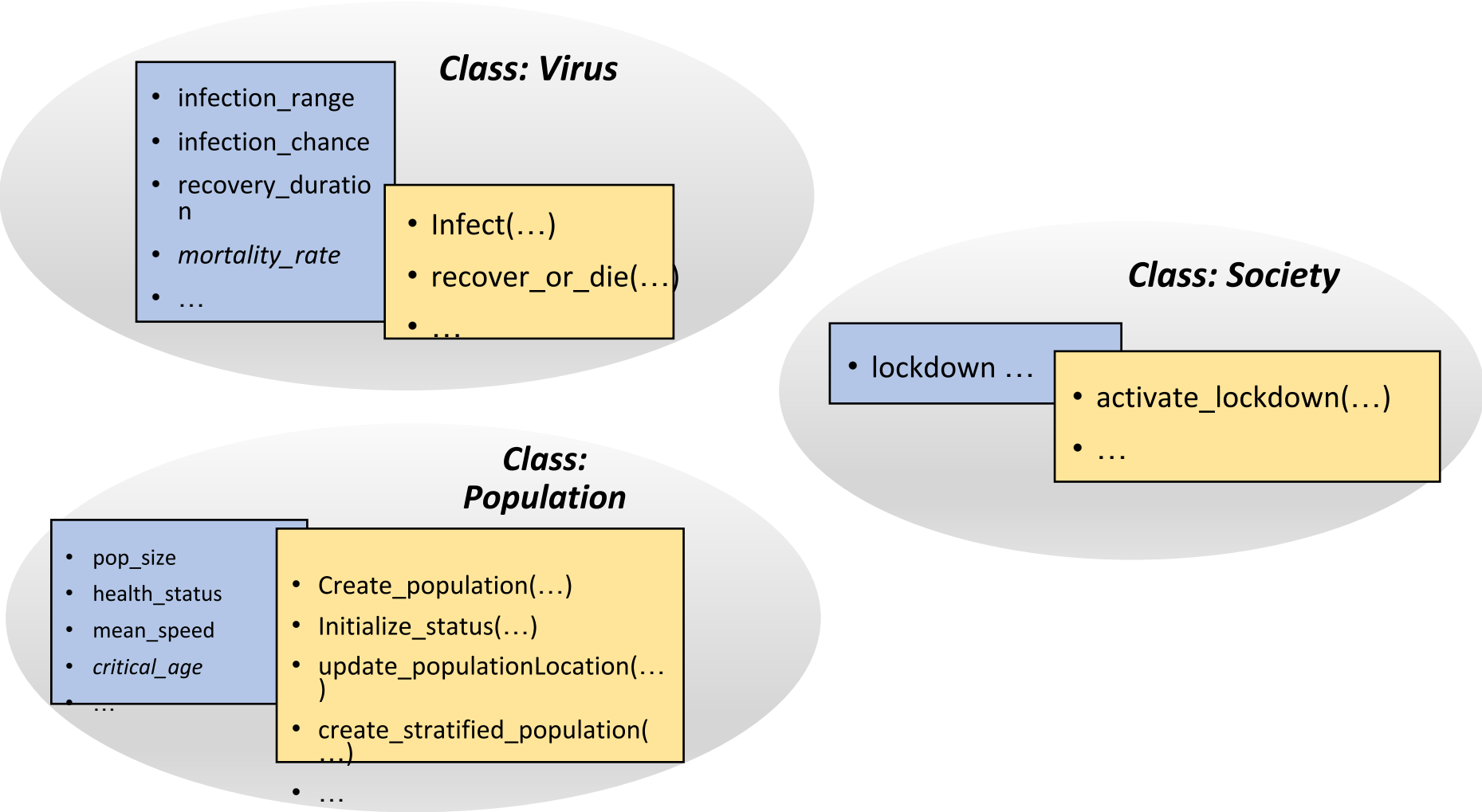
- pop_size
- health_status
- mean_speed
- *critical_age*
- …

Society

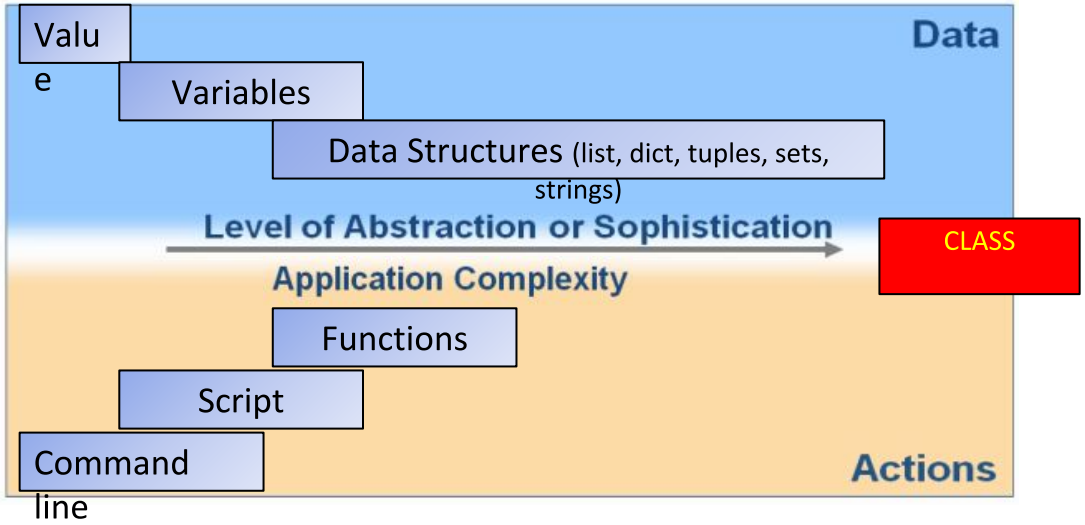
- lockdown …

Related data and actions

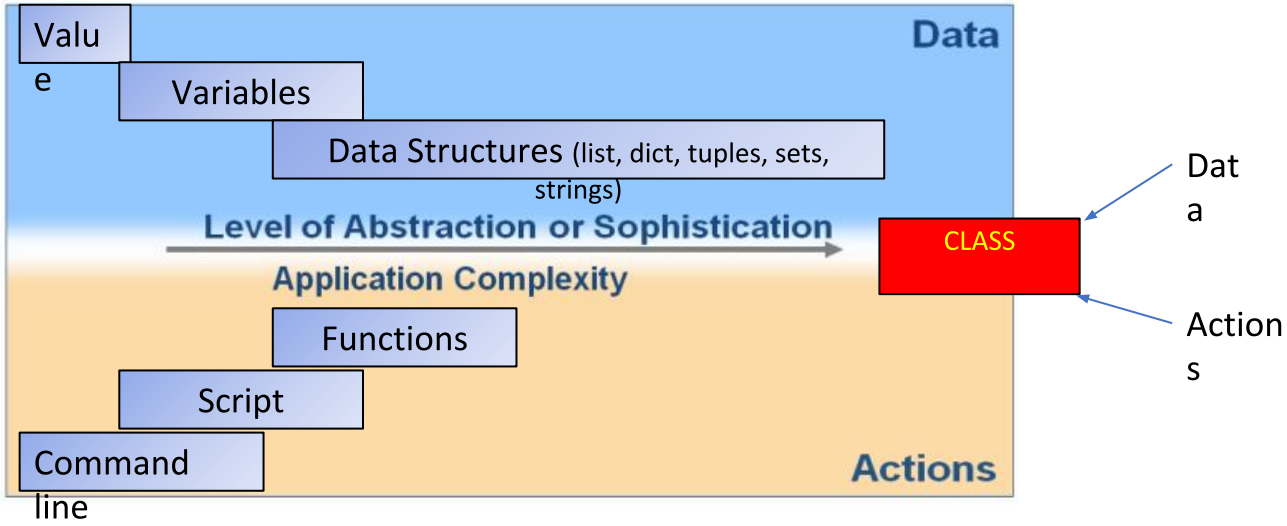



Class: Virus
infection_range
infection_chance
recovery_duration
mortality_rate
…
Infect(…)
recover_or_die(…)
…
Class: Society
lockdown …
activate_lockdown(…)
…
Class: Population
pop_size
health_status
mean_speed
critical_age
…
Create_population(…)
Initialize_status(…)
update_populationLocation(…)
create_stratified_population(…)
…

Progression of programming techniques



Progression of programming techniques



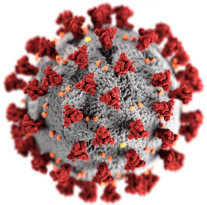
Based on

Object-Oriented Terminology

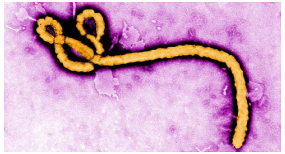
- Class
  - Outline of data
  - *Properties* (data)
  - *Methods* (actions)
- Object
  - Specific example of a class
  - *Instance*

Class: Virus

Properties (data)
Methods (actions)



object SARS-CoV-2
is instance or of class Virus



object ebola
is instance or of class Virus

Demo: Building a Simple Class: Virus

- Define a virus *class*

Properties

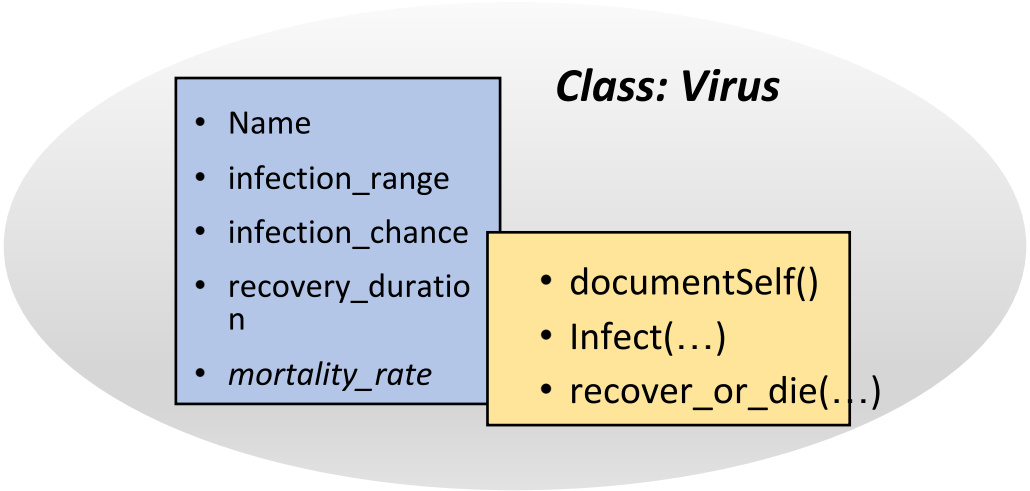
1. name
2. infection_range
3. infection_chance
4. recovery_duration

Methods

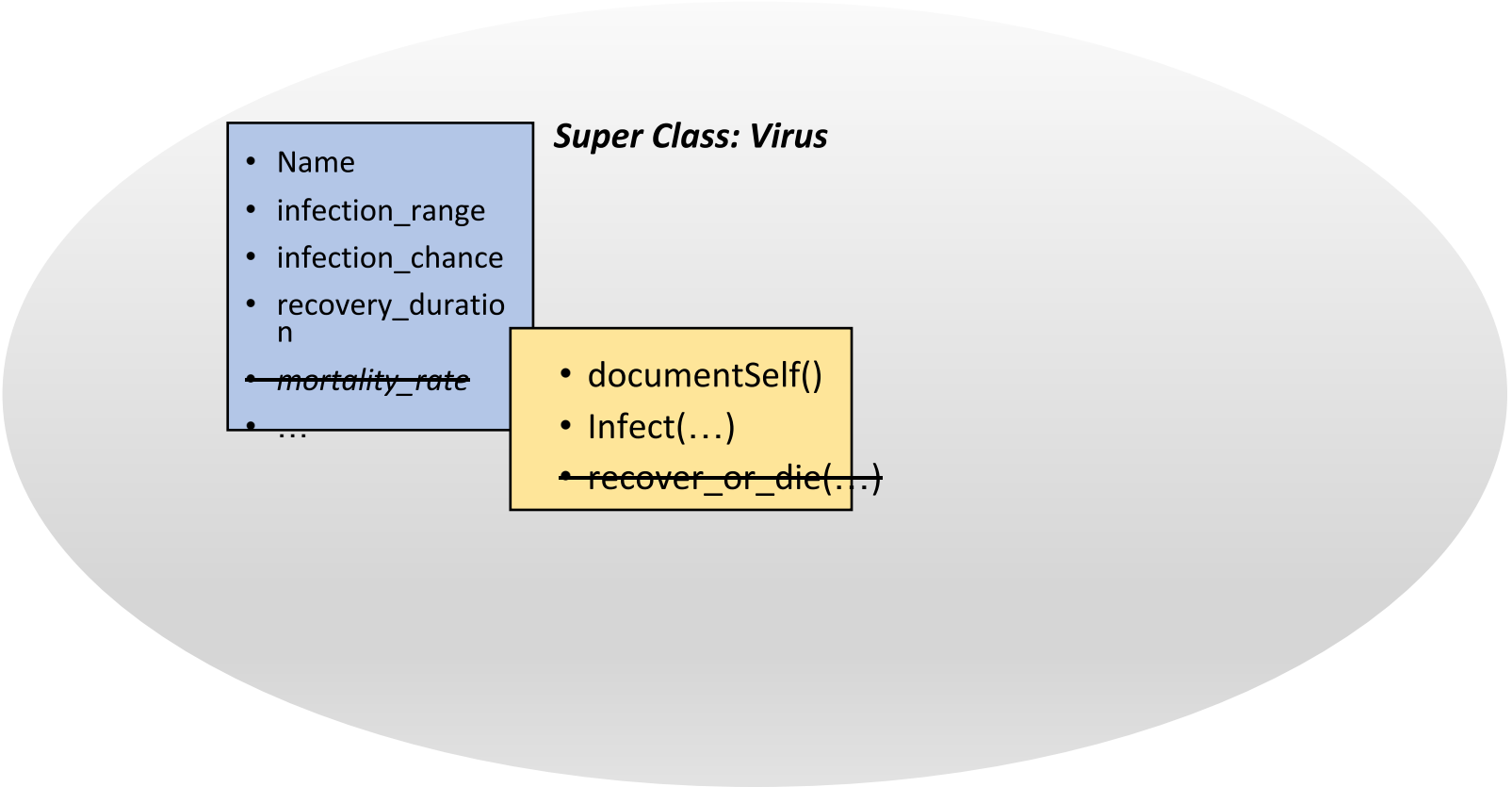
5. documentSelf()
6. infect()

- Create two virus objects (instances of the class Virus) named
  - *virus1*
  - *SARS*

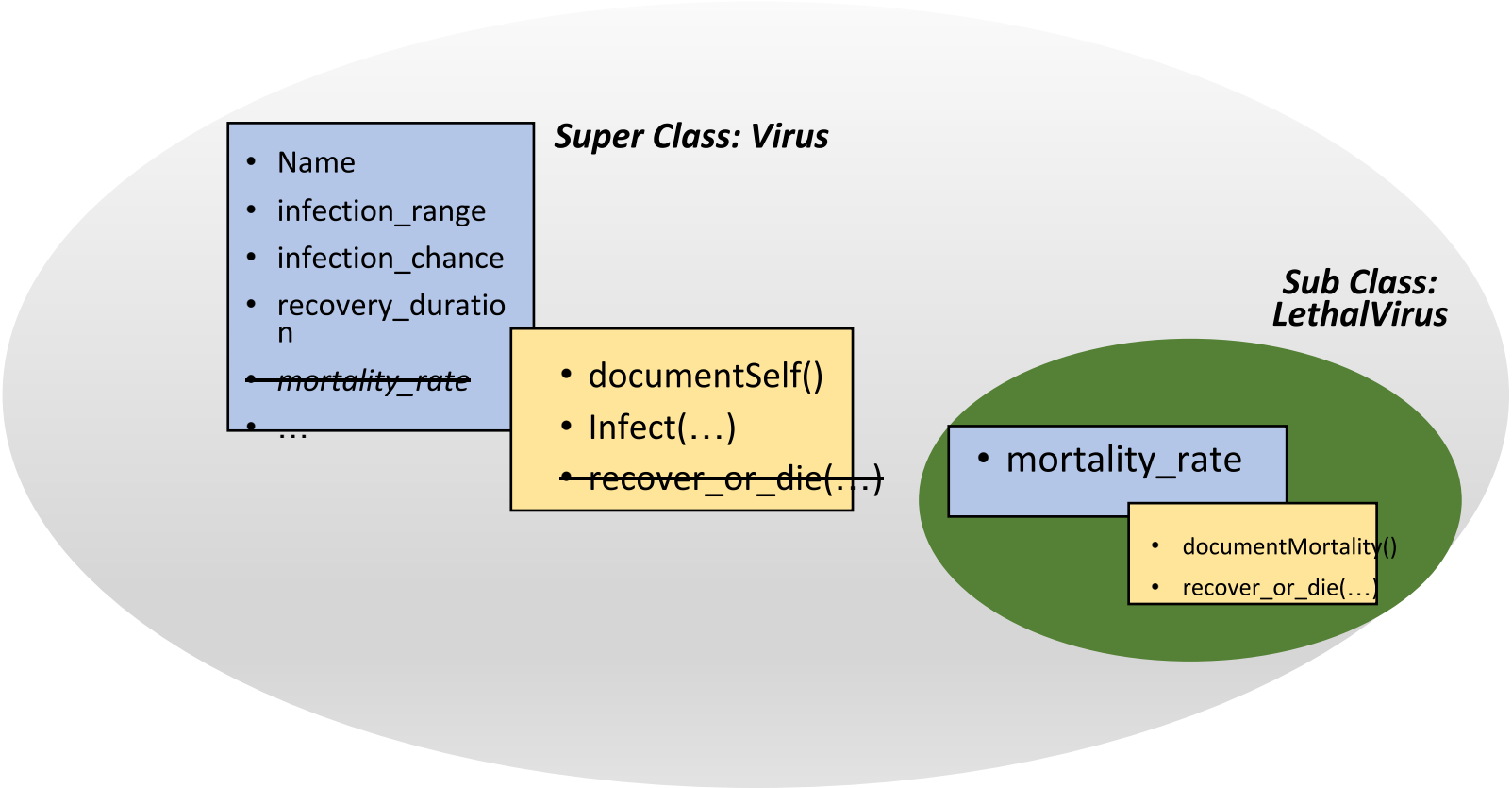
The class Virus



What we actually implemented



What we actually implemented



NextLevel: Building a Simple Class: Virus

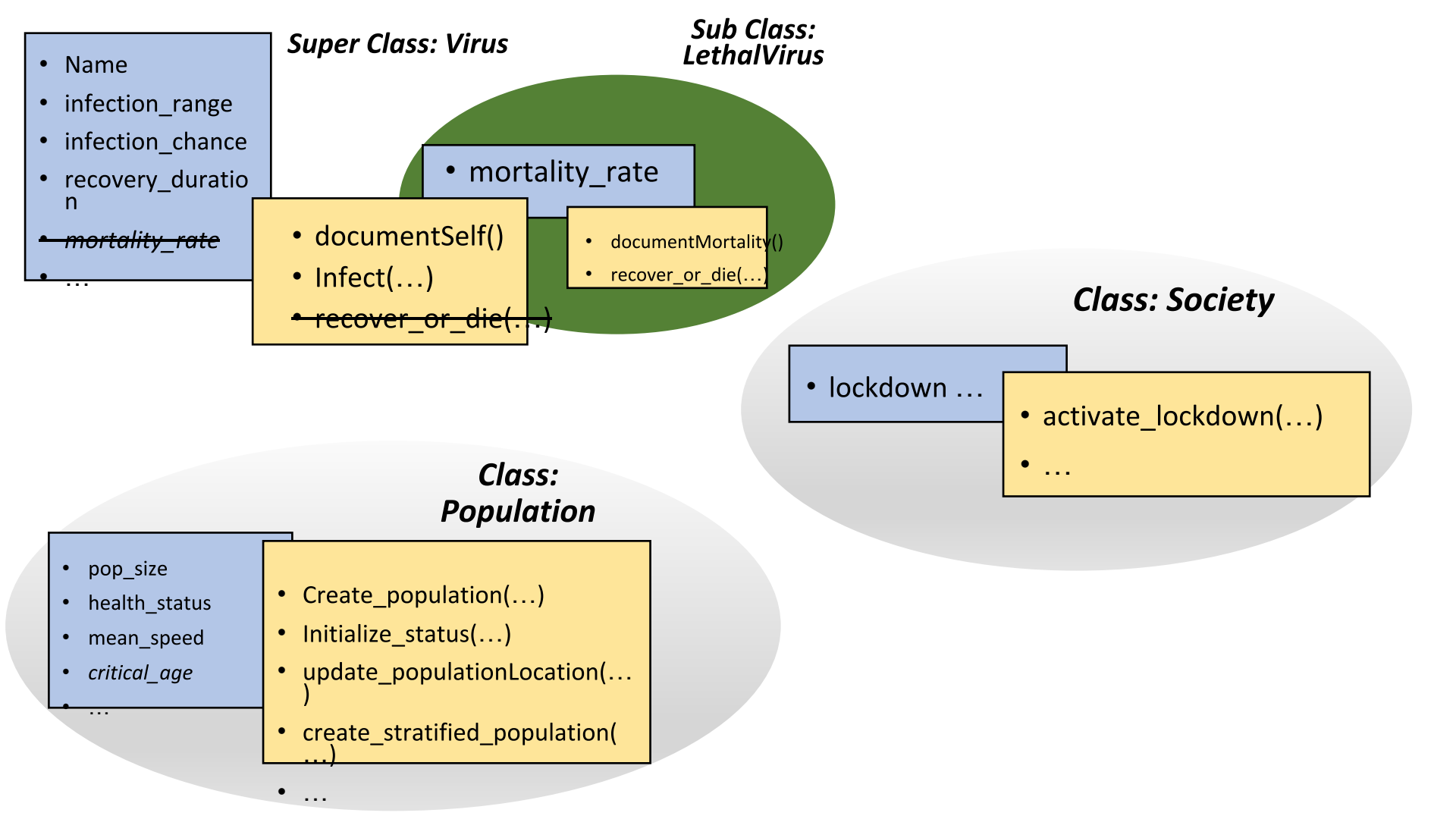
- *Create a subclass LethalVirus*

Properties

1. mortality_rate

Methods

2. documentMortality()
3. recover_or_die(…)


Super Class: Virus
Name
infection_range
infection_chance
recovery_duration
mortality_rate
…
documentSelf()
Infect(…)
recover_or_die(…)
Sub Class: LethalVirus
mortality_rate
documentMortality()
recover_or_die(…)
Class: Society
lockdown …
activate_lockdown(…)
…
Class: Population
pop_size
health_status
mean_speed
critical_age
…
Create_population(…)
Initialize_status(…)
update_populationLocation(…)
create_stratified_population(…)
…

Thank you!

Thanks for watching this webinar. We hope you enjoyed it!

- You'll get a reminder when webinar recording is posted
- Example repository
  github.com/csdms/level-up
- Let us know what you think about these webinars through the CSDMS Help Desk:
  github.com/csdms/help-desk