voyage

Python

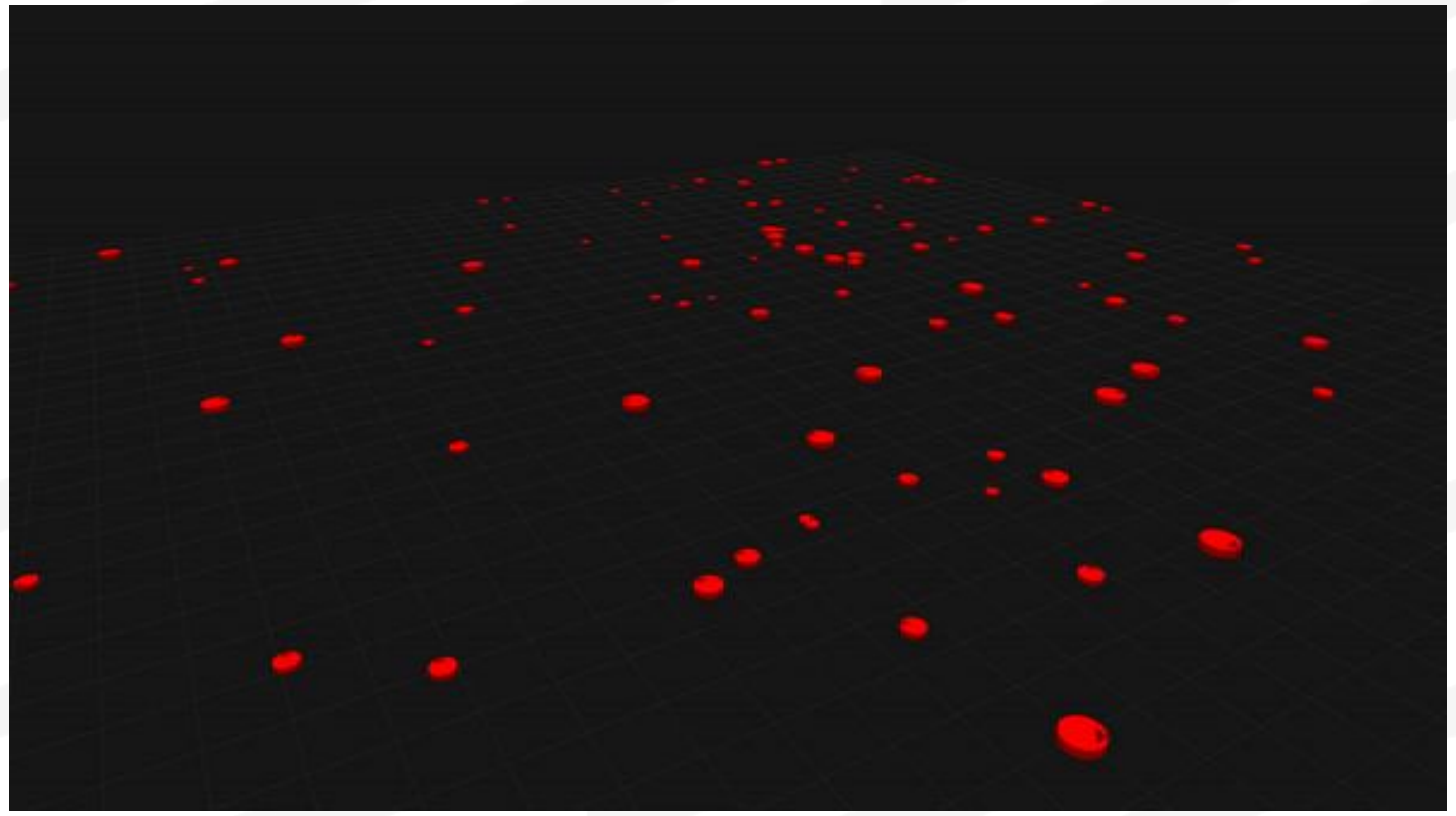
Hands-on Introductory Workshop

voyage

Agenda

- Part 0: Basics
  - python as a scripting/programming language
  - docs and links
  - basic constructs
- Part 1: Create a simple 2D Vector object
  - object-oriented programming (OOP)
  - test driven development (TDD)
- *Small Break*
- Part 2: Create a simple physics system
  - collections, iterators
  - packages
- Part 3: Performance (if we have time)
  - profiling
  - spatial acceleration structure

we will build this:



Basics

- interpreted vs compiled languages
- static vs dynamic typing
- brief history of the language
- tools and IDEs
- the python documentation
- basic constructs:
  - variables
  - expressions
  - functions
  - conditions
  - loops

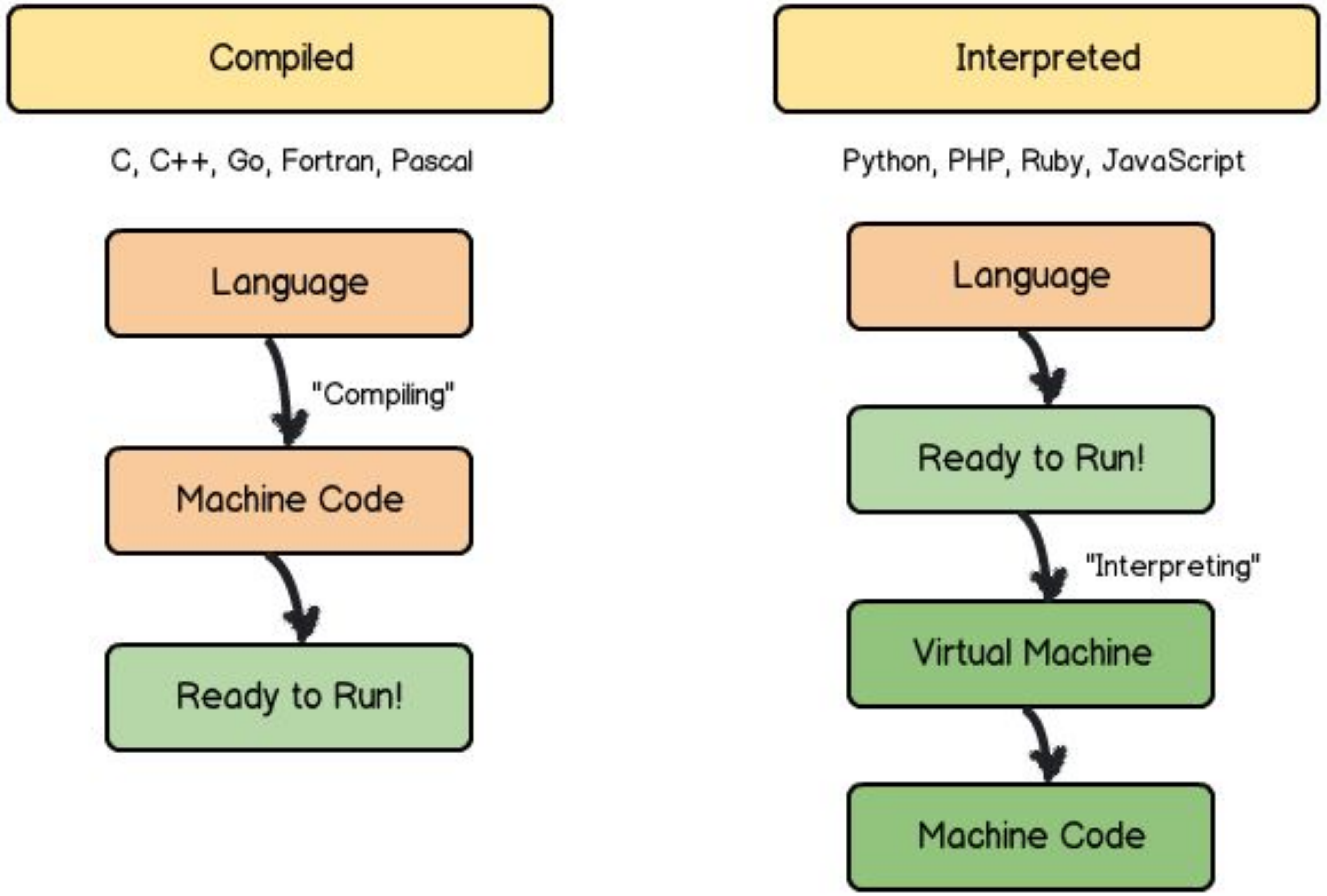
interpreted and compiled languages

Compiled Language

the source code is processed by a compiler, that builds an independent executable program

Interpreted Language

the source code is read by an interpreter, that executes the instructions in the script



voyage

static and dynamic typing

“value”: an object of a specific type

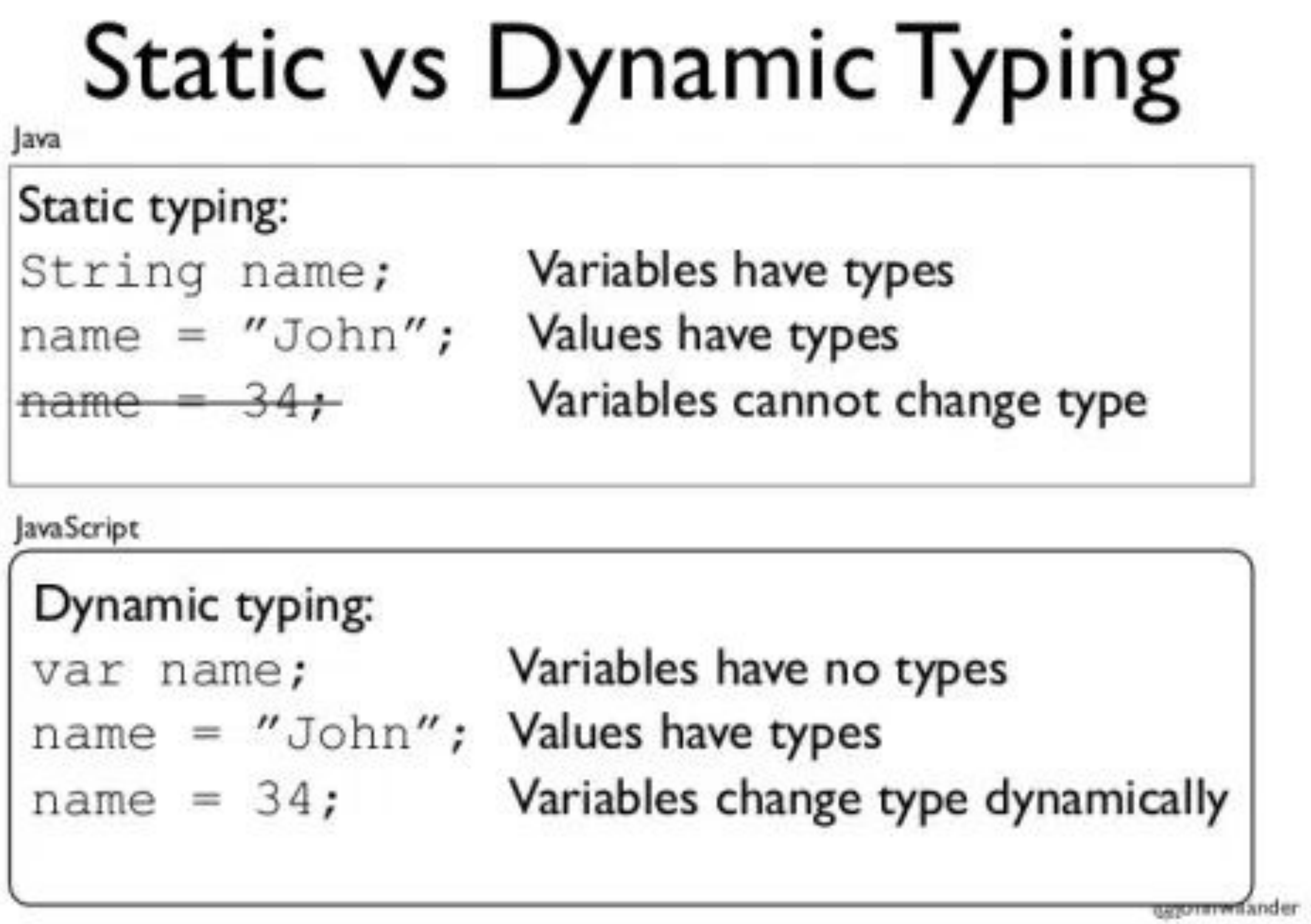
“variable”: a label used to refer to a specific value

Static Typing

“variables” and “values” are forced to have the same type for the entire lifetime

Dynamic Typing

“variables” can refer to “values” of different types during their lifetime



Static vs Dynamic Typing

Java

Static typing:

`String name;`	Variables have types
`name = ”John”;`	Values have types
~~`name = 34;`~~	Variables cannot change type

JavaScript

Dynamic typing:

`var name;`	Variables have no types
`name = ”John”;`	Values have types
`name = 34;`	Variables change type dynamically

@johnwilander

History

- created in the late 1980s by Guido Van Rossum
- Python v2 was released in 2000
- Python v3 was released in 2008, and was *not* backward compatible
- Python v2 has been discontinued on Jan 1st, 2020

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

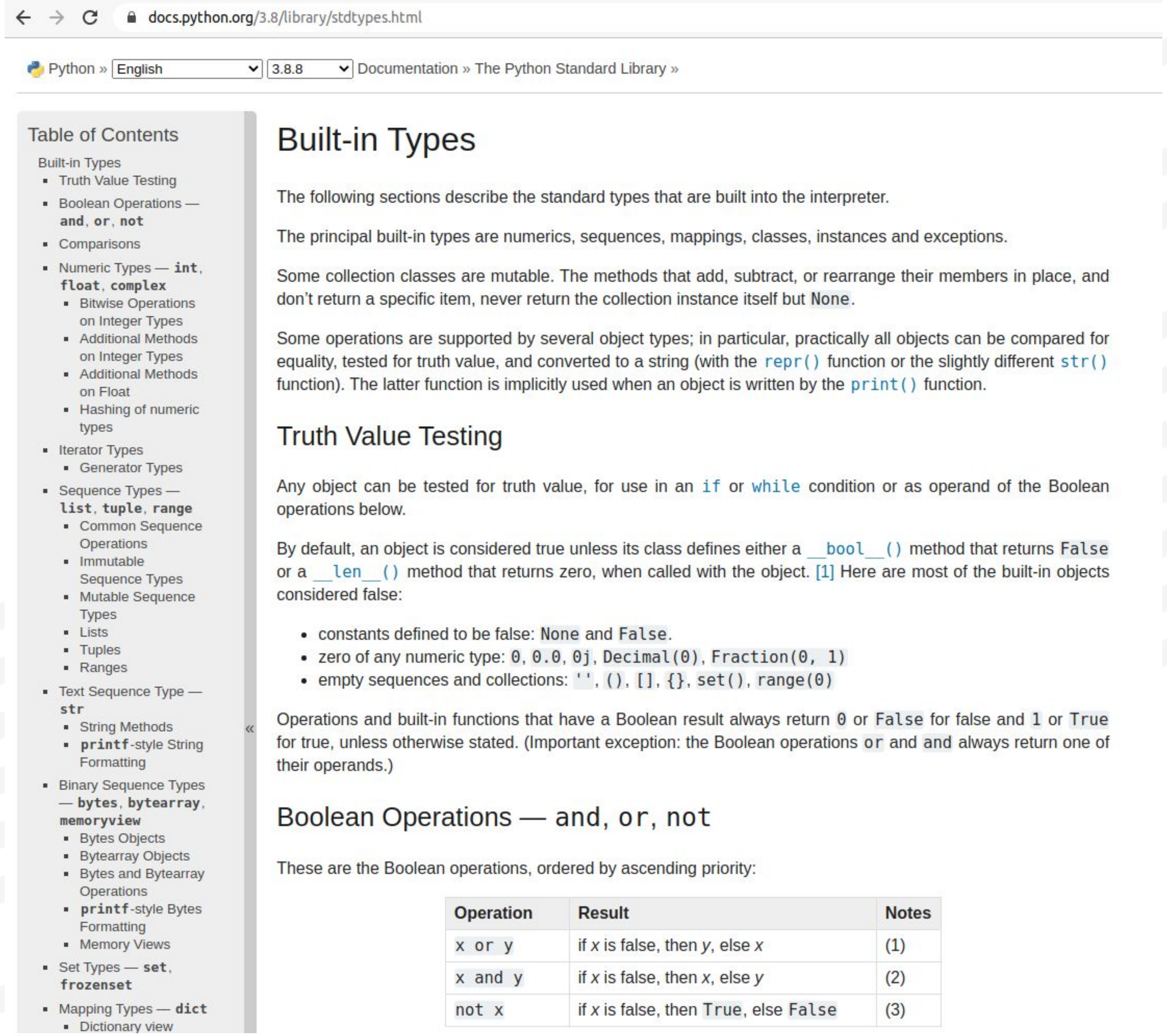
Some of the strengths of the language are its extensive standard library and the huge ecosystem of tools that are available

Documentation

Python comes with an extensive amount of documentation:

docs.python.org/3

Go there before blindly googling for solutions! :)



docs.python.org/3.8/library/stdtypes.html
Python » English 3.8.8 Documentation » The Python Standard Library »

Table of Contents

Built-in Types

The following sections describe the standard types that are built into the interpreter.

The principal built-in types are numerics, sequences, mappings, classes, instances and exceptions.

Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but `None`.

Some operations are supported by several object types; in particular, practically all objects can be compared for equality, tested for truth value, and converted to a string (with the `repr()` function or the slightly different `str()` function). The latter function is implicitly used when an object is written by the `print()` function.

Truth Value Testing

Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations below.

By default, an object is considered true unless its class defines either a `__bool__()` method that returns `False` or a `__len__()` method that returns zero, when called with the object. [1] Here are most of the built-in objects considered false:

- constants defined to be false: `None` and `False`.
- zero of any numeric type: `0`, `0.0`, `0j`, `Decimal(0)`, `Fraction(0, 1)`
- empty sequences and collections: `''`, `()`, `[]`, `{}`, `set()`, `range(0)`

Operations and built-in functions that have a Boolean result always return `0` or `False` for false and `1` or `True` for true, unless otherwise stated. (Important exception: the Boolean operations `or` and `and` always return one of their operands.)

Boolean Operations — `and`, `or`, `not`

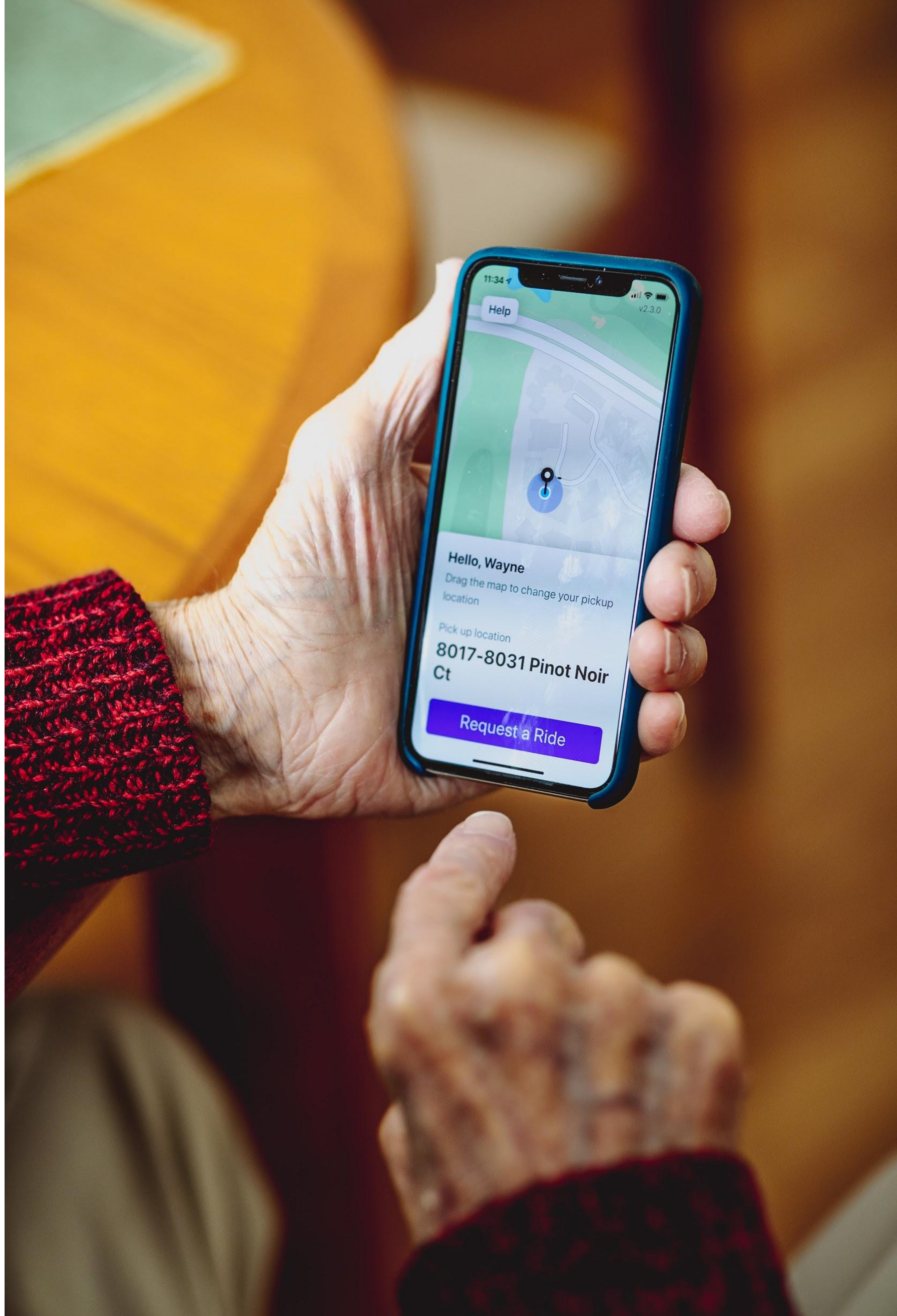
These are the Boolean operations, ordered by ascending priority:

Operation	Result	Notes
`x or y`	if *x* is false, then *y*, else *x*	(1)
`x and y`	if *x* is false, then *x*, else *y*	(2)
`not x`	if *x* is false, then `True`, else `False`	(3)

Hands-on

basic concepts:

- running a script
- variables
- expressions
- functions
- conditions
- loops



voyage

2D Vector class

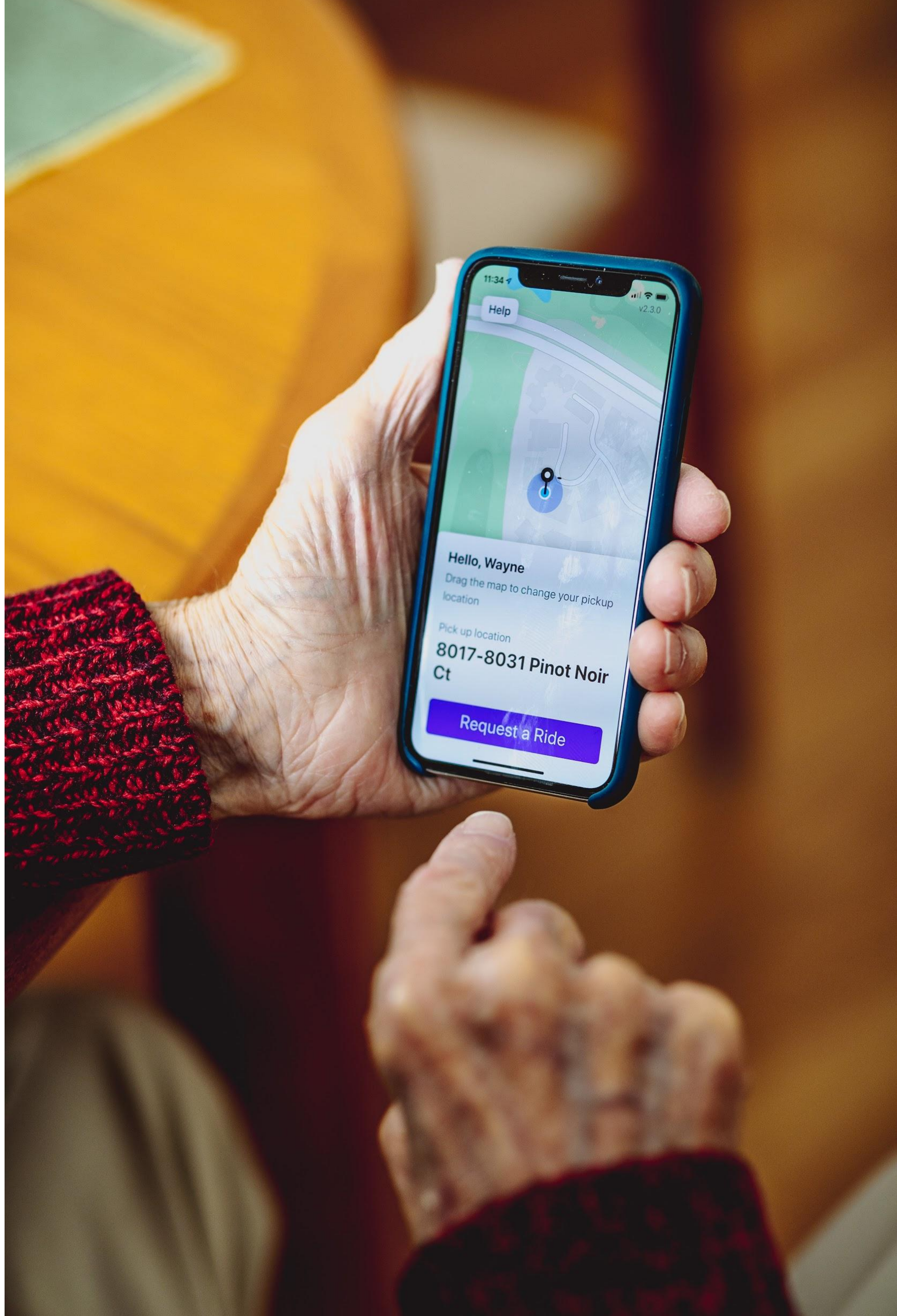
- what’s a “test”
- unit testing
- python test package
- what’s an “object”
- OOP in Python

voyage

Hands-on

Let’s create a 2D Vector using Test-Driven-Development:

- unit tests in python
- objects
- operators



voyage

Physics System

- create a “physics” object
- iterate over a collection of objects
- calculate/update dynamics
- process collisions

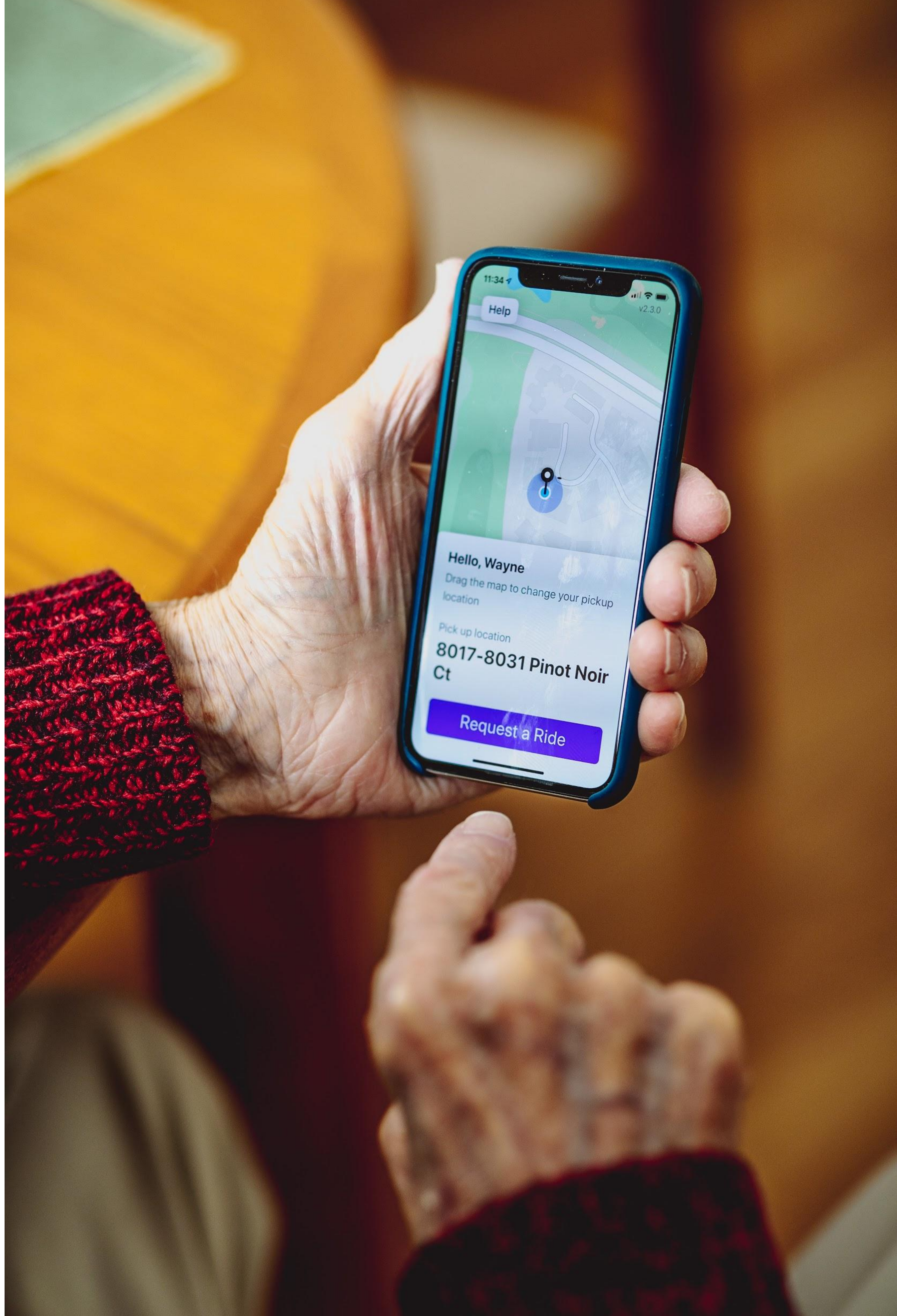
voyage

Hands-on

Let’s build a Physics system using the 2D Vector object

- collections
- iterators
- collisions

Bonus: visualizing the scene in 3D



voyage

Performance

- profiling
  - identify bottlenecks
- spatial acceleration structure
- comparison with C++

voyage

Q&A