Introduction to Programming - Day 2

Ben Evans

kathik

Going Further With Python

- Introducing OO
- OO in Python
  - Classes
  - Design Patterns

Introducing OO

- Object Oriented Programming
- Basic Definitions
- Thinking About Data
- State Models

A Warning

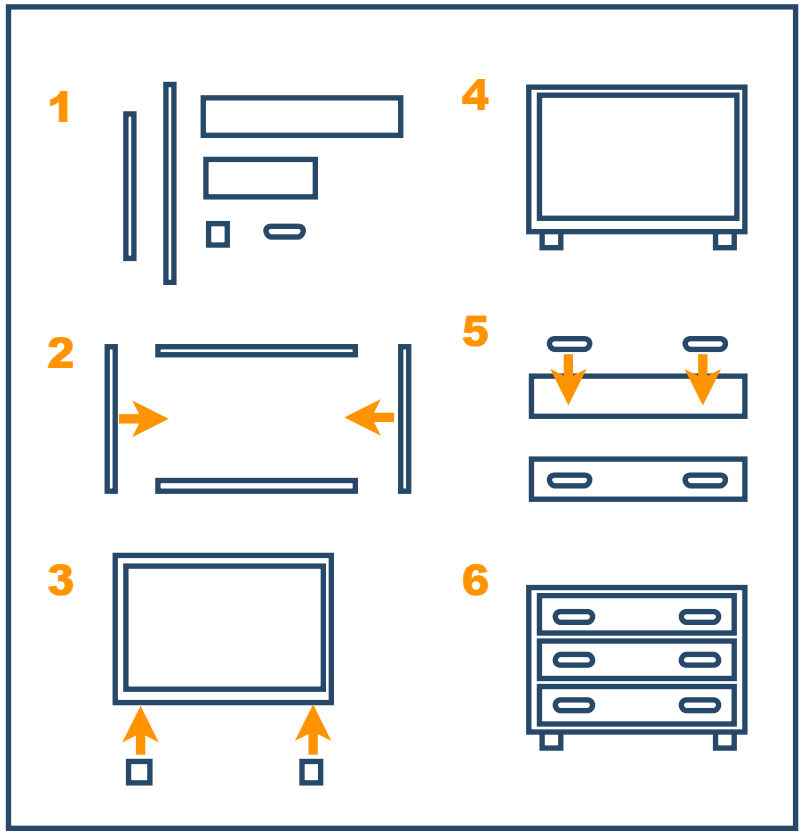
- If you have experience with OO programming, be careful
- "Object-oriented" has different meanings in different languages.
- Don't assume that Python works the same way as other OO languages
  - Particularly true for C++ programmers

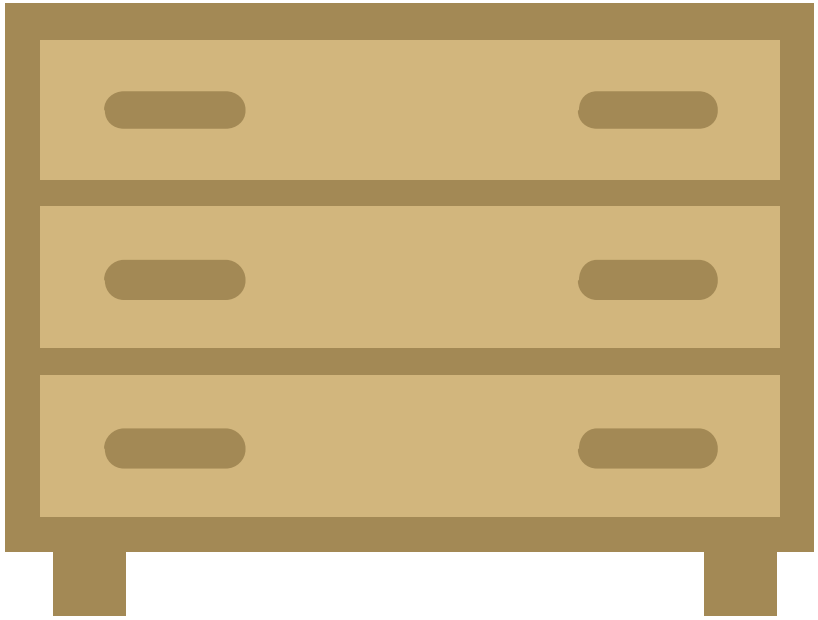
Definitions

- Class
  - A collection of
    - Data fields that hold values
    - Methods that operate on those values

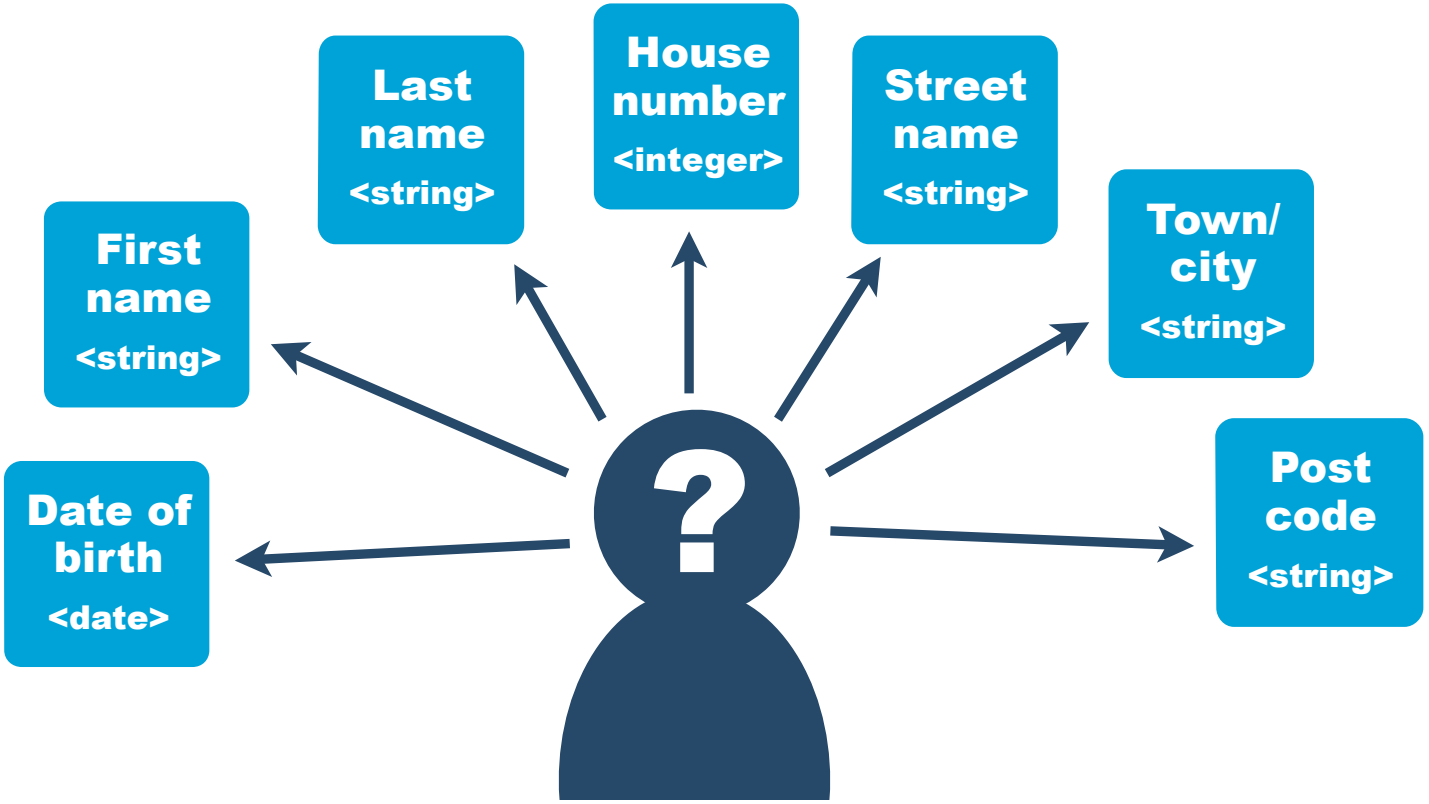
- Object
  - An instance of a class
  - “A bundle of state and methods to act on that state”

Class vs Object

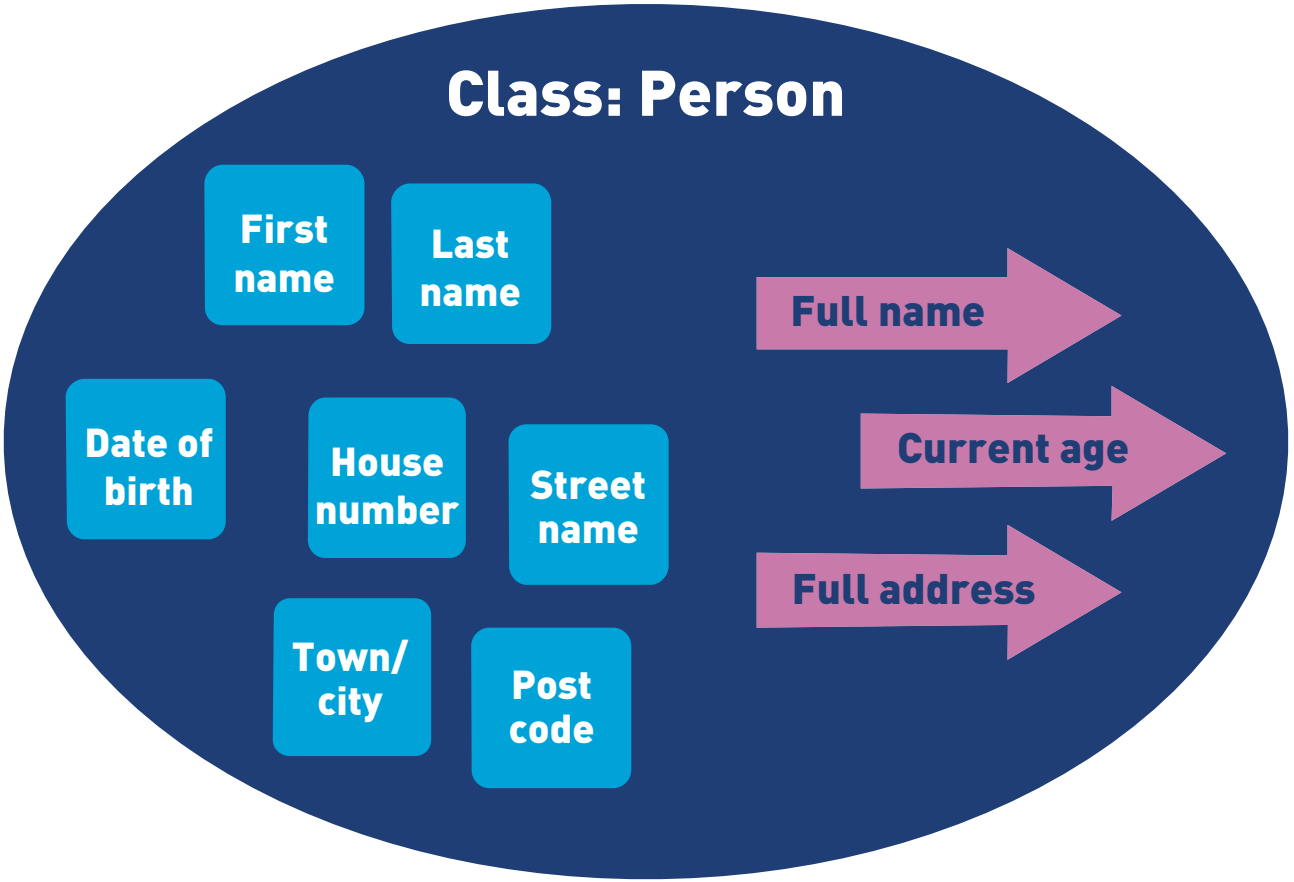




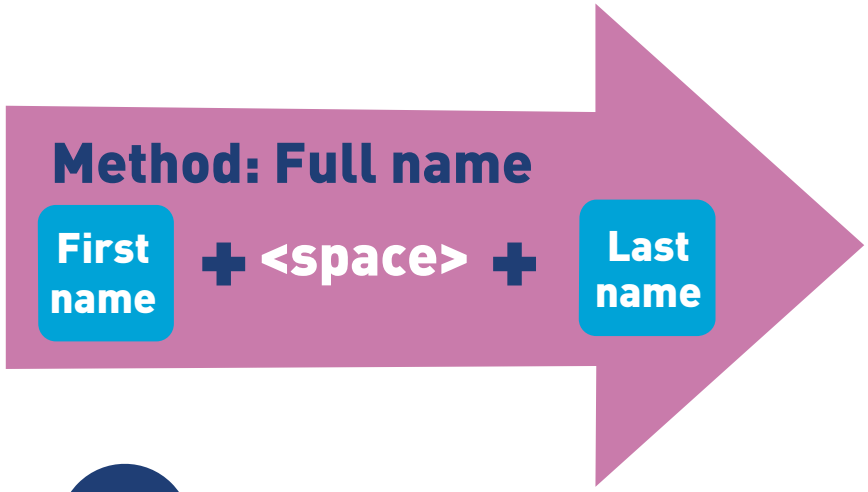
Thinking about data

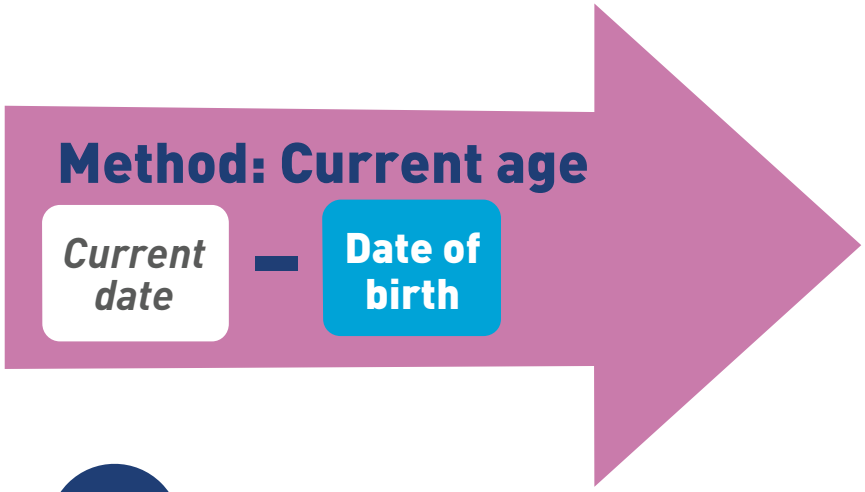


Classes


Class: Person
First name
Last name
Date of birth
House number
Street name
Town/ city
Post code
Full name
Current age
Full address

Methods

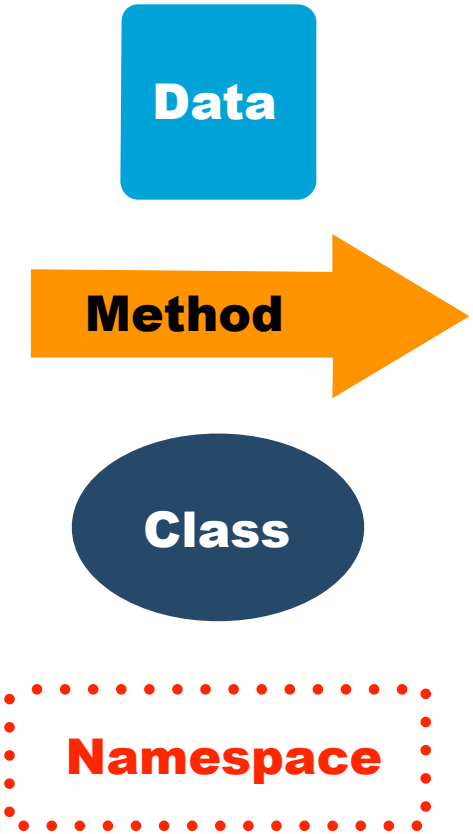




First name: Bob
Last name: Smith
Full name: **Bob Smith**

Current date: 24th May 2015
Date of birth: 21st May 1976
Current age: **38 years old**

Summary: Building blocks



OO in Python

- Classes in Python
- Composition
- Inheritance

OO in Python

- Let's see how to do this in Python
  - __init__ is the constructor

```
>>> class Address:
...     'This class represents an address'
...     def __init__(self, number, street, city, postcode):
...         self.number = number
...         self.street = street
...         self.city = city
...         self.postcode = postcode
...     def address(self):
...         return self.number + ' ' + self.street + ' ' + self.city + ' ' + self.postcode
...
>>> hudson_flat = Address("221b", "Baker St", "London", "NW1 1XX")
>>> hudson_flat.address()
'221b Baker St London NW1 1XX'
```

OO in Python

- Methods are public by default
- Methods are virtual by default
- Explicit ‘self’ parameter (provided via syntactic sugar)
- Classes are themselves objects
- Classes are created at runtime
- Python supports class fields as well as instance

OO in Python

- Extend from addresses to people...

```
>>> class Person:
...     'This class represents a person'
...     def __init__(self, first_name, surname, dob, address):
...         self.first_name = first_name
...         self.surname = surname
...         self.dob = dob
...         self.address = address
...     def full_name(self):
...         return self.first_name +' '+ self.surname
```

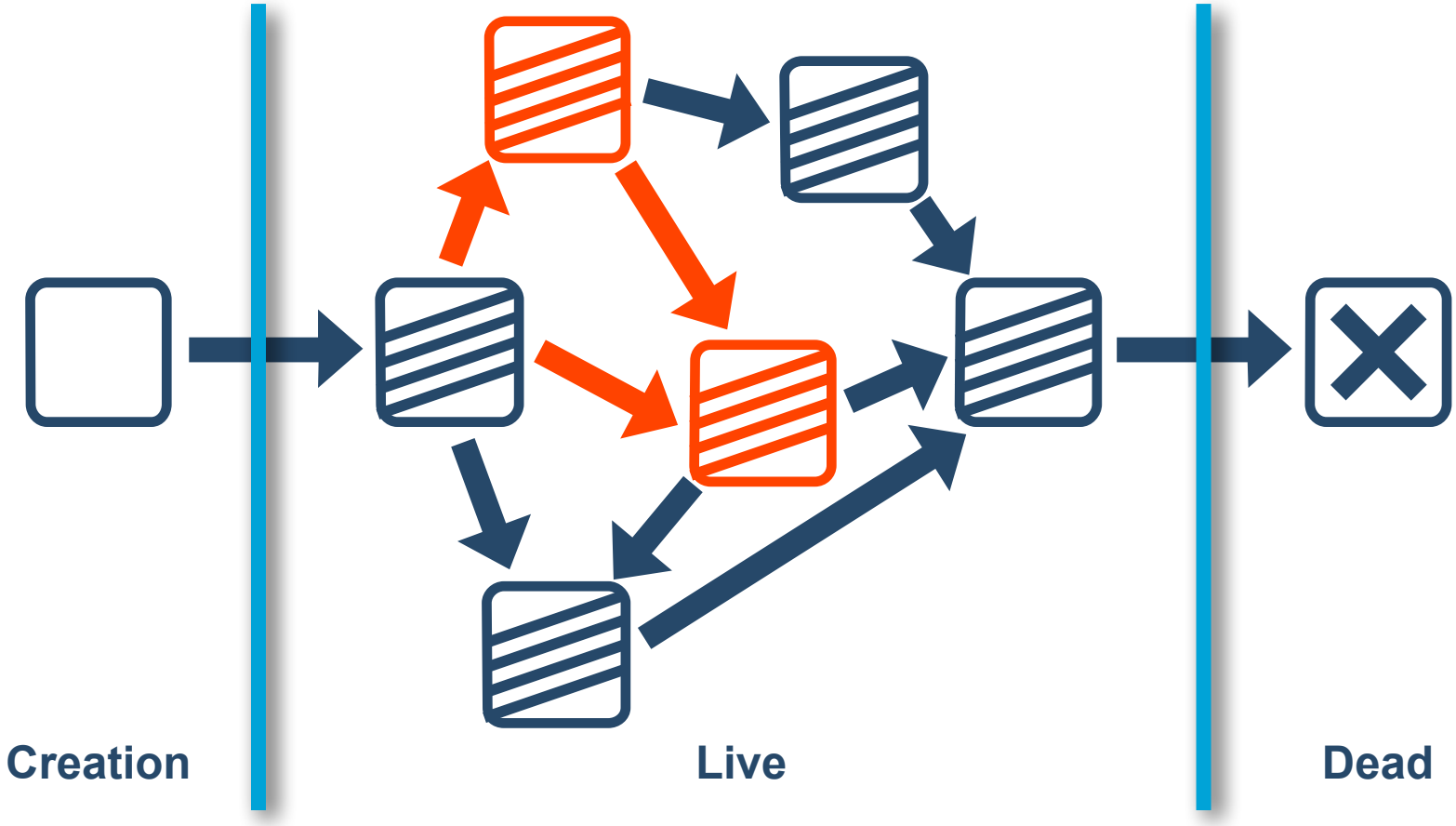
OO in Python

- Let's do some inheritance...
  - Note the explicit super call

```
class Detective(Person):
    def __init__(self, first_name, surname, dob, address, assistant):
        Person.__init__(self, first_name, surname, dob, address)
        self.assistant = assistant

>>> watson = Person("John", "Watson", "1974-03-22", hudson_flat)
>>> holmes = Detective("Sherlock", "Holmes", "1976-07-19", hudson_flat, watson)
```

OO State Transition Model



Group Exercise A

- Design an object system to represent iTunes listens
- Group Discussion: 20-30 minutes

Patterns & AntiPatterns

- Pattern
  - A small-scale best practice in code
  - E.g. Iterator, Adapter, Singleton, Write-behind journal
- Antipattern
  - A larger-scale dysfunction of a project or team
  - E.g. Not Invented Here

Patterns: Recurring Themes

- How can we learn how to do better?
- Pattern Languages
  - A way of talking about software design
  - Higher level of abstraction
  - The essence of the software
  - Not tied to specific projects
  - Developers love war stories

Adapter

kathik

Adapter

Simplicity?

Londoner (proud of the Tube system, to friends from the country). "THERE'S THE WHOLE THING, YOU SEE! *ABSOLUTELY SIMPLE!*"

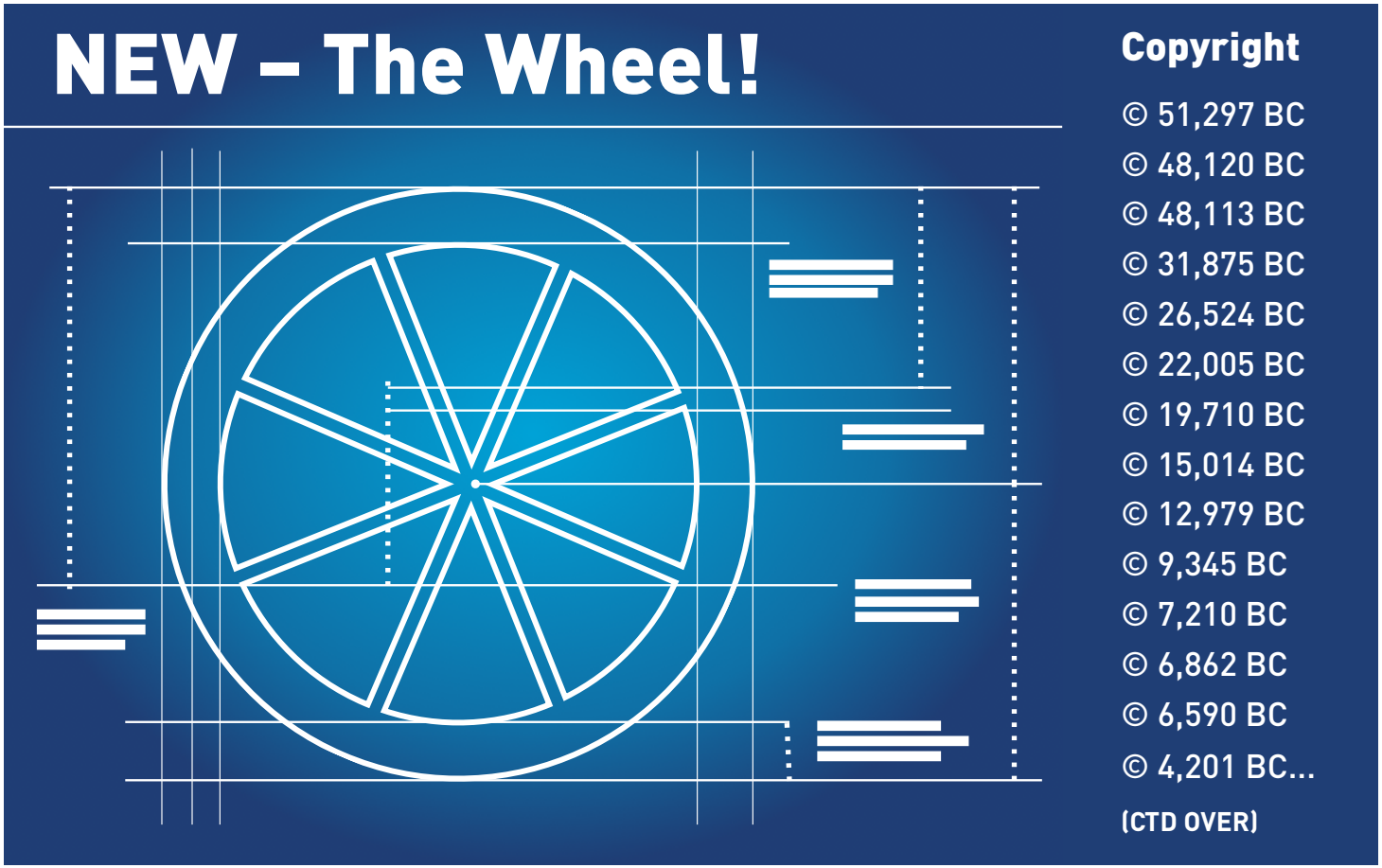
kathik

Antipatterns: Study The Problems

- How can we learn how to do better?
- Study the problem
- Look at failure cases
- Learn from them
- See how to avoid them

AntiPatterns

Not Invented Here



Blame Donkey

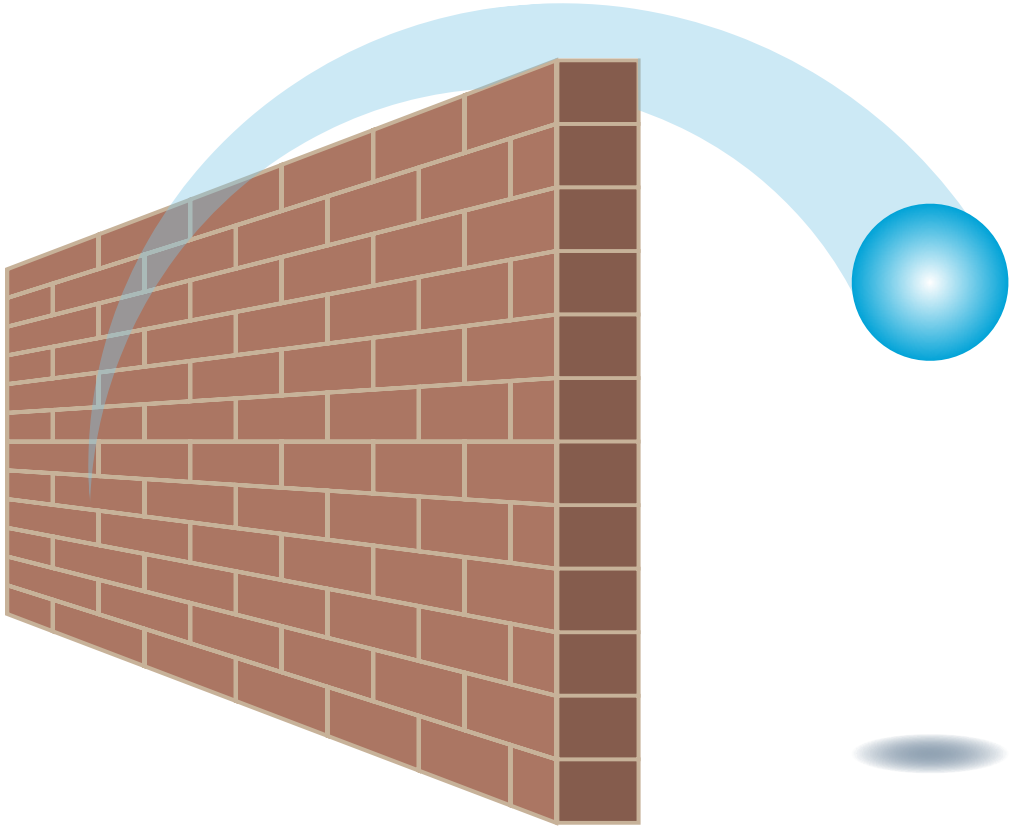
Blame Donkey

- Certain components are always identified as the issue
  - "It's always JMS / Hibernate / ANOTHERTECH"
    - AntiPattern often displayed by management / biz
    - But Technology is not immune

- Reality
  - Insufficient analysis has been done to reach this conclusion

- Solutions
  - Resist pressure to rush to conclusions
  - Perform analysis as normal
  - Communicate the results of analysis to all stakeholders

Throw It Over The Wall



kathik

Throw It Over The Wall

- Teams focus on their immediate domain
  - "It has plenty of unit tests. QA should take over from here"
- Reality
  - Preventable problems pass into production
- Building better cross-team relationships helps
  - Seek to understand what other teams actually do
  - Outages and releases are much less painful
  - Hard to do with remote L1 operations teams