Programming Paradigms
&
Object-Oriented Programming

Programming paradigms

Way to approach and structure your code

Procedural
Object-Oriented
Functional

Many programming languages are multi-paradigm --
-- they support multiple paradigms

Python, JavaScript, Java, Ruby, many others

Procedural Programming

Easiest to understand at first

Linear, step-by-step actions to perform tasks

Tasks are often encapsulated into modular procedures
-- get user input, write user input to a file, perform a calculation, etc

The abstract concept of a procedure is implemented
as functions in Python - a modular, call-able block of code

Other programming languages may use other terms:
procedure, routine, subroutine, etc.

Object-oriented programming (OOP)

Objects are the center of focus

Concept of an object:
A unit that bundles together related data and executable code

In Python, we call these attributes (data) and methods (executable code)
Python object *attributes* are analogous to JavaScript object properties

Functional programming

Like procedural programming, focuses on functions

But don't confuse with procedural programming!

Tenets: Pure functions, avoidance of side effects, use of immutable data

OOP: Classes

Class-based approach to OOP:

A class is a blueprint for an object, like an architectural blueprint

Objects are instances of a class,
like houses built from an architectural blueprint

Many objects can be created from the same class

Can think of classes as a custom data type/data structure

OOP: Class inheritance

Class inheritance – child classes created based on parent classes

Usually more complex classes from simpler classes, share base code

Example:

Parent class: House
Child classes: Cottage, Mansion, etc, all inherit from House class

OOP example: Red Alert

Game with different types of buildings that can produce soldiers, warships, and tanks

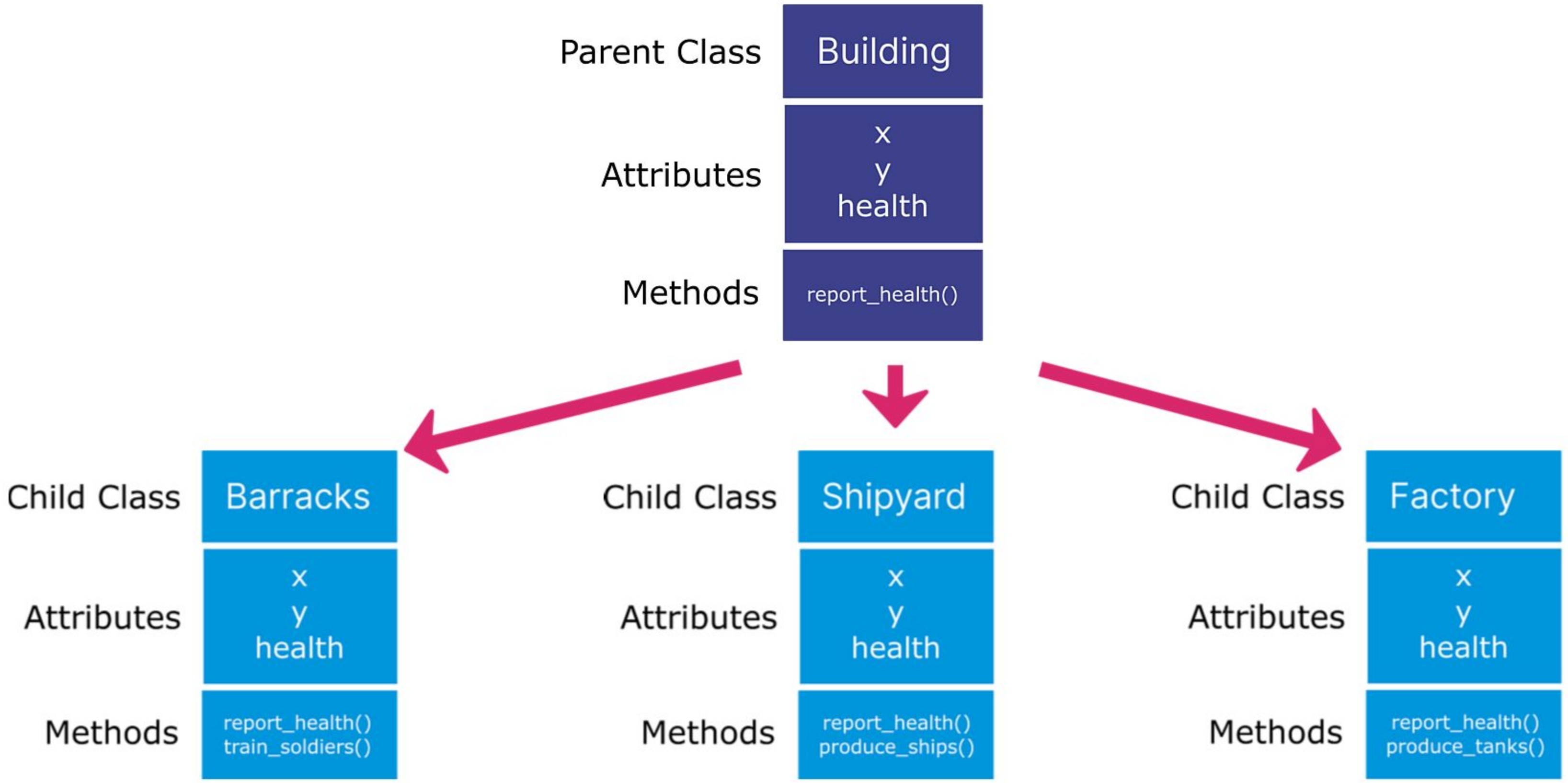
Types of buildings:
barracks, shipyard, factory

Parent class of Building:
Attributes: x coordinate, y coordinate, health, etc
Methods (actions): report_health(), etc

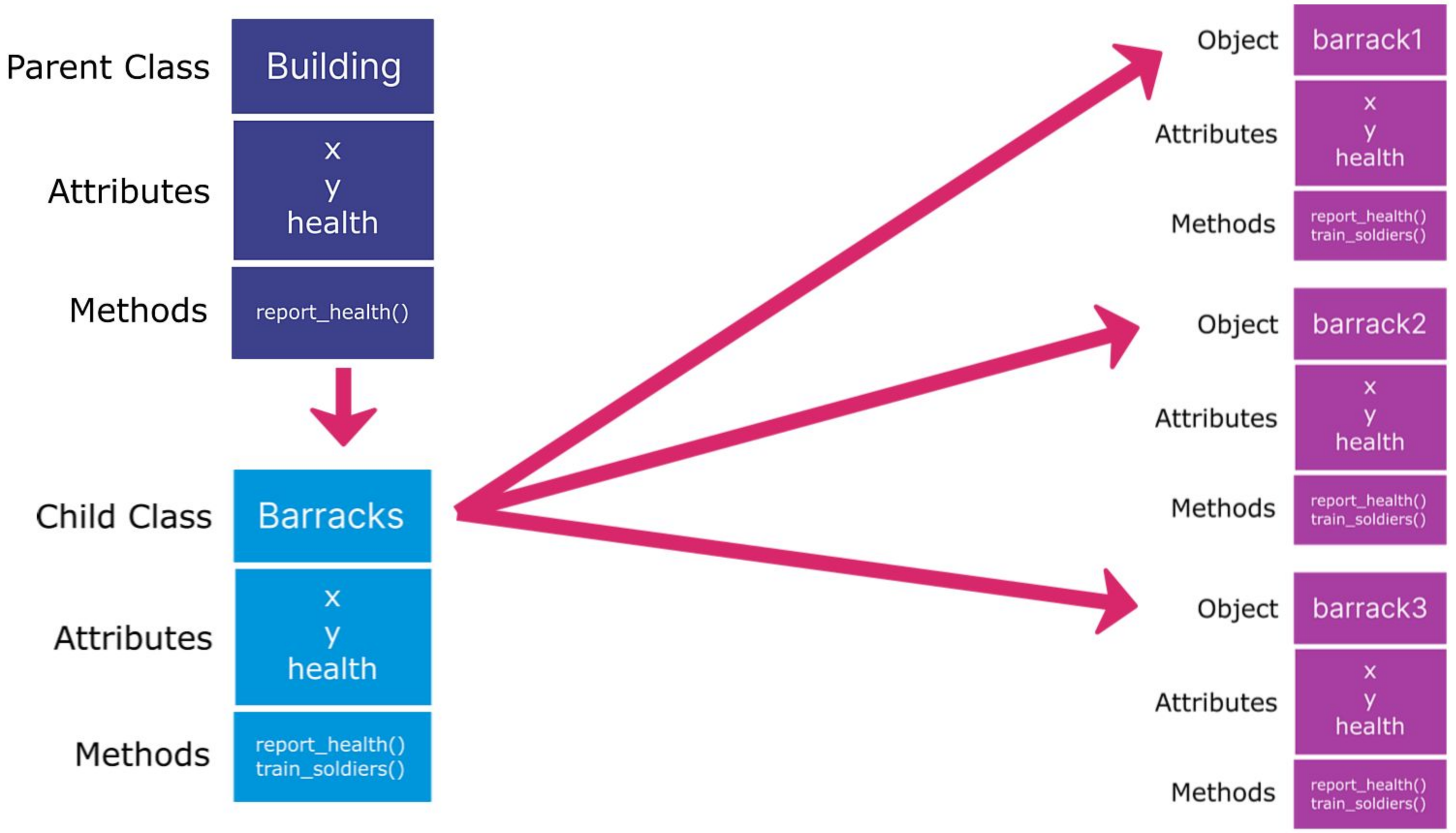
OOP example: Red Alert

Parent Class	Building
Attributes	x y health
Methods	report_health()

OOP example: Red Alert



Object instantiation



Python and OOP

Python is multi-paradigm but often considered best for OOP

Everything is an object in Python

All data types can be considered built-in classes

All values with a data type can be considered objects –
instances of their data type/class

Python and OOP

A string is a String object – an instance of the class of String, with methods such as lower()

A list is a List object – an instance of the class of List, with methods such as pop()

In Python, anything that can be assigned as the value of a variable can be considered an object, capable of having attributes and methods

Functions are also objects