Object-Orientation

Carl Friedrich Bolz, David Schneider
Dynamische Programmiersprachen
Heinrich-Heine-Universität Düsseldorf
Sommersemester 2010

What is an Object?

An object consist of two elements:

- State + Behaviour

Why Object-Orientation?

Abstraction
To use an object, we don't need to know how it is implemented.
Modularity
Programs are split up into independent parts.
Extensibility
Can enhance a program without changing existing parts.

Object-Orientation in Dynamic Languages

Two important principles for OO in dynamic languages:

- Everything is an object
- Every object is an instance of a class (a type)

Messages

- Objects interact by sending messages to each other
- The implementation of a message is called a method
- message send consists of two steps: *lookup* and *call*
- lookup finds which method to use, call calls it
- lookup happens fully at runtime!
- the message send is the primitive operation of program execution

Inheritance

- inheritance (or subclassing) is the way in which classes can be related to each other
- every class can have a superclass
- determines how message sends behave, thus can be used for code-sharing

Python: Instance-Attributes

- Instance-attributes are the state part of an object
- they are stored in a dictionary attached to the instance
- the instance attributes can be accessed via the `__dict__` attribute

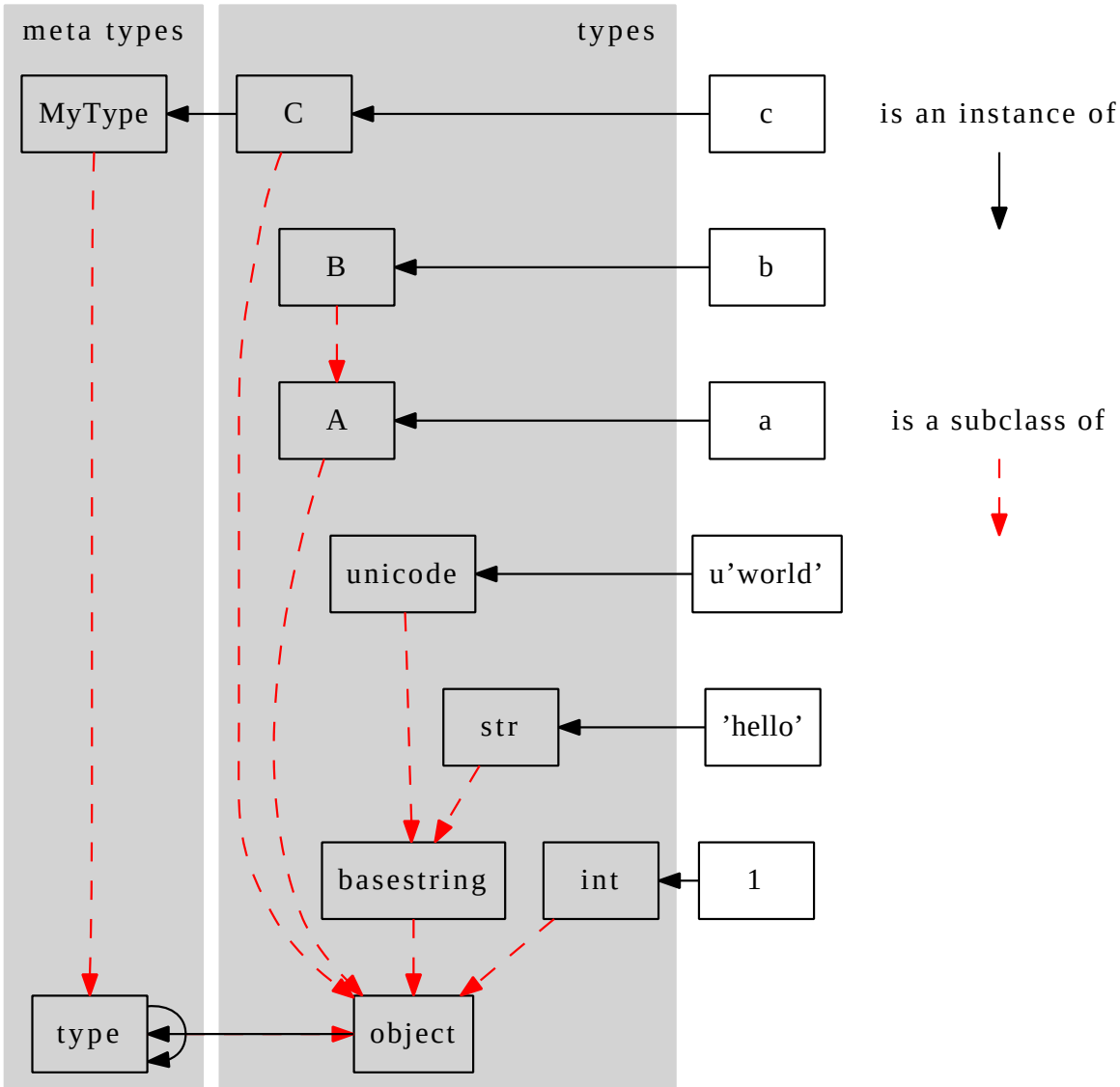
Python: Object Creation

- instances are created by calling a type
- classes are normal instances of type `type`

Python: Introspecting Objects

- the class of an object can be found via the `__class__` attribute
- the base class of a type can be accessed with the `__base__` attribute

Python: Basics of the Object Model



(many of the arrows from classes to `type` left out)