Python's Object Model

Petr Zemek

March 26, 2020

TI Systems Seminar @ Avast

Before we Begin

- The talk is a live demo accompanied by examples:

  https://github.com/s3rvac/talks/tree/master/2020-03-26-Python-Object-Model/examples

- Disclaimer: In what follows, *Python* means Python 3.6 or newer.

Getting Started

- What is an object?
- What is an object model?
- The basics (classes, methods, properties, inheritance).
- Very little of Python is truly magical.
- In Python, everything is an `object`.
- In Python, every object has an identity, type, and value.
- Objects in Python do not generally have a fixed layout.
- Construction and finalization.
- Multiple inheritance, MRO.
- Methods vs functions.

Continuing...

- The underlying storage is a `dict`.
- `__dict__` vs `__slots__`
- What happens when you access an attribute of an object?
- Hooking into attribute access (`__getattr__`, `__getattribute__`).
- Descriptors: The mechanism behind methods, properties, static/class methods.

...and Finishing with Metaclasses

- What is metaprogramming?
- Classes are instances of *metaclasses*.
- `type`: the default metaclass.
- Creating a class manually via `type()`.
- The mysterious relationship between `type` and `object`.
- What happens when you create and instantiate a class?
- What happens when you access an attribute of a class?
- Alternatives to metaclasses:
  - class decorators
  - `__init_subclass__`
  - `__set_name__`
  - code generation

Further Reading and Watching

- Python 3 Docs: Data model
- Python 3 Docs: Descriptors
- Python 3 Docs: Types
- eev.ee: Object models
- blog.ionelmc.ro: Understanding Python metaclasses
- marco-buttu.github.io: Python's object model
- stackoverflow.com: Usage of slots
- David Beazley: Python 3 Metaprogramming (video, 3 hours)
- Mark Smith: Python Types & Metaclasses Made Simple (video, 1 hour)