Import mangling in `torch.package`

Mangling rules

These are the core invariants; if you are changing mangling code please preserve them.

1. For every module imported by `PackageImporter`, two attributes are mangled:
   - `__module__`
   - `__file__`
2. Any `__module__` and `__file__` attribute accessed inside `Package{Ex|Im}porter` should be demangled immediately.
3. No mangled names should be serialized by `PackageExporter`.

Why do we mangle imported names?

To avoid accidental name collisions with modules in `sys.modules`. Consider the following:

```
from torchvision.models import resnet18
local_resnet18 = resnet18()

# a loaded resnet18, potentially with a different implementation than the local one!
i = torch.PackageImporter('my_resnet_18.pt')
loaded_resnet18 = i.load_pickle('model', 'model.pkl')

print(type(local_resnet18).__module__)  # 'torchvision.models.resnet18'
print(type(loaded_resnet18).__module__)  # ALSO 'torchvision.models.resnet18'
```

These two model types have the same originating `__module__` name set. While this isn't facially incorrect, there are a number of places in `cpython` and elsewhere that assume you can take any module name, look it up `sys.modules`, and get the right module back, including: - `import_from` - `inspect`: used in TorchScript to retrieve source code to compile - …probably more that we don't know about.

In these cases, we may silently pick up the wrong module for `loaded_resnet18` and e.g. TorchScript the wrong source code for our model.

How names are mangled

On import, all modules produced by a given `PackageImporter` are given a new top-level module as their parent. This is called the `mangle parent`. For example:

`torchvision.models.resnet18`

becomes

`<torch_package_0>.torchvision.models.resnet18`

The mangle parent is made unique to a given `PackageImporter` instance by bumping a process-global `mangle_index`, i.e. `<torch__package{mangle_index}>`.

The mangle parent intentionally uses angle brackets (`<` and `>`) to make it very unlikely that mangled names will collide with any "real" user module.

An imported module's `__file__` attribute is mangled in the same way, so:

`torchvision/modules/resnet18.py`

becomes

`<torch_package_0>.torchvision/modules/resnet18.py`

Similarly, the use of angle brackets makes it very unlikely that such a name will exist in the user's file system.

Don't serialize mangled names

Mangling happens `on import`, and the results are never saved into a package. Assigning mangle parents on import means that we can enforce that mangle parents are unique within the environment doing the importing.

It also allows us to avoid serializing (and maintaining backward compatibility for) this detail.