Warning: This doc presents the end state where we want to be with our public API. And as of writing, there are still some function that "look public" even though they should be private. As of writing, if a function "looks public" and is documented on https://pytorch.org/docs, then you can rely on it being public. Otherwise, if you plan to rely on a function that is not documented on the website, please open an issue first to ask if this function is indeed public or unexpectedly exposed and might get removed in the future.

What is public API within the PyTorch project

A submodule is considered public if no name in its path starts with `_`. Other objects, in particular functions and classes, are public if they have a `__module__` attribute, it starts with `torch.`, corresponds to a public submodule and the object's name itself does not start with "_".

To ensure that we can check that all objects that "looks public" are expected, we also have the following constraints for public submodules within the torch namespace. They should either:

- Define `__all__`
  - In this case, all callable and classes in `__all__` must be public and have their `__module__` start with the current submodule.
  - All attributes that are not in `__all__` must NOT be public and have their `__module__` start with the current submodule.
- Not define `__all__`
  - In this case, all the objects in `dir(submod)` that do not start with `_` must have their `__module__` start with the current submodule.

How do we check compliance of submodules?

The submodule compliance is tested in https://github.com/pytorch/pytorch/blob/master/test/test_public_bindings.py This ensures that:

- Our C++ bindings do not expose private API in `torch.` namespace (for example JIT types or internal objects that are missing a leading `_`).
- Ensure that all submodules within the `torch.` namespace follow the rules above.

If a newly added function breaks any of these tests, you should make sure that the submodule in question is compliant. During the transition period while we fix many submodules, it might be simpler to first fix the submodule (adding a proper `__all__` field) before adding your change.

How do we check docs?

We use sphinx's coverage tool doc to check that the documentation on our website covers all the public APIs. If your change breaks this check when building the doc, you should make sure that the newly added public function has the right docstring and that it is properly referenced in a `.rst` file in the `docs/` folder.

On top of that, we also have a special check that runs alongside the basic coverage tool src that ensures that all public submodules of the `torch.` namespace is properly referenced in the online documentation. If your change break this test, follow the hints in the error message to fix it by properly documenting it in a `.rst` file.