

# Tensor Indexing API

Indexing a tensor in the PyTorch C++ API works very similar to the Python API. All index types such as `None` / `...` / `integer` / `boolean` / `slice` / `tensor` are available in the C++ API, making translation from Python indexing code to C++ very simple. The main difference is that, instead of using the `[]`-operator similar to the Python API syntax, in the C++ API the indexing methods are:

- `torch::Tensor::index` ([link](#))

- **System Message: WARNING/2** (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\cpp\source\notes\[pytorch-master][docs][cpp][source][notes]tensor\_indexing.rst, line 2); [backlink](#)

Duplicate explicit target name: "link".

`torch::Tensor::index_put_` ([link](#))

It's also important to note that index types such as `None` / `Ellipsis` / `Slice` live in the `torch::indexing` namespace, and it's recommended to put using namespace `torch::indexing` before any indexing code for convenient use of those index types.

Here are some examples of translating Python indexing code to C++:

## Getter

Python	C++ (assuming using namespace torch::indexing)
<code>tensor[None]</code>	<code>tensor.index({None})</code>
<code>tensor[Ellipsis, ...]</code>	<code>tensor.index({Ellipsis, "..."})</code>
<code>tensor[1, 2]</code>	<code>tensor.index({1, 2})</code>
<code>tensor[True, False]</code>	<code>tensor.index({true, false})</code>
<code>tensor[1::2]</code>	<code>tensor.index({Slice(1, None, 2)})</code>
<code>tensor[torch.tensor([1, 2])]</code>	<code>tensor.index({torch::tensor({1, 2})})</code>
<code>tensor[..., 0, True, 1::2, torch.tensor([1, 2])]</code>	<code>tensor.index({ "...", 0, true, Slice(1, None, 2), torch::tensor({1, 2}) })</code>

## Setter

Python	C++ (assuming using namespace torch::indexing)
<code>tensor[None] = 1</code>	<code>tensor.index_put_({None}, 1)</code>
<code>tensor[Ellipsis, ...] = 1</code>	<code>tensor.index_put_({Ellipsis, "..."}, 1)</code>
<code>tensor[1, 2] = 1</code>	<code>tensor.index_put_({1, 2}, 1)</code>
<code>tensor[True, False] = 1</code>	<code>tensor.index_put_({true, false}, 1)</code>
<code>tensor[1::2] = 1</code>	<code>tensor.index_put_({Slice(1, None, 2)}, 1)</code>
<code>tensor[torch.tensor([1, 2])] = 1</code>	<code>tensor.index_put_({torch::tensor({1, 2})}, 1)</code>
<code>tensor[..., 0, True, 1::2, torch.tensor([1, 2])] = 1</code>	<code>tensor.index_put_({ "...", 0, true, Slice(1, None, 2), torch::tensor({1, 2}) }, 1)</code>

## Translating between Python/C++ index types

The one-to-one translation between Python and C++ index types is as follows:

Python	C++ (assuming using namespace torch::indexing)
<code>None</code>	<code>None</code>
<code>Ellipsis</code>	<code>Ellipsis</code>
<code>...</code>	<code>"..."</code>
<code>123</code>	<code>123</code>
<code>True</code>	<code>true</code>
<code>False</code>	<code>false</code>
<code>: or ::</code>	<code>Slice()</code> or <code>Slice(None, None)</code> or <code>Slice(None, None, None)</code>
<code>1: or 1::</code>	<code>Slice(1, None)</code> or <code>Slice(1, None, None)</code>
<code>:3 or :3:</code>	<code>Slice(None, 3)</code> or <code>Slice(None, 3, None)</code>
<code>::2</code>	<code>Slice(None, None, 2)</code>
<code>1:3</code>	<code>Slice(1, 3)</code>
<code>1::2</code>	<code>Slice(1, None, 2)</code>
<code>:3:2</code>	<code>Slice(None, 3, 2)</code>
<code>1:3:2</code>	<code>Slice(1, 3, 2)</code>

Python	C++ (assuming using namespace torch::indexing)
<code>torch.tensor([1, 2])</code>	<code>torch::tensor({1, 2})</code>