Memory format aware operators are the operators which satisfy two requirements:

- they generate output in same memory format as inputs
- they use the most efficient kernels for each different memory formats

Let say we want to add/modify `operator` to support `torch.channels_last` memory format.

```
in_tensor = x.contiguous(memory_format=torch.channels_last)
out_tensor = torch.operator(in_tensor)
print(out_tensor.is_contiguous(memory_format=torch.channels_last)) # True
```

To do so, we need to modify the operator's CPP code. An old version of operator might look similar to this:

```
auto output_tensor = at::empty_like(input_tensor);
// .... standard kernel for contiguous or strided tensors
return output_tensor;
```

The preferred way of writing memory format aware operators is to use the `switch` operator. This approach allows us to expand memory formats support in the future.

```
// ...
auto memory_format = input_tensor.suggest_memory_format();
auto output_tensor = at::empty(output_shape, memory_format);

switch (memory_format) {
  case MemoryFormat::ChannelsLast: {
    auto input_cl_contiguous = input_tensor.contiguous(
        MemoryFormat::ChannelsLast); // if kernel requires memory dense
                                     // tensor
    // .... kernel code
    break;
  }
  case MemoryFormat::Contiguous: {
    // .... standard kernel for contiguous or strided tensors
    break;
  }
  default:
    TORCH_CHECK(
        false,
        "Unsupported memory format. Supports only ChannelsLast, Contiguous");
}
// ...
```

Important to learn that `suggest_memory_format` is not similar to `input_tensor.is_contiguous(...)`, see function comments.

More memory format handling required when you are writing `_out` operator implementation.

```
in_tensor = x.contiguous(memory_format=torch.channels_last)
out_tensor = o.contiguous(memory_format=torch.contiguous_format)
```

```
torch.operator(in_tensor, out=out_tensor)
print(out_tensor.is_contiguous(memory_format=torch.contiguous_format)) # True
```

Keeping the memory format of the output is essential. However, some performant algorithms require matching formats of inputs and outputs. In this case, it is possible to do a `copy_` trick.

```
Tensor self_or_new_memory_format(Tensor& self, MemoryFormat memory_format) {
    if (self.is_contiguous(memory_format)) {
        return self;
    }
    return at::empty_like(self, self.options(), memory_format);
}
```

```
// ...
auto memory_format = input_tensor.suggest_memory_format();

assert_no_internal_overlap(output);
if (output_shape != output.sizes()) {
    output.resize_(output_shape, memory_format);
}

auto temporary_output_tensor = self_or_new_memory_format(output, memory_format);

switch (memory_format) {
  case MemoryFormat::ChannelsLast: {
    auto input_cl_contiguous = input_tensor.contiguous(
        MemoryFormat::ChannelsLast); // if kernel requires memory dense
                                     // tensor
    // .... kernel code
    break;
  }
  case MemoryFormat::Contiguous: {
    // .... standard kernel
    break;
  }
  default:
    TORCH_CHECK(
        false,
        "Unsupported memory format. Supports only ChannelsLast, Contiguous");
}

if (!output.is_same(temporary_output_tensor)) {
    output.copy_(temporary_output_tensor);
}
// ...
```

In some cases, there is no performant algorithm for contiguous or channels last inputs, so the same trick with temporary tensors and `copy_` can be applied.

```
// ...
auto memory_format = input_tensor.suggest_memory_format();

assert_no_internal_overlap(output);
if (output_shape != output.sizes()) {
    output.resize_(output_shape, memory_format);
}

auto temporary_output_tensor = self_or_new_memory_format(output,
MemoryFormat::ChannelsLast);
auto input_cl_contiguous = input_tensor.contiguous(MemoryFormat::ChannelsLast);
// .... channels last kernel code

if (!output.is_same(temporary_output_tensor)) {
    output.copy_(temporary_output_tensor);
}
// ...
```

Or you can do hard exit with unsupported memory format message (this is least preferred way, and we consider such operators incomplete).

```
// ...
switch (memory_format) {
  case MemoryFormat::ChannelsLast: {
    auto input_cl_contiguous = input_tensor.contiguous(
        MemoryFormat::ChannelsLast); // if kernel requires memory dense
                                     // tensor
    // .... kernel code
    break;
  }
  case MemoryFormat::Contiguous:
  default:
    TORCH_CHECK(
        false,
        "Unsupported memory format. Supports only ChannelsLast");
}
// ...
```

Please do not forget to cover all scenarios with unit tests. We had seen countless cases when simple test saved hours of debugging.