CNTK Layers vs. CNTK Graph API

CNTK Layers	CNTK Graph API
`z = f(x,y)`	`z = f.clone(CloneMethod.share, {f.arguments[0]: x, f.arguments[1]: y})`

`z = f(lbl=y, qry=x)`	`params_dict = { arg.name: arg for arg in f.arguments }`
	`z = f.eval({params_dict['lbl']: y, params_dict['qry']: x})`

`z = f([[2, 3]], [[4, 5]])`	`z = f.eval({f.arguments[0]: [[2, 3]], f.arguments[1]: [[4, 5]]})`
`s = v[...,13:42,:]`	`s = slice(v, begin_index=13, end_index=42, axis=-2)`
`E = model.embed.E`	`E = find_by_name(find_by_name(model, 'embed').block_root, 'E')`

`@Function`
`def f(x, y):`
`    return (`*expr1*`(x,y), `*expr2*`(x,y))`
	`x, y = ordered(placeholder_variable('x'),`
	`               placeholder_variable('y'))`
	`f = combine([`*expr1*`(x,y), `*expr2*`(x,y)])`

`@BlockFunction('f', name)`
`def f(x, y):`
`    return `*expr*`(x, y)`
	`x, y = (placeholder_variable('x'), placeholder_variable('y'))`
	`x1, y1 = [placeholder_variable(arg.name) for arg in (x,y)]`
	`f = `*expr*`(x1, y1)`
	`f = as_block(f, {x1: x, y1: y}, 'f', name)`

`@Function`
`def f(x: Sequence[Tensor[(13,42)]])`
`    return `*expr*`(x)`
	`x = input_variable((13,42),`
	`              dynamic_axes=Axis.default_input_variable_dynamic_axes())`
	`f = `*expr*`(x)`