Toy PyG Example

```
import torch_geometric as pyg
from torch_geometric.datasets import Planetoid
```

- load the dataset

```
dataset = Planetoid(root="myprj", name = "Cora")
```

↳ dataset.{num_classes, num_node_features, num_edge_features}

dataset.data

```
import torch
import torch.nn.functional as F
from torch_geometric.nn import SAGEConv
```

```
data = dataset[0]
```

- define the neural net

```
class Net(torch.nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.Conv = SAGEConv(dataset.num_features,  → #in_channels
                             dataset.num_classes,   → #out_channels
                             aggr = "max")
                                    ↳ mean, add, ...

    def forward(self):
        x = self.Conv(data.x, data.edge_index)
        return F.log_softmax(x, dim=1)
```

- do some setup

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model, data = Net().to(device), data.to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01,
                             weight_decay=5e-4)
```

- training loop

```
def train():
    model.train()
    optimizer.zero_grad()
    F.nll_loss(model()[data.train_mask], data.y[data.train_mask]).
                                                          backward()
    optimizer.step()
```

- test loop

```
def test():
    model.eval()
    logits, accs = model(), []
    for _, mask in data('train_mask', 'val_mask', 'test_mask'):
        pred = logits[mask].max(1)[1]
        acc = pred.eq(data.y[mask]).sum().item()/
                                        mask.sum().item()
        accs.append(acc)
    return accs
```

- train

```
best_val_acc = test_acc = 0
for epoch in range(epochs):
    train()
    _,val_acc, tmp_test_acc = test()
    if val_acc > best_val_acc:
        best_val_acc = val_acc
        test_acc = tmp_test_acc
```