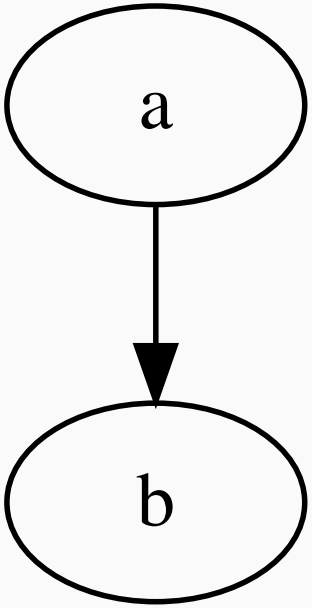
The DOT language

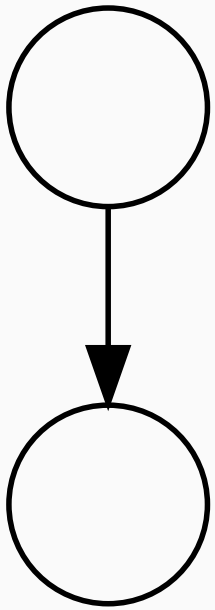
Pierce Edmiston

```
digraph {
  a -> b;
}
```

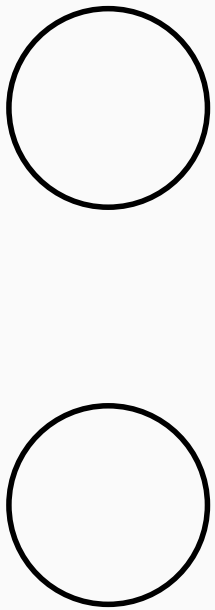


Node shapes

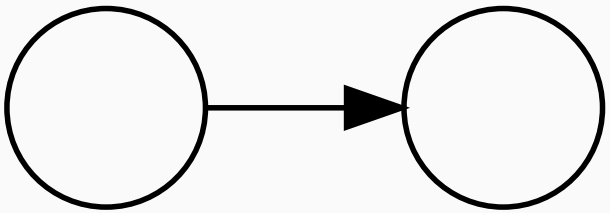
```
digraph {
  node[shape=circle label=""];
  a -> b;
}
```



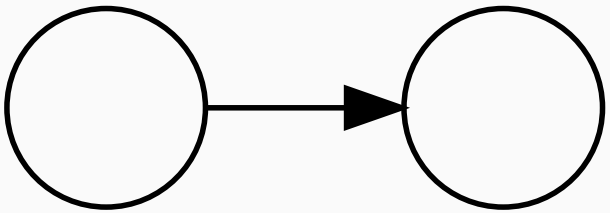
```
digraph {
  node[shape=circle label=""];
  edge[style=invis];
  a -> b;
}
```



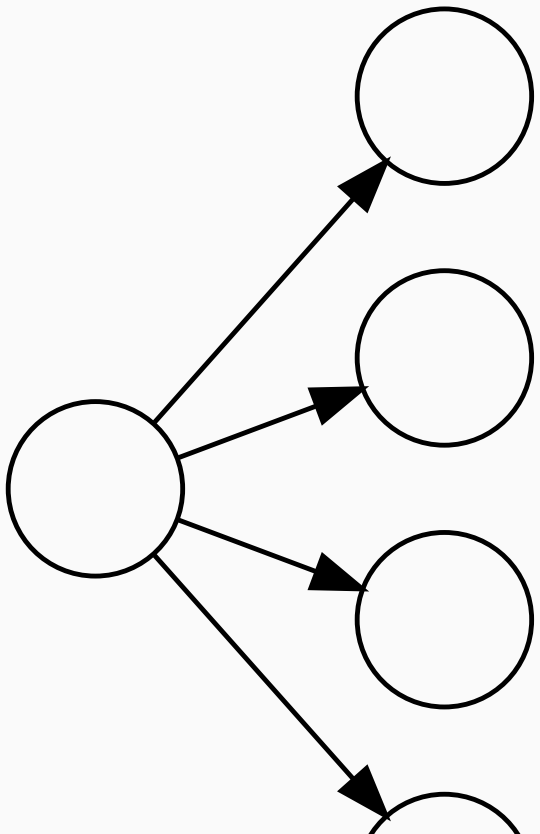
```
digraph {
  graph[rankdir=LR];
  node[shape=circle label=""];
  a -> b;
}
```



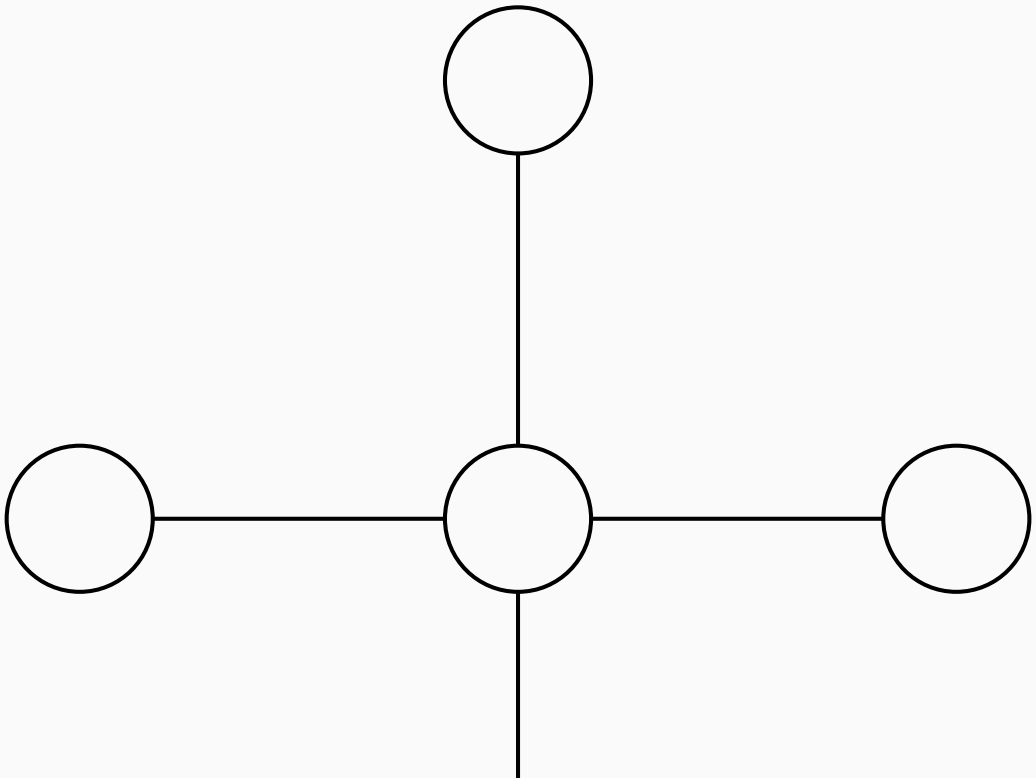
```
digraph {
  rankdir=LR;
  node[shape=circle label=""];
  a -> b;
}
```



```
digraph {
  rankdir=LR;
  node[shape=circle label=""];
  a -> {b, c, d, e};
}
```



```
graph {
  rankdir=LR;
  layout=circo;
  node[shape=circle label=""];
  a -- {b, c, d, e};
}
```

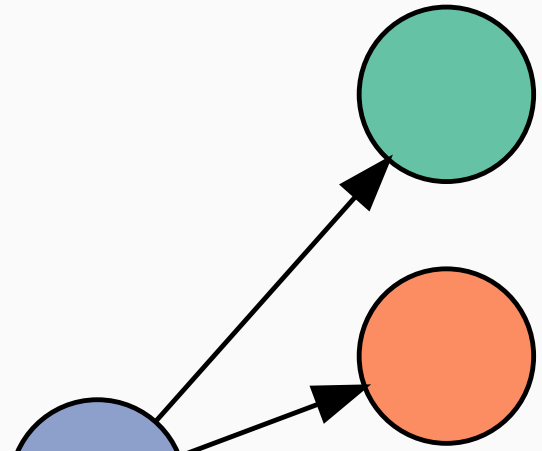


```
graph {
  rankdir=LR;
  layout=circo;
  node[shape=circle label=""];
  a -- {b, c, d, e};
  b -- {c, d, e};
  c -- {d, e};
  d -- e;

  a[label="Pierce"];
  b[label="Willy"];
  c[label="Ed"];
  d[label="Dan"];
  e[label="Josh"];
}
```

Graph, Node, Edge attrs

```
digraph {
  rankdir=LR;
  node[shape=circle label="" style=filled];
  a -> {b, c, d, e};
  a[fillcolor="#8da0cb"]
  c[fillcolor="#fc8d62"]
  b, d, e[fillcolor="#66c2a5"]
}
```



```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier, export_gra
import graphviz

# ...

def plot_decision_tree(model: DecisionTreeClassifier):
    iris = datasets.load_iris()
    dot_data = export_graphviz(model, out_file=None,
                               feature_names=iris.feature_
                               class_names=iris.target_nam
                               impurity=False,
                               filled=True,
                               rounded=True,
                               special_characters=True)
    graph = graphviz.Source(dot_data)
```

Visualizing Wikipedia article revision history, graph function

```
def graph(edges, nodes=None, remove_labels=False):
    """Create a revision history Digraph from a pandas Data
    g = graphviz.Digraph(graph_attr={'rankdir': 'LR'})

    if nodes is None:
        labels = set(edges.iloc[:, 0]).union(set(edges.ilo
        nodes = pd.DataFrame({'name': list(labels), 'label

    node_data = nodes.to_dict('index')
    for _, attrs in node_data.items():
        if remove_labels:
            attrs['label'] = ''
        g.node(**attrs)

    g.edges([(from_node, to_node) for _, (from_node, to_no
```