Golang

September 10, 2018

```
In [1]: display.HTML(`<h1 style="color:green;">Hello, World</h1>`)
```

```
Out[1]: <h1 style="color:green;">Hello, World</h1>
```

```
In [2]: {
            Display(display.Markdown("* hello from markdown"))
            Display(display.Math(`e^{i\Pi}+1=0`))
        }
```

- hello from markdown

$$e^{i\Pi} + 1 = 0$$

```
In [3]: json := make(map[string]interface{})
        json["id"] = 1
        json["name"] = "gucci"
        json["action"] = "perfect"
```

```
In [4]: display.JSON(json)
```

```
Out[4]: map[id:1 name:gucci action:perfect]
```

```
In [5]: import (
            "net/http"
            "io/ioutil"
        )
        resp, err := http.Get("https://github.com/gopherdata/gophernotes/raw/master/files/gopher
        bytes, err := ioutil.ReadAll(resp.Body)
        resp.Body.Close()
        display.PNG(bytes)
```

```
Out[5]:
```

In [6]:
```
////  downloaddownloa  and display an SVG
resp, err := http.Get("http://jupyter.org/assets/nav_logo.svg")
bytes, err := ioutil.ReadAll(resp.Body)
resp.Body.Close()
display.SVG(string(bytes))
```

Out[6]:

In [7]:
```
// download and display a JPEG
resp, err := http.Get("https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/Gopherc
bytes, err := ioutil.ReadAll(resp.Body)
resp.Body.Close()
display.JPEG(bytes)
```

Out[7]:

#
GRAPH
EX-
AM-
PLE
IN
GOLANG

In [8]:
```
// Downloading graph
import ("os/exec"
        "log"
       )
get := exec.Command("go", "get", "github.com/twmb/algoimpl/go/graph")
_, e := get.CombinedOutput()
if e != nil {
    log.Print(e.Error())
```

```
        }
```

In [9]:
```
// imports
import (
    "fmt"
    "github.com/twmb/algoimpl/go/graph"
)
```

In [10]:
```
g := graph.New(graph.Undirected)
clothes := make(map[string]graph.Node)
// Make a mapping from strings to a node
clothes["shirt"] = g.MakeNode()
clothes["tie"] = g.MakeNode()
clothes["jacket"] = g.MakeNode()
clothes["belt"] = g.MakeNode()
clothes["watch"] = g.MakeNode()
clothes["undershorts"] = g.MakeNode()
clothes["pants"] = g.MakeNode()
clothes["shoes"] = g.MakeNode()
clothes["socks"] = g.MakeNode()
```

In [11]:
```
// Make references back to the string values
for key, node := range clothes {
    *node.Value = key
    fmt.Println(*node.Value)
}
```

```
jacket
belt
watch
shirt
tie
undershorts
pants
shoes
socks
```

In [12]:
```
g.MakeEdge(clothes["shirt"], clothes["tie"])
g.MakeEdge(clothes["tie"], clothes["jacket"])
g.MakeEdge(clothes["shirt"], clothes["belt"])
g.MakeEdge(clothes["belt"], clothes["jacket"])
g.MakeEdge(clothes["undershorts"], clothes["pants"])
g.MakeEdge(clothes["undershorts"], clothes["shoes"])
g.MakeEdge(clothes["pants"], clothes["belt"])
g.MakeEdge(clothes["pants"], clothes["shoes"])
g.MakeEdge(clothes["socks"], clothes["shoes"])
```

In [14]:
```
sorted := g.TopologicalSort()
for i := range sorted {
```

```
    fmt.Println(*sorted[i].Value)
}
```