10.Visualizacao.Rica

October 27, 2019

1 Visualização Rica

Este notebook apresenta os seguintes tópicos

1.1 Visualizacoes ricas

Como dito anteriormente, o Jupyter permite misturar texto formatado com código e com visualizações.

As visualizações até o momento foram feitas usando as bibliotecas `matplotlib` e `pandas` (que também utiliza a `matplotlib`), mas podemos chegar em situações em que queremos criar nossas próprias visualizações para objetos próprios.

Para fazer isso, podemos definir os métodos *repr**_(self) em classes, on * pode ser algum dos seguintes formatos suportados pelo Jupyter:

- svg
- png
- jpeg
- html
- javascript
- latex

Por exemplo, podemos definir uma classe `SQRT` que represente uma raiz quadrada de um número usando Latex.

[1]:
```
class SQRT:
    def __init__(self, number):
        self.number = number

    def formula(self):
        text = self.number
        if hasattr(self.number, "formula"):
            text = self.number.formula()
        return "\sqrt{%s}" % (text,)
```

```
    def _repr_latex_(self):
        return "$${}$$".format(self.formula())

SQRT(25)
```

[1]:

$$\sqrt{25}$$

Essa classe pode ser usada em conjunto com outra.

[2]:
```
SQRT(SQRT(25))
```

[2]:

$$\sqrt{\sqrt{25}}$$

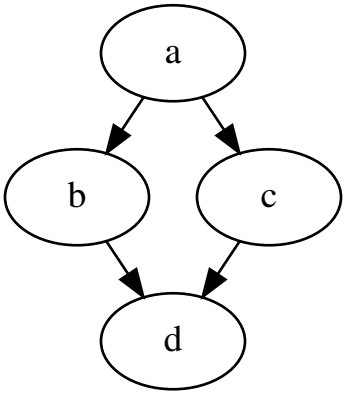
Também podemos chamar programas externos para construir imagens. A seguir usaremos GraphViz (dot) para construir tanto uma imagem SVG quanto uma imagem PNG.

[3]:
```
import os
from subprocess import Popen, PIPE as P

class Graph:
    def __init__(self, definition):
        self.definition = definition

    def dottext(self):
        result = [
            "  {} -> {};".format(node, other)
            for node, edges in self.definition.items()
            for other in edges
        ]
        return "digraph G {{\n  ranksep=0.25;\n{}\n}}".format("\n".join(result))

    def dot(self, format="png"):  # ToDo: Tratar erro
        kwargs = {} if os.name != 'nt' else {"creationflags": 0x08000000}
        p = Popen(['dot', '-T', format], stdout=P, stdin=P, stderr=P, **kwargs)
        return p.communicate(self.dottext().encode('utf-8'))[0]

    def __repr__(self):
        return self.dottext()

    def _repr_svg_(self):
        return self.dot("svg").decode("utf-8")

    def _repr_png_(self):
        return self.dot("png")
```

```
graph = Graph({"a": ["b", "c"], "b": ["d"], "c": ["d"]})
```

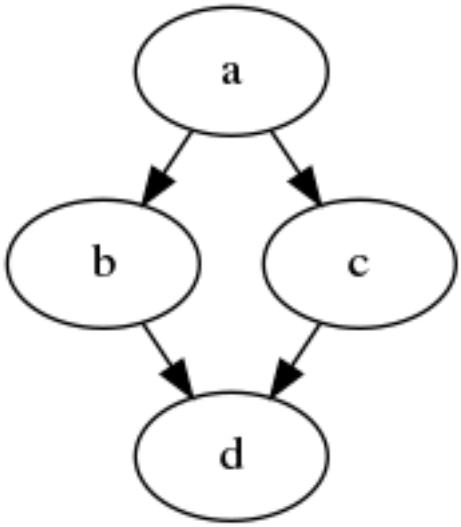
[4]:
```
graph
```

[4]:



Por padrão, objetos que oferecem tanto visualização png quanto svg são visualizados como SVG no Jupyter. Porém, podemos forçar que sejam visualizados como PNG.

[5]:
```
from IPython.display import display_png
display_png(graph)
```



Ou podemos forçar o uso do __repr__ do Python.

[6]:
```
from IPython.display import display_pretty
display_pretty(graph)
```

```
digraph G {
  ranksep=0.25;
  a -> b;
  a -> c;
  b -> d;
  c -> d;
}
```

Aproximando do assunto da apresentação, podemos usar esse grafo para exibir a AST.

[7]:
```
import ast
from collections import defaultdict

class GraphVisitor(ast.NodeVisitor):

    def __init__(self):
        self.parent = []
        self.graph = defaultdict(list)
        self.id = 0

    def generic_visit(self, node):
        old_parent = self.parent
        name = type(node).__name__ + str(self.id)
        self.id += 1
        if old_parent:
            self.graph[old_parent].append(name)
        self.parent = name
        super().generic_visit(node)
        self.parent = old_parent
```

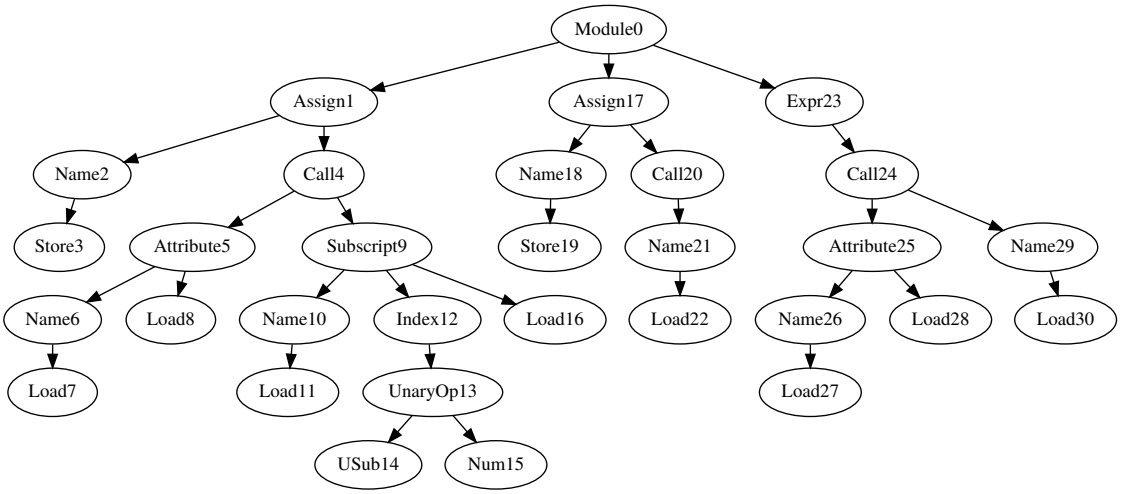
Executando para a própria célula.

[8]:
```
tree = ast.parse(In[-1])
visitor = GraphVisitor()
visitor.visit(tree)
```

Visualização da AST.

[9]:
```
Graph(visitor.graph)
```

[9]:



1.2 Exercicio 12

Modifique a classe GraphVisitor para exibir valores nos nós do tipo Name e Num.

[ ]:
```
...
```

[ ]:
```
tree = ast.parse(In[-1])
visitor = GraphVisitor()
visitor.visit(tree)
```

[ ]:
```
Graph(visitor.graph)
```

1.3 Exercicio 13

Implemente uma cell magic para visualizar a AST.

[ ]:
```
import ast
from IPython.core.magic import Magics, magics_class, cell_magic

@magics_class
class ASTMagic(Magics):
    @cell_magic
    def view_ast(self, line, cell):
        ...

shell = get_ipython()
shell.register_magics(ASTMagic)
```

[ ]:
```
%%view_ast
a = 1 + 2
b = a + 3
```

Continua: 11.Widgets.pdf