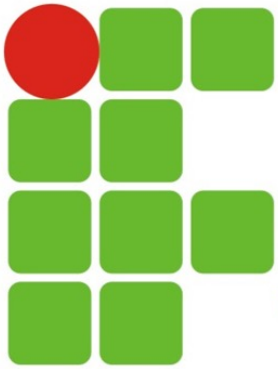

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação Orientada a Objetos

APRESENTAÇÃO DA DISCIPLINA

PROFESSOR:
Demetrios Coutinho
Demetrios.coutinho@ifrn.edu.br

AGENDA

- Módulos
- Biblioteca de terceiros
- Docstrings
- Sobrecarga de Operadores

Módulos

- Agora que sabemos como criar classes e instanciar objetos precisamos saber como organizá-los;
  - Para programas pequenos podemos colocar todas as classes em um único arquivo e apenas adicionar um script ao final do arquivo para fazer os objetos interegirem;
  - No entanto, quando o projeto começa a crescer pode ficar difícil achar classes que precisam ser modificadas entre tantas classes definidas em um único lugar;
  - Os módulos são simples arquivos python (*.py) onde essas classes podem ser organizadas;
    - Ex.: dois arquivos python são dois módulos ;-)
    - Se tivermos dois módulos no mesmo diretório podemos carregar classes, funções e métodos de um módulo para outro facilmente;

Módulos

- Para ilustrar, vamos hipoteticamente implementar um sistema de eccomerce;
- Devemos armazenar muitos dados no database;
  - Assim sendo, podemos colocar todas as classes e funções relacionadas à base de dados dentro do módulo *database.py*;
- Assuma que existe uma classe chamada *Database* dentro do módulo *database.py*;
- Assuma ainda que existe um módulo chamado *products.py* reponsável por fazer consultas relacionadas aos produtos;
- Existem algumas variações de sintaxe para importarmos a classe Database dentro do *products.py*:

In [ ]:

```
1  import database
2  db = database.Database()
3  # Do queries on db
```

Módulos

```
1  import database
2  db = database.Database()
3  # Do queries on db
```

- Na versão acima importamos o módulo database para o *namespace* do *products.py*
  - Namespace é uma lista de identificadores acessíveis para um módulo ou função;
- Assim, podemos acessar qualquer classe ou função de database usando a notação

database.< something >

- Alternativamente, podemos importar coisas específicas usando a notação from...import:

In [ ]:

```
1  from database import Database
2  db = Database()
3  # Do queries on db
```

Módulos

- Se, por algum motivo, o módulo *products* já possuir uma classe chamada *Database* e não quisermos que haja conflito de nomes, então podemos colocar um apelido no *database.Database*:

In [ ]:

```
1  from database import Database as DB
2  db = DB()
3  # Do queries on db
```

- Podemos ainda importar vários itens em uma única instrução:

In [ ]:

```
1  from database import Database, Query #suponha que existe Query no módulo data
```

Módulos

- Apesar de não recomendado, podemos também importar todas as classes e funções de um módulo usando o caractere *****

In [ ]:

```
1  from database import *
```

- O importe de tudo não é recomendado, pois:
  - Prejudica a legibilidade do código;
  - Pode gerar problemas evitáveis no namespace (como conflito de identificadores), em função de objetos indesejáveis no namespace;

Bibliotecas de Terceiros

- O Python vem com uma adorável biblioteca padrão, que é uma coleção de pacotes e módulos que estão disponíveis em todas as máquinas que executam o Python.
- Entretanto, às vezes precisamos de bibliotecas de terceiros;
  - Para procurar bibliotecas de terceiros use o Python Package Index (**PyPi**) (http://pypi.python.org/)
  - Uma vez que você identificou a biblioteca que você quer usar, basta usar o *pip* para instalá-la;

Docstrings

- Apesar do python ser uma linguagem de fácil interpretação, precisamos documentar nossos códigos;
  - Principalmente quando estamos trabalhando em equipe ou fazendo projetos que podem ser escalados;
  - Além disso, em orientação a objetos é importante escrever a documentação das APIs da forma mais clara e concisa possível, explicando o que cada objeto, atributo e método faz;
- Através do uso de docstrings podemos adicionar a documentação dentro do próprio código;
  - Para tanto, colocamos a documentação em aspas simples ou dupla para documentações de linhas únicas ou as aspas simples ou duplas três vezes para textos que ficam em mais de uma linha;
  - A documentação deve seguir a identação da classe, método, etc.
- Para ilustrar o uso do docstring, vamos documentar nossa classe Point:

https://www.python.org/dev/peps/pep-0257/

Sobrecarga de Operadores

- **__str__** é um método especial, como **__init__,** usado para retornar uma representação de string de um objeto;
- **Recomendação**: Codifique classes escrevendo **__init__**, o que facilita a instanciação de objetos, e **__str__,** que é útil para a depuração;
- Alterar o comportamento de um operador para que funcione com tipos definidos pelo programador chama-se **sobrecarga de operadores.**

Sobrecarga de Operadores

- Ao definir outros métodos especiais, você pode especificar o comportamento de operadores nos tipos definidos pelo programador;
- Por exemplo, se você definir um método chamado **__add__** para a classe Point;
- Há vários outros métodos especiais:
  - http://docs.python.org/3/reference/datamodel.html#specialnames.

Exercício

DÚVIDAS?