Algoritmos e Estrutura de Dados

Aula 2

Introdução a Linguagem Python
(Parte II)

Prof. Tiago A. E. Ferreira

Primeiro Contado com Classes

- A definição de prática de uma classe é

```
class NomeDaClasse:
    <comando-1>
    .
    .
    .
    <comando-N>
```

  - Na prática os comandos dentro de uma classe serão funções!
    - Quando se cria uma classe um novo espaço de nomes é criado, sendo todos as atribuições de variáveis da classe atribuídas a este escopo local.
    - Ao término do processo de definição da Classe (sem erros), um objeto de classe é criado.

Objetos de Classe

- Suportam dois tipos de operações:
  - Referência a atributos; e
  - Instanciação.
- Referência a Atributos:
  - Sintaxe: ***objeto.atributo***
    - Atributos válidos são todos os nomes presente o espaço de nomes do objeto. Ex.:

```
class MyClass:
    "Um exemplo simples de classe"
    i = 12345
    def f(x):
        return 'hello world'
```

    - ***MyClass.i*** e ***MyClass.f*** são referências válidas.
    - O atributo **__doc__** também é válido, retornando o *docstring*: "Um exemplo simples de classe"

Objetos de Classe

- Instanciação de Classe

```
x = MyClass()
```

  - Cria uma nova instância da Classe, atribuindo o novo objeto a variável *X*
    - De forma padrão, esta operação cria um objeto vazio (NULL)
    - Contudo é possível criar um objeto em um estado já pré-definido. Para tanto, existe um método especial **__init__()**

```
def __init__(self):
    self.data = []
```

    - Quando uma classe é instanciada, automaticamente o método **__init__()** é invocado!

Objetos de Classe

- Assim, é possível modificar o método **__init__()** para determinar como um objeto deve ser inicializado.
  - Exemplo:

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0,-4.5)
>>> x.r, x.i
(3.0, -4.5)
```

Instâncias

- As únicas operações reconhecidas por instâncias são as referências aos atributos
  - Existem dois atributos:
    - Atributos de dados (ou simplesmente atributos)
    - Atributos de referência (ou métodos)
- Atributos de Dados:
  - Estes não precisam ser declarados.
    - Exemplo: seja ***x*** uma instância da classe ***MyClass***

```
x.counter = 1
while x.counter < 10:
    x.counter = x.counter * 2
print x.counter
del x.counter
```

Instâncias

- Métodos
  - Os métodos são as funções que pertencem as instâncias das Classes
  - Seja ***x*** uma instância da classe ***MyClass***, então,
    - x.f(arg) estará invocando o método f.
  - Por convenção, o primeiro argumento de qualquer método é frequentement ***self***
  - Métodos podem invocar outros métodos através do argumento ***self***:

```
class Bag:
    def __init__(self):
        self.data = []
    def add(self, x):
        self.data.append(x)
    def addtwice(self, x):
        self.add(x)
        self.add(x)
```

Herança

- Uma Classe pode herdar os atributos de uma outra Classe através do processo de Herança!
  - A sintaxe de uma classe derivada é:

```
class DerivedClassName(BaseClassName):
    <statement-1>
    .
    .
    .
    <statement-N>
```

  - Se um atributo invocado não for da classe derivada, este será procurado na classe base
    - Tal processo é recursivo!

Herança Múltipla

- Em Python é possível ter heranças múltiplas:

```
class DerivedClassName(Base1, Base2, Base3):
    <statement-1>
    .
    .
    .
    <statement-N>
```

- Contudo, é sabido que a herança múltipla pode vir a gerar muita “dor de cabeça”
  - Além do que em teoria os problemas podem ser resolvidos por herança simples!

Variáveis Privadas

- Qualquer identificador do tipo ***__nome__*** ou ***__nome_*** é substituido por
  - ***_className_nome***, onde *className* é o nome da classe corrente.
  - Esta construção pode ser utilizada para tornar **privada:**
    - **Instâncias;**
    - **Variáveis da Classe; e**
    - **Métodos**
  - Fora de uma Classe não se aplica tal procedimento
  - Haverá truncamento no identificador se ***_className_nome*** tiver mais de 255 caracteres

Alguns Atributos Especiais

- Além do construtor de uma Classe:
  - O método ***__init__(self)***
  - É possível citar:

Atributo	Descrição
__bases__	Lista de classes bases que a classe atual herda. Caso a classe atual não herde de nenhuma outra classe, está lista será vazia
__dict__	Dicionário com o espaço de nomes da classe. Cada para de valor chave representa um identificador e o seu valor no espaço de nomes
__doc__	É o *docstring* da classe. Se a classe não especificar nenhum *docstring*, seu valore será **None**
__module__	Uma string que contem o nome do módulo (file) no qual a classe está definida
__name__	Um string que contem o nome da classe
__class__	Uma referência para a classe da qual o objeto foi instanciado
__dict__	Um dicionário que corresponde ao espaço de nomes do objeto. Cada par de valores chaves representa um identificador e seu valor no espaço de nomes.
__str__	Retorna uma string invocada por ***print objeto***

Exercício

- Crie as classes:
  - Cliente:
    - Nome
    - CPF
    - Endereço
    - Telefone
  - Conta Corrente:
    - Cliente
    - Agencia
    - Saldo

Exercícios

- Crie as classes:
  - Cliente:
    - Nome
    - CPF
    - Endereço
    - Telefone
  - Conta Corrente:
    - Número
    - Agencia
    - Saldo
    - Saque
    - Depósito
  - Banco
    - Cliente
    - Conta Conta-Corrente
- Com estas três classes crie um programa que cadastre clientes no banco e estes possam movimentar as contas
  - Quando ***print OBJ-Banco*** gerar um relatório com todas as contas!