SCC 124 - Introdução à Programação para Engenharias

Classes e Objetos

python

Professor: André C. P. L. F. de Carvalho, ICMC-USP
Pos-doutorando: Isvani Frias-Blanco
Monitor: Henrique Bonini de Britto Menezes

Aula de hoje

- Introdução
- Linguagens imperativas
- Linguagens procedurais
- Linguagens orientadas a objetos
- Classes e objetos

Introdução

- Existem mais de 500 de linguagens de programação
- Paradigmas de programação
  - Classifica as linguagens de programação de acordo com o estilo de programação
    - Imperativas
      - Computação ocorre por mudança de estados (Ex.: variáveis)
    - Declarativas
      - Programa descreve o que faz, ao invés de como faz
      - Ex.: SQL

Linguagens imperativas

- Possuem os seguintes componentes:
  - Variáveis
  - Atribuições
  - Operações de entrada e de saída
  - Comandos de controle de fluxo
- Programa pode ser visto como um grande comando

Linguagens imperativas

- Podem ser:
  - Procedurais
  - Funcionais
  - Concorrentes
  - Lógicas
  - Orientada a objetos
  - ...
  - Multiparadigma

Linguagens procedurais

- A maioria das primeiras linguagens de programação é procedural
  - Programação orientada a procedimentos
  - Programa é decomposto em passos
  - Exemplos:
    - Fortran
    - Pascal
    - C

Linguagens procedurais

- Problema das linguagens procedurais
  - Variáveis globais podem ser acessadas de qualquer parte de um programa
    - Grandes programas sem disciplina para acesso a variáveis globais tendem a ser impraticáveis
    - Módulo que acessa uma variável global não pode ser desenvolvido e entendido de forma independente de outros módulos que também acessam essa variável

Linguagens procedurais

- "O remédio é esconder informação" (David Parnas, 1970)
  - Encapsular cada variável em um objeto
    - Junto com um grupo de operações que sozinhas têm acesso direto à variável
  - Outros objetos podem acessar a variável apenas indiretamente
    - Utilizando grupos de operações

Encapsulamento

- Junta dados e as operações sobre os dados em uma única unidade: classe
  - Dados e funções são combinados
    - "Isso é como um objeto X é representado e essas são as únicas operações que podem ser aplicadas a X"
  - Programação orientação a objetos

Orientação a objetos

- Até agora todos os programas vistos no curso foram baseados em funções
  - Blocos de comandos que manipulam dados
  - Programação orientada-a-procedimento
- Existe outra forma de organizar programas
  - Combinar estruturas de dados e funções e colocar dentro de objetos
  - É o que faz a programação orientada a objetos

Programação orientada a objetos

- Python é multiparadigma
  - Procedural, orientada a objetos e funcional
- Programação orientada a objeto é usada para:
  - Reduzir efeitos colaterias
  - Tornar programas mais claros
  - Escrever programas de grande porte

Programação orientada a objetos

- Principais aspectos de programação orientada a objeto (POO):
  - Classe
    - Criar uma classe significa criar um novo tipo
  - Objeto
    - É uma instância (variável) de uma classe
      - Ex.: variáveis do tipo *int* são instâncias da classe *int*

Programação orientada a objetos

- Toda variável é um objeto
  - Por isso pertence a uma classe de objetos
  - Classe define para os objetos da classe:
    - O formato dos dados e
    - As operações permitidas

Programação orientada a objetos

- Objetos podem armazenar dados
  - Usando variáveis que pertencem ao objeto ou sua classe
    - Variáveis que pertencem a um objeto ou sua classe são chamadas de campos (*fields*)
- Objetos podem usar funções que pertencem à sua classe
  - São conhecidas como métodos (*methods*)

Programação orientada a objetos

- Uso da terminologia anterior
  - Permite diferenciar métodos e funções que são:
    - Funções: independentes de classe ou objeto
    - Métodos: pertencem a uma classes ou objeto
- Campos e métodos de uma classe são chamados de atributos da classe

Programação orientada a objetos

- Existem dois tipos de campos
  - Campos que pertencem à uma classe
    - Variáveis da classe (*class variables*)
  - Campos que pertencem a cada objeto da classe
    - Variáveis do objeto (*object variables*)
- Uma classe é criada usando a palavra chave *class*
  - Campos e métodos são escritos em um bloco indentado

Variável *self*

- Diferença entre métodos e funções comuns:
  - Definição de um método inclui um nome (variável) extra no início da lista de parâmetros
    - Por convenção, chamada de *self*
      - Programador pode dar outro nome
- Variável *self*
  - Não recebe valor na chamada do método
    - Interpretador Python se encarrega de dar um valor
  - Se refere ao próprio objeto

Exemplo 1

```
class Pessoa:
    pass  # Bloco de comandos vazio

p = Pessoa()
print(p)
```

Instância da classe Pessoa no módulo *__main__* Armazenado no endereço de memória listado

Saída:

<__main__.Pessoa object at 0x00E459D0>

Exemplo 2

```
class Pessoa:
    def diga_ola (self):
        print('Ola, como vai?')
p = Pessoa()
p.diga_ola()
# As 2 últimas linhas pode ser
# trocadas por Pessoa().diga_ola()
```

Saída:

Ola, como vai?

Métodos

- Alguns nomes de métodos possuem um significado em Python
  - Método *__init__*
    - Executado assim que um objeto é instanciado para uma classe
    - Útil para inicializar um objeto

Exemplo

```
class Pessoa:
    def __init__(self, nome):
        self.name = nome

    def diga_ola(self):
        print('Ola, meu nome é', self.name)

p = Pessoa('André')
p.diga_ola()
# As 2 últimas linhas pode ser escritas
# como Pessoa('André').diga_ola()
```

Saída:

Ola, meu nome é André

Comparação de exemplos

```
class Pessoa:
    def diga_ola (self):
        print('Ola, como vai?')
p = Pessoa()
p.diga_ola()
# As 2 últimas linhas pode ser
# trocadas por Pessoa().diga_ola()
```

```
class Pessoa:
    def __init__(self, nome):
        self.name = nome

    def diga_ola(self):
        print('Ola, meu nome é', self.name)

p = Pessoa('André')
p.diga_ola()
# As 2 últimas linhas pode ser escritas
# como Pessoa('André').diga_ola()
```


Variáveis

- Campos são variáveis comuns, atreladas ao espaço de nomes de uma classe ou objeto
  - Válidos apenas no contexto da classe e do objeto
  - Variáveis classe
    - Compartilhadas pela classe e seus objetos
    - Uma única cópia é armazenada na memória
  - Variáveis objeto
    - Válidas apenas no objeto
    - Uma cópia (espaço de memória) para cada objeto

Exemplo

Variável classe

Variável objeto

```
class NumComplexos:
    def __init__(self, parteReal, parteImag):
        self.r = parteReal
        self.i = parteImag

x = NumComplexos(2.0, -4.5)
x.inc = 1
while (x.inc < 5):
    x.r = x.r + 1
    x.i = x.i + 1
    print("Valores para", x.inc, "-esimo x são: ", x.r, x.i)
    x.inc = x.inc + 1
```

Exemplo

Variável classe

Variável objeto

```
class NumComplexos:
    def __init__(self, parteReal, parteImag):
        self.r = parteReal
        self.i = parteImag

x = NumComplexos(2.0, -4.5)
x.inc = 1
while (x.inc < 5):
    x.r = x.r + 1
    x.i = x.i + 1
    print("Valores para", x.inc, "-esimo x são: ", x.r, x.i)
    x.inc = x.inc + 1
```

Saída:

```
Valores para 1 -esimo x são:  3.0 -3.5
Valores para 2 -esimo x são:  4.0 -2.5
Valores para 3 -esimo x são:  5.0 -1.5
Valores para 4 -esimo x são:  6.0 -0.5
```


Conclusão

- Paradigmas de linguagens de programação
- Linguagens imperativas
- Linguagens procedurais
- Linguagens orientadas a objetos
- Classes e objetos

Perguntas