F<BR/C> DE SOFTWARE

Classes em Python

PROFESSOR: Maurício de Souza Estevam

1 - Introdução às classes

1.1 O que são classes e por que são importantes em Python:

As classes são uma estrutura fundamental na programação orientada a objetos. Elas permitem que os desenvolvedores criem modelos que representam objetos reais ou abstratos. As classes são importantes em Python porque fornecem uma maneira de organizar e estruturar código, tornando-o mais legível, modular e reutilizável.

1.2 Criando uma classe simples em Python:

Para criar uma classe em Python, use a palavra-chave "class" seguida pelo nome da classe. Dentro da classe, você pode definir atributos e métodos que descrevem o comportamento do objeto representado pela classe. Por exemplo, você pode criar uma classe "Pessoa" com atributos como nome, idade e altura, e métodos para obter e definir esses atributos.

1.3 Atributos e métodos de classe:

Atributos são variáveis que descrevem o estado de um objeto, enquanto métodos são funções que definem o comportamento do objeto. Você pode definir atributos e métodos dentro de uma classe para criar objetos que possuem essas características. Por exemplo, você pode definir um atributo "idade" em uma classe Pessoa e um método "envelhecer" que aumenta a idade em um ano.

1.4 Instanciando objetos a partir de uma classe:

Para criar um objeto a partir de uma classe em Python, você usa a sintaxe "nome_da_classe()" para chamar o construtor da classe. Isso cria uma instância da classe, que é um objeto que possui os atributos e métodos definidos na classe. Por exemplo, você pode criar uma instância da classe Pessoa chamada "pessoa1" e definir seu nome como "Ana" e sua idade como 25.

1.5 Herança de classes e polimorfismo:

Herança de classes é uma técnica em que uma nova classe é criada a partir de uma classe existente, herdando seus atributos e métodos. Isso permite que você crie classes mais especializadas que possuem o comportamento da classe pai. Por exemplo, você pode criar uma classe "Estudante" que herda da classe "Pessoa" e adiciona um atributo "curso". Polimorfismo é outra técnica que permite que objetos de diferentes classes sejam tratados da mesma maneira. Isso é útil quando você deseja executar um comportamento em diferentes tipos de objetos, como quando você quer imprimir uma mensagem de saudação para uma pessoa ou para um estudante.

2: Atributos e métodos

2.1 Atributos de instância e de classe:

Os atributos de instância são variáveis associadas a um objeto específico de uma classe. Eles podem ser acessados e modificados usando a notação ponto, após a criação da instância. Os atributos de classe são variáveis que são compartilhadas entre todas as instâncias da classe. Eles podem ser acessados usando a notação ponto, mas sua modificação afeta todas as instâncias da classe. Por exemplo, você pode ter uma classe "Círculo" com um atributo de instância "raio" e um atributo de classe "pi".

2.2 Métodos de instância e de classe:

Os métodos de instância são funções que são definidas dentro da classe e que podem ser executadas em um objeto específico dessa classe. Eles têm acesso aos atributos de instância do objeto e podem modificá-los. Os métodos de classe são funções que são definidas dentro da classe, mas que não têm acesso aos atributos de instância. Eles são executados em toda a classe e podem ser úteis para realizar operações que não dependem de nenhum objeto específico. Por exemplo, você pode ter um método de instância "calcular_circunferencia" em uma classe "Círculo" e um método de classe "calcular_area" na mesma classe.

2.3 Propriedades em Python:

As propriedades em Python são um recurso que permite controlar o acesso e a modificação de atributos de instância de uma classe. Elas fornecem uma maneira de definir um método de leitura e um método de escrita para um atributo, para que você possa garantir que ele seja sempre lido e modificado da maneira desejada. Por exemplo, você pode definir uma propriedade "idade" em uma classe "Pessoa" para garantir que a idade seja sempre um número positivo.

2.4 Métodos especiais em Python:

Os métodos especiais em Python são métodos que começam e terminam com dois caracteres de sublinhado, como "init" ou "str". Eles fornecem um comportamento especial para objetos de uma classe. Por exemplo, o método "init" é usado como um construtor de uma classe, e é chamado quando um objeto é criado a partir da classe. O método "str" é usado para retornar uma representação em formato de string de um objeto. Você pode personalizar esses métodos para adaptar o comportamento de sua classe às suas necessidades.

3: Herança e polimorfismo

3.1 Herança simples e múltipla em Python:

A herança em Python permite que uma classe herde atributos e métodos de outra classe. A herança simples envolve uma classe derivada que herda de uma única classe base. Já a herança múltipla envolve uma classe derivada que herda de duas ou mais classes base. Por exemplo, você pode ter uma classe "Animal" como classe base, e as classes derivadas "Cachorro" e "Gato" que herdam de "Animal".

3.2 Sobrescrita de métodos em Python:

A sobrescrita de métodos em Python ocorre quando uma classe derivada redefine um método que já foi definido em sua classe base. Isso permite que a classe derivada personalize ou altere o comportamento do método. Por exemplo, a classe "Gato" pode sobrescrever o método "emitir_som" da classe "Animal" para retornar o som "miau".

3.3 Herança e encapsulamento em Python:

O encapsulamento em Python é um conceito que se refere à ocultação de detalhes de implementação dentro de uma classe. Isso é feito para proteger os atributos e métodos de uma classe de acessos não autorizados. Quando uma classe é derivada de uma classe base, ela herda seus atributos e métodos, incluindo aqueles que estão encapsulados. Isso significa que a classe derivada pode acessá-los, mas não pode modificá-los diretamente. Por exemplo, a classe "Cachorro" pode herdar o atributo encapsulado "idade" da classe base "Animal", mas não pode modificá-lo diretamente.

3.4 Polimorfismo em Python:

O polimorfismo em Python é a capacidade de um objeto de uma classe derivada ser tratado como um objeto de sua classe base. Isso significa que um método que é definido na classe base pode ser chamado em um objeto de uma classe derivada, desde que a classe derivada tenha implementado o método. Isso permite que o mesmo método seja chamado em diferentes objetos de diferentes classes, produzindo comportamentos diferentes, dependendo da classe do objeto em questão. Por exemplo, um método "fazer_som" definido na classe "Animal" pode ser chamado em um objeto "Cachorro" ou "Gato", produzindo sons diferentes, dependendo da classe do objeto.

Aula 4: Módulos, pacotes e tratamento de exceções

4.1 Criando e importando módulos em Python:

Um módulo em Python é um arquivo que contém definições de funções, classes e variáveis. Criar um módulo em Python envolve simplesmente escrever o código desejado em um arquivo com extensão .py. Para importar um módulo em Python, basta usar a palavra-chave "import" seguida do nome do módulo. Por exemplo, se você tiver um módulo chamado "meumodulo.py", pode importá-lo em outro arquivo Python usando "import meumodulo".

4.2 Criando e usando pacotes em Python:

Um pacote em Python é um diretório que contém um ou mais módulos relacionados. Para criar um pacote, basta criar um diretório com um arquivo "init.py". Esse arquivo de inicialização será executado quando o pacote for importado e pode ser usado para definir variáveis e funções para o pacote. Para usar um pacote em Python, basta importá-lo como faria com um módulo. Por exemplo, se você tiver um pacote chamado "meupacote" que contenha um módulo "meumodulo.py", pode importar o módulo usando "import meupacote.meumodulo".

4.3 Tratamento de exceções em Python:

O tratamento de exceções em Python é uma técnica usada para lidar com erros durante a execução de um programa. Quando uma exceção é gerada, o programa pode ser interrompido se não houver um tratamento adequado. Para tratar exceções em Python, você pode usar blocos "try" e "except". O bloco "try" contém o código que pode gerar uma exceção, e o bloco "except" contém o código que será executado se uma exceção for gerada. Por exemplo, se você estiver tentando dividir um número por zero, pode usar um bloco "try" para tentar realizar a operação de divisão e um bloco "except" para lidar com a exceção caso ocorra.

4.4 Exemplos de aplicação de classes em Python:

As classes em Python podem ser usadas em uma variedade de aplicações, desde modelagem de objetos do mundo real até criação de jogos e aplicativos. Algumas aplicações comuns de classes em Python incluem criar objetos que representam pessoas, animais, veículos, itens de jogo e muito mais. As classes também podem ser usadas para criar aplicativos que envolvem gráficos, banco de dados e outras funcionalidades avançadas. Por exemplo, um jogo de aventura em texto pode usar classes para representar o jogador, o ambiente do jogo e os itens que o jogador pode encontrar.

Exercício 1

Fazer um programa para ler os dados de duas pessoas, depois mostrar o nome da pessoa mais velha.

Exemplo:

Dados da primeira pessoa:

Nome: Maria

Idade: 17

Dados da segunda pessoa:

Nome: Joao

Idade: 16

Pessoa mais velha: Maria

Exercício 2

Fazer um programa para ler nome e salário de dois funcionários. Depois, mostrar o salário médio dos funcionários.

Exemplo:

Dados do primeiro funcionário:

Nome: Carlos Silva

Salário: 6300.00

Dados do segundo funcionário:

Nome: Ana Marques

Salário: 6700.00

Salário médio = 6500.00

Exercício 3

Fazer um programa para ler os valores da largura e altura de um retângulo. Em seguida, mostrar na tela o valor de sua área, perímetro e diagonal.

Exemplo

Entre a largura e altura do retângulo:

3.00

4.00

AREA = 12.00

PERÍMETRO = 14.00

DIAGONAL = 5.00

Exercício 4

Fazer um programa para ler os dados de um funcionário (nome, salário bruto e imposto). Em seguida, mostrar os dados do funcionário (nome e salário líquido). Em seguida, aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário.

Exemplo:

Nome: Joao Silva

Salário bruto: 6000.00

Imposto: 1000.00

Funcionário: Joao Silva, $ 5000.00

Digite a porcentagem para aumentar o salário: 10.0

Dados atualizados: Joao Silva, $ 5600.00

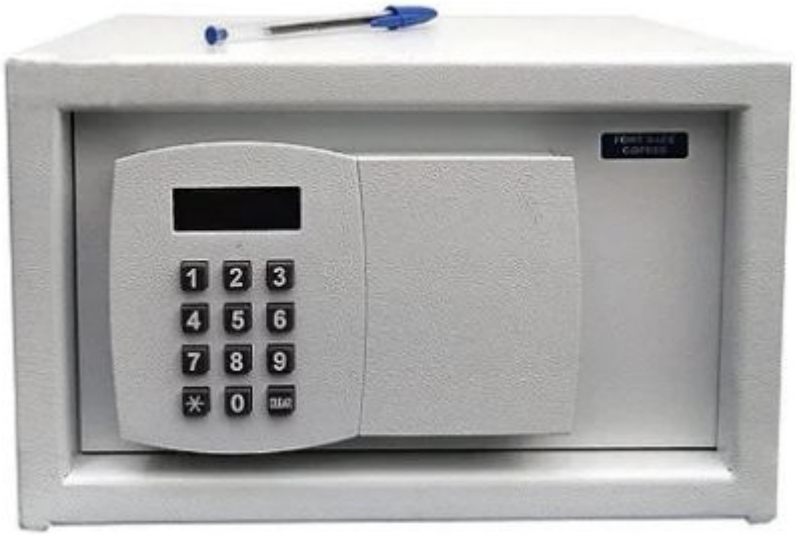
Exercício 5

Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos e dizer qual dos dois triângulos possui a maior área.

A fórmula para calcular a área de um triângulo a partir das medidas de seus lados a, b e c é a seguinte (fórmula de Heron):

$$area = \sqrt{p(p-a)(p-b)(p-c)} \qquad \text{onde} \qquad p = \frac{a+b+c}{2}$$

Jogo do Cofre



Neste exercício, você vai criar um jogo chamado Jogo do Cofre, onde o objetivo do jogador é acertar a senha de um cofre. A cada tentativa, o jogo informará quantos números o jogador acertou e quantos números o jogador errou.

Step - 1

Crie uma classe com um método que recebe uma tentativa de senha como parâmetro e retorna True se a tentativa estiver correta ou False caso contrário. A senha deve ser uma lista de 3 números inteiros aleatórios entre 1 e 5.

Step - 2

Modifique o método verificar_senha para que, caso a tentativa esteja incorreta, ele retorne uma mensagem informando quantos números o jogador acertou e quantos números o jogador errou. Para isso, percorra os números da tentativa e compare com os números da senha, verificando se eles estão na mesma posição. Se um número estiver na mesma posição, ele é um acerto. Caso contrário, é um erro. Armazene os acertos e erros em uma lista e imprima a mensagem informando quais números foram acertados e quais foram errados.

Step - 3

Crie uma classe chamada Jogo com um método jogar que recebe uma instância da classe Cofre como parâmetro e controla as tentativas do jogador. O jogador terá n tentativas para acertar a senha. A cada tentativa incorreta, o jogo informará quantas tentativas ainda restam.

Step - 4

Modifique o método jogar para que, caso a tentativa esteja incorreta, ele imprima a mensagem informando quantos números o jogador acertou e quantos números o jogador errou. Caso a tentativa esteja correta, ele deve imprimir uma mensagem informando que a senha foi correta e encerrar o jogo.

Step - 5

Crie uma instância da classe Jogo e chame o método jogar para iniciar o jogo.