Aula 01 - - POO em Python: Classes, Objetos, Métodos e Atributos ## Introdução ao tema A abordagem de Programação Orientada a Objetos (POO) foi inicialmente proposta no fim dos anos 1960. Levou quase 20 anos para que a tecnologia de objetos se tornasse amplamente usada. Durante os anos 1990, a engenharia de software POO tornou-se o paradigma de escolha para muitos construtores de produtos de software. ### O que é POO e por que ela é importante? Programação Orientada a Objetos é uma abordagem de desenvolvimento de software que organiza os problemas e suas soluções como um conjunto de **objetos** distintos. Um objeto **encapsula** tanto os dados quanto o processamento que é aplicado aos dados. O que leva ao **REUSO**, aumento de a produtividade, diminuição do custo de desenvolvimento e manunteção. ### Conceitos Fundamentais * **Objeto** * Qualquer coisa existente no mundo real, em formato concreto ou abstrato (que exista física ou conceitualmente). * Ex: aluno, mesa, professor, cadeira, caneta, conta bancária, tela, botão em uma tela etc. * Estrutura computacional que representa um objeto do mundo real. * **Classe** * Quando identificamos características e operações similares em objetos distintos, estamos realizando sua classificação, ou seja, identificando classes. * É uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. ## POO em Python O Python é uma linguagem de programação amplamente usada em aplicações da Web, desenvolvimento de software, ciência de dados e machine learning (ML). ([*aws - Amazon Web Services*](https://aws.amazon.com/pt/what-is/python/)) Python é uma linguagem de programação que permite trabalhar com vários paradigmas de programação como Orietação a Objeto, Progrmação Funcional, Programação Estruturada, etc. Como dito anteriormente Python é uma linguagem de programação que permite trabalhar com diferentes paradigmas, no entanto, **Python é uma linguagem de Programação Orientada a Objetos**, então tudo em Python **é tratado como um objeto**. ### Classes em Python Por definição as classes serão constituidas por Métodos e Atributos. * **Métodos** são as ações que podem ser executadas por um objeto da classe. * **Atributos** são as propriedades relacionadas ao objeto da classe. Objetos são **instâncias** de uma classe. Instâncias são o processo de materialização de uma classe em objeto, ou seja, reservar um espaço de memória para armazenamento da estrutra de uma classe. Um exemplo real de como classe e objetos seriam; **Classe**: Pessoa * **Atributos**: Nome, Sexo, Profissão * **Métodos**: Trabalhando, Estudando, Movimentando Com a definição da classe acima, podemos criar vários objetos com atributos e ações específicas. * **Objeto 1**: Reinaldo * **Atributos**: * Nome: Reinaldo C. Mendes * Sexo: Masculino * Profissão: Cientista de Dados * **Métodos**: * Trabalhando: Testando um algoritmo escrito em Python * Estudando: Ele estuda 3 horas durante a noite. * **Objeto 2**: Wagner * **Atributos**: * Nome: Wagner Coutinho de Moura Filho * Sexo: Masculino * Profissão: Desenvolvedor Backend * **Métodos**: * Trabalhando: Aguardando deploy de código em Python * Estudando: Ele estuda 2 horas durante todas as manhãs. Podemos observar que, Reinaldo é do sexo masculino e trabalha com Ciência de Dados. No momento ele está realizando um teste de código. Já Wagner que também é do sexo masculino, trabalha com Desenvolvimento BackEnd e no momento ele está aguardando o deploy do código. ***"Objetos são criados de uma mesma classe, no entanto, possuem comportamentos e estados diferentes".*** #### Criando uma Classe em Python Em Python, uma classe é definida usando a palavra-chave `class`. A sintaxe para criar uma classe é dada abaixo. **Sintaxe** ```Python class class_name: '''Isso é uma docstring. Eu criei uma nova classe''' . . ``` * **class_name**: É o nome da classe. * **Docstring**: É a primeira string dentro da classe e tem uma breve descrição da classe. Embora não seja obrigatório, é altamente recomendado. * **declaração**: Atributos e Métodos. **Exemplo Real** Neste exemplo, estamos criando uma classe Pessoa com variáveis de instância de nome, sexo, profissão e comportamento atual. ```Python class Person: def __init__(self, name, sex, profession): self.name = name self.sex = sex self.profession = profession def show(self): print("Nome: ", self.name, "Sexo: ", self.sex, "Profissão: ", self.profession) def work(self): print("Trabalha como ", self.profession) ``` #### Instânciando uma Classe e criando um Objeto em Python O objeto é criado usando o nome da classe. Objetos de classe suportam dois tipos de operações: * **Referência de Atributos**: Usam a sintaxe padrão usada para todas as referencias em Python. . * **Instânciação de Classe**: Cria uma nova *instância* de uma classe e atribui esse objeto à uma variável. Em Python a criação de um objeto é dívidida em duas partes: * **Criação de objetos**:

Internamente, o `__new__` é o método que cria o objeto. * **Inicialização de objetos**: O método `__init__()`, nos permite implementar o construtor para inicializar o objeto. **Sintaxe** ```Python = () ``` **Exemplo Real** ```Python reinaldo = Person('Reinaldo', 'Masculino', 'Cientista de Dados') ``` **Exemplo Completo** ```Python # Criando a classe pessoa class Person: def __init__(self, name, sex, profession): self.name = name self.sex = sex self.profession = profession def show(self): print("Nome: ", self.name, "Sexo: ", self.sex, "Profissão: ", self.profession) def work(self): print("Trabalha como ", self.profession) # Criando objetos da classe pessoa reinaldo = Person('Reinaldo Carlos Mendes', 'Masculino', 'Cientista de Dados') wagner = Person('Wagner Coutinho de Moura Filho', 'Masculino', 'Desenvolvedor Backend') # Chamadas dos métodos e atributos reinaldo.show() print(reinaldo.name) print("\n") print(wagner.name) wagner.work() print(wagner.profession) ``` **Saída** ``` Nome: Reinaldo Carlos Mendes Sexo: Masculino Profissão: Cientista de Dados Reinaldo Carlos Mendes Wagner Coutinho de Moura Filho Trabalha como Desenvolvedor Backend Desenvolvedor Backend ``` #### Declaração `pass` em Classes Pyhon No Python a palavra-chave `pass` é uma declaração nula. Ou seja, ao criar uma classe com a declaração `pass`, nada acontecerá quando executado. A declaração `pass` é usada para ter um ***bloco vazio em um código*** porque o ***código vazio não é permitido*** em ***loops***, ***definição de função*** e ***definição de classe***. Sendo assim, a declaração `pass` resultará em nenhuma operação (NOP). *Geralmente, nós a usamos como um espaço reservado quando não sabemos qual código escrever ou adicionar código em uma versão futura.* **Exemplo** ```Python class Protocolo: pass ``` #### Nomenclatura em Classes Python * Os nomes de classes têm a primeira letra de cada palavra maiúscula (CamelCase). * **Sintaxe**: `class NomeDeUmaClasse` * Exemplo: `class RelatorioSemestral` * Nomes de métodos (e funções) devem estar em letras minúsculas, com palavras separadas por underscores conforme seja útil para a legibilidade. * **Sintaxe**: `def nome_de_um_metodo` * **Exemplo**: `def exibir_salario_bruto(self)` * Obs: Usar ***self*** como primeiro parâmetro de um método. * Constantes são geralmente definidas em um nível de módulo e escritas em letras maiúsculas com underscores separando as palavras. * **Exemplos**: MAX_VALOR e TOTAL_SALARIO. * Nomes de variáveis e parâmetros de funções e métodos gralmente seguem a mesma regra dos métodos, devendo estar em letras minúsculas, com palavras separadas por underscores conforme seja útil para a legibilidade. * **Sintaxe**: nome_de_uma_variavel * **Exemplo**: salario_total_funcionario * A identação deve ser feita usando quatro espaços por nível. * Linhas em branco são recomendadas para separar funções e definições de classes (duas linhas), além de definições de métodos (uma linha). * O espaço em branco deve ser usado para separar operadores matemáticos, binários, de comparação e de atribuição de outros elementos. * **Exemplo**: ```python if status_variavel == False: print("Exibindo informação condicional") ``` :::note Para saber mais sobre nomenclartura e padrão de escrita de código python acesse a documentão da [**PEP8**](https://peps.python.org/pep-0008/ "Clique aqui para acessar a documentação oficial da PEP8") :::