1 - O que é orientação a objetos em Python?

R: Orientação a objetos em Python é um paradigma de programação que permite modelar objetos do mundo real por meio de classes, facilitando a reutilização de código, a organização do programa e a encapsulação de dados.

2 - Qual a diferença entre uma classe e um objeto em Python?

R: Classe é uma estrutura que define as características e comportamentos de um objeto. Já objeto é uma instância de uma classe, criado a partir da definição de suas características.

3 - Como criar uma classe em Python?

R: Para criar uma classe em Python, utiliza-se a palavra reservada "class", seguida do nome da classe e um bloco de código que define suas propriedades e métodos. Por exemplo: ```class Pessoa: def __init__(self, nome, idade): self.nome = nome self.idade = idade def apresentar(self): print("Olá, meu nome é", self.nome)```O código acima cria uma classe Pessoa, com dois atributos (nome e idade) e um método (apresentar()).

4 - Como criar um objeto a partir de uma classe em Python?

R: Para criar um objeto a partir de uma classe em Python, utiliza-se a palavra reservada "new" seguida do nome da classe e os argumentos necessários para inicializar seus atributos. Por exemplo:```pessoa1 = Pessoa("João", 25)```O código acima cria um objeto da classe Pessoa, chamado pessoa1, com os valores "João" e 25 para os atributos nome e idade, respectivamente.

5 - Como criar um construtor em Python?

R: Para criar um construtor em Python, utiliza-se o método especial __init__, que é chamado automaticamente quando um objeto da classe é criado. Por exemplo:```class Pessoa: def __init__(self, nome, idade): self.nome = nome self.idade = idade```O código acima cria um construtor para a classe Pessoa, que espera dois argumentos (nome e idade) e inicializa seus atributos correspondentes.

6 - Como criar um método dentro de uma classe em Python?

R: Para criar um método dentro de uma classe em Python, utiliza-se a definição de uma função dentro da classe, seguida da palavra reservada "self" como primeiro argumento, que representa o próprio objeto. Por exemplo:```class Pessoa: def __init__(self, nome, idade): self.nome = nome self.idade = idade def apresentar(self): print("Olá, meu nome é", self.nome)```O código acima cria um método chamado apresentar(), que imprime uma mensagem com o nome da pessoa.

7 - O que é herança em Python? Como implementar?

R: Herança em Python é um recurso que permite criar uma classe base e outras classes que herdam suas características e comportamentos. Para implementar a herança, utiliza-se a palavra reservada "class", seguida do nome da classe filha e o nome da classe pai dentro de parênteses. Por exemplo:```class Pessoa: def __init__(self, nome): self.nome = nome def apresentar(self): print("Olá, meu nome é", self.nome)class Funcionario(Pessoa): def __init__(self, nome, salario): super().__init__(nome) self.salario = salario def apresentar(self): super().apresentar() print("e meu salário é", self.salario)```O código acima cria as classes Pessoa e Funcionario, onde Funcionario herda as características de Pessoa.