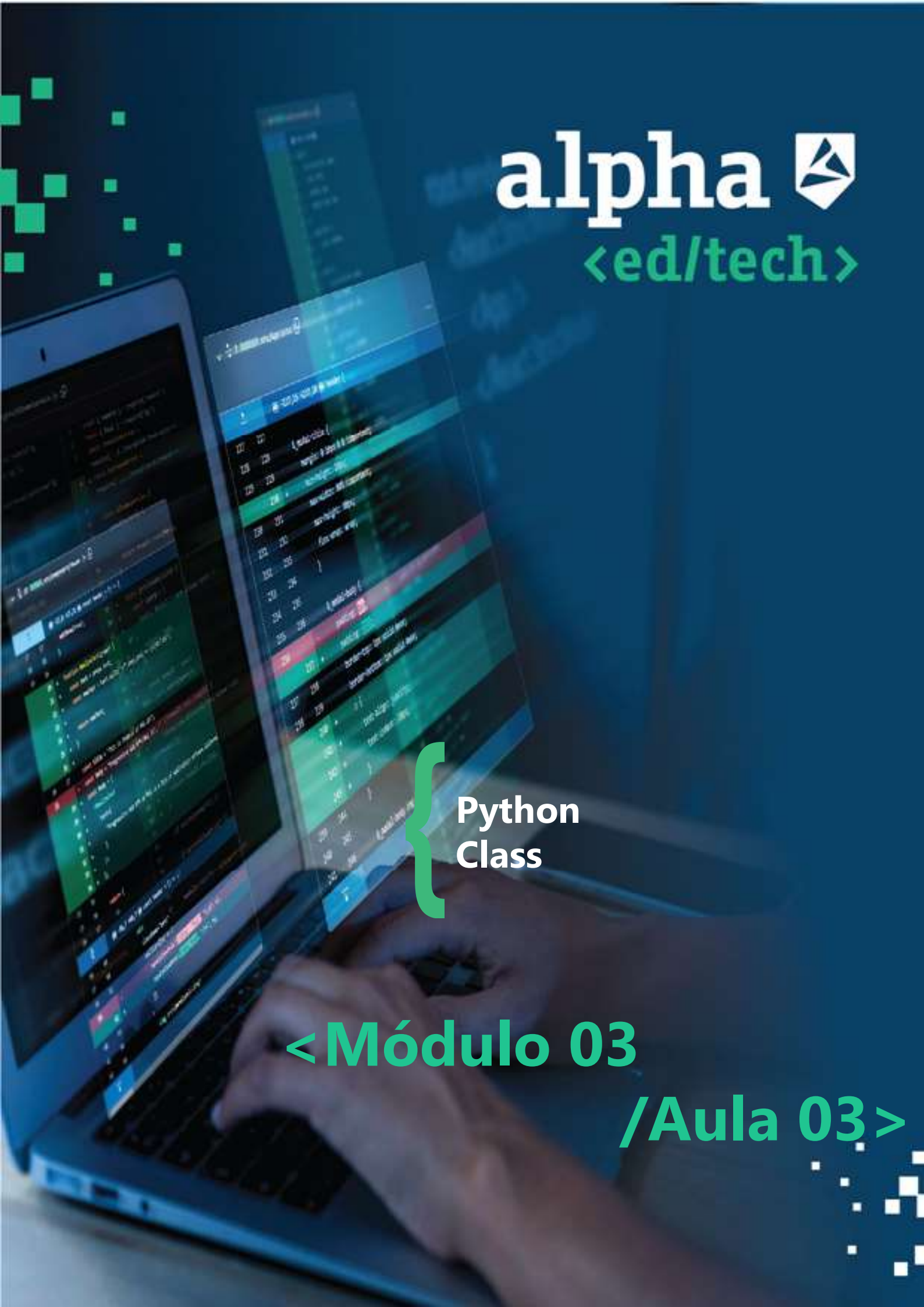
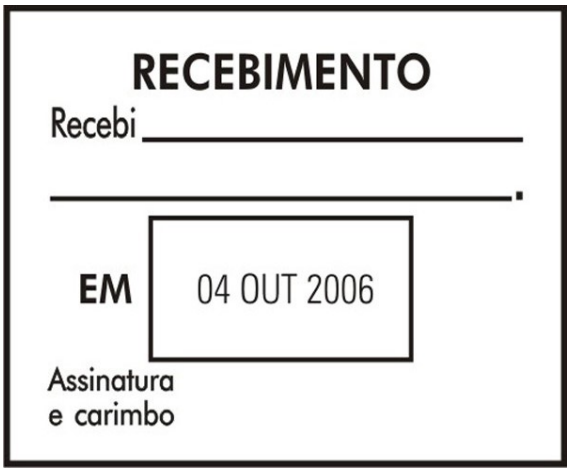

alpha
<ed/tech>
Python
Class
<Módulo 03
/Aula 03>

Funções Definidas como Classes

Introdução

Bem-vindos ao capítulo sobre **Classes** em Python, mas ainda não vamos nos aprofundar em **Orientação a Objetos**.

Para entender o uso de classes em Python, vamos considerar que uma classe é como um modelo (um carimbo) ou um plano para criar objetos.
Exemplo simples de uma classe `Carro` para ilustrar os conceitos.



Definindo uma Classe

Em Python, você pode definir uma classe usando a palavra-chave `**class**`.

- `**tipo**` é um **atributo da classe**. Todos os objetos criados a partir dessa classe terão esse atributo.

- `**__init__**` é um **método especial** chamado de **construtor**. Ele é chamado automaticamente quando um **novo objeto é criado** a partir da classe e é usado para **inicializar os atributos do objeto**.

- `**dirigir**` e `**obter_informacoes**` são **métodos da classe**. Eles representam **comportamentos** que um objeto desse novo tipo da `Class` pode ter.

Uma `**Class**` oferece uma maneira de **armazenar informações e comportamentos no Python (outras linguagens OO também tem o mesmo conceito)**.

Exemplo de Classe Carro

```
class Carro:
    # Atributo da classe
    tipo = "Veículo"

    # Método inicializador (construtor)
    def __init__(self, cor, marca):
        self.cor = cor
        self.marca = marca

    # Método da classe
    def dirigir(self):
        return "Vrum Vrum!"

    # Método da classe
    def obter_informacoes(self):
        return f"Um carro {self.cor} da marca {self.marca}."
```

Criando Objetos da Classe Carro

Uma vez que você tenha definido a classe, você pode **criar objetos** (também chamados de **instâncias**) dessa classe.

Cada objeto tem seus próprios **atributos**, mas compartilha os **métodos** da classe.

```
# Criando objetos (instâncias) da classe Carro
carro1 = Carro(cor="vermelho", marca="Toyota")
carro2 = Carro(cor="azul", marca="Honda")

carrosEstacionados = [ Carro(cor="vermelho", marca="Fiat"),
          Carro(cor="marron", marca="Honda"),
          Carro(cor="marron", marca="Honda") ]
```

Em resumo, em Python, você pode utilizar a **tipagem dinâmica** para a maioria das situações, enquanto a **tipagem estática** (com `typing`) pode ser usada **opcionalmente** para ajudar na legibilidade e na manutenção do código, principalmente em projetos maiores.

Acessando Atributos e Métodos

Você pode acessar os atributos e métodos de um objeto usando a notação de ponto (`.`).

```
# Acessando atributos
print(carro1.cor)      # Saída: vermelho
print(carro2.marca)    # Saída: Honda

# Acessando métodos
print(carro1.dirigir())             # Saída: Vrum Vrum!
print(carro2.obter_informacoes())   # Saída: Um carro azul da marca Honda.
```

Esses são conceitos básicos de classes em Python.
Elas fornecem uma maneira de organizar e estruturar o código de forma mais modular e orientada a objetos.