



P00 Herança e Polimorfismo



Prof. Cloves Rocha
car@etepd.com

LP

Roteiro da aula

PARTE 1 DE 2

- Revisando o paradigma orientado a procedimentos;
- Paradigma orientado a objetos;
- Os quatro pilares;
- Abstração;
- Instância;
- Métodos;
- Método Construtor;

PARTE 2 DE 2

- Mão no código!
 - Prática no laboratório;
 - Herança e Polimorfismo;
 - Desafio prático;
- Referências.
- Material Complementar.

```
var ax = settings.accX;  
var ay = settings.accY;  
var th = t.height();  
var wh = w.height();  
var tw = t.width();  
var ww = w.width();
```

```
if (y + th + ay >= b &&  
    y <= b + wh + ay &&  
    x + tw + ax >= a &&  
    x <= a + ww + ax) {
```

```
    //trigger the custom event  
    if (!t.appeared) t.trigger('appear', settings.delay);
```

```
  } else {
```

```
    //it scrolled out of view  
    t.appeared = false;
```

```
  }
```

```
};
```

```
//create a modified fn with some additional logic  
var modifiedFn = function() {
```

```
    //mark the element as visible
```

Revisando o paradigma orientado a procedimentos.

O que é um objeto então?



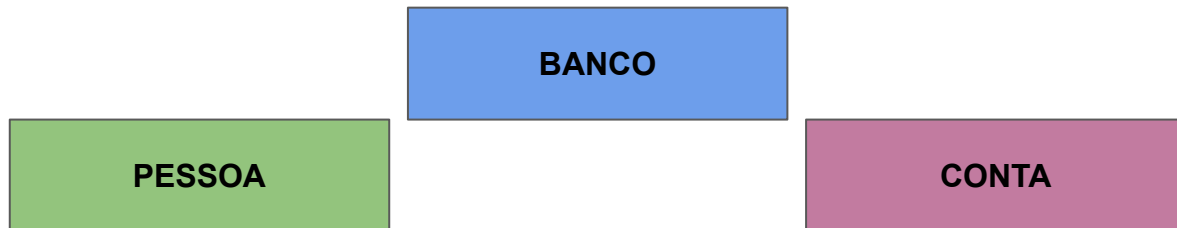
1. Baseado em chamadas de funções ou sub-rotinas que operam sobre elas.
2. O fluxo de dados concentra todas as variáveis.
3. Uma função toma um conjunto de variáveis como argumento e retorna o resultado para o fluxo de dados, para ser usado por outra função ou simplesmente ser exibido para o usuário.



Paradigma orientado a objetos



- Por todo lugar tem objetos: carro, mesa, livro, pessoa, e muito mais;
 - Os objetos do mundo real têm duas características em comum:
 - Estado = propriedades(nome, peso, altura, cor, etc.);
 - Comportamento = ações(andar, falar, calcular, etc.).
- Pensaram em mais algum?



Definições



1. Paradigma para desenvolvimento de software que baseia-se na utilização de componentes individuais (objetos) que colaboram para construir sistemas mais complexos.
2. A colaboração entre os objetos é feita através do envio de mensagens.
3. Um paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver problemas dentro desta fronteira.

Vantagens



1. Facilita a reutilização de código;
2. Os modelos refletem o mundo real de maneira mais aproximada;
3. Descrevem de maneira mais precisa os dados;
4. Mais fáceis de entender e manter;
5. Pequenas mudanças nos requisitos não implicam em grandes alterações no sistema em desenvolvimento.

Os quatro pilares



ABSTRAÇÃO

ENCAPSULAMENTO

HERANÇA

POLIMORFISMO

Abstração - Classe em Python



```
#ETE Porto Digital
#Prof. Cloves Rocha
class Estudante:
    def __init__(self, nome_completo, nome, origem, tipo, nivel_dificuldade):
        self.nome_completo = nome_completo
        self.nome = nome
        self.origem = origem
        self.tipo = tipo
        self.nivel_dificuldade = nivel_dificuldade

maria = Estudante('Maria dos Santos', 'Mari', 'Trabalha', 'Maga', 2)
print( maria.nome_completo + "\n" + maria.origem + "\n" + maria.tipo)
```



Instância



1. Uma instância é um objeto criado com base em uma classe definida;
2. Classe é apenas uma estrutura, que especifica objetos, mas que não pode ser utilizada diretamente;
3. Instância representa o objeto concretizado a partir de uma classe;
4. Uma instância possui um ciclo de vida:

ESTRUTURA

variável = **Classe()**

Exemplo:

```
if __name__ == 'main':  
    conta = Conta()  
    conta.saldo = 20  
    conta.numero = '22123-7'  
    print(conta.saldo)  
    print(conta.numero)
```



1. Representa os comportamentos de uma classe;
2. Permitem acesso a atributos, tanto para recuperar os valores, como para alterá-los caso necessário;
3. Podem retornar ou não algum valor; e
4. Podem possuir ou não parâmetros.

ESTRUTURA

def nome_do_metodo(self, parametros)

Importante: o parâmetro self é obrigatório.

Exemplo:

```
class Conta:
    numero = "00000-0"
    saldo = 0.0

    def deposito(self, valor):
        self.saldo += valor

    def saque(self, valor):
        if (self.saldo > 0):
            self.saldo -= valor
        else:
            print("Saldo insuficiente")

if __name__ == '__main__':
    conta = Conta()
    conta.saldo = 20
    conta.numero = "13131-2"
    print(conta.saldo)
    print(conta.numero)
```

Método construtor



1. Determina que ações devem ser executadas quando da criação de um objeto; e
2. Pode possuir ou não parâmetros.

ESTRUTURA

```
def __init__(self, parametros)
```

Importante: o parâmetro self é obrigatório.

Exemplo:

```
class Conta:
    numero = "00000-0"
    saldo = 0.0

    def __init__(self, numero, saldoInicial):
        self.numero = numero
        self.saldo = saldoInicial

conta = Conta("12345-1", 0)
print(conta.numero)
print(conta.saldo)
```

Mão no código!



PARTE 2 DE 2

Prática no laboratório

Herança e Polimorfismo



Acesse o **GitHub** da disciplina [CLIQUE AQUI](#)

Agora que vocês
conheceram melhor -
POO - Herança e
Polimorfismo.

**Vamos praticar um
pouco!**





Desafio Prático

Conforme classroom da disciplina.



INSTRUÇÕES

- No classroom da disciplina.



DÚVIDAS?



Referências

- Material [GitHub](#).

Material
Complementar
No classroom.