



POO II

Revisão da Linguagem Python

Prof. Romuere Silva

Orientação a Objetos

```
class Conta:
    pass

>>> from conta import Conta
>>> conta = Conta()
>>> type(conta)
<class 'conta.Conta'>
```

```
class Conta:
    def __init__(self, numero, titular, saldo, limite):
        self.numero = numero
        self.titular = titular
        self.saldo = saldo
        self.limite = limite
```

```
>>> from conta import Conta
>>> conta = Conta()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __init__() missing 4 required positional arguments: 'numero', 'titular', 'saldo', and 'limite'
```

```
>>> conta = Conta('123-4', 'João', 120.0, 1000.0)
```

Métodos

```
class Conta:
```

```
# método __init__() omitido
```

```
def deposita(self, valor):  
    self.saldo += valor
```

```
>>> conta.deposita(20.0)
```

```
class Conta:
```

```
# outros métodos omitidos
```

```
def saca(self, valor):  
    self.saldo -= valor
```

```
def extrato(self):  
    print("numero: {} \nsaldo: {}".format(self.numero, self.saldo))
```

```
>>> from conta import Conta
```

```
>>>
```

```
>>> conta = Conta('123-4', 'João', 120.0, 1000.0)
```

```
>>> conta.deposita(20.0)
```

```
>>> conta.extrato()
```

```
numero: '123-4'
```

```
saldo: 140.0
```

```
>>> conta.saca(15)
```

```
>>> conta.extrato()
```

```
numero: '123-4'
```

```
saldo: 125.0
```



Atividade

- Construa a classe Conta com:
 - os métodos deposita, saca e extrato;
 - os atributos numero, titular, saldo e limite.

Métodos com Retorno

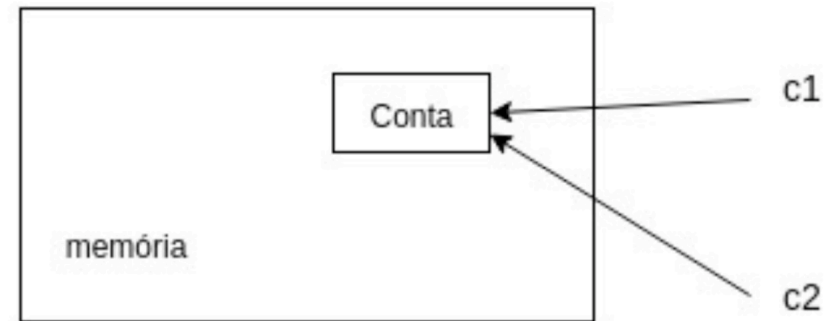
```
def saca(self, valor):  
    if (self.saldo < valor):  
        return False  
    else:  
        self.saldo -= valor  
        return True
```

```
>>> from conta import Conta  
>>> minha_conta.saldo = 1000  
>>> consegui = minha_conta.saca(2000)  
>>> if(conseguir):  
...     print("consegui sacar")  
... else:  
...     print("não consegui sacar")  
>>>  
'não consegui sacar'
```

```
>>> from conta import Conta  
>>> minha_conta.saldo = 1000  
>>> if(minha_conta.saca(2000)):  
...     print("consegui sacar")  
... else:  
...     print("não consegui sacar")  
>>>  
'não consegui sacar'
```

Objetos são acessados por Referência

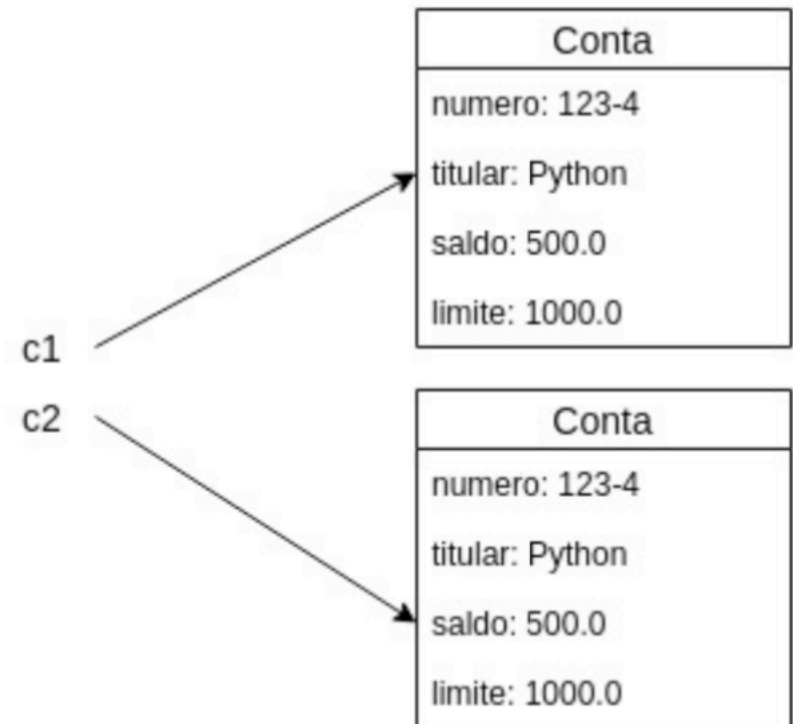
```
>>> from conta import Conta
>>> c1 = Conta('123-4', 'João', 120.0, 1000.0)
>>> c2 = c1
>>> c2.saldo
120.0
>>> c1.deposita(100.0)
>>> c1.saldo
220.0
>>> c2.deposita(30.0)
>>> c2.saldo
250.0
>>> c1.saldo
250.0
```



```
>>> id(c1) == id(c2)
True
>>> c1 == c2
True
```

Objetos são acessados por **Referência**

```
>>> c1 = Conta("123-4", "Python", 500.0, 1000.0)
>>> c2 = Conta("123-4", "Python", 500.0, 1000.0)
>>> if(c1 == c2):
...     print("contas iguais")
>>>
```





Método Transfere



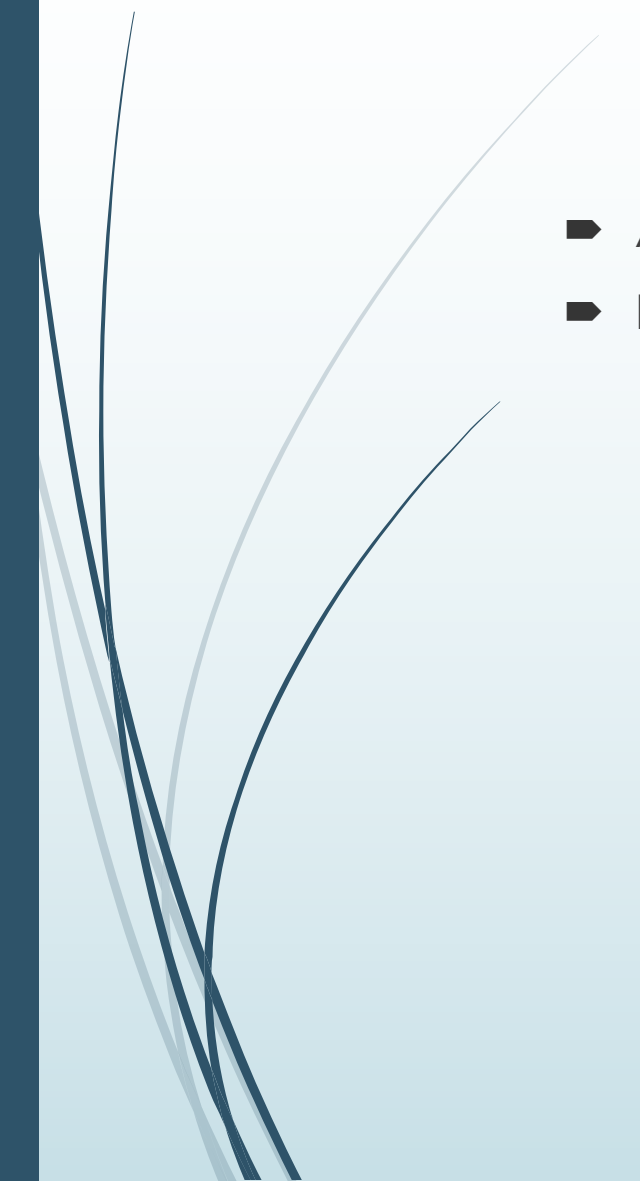
```
class Conta:

    # código omitido

    def transfere(self, destino, valor):
        retirou = self.saca(valor)
        if (retirou == False):
            return False
        else:
            destino.deposita(valor)
            return True
```




Atividade

- Adicione a função transfere na sua conta;
 - Faça testes para transferir entre contas, com e sem saldo suficiente.
- 

Atributos com valor padrão

```
class Conta:
```

```
    def __init__(self, numero, titular, saldo, limite=1000.0):  
        self.numero = numero  
        self.titular = titular  
        self.saldo = saldo  
        self.limite = limite
```

```
>>> conta = Conta('123-4', 'joão', 120.0)
```

Atributos como objetos de outras classes - Agregação

```
class Cliente:

    def __init__(self, nome, sobrenome, cpf):
        self.nome = nome
        self.sobrenome = sobrenome
        self.cpf = cpf

class Conta:

    def __init__(self, numero, cliente, saldo, limite):
        self.numero = numero
        self.titular = cliente
        self.saldo = saldo
        self.limite = limite

>>> from conta import Conta, Cliente
>>> cliente = Cliente('João', 'Oliveira', '111111111-1')
>>> minha_conta = Conta('123-4', cliente, 120.0, 1000.0)
```



Atividade

- Edite seu projeto e crie a classe Cliente com nome, sobrenome e cpf;
- Toda conta agora deverá receber um cliente como parâmetros;



Tudo é Objeto

```
>>> type(conta.numero)
<class 'str'>
>>> type(conta.saldo)
<class 'float'>
>>> type(conta.titular)
<class '__conta__.Cliente'>
```

COMPOSIÇÃO

```
import datetime
```

```
class Historico:
```

```
    def __init__(self):
        self.data_abertura = datetime.datetime.today()
        self.transacoes = []

    def imprime(self):
        print("data abertura: {}".format(self.data_abertura))
        print("transações: ")
        for t in self.transacoes:
            print("-", t)
```

```
class Conta:
```

```
    def __init__(self, numero, cliente, saldo, limite=1000.0):
        self.numero = numero
        self.cliente = cliente
        self.saldo = saldo
        self.limite = limite
        self.historico = Historico()
```

```
class Conta:
```

```
    #código omitido
```

```
    def deposita(self, valor):
        self.saldo += valor
        self.historico.transacoes.append("depósito de {}".format(valor))

    def saca(self, valor):
        if (self.saldo < valor):
            return False
        else:
            self.saldo -= valor
            self.historico.transacoes.append("saque de {}".format(valor))

    def extrato(self):
        print("numero: {} \nsaldo: {}".format(self.numero, self.saldo))
        self.historico.transacoes.append("tirou extrato - saldo de {}".format(self.saldo))

    def transfere_para(self, destino, valor):
        retirou = self.saca(valor)
        if (retirou == False):
            return False
        else:
            destino.deposita(valor)
            self.historico.transacoes.append("transferencia de {} para conta {}".format(valor, destino.numero))
            return True
```

Testando...

```
$python3.6
>>> from conta import Conta, Cliente
>>> cliente1 = Cliente('João', 'Oliveira', '1111111111-11')
>>> cliente2 = Cliente('José', 'Azevedo', '22222222-22')
>>> conta1 = Conta('123-4', cliente1, 1000.0)
>>> conta2 = Conta('123-5', cliente2, 1000.0)
>>> conta1.deposita(100.0)
>>> conta1.saca(50.0)
>>> conta1.transfere_para(conta2, 200.0)
>>> conta1.extrato
numero: 123-4
saldo: 850.0
>>> conta1.historico.imprime()
data abertura: 2018-05-10 19:44:07.406533
transações:
- depósito de 100.0
- saque de 50.0
- saque de 200.0
- transferencia de 200.0 para conta 123-5
- tirou extrato - saldo de 850.0
>>> conta2.historico.imprime()
data abertura: 2018-05-10 19:44:07.406553
transações:
- depósito de 200.0
```



Atividade

- Inclua a classe Histórico na sua conta;
 - Faça testes com depósitos, transferências e saques.
- 