



**INSTITUTO FEDERAL DE ALAGOAS
CAMPUS PALMEIRA DOS ÍNDIOS
CURSO TÉCNICO INTEGRADO AO ENSINO MÉDIO EM INFORMÁTICA**

VITOR VINÍCIUS PORANGABA TORRES

9.5 EXERCÍCIOS:

**PALMEIRA DOS ÍNDIOS-AL
2025**

VITOR VINÍCIUS PORANGABA TORRES

9.5 EXERCÍCIOS:

Trabalho elaborado na disciplina de
Programação orientada á objetos para obtenção
de nota.

Professor: Carlos Jean

PALMEIRA DOS ÍNDIOS-AL
2025

9.5 EXERCÍCIOS:

1. Adicione o modificador de visibilidade privado (dois underscores: __) para cada atributo e método da sua classe Conta .

```
You, 1 second ago | 1 author (You)
class Conta:
    def __init__(self, numero, cliente, saldo, limite = 2000.0):
        self.__numero = numero
        self.__cliente = cliente
        self.__saldo = saldo
        self.__limite = limite
        self.__historico = Historico()
```

Tente criar uma Conta e modificar ou ler um de seus atributos "privados". O que acontece?

- Não acontece nada com os atributos privados, somente cria outros atributos

2. Sabendo que no Python não existem atributos privados, como podemos modificar e ler esses atributos? É uma boa prática fazer isso?

Dica: teste os comandos print(conta.__numero) e print(conta._Conta__numero) . O que ocorre? - O primeiro print não funcionou porque o atributo __numero não existe, pois como ele é privado o nome dele muda automaticamente para _Conta__numero.

```
10
11 print(conta1.__numero)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    JUPYTER

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData/Local/Programs/Python/GitHub/Atividades-de-poo/00/conta_teste.py
Traceback (most recent call last):
  File "c:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo\00\conta_teste.py", line 11, in <module>
    print(conta1.__numero)
    ^^^^^^^^^^^^^^^^^^
AttributeError: 'Conta' object has no attribute '__numero'
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

```
10
11 print(conta1._Conta__numero)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    JUPYTER

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
tos/GitHub/Atividades-de-poo/00/conta_teste.py
123-45
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

3. Modifique o acesso para 'protegido' seguindo a convenção do Python e modifique o prefixo __ por apenas um underscore _ . Crie métodos de acesso em sua classe Conta através do decorator @property .

```
You, 39 minutes ago | 1 author (You)
22 class Conta:
23     def __init__(self, numero, cliente, saldo, limite = 2000.0):
24         self._numero = numero
25         self._cliente = cliente
26         self._saldo = saldo
27         self._limite = limite
28         self._historico = Historico()
29
30     @property
31     def saldo(self):
32         return self._saldo
33
34     @saldo.setter
35     def saldo(self, saldo):
36         if (saldo < 0):
37             print("saldo não pode ser negativo")
38         else:
39             self._saldo = saldo
You, 39 minutes ago • Uncommitted changes

14 conta1.saldo = 120
15 print(conta1.saldo)
16 You, 5 minutes ago • Uncommitted changes
17 conta1.saldo = -30

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    JUPYTER    GITL
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
yhton.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-
120
saldo não pode ser negativo
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

4. Crie novamente uma conta e acesse e modifique seus atributos. O que mudou?

Dica: tente os comandos na seguinte ordem: print(conta._numero) , conta._numero= '50' e print(conta._numero) . O que ocorre? Agora os atributos podem ser alterados porque eles só são privados por convenção.

```
15 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
16 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
17
18 print(conta1._numero, '\n')
19 conta1._numero= '50'
20 print(conta1._numero)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    JUPYTER    GITL
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
yhton.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-
123-45

50
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

5. Modifique sua classe Conta de modo que não seja permitido criar outros atributos além dos definidos anteriormente utilizando `__slots__`.

The screenshot shows a code editor with Python code and a terminal window.

```
class Conta:
    __slots__ = ['_numero', '_cliente', '_saldo', '_limite', '_historico']

    def __init__(self, numero, cliente, saldo, limite = 2000.0):
        self._numero = numero
        self._cliente = cliente
        self._saldo = saldo
        self._limite = limite
        self._historico = Historico()

18 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
19 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
20
21 conta1.dinheiro = 'real'
```

TERMINAL

```
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData/Local/Programs/Python/python.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
Traceback (most recent call last):
  File "c:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo\00\conta_teste.py", line 21, in <module>
    conta1.dinheiro = 'real'
    ^^^^^^^^^^^^^^
AttributeError: 'Conta' object has no attribute 'dinheiro' and no __dict__ for setting new attributes
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

- Com essa mudança não é mais possível criar atributos de forma interativa.

6. (Opcional) Adicione um atributo identificador na classe Conta . Esse identificador deve ter um valor único para cada instância do tipo Conta . A primeira Conta instanciada tem identificador 1, a segunda 2, e assim por diante.

```
1  # Vitor Vinicius Porangaba Torres - 512
2
3  import datetime
4  You, 6 days ago | 1 author (You)
5  > class Historico: ...
6
7
8  You, 2 weeks ago | 1 author (You)
9  > class Cliente: ...
10
11 You, 1 second ago | 1 author (You)
12 class Conta:
13     __slots__ = ['_numero', '_cliente', '_saldo', '_limite', '_historico', '_identificador']
14     identificador = 1
15     You, 21 seconds ago • Uncommitted changes
16     def __init__(self, numero, cliente, saldo, Limite = 2000.0):
17         self._numero = numero
18         self._cliente = cliente
19         self._saldo = saldo
20         self._limite = limite
21         self._historico = Historico()
22         self._identificador = Conta.identificador
23         Conta.identificador += 1
```

```
1  # Vitor Vinicius Porangaba Torres - 512
2
3  from conta import Conta, Cliente, Historico
4  ...
5  print(conta1.__numero)
6  print(conta1._Conta__numero)
7  conta1.saldo = 120
8  print(conta1.saldo)
9  conta1.saldo = -30
10 print(conta1.__numero, '\n')
11 conta1._numero= '50'
12 print(conta1.__numero)
13 conta1.dinheiro = 'real'
14 ...
15 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
16 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)    You, 2
17 cliente2 = Cliente("Vinícius", "Porangaba", "555.666.777-88")
18 conta2 = Conta('678-90', cliente2, 300.0, 2000.0)
19
20 print(conta1._identificador)
21 print(conta2._identificador)
```

Python

```
#conta.py
#Vitor Vinícius Porangaba Torres - 512

import datetime
class Historico:
    def __init__(self):
        self.data_abertura = datetime.datetime.today()
        self.transacoes = []

    def imprime(self, conta):
        print(f"\nData abertura da conta {conta.numero}:
{self.data_abertura}")
        print("transações: ")
        for t in self.transacoes:
            print("-", t)

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

class Conta:
    __slots__ = ['_numero', '_cliente', '_saldo', '_limite',
    '_historico', '_identificador']
    identificador = 1
    def __init__(self, numero, cliente, saldo, limite = 2000.0):
        self._numero = numero
        self._cliente = cliente
        self._saldo = saldo
        self._limite = limite
        self._historico = Historico()
        self._identificador = Conta.identificador
        Conta.identificador += 1

    @property
    def saldo(self):
        return self._saldo

    @saldo.setter
```

```
def saldo(self, saldo):
    if (saldo < 0):
        print("saldo não pode ser negativo")
    else:
        self._saldo = saldo

def imprime_historico(self):
    self.historico.imprime(self)

def deposita(self, valor):
    self.saldo += valor
    self.historico.transacoes.append(f"Depósito de {valor} em
{datetime.datetime.today()}")

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

def extrato(self):
    print(f"Nome: {self.cliente.nome} \nSobrenome:
{sself.cliente.sobrenome} \nCPF: {self.cliente.cpf} \nNúmero:
{sself.numero} \nSaldo: {self.saldo}\n")
    self.historico.transacoes.append(f"tirou extrato em
{datetime.datetime.today()} - saldo de {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(f"transferencia de {valor}
para conta {destino.numero}")
        return True
```

Python

```
# conta_teste.py
# Vitor Vinícius Porangaba Torres - 512

from conta import Conta, Cliente, Historico
'''

print(conta1.__numero)
print(conta1._Conta__numero)
conta1.saldo = 120
print(conta1.saldo)
conta1.saldo = -30
print(conta1._numero, '\n')
conta1._numero= '50'
print(conta1._numero)
conta1.dinheiro = 'real'
'''

cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
cliente2 = Cliente("Vinícius", "Porangaba", "555.666.777-88")
conta2 = Conta('678-90', cliente2, 300.0, 2000.0)

print(conta1._identificador)
print(conta2._identificador)
```