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

VITOR VINÍCIUS PORANGABA TORRES

EXERCÍCIOS 8.13

PALMEIRA DOS ÍNDIOS-AL
2025

VITOR VINÍCIUS PORANGABA TORRES

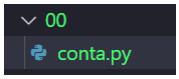
EXERCÍCIOS 8.13

Trabalho elaborado na disciplina de Programação Orientada á objetos para obtenção de nota.
Professor: Carlos Jean

PALMEIRA DOS ÍNDIOS-AL
2025

8.13 EXERCÍCIO: PRIMEIRA CLASSE PYTHON:

1. Crie um arquivo chamado conta.py na pasta oo criada no exercício anterior:



2. Crie a classe Conta sem nenhum atributo e salve o arquivo:

```
00 > conta.py
1   # Vitor Vinícius Porangaba Torres - 512
2   class Conta:
3       pass
```

3. Vá até a pasta onde se encontra o arquivo conta.py e crie um novo arquivo, chamado conta_teste.py . Dentro deste arquivo importe a classe Conta do módulo conta:

```
00 > conta_teste.py
1   # Vitor Vinícius Porangaba Torres - 512
2   from conta import Conta
```

4. Crie uma instância (objeto) da classe Conta e utilize a função type() para verificar o tipo do objeto:

```
00 > conta_teste.py > ...
1   # Vitor Vinícius Porangaba Torres - 512
2   from conta import Conta
3   conta = Conta()
4   print(type(conta))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData
ython.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
<class 'conta.Conta'>
```

5. Abra novamente o arquivo conta.py e escreva o método __init__() , recebendo os atributos anteriormente definidos por nós que toda conta deve ter (numero titular, saldo e limite):

```
@@ -1,4 +1,8 @@
1  1   # Vitor Vinícius Porangaba Torres - 512
2  2
3  3   class Conta:
4     -     pass
      4 +   def __init__(self, numero, titular, saldo, limite):
      5 +       self.numero = numero
      6 +       self.titular = titular
      7 +       self.saldo = saldo
      8 +       self.limite = limite
```

6. No arquivo conta_teste.py , Tente criar uma conta sem passar qualquer argumento no construtor:

```
00 > conta_teste.py > ...
1   # Vitor Vinícius Porangaba Torres - 512
2
3   from conta import Conta
4   conta = Conta()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                Python

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData/Local/Programs/Python/Pyt
ython.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 4, in <module>
    conta = Conta()
TypeError: Conta.__init__() missing 4 required positional arguments: 'numero', 'titular', 'saldo', and 'limite'
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

7. Agora vamos seguir o exigido pela classe, pela receita de uma conta:

```
00 > conta_teste.py > conta
1   # Vitor Vinícius Porangaba Torres - 512
2
3   from conta import Conta
4   conta = Conta('123-45', 'Vini', 120.0, 1000.0)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                Python

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData/Local/Programs/Python/Python313/p
ython.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

8. O interpretador não acusou nenhum erro. Vamos imprimir o número e titular da conta:

```
00 > conta_teste.py > ...
1   # Vitor Vinícius Porangaba Torres - 512
2
3   from conta import Conta
4   conta = Conta('123-45', 'Vini', 120.0, 1000.0)
5   print(f'Número: {conta.numero}')
6   print(f'Titular: {conta.titular}')
```

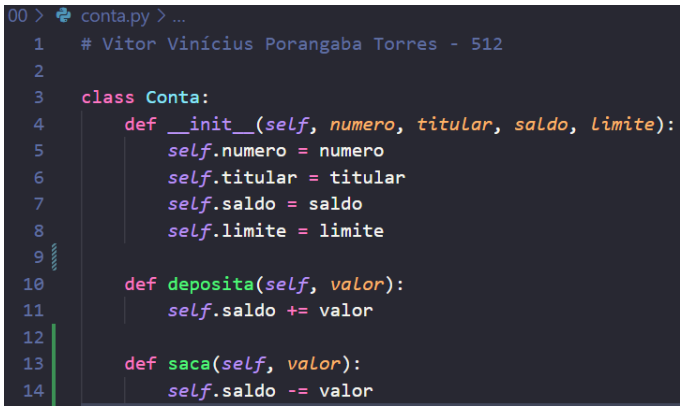
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppDat
ython.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
Número: 123-45
Titular: Vini
```

9. Crie o método deposita() dentro da classe Conta . Esse método deve receber uma referência do próprio objeto e o valor a ser adicionado ao saldo da conta.

```
00 > conta.py > Conta
1   # Vitor Vinícius Porangaba Torres - 512
2
3   class Conta:
4       def __init__(self, numero, titular, saldo, limite):
5           self.numero = numero
6           self.titular = titular
7           self.saldo = saldo
8           self.limite = limite
9
10      def deposita(self, valor):
11          self.saldo += valor
```

10. Crie o método saca() que recebe como argumento uma referência do próprio objeto e o valor a ser sacado. Esse método subtrai o valor do saldo da conta.



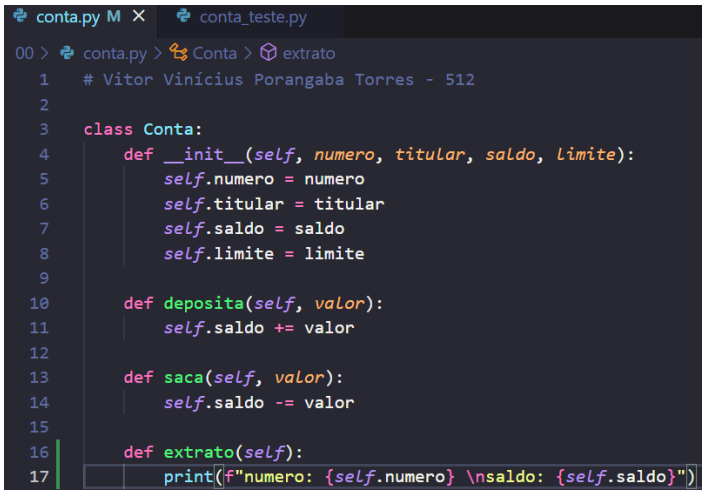
```python
# Vitor Vinícius Porangaba Torres - 512

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

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

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

11. Crie o método extrato() , que recebe como argumento uma referência do próprio objeto. Esse método imprimirá o saldo da conta:



```python
# Vitor Vinícius Porangaba Torres - 512

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

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

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

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

12. Modifique o método saca() fazendo retornar um valor que representa se a operação foi ou não bem sucedida. Lembre que não é permitido sacar um valor maior do que o saldo.

00\conta.py

```
@@ -1,17 +1,21 @@
# Vitor Vinícius Porangaba Torres - 512

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

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

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

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

13. Crie o método transfere_para() que recebe como argumento uma referência do próprio objeto, uma Conta destino e o valor a ser transferido. Esse método deve sacar o valor do próprio objeto e depositar na conta destino:

00 > conta.py > Conta > transfere_para

```python
# Vitor Vinícius Porangaba Torres - 512

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

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

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

    def extrato(self):
        print(f"numero: {self.numero} \nsaldo: {self.saldo}")

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

14. Por fim, crie duas contas no arquivo teste-conta.py e verifique os métodos criados.

```python
# Vitor Vinícius Porangaba Torres - 512

from conta import Conta
conta1 = Conta('123-45', 'Vini', 120.0, 1000.0)
conta2 = Conta('678-90', 'Vitor', 300.0, 2000.0)

conta1.deposita(100)
conta1.extrato()
conta1.saca(200)
conta1.extrato()
conta2.deposita(50)
conta2.extrato()
conta2.saca(300)
conta2.extrato()
conta1.transfere_para(conta2, 20)
conta1.extrato()
conta2.extrato()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/AppData/Local/Programs/Python/Python313/python.exe c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
numero: 123-45
saldo: 220.0
numero: 123-45
saldo: 20.0
numero: 678-90
saldo: 350.0
numero: 678-90
saldo: 50.0
numero: 123-45
saldo: 0.0
numero: 678-90
saldo: 70.0
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

1. (Opcional) Crie uma classe para representar um cliente do nosso banco que deve ter nome, sobrenome e cpf . Instancie uma Conta e passe um cliente como titular da conta. Modifique o método extrato() da classe Conta para imprimir, além do número e o saldo, os dados do cliente. Podemos criar uma Conta sem um Cliente ? E um Cliente sem uma Conta ? Resposta: Não podemos criar uma conta sem cliente, mas podemos criar um cliente sem uma conta, isso se chama agregação.

00\conta.py

```
   1   1   # Vitor Vinícius Porangaba Torres - 512
   2   2
   3   + class Cliente:
   4   +     def __init__(self, nome, sobrenome, cpf):
   5   +         self.nome = nome
   6   +         self.sobrenome = sobrenome
   7   +         self.cpf = cpf
   8   +
   3   9   class Conta:
   4       -     def __init__(self, numero, titular, saldo, limite):
       10  +     def __init__(self, numero, cliente, saldo, limite):
   5   11          self.numero = numero
   6       -         self.titular = titular
       12  +         self.titular = cliente
   7   13          self.saldo = saldo
   8   14          self.limite = limite
   9   15
       @@ -17,8 +23,8 @@ class Conta:
  17   23              self.saldo -= valor
  18   24              return True
  19   25
  20       -     def extrato(self):
  21       -         print(f"numero: {self.numero} \nsaldo: {self.saldo}")
       26  +     def extrato(self, cliente):
       27  +         print(f"Nome: {cliente.nome} \nSonbrenome: {cliente.sobrenome} \nCPF: {cliente.cpf} \nnumero: {self.numero} \nsaldo: {self.saldo}\n")
  22   28
  23   29      def transfere_para(self, destino, valor):
  24   30          retirou = self.saca(valor)
```

00 > conta_teste.py > ...

```python
# Vitor Vinícius Porangaba Torres - 512

from conta import Conta, Cliente

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)

conta1.deposita(1000)
conta1.extrato(cliente1)
conta1.saca(200)
conta1.extrato(cliente1)
conta2.deposita(50)
conta2.extrato(cliente2)
conta2.saca(300)
conta2.extrato(cliente2)
conta1.transfere_para(conta2, 200)
conta1.extrato(cliente1)
conta2.extrato(cliente2)
```

Saída:

```
None
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
C:/Users/vvini/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 1120.0

Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 920.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 350.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 50.0

Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 720.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 250.0
```

2. (Opcional) Crie uma classe que represente uma data, com dia, mês e ano. Crie um atributo data_abertura na classe Conta . Crie uma nova conta e faça testes no console do Python.

00\conta.py

```
@@ -1,17 +1,28 @@
# Vitor Vinícius Porangaba Torres - 512

+ import datetime
+
+ class Historico:
+     def __init__(self, numero):
+         self.data_abertura = datetime.datetime.today()
+         self.numero = numero
+
+     def imprime(self, numero):
+         print(f"Data de abertura da conta {self.numero} é: {self.data_abertura}")
+

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
+         self.historico = Historico(numero)
```

conta_teste.py > ...

```python
# Vitor Vinícius Porangaba Torres - 512

from conta import Conta, Cliente, Historico

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)

conta1.deposita(1000)
conta1.extrato(cliente1)
conta1.saca(200)
conta1.extrato(cliente1)
conta2.deposita(50)
conta2.extrato(cliente2)
conta2.saca(300)
conta2.extrato(cliente2)
conta1.transfere_para(conta2, 200)
conta1.historico.imprime(conta1.numero)
conta1.extrato(cliente1)
conta2.historico.imprime(conta2.numero)
conta2.extrato(cliente2)
```

Saída:

```
None
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
C:/Users/vvini/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 1120.0

Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 920.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 350.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 50.0

Data de abertura da conta 123-45 é: 2025-10-28 11:02:03.321579
Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
numero: 123-45
saldo: 720.0

Data de abertura da conta 678-90 é: 2025-10-28 11:02:03.321633
Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
numero: 678-90
saldo: 250.0
```

3. (Desafio) Crie uma classe Historico que represente o histórico de uma Conta seguindo o exemplo da apostila. Faça testes no console do Python criando algumas contas, fazendo operações e por último mostrando o histórico de transações de uma Conta . Faz sentido criar um objeto do tipo Historico sem uma Conta?
   Resposta: Não faz sentido, por isso o histórico fizemos uma composição

Python

```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:
    def __init__(self, numero, cliente, saldo, limite = 2000.0):
        self.numero = numero
        self.titular = cliente
        self.saldo = saldo
        self.limite = limite
        self.historico = Historico()

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

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

    def extrato(self, cliente):
        print(f"Nome: {cliente.nome} \nSonbrenome: {cliente.sobrenome} \nCPF:
{cliente.cpf} \nNumero: {self.numero} \nSaldo: {self.saldo}\n")
```

```
        self.historico.transacoes.append(f"tirou extrato - 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
```

00 > conta_teste.py > ...

```
1   # Vitor Vinícius Porangaba Torres - 512
2
3   from conta import Conta, Cliente, Historico
4
5   cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
6   conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
7
8   cliente2 = Cliente("Vinícius", "Porangaba", "555.666.777-88")
9   conta2 = Conta('678-90', cliente2, 300.0, 2000.0)
10
11  conta1.deposita(1000)
12  conta1.extrato(cliente1)
13  conta1.saca(200)
14  conta1.extrato(cliente1)
15  conta2.deposita(550)
16  conta2.extrato(cliente2)
17  conta2.saca(300)
18  conta2.extrato(cliente2)
19  conta1.transfere_para(conta2, 200)
20  conta2.transfere_para(conta1, 100)
21  conta1.extrato(cliente1)
22  conta2.extrato(cliente2)
23  conta1.historico.imprime(conta1)
24  conta2.historico.imprime(conta2)
```

Saída:

```
None
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
C:/Users/vvini/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/vvini/OneDrive/Documentos/GitHub/Atividades-de-poo/00/conta_teste.py
Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
Numero: 123-45
Saldo: 1120.0

Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
Numero: 123-45
Saldo: 920.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
Numero: 678-90
Saldo: 850.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
Numero: 678-90
Saldo: 550.0

Nome: Vitor
Sonbrenome: Torres
CPF: 111.222.333-44
Numero: 123-45
Saldo: 820.0

Nome: Vinícius
Sonbrenome: Porangaba
CPF: 555.666.777-88
Numero: 678-90
Saldo: 650.0


Data abertura da conta 123-45: 2025-10-28 13:27:56.240628
transações:
- Depósito de 1000
- tirou extrato - saldo de 1120.0
- tirou extrato - saldo de 920.0
- transferencia de 200 para conta 678-90
- Depósito de 100
- tirou extrato - saldo de 820.0

Data abertura da conta 678-90: 2025-10-28 13:27:56.240662
transações:
- Depósito de 550
- tirou extrato - saldo de 850.0
- tirou extrato - saldo de 550.0
- Depósito de 200
- transferencia de 100 para conta 123-45
- tirou extrato - saldo de 650.0
```