



**INSTITUTO
FEDERAL**

Alagoas

Campus
Palmeira
dos Índios

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

VITOR VINÍCIUS PORANGABA TORRES

EXERCÍCIOS 11.7 E 11.9

**PALMEIRA DOS ÍNDIOS-AL
2025**

VITOR VINÍCIUS PORANGABA TORRES

EXERCÍCIOS 11.7 E 11.9

Trabalho elaborado na disciplina de
Programação Orientada à Objetos para
obtenção de nota.
Professor: Carlos Jean

11.7 EXERCÍCIO: HERANÇA E POLIMORFISMO:

1. Adicione na classe Conta um novo método chamado atualiza() que atualiza a conta de acordo com a taxa percentual:

```
You, 1 second ago | 1 author (You)
21 class Conta:
22     __slots__ = ['_numero', '_cliente', '_saldo', '_limite', '_historico', '_identificador']
23     identificador = 1
24     def __init__(self, numero, cliente, saldo, limite = 2000.0):
25         self._numero = numero
26         self._cliente = cliente
27         self._saldo = saldo
28         self._limite = limite
29         self._historico = Historico()
30         self._identificador = Conta.identificador
31         Conta.identificador += 1
32
33     def atualiza(self, taxa):
34         self._saldo += self._saldo * taxa
35
You, 1 second ago • Uncommitted changes

17 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
18 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
19 cliente2 = Cliente("Vinícius", "Porangaba", "555.666.777-")
20 conta2 = Conta('678-90', cliente2, 300.0, 2000.0)
21
22 print(conta1.saldo)
23 conta1.atualiza(0.10)
24 print(conta1.saldo)

You, 1 second ago • Uncommitted changes

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:\Python313\python.exe c:/Users/vini/OneDrive/Documentos/GitHub
120.0
132.0
```

2. Crie duas subclasses da classe Conta : ContaCorrente e ContaPoupanca . Ambas terão o método atualiza() reescrito: a ContaCorrente deve atualizar-se com o dobro da taxa e a ContaPoupanca deve atualizar-se com o triplo da taxa. Além disso, a ContaCorrente deve reescrever o método deposita() a fim de retirar uma taxa bancária de dez centavos de cada depósito.
- Crie a classe ContaCorrente no arquivo conta.py e faça com que ela seja subclasse (filha) da classe Conta:

```
74 class ContaCorrente(Conta):
75     pass

You, 1 second ago • Uncommitted changes
```

- Crie a classe ContaPoupanca no arquivo conta.py e faça com que ela seja subclasse (filha) da classe Conta :

```
77 class ContaPoupanca(Conta):
78     pass
```

- Reescreva o método atualiza() na classe ContaCorrente , seguindo o enunciado:

```
74 class ContaCorrente(Conta):
75     def atualiza(self, taxa):
76         self._saldo += self._saldo * taxa * 2
```

- Reescreva o método atualiza() na classe ContaPoupanca , seguindo o enunciado:

```
78 class ContaPoupanca(Conta):
79     def atualiza(self, taxa):
80         self._saldo += self._saldo * taxa * 3
```

- Na classe ContaCorrente , reescreva o método deposita() para descontar a taxa bancária de dez centavos:

```
74 class ContaCorrente(Conta):
75     def atualiza(self, taxa):
76         self._saldo += self._saldo * taxa * 2
77     def deposita(self, valor):
78         self._saldo += valor - 0.10
```

- Saída:

```
19 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
20 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
21
22 print(conta1.saldo)
23 ContaCorrente.atualiza(conta1, 0.10)
24 print(conta1.saldo)
25 ContaCorrente.deposita(conta1, 100)
26 print(conta1.saldo)
27 ContaPoupanca.atualiza(conta1, 0.20)
28 print(conta1.saldo)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:\Python313\python.exe c:/Users/vini/OneDrive/Documentos/GitHub/Ativ
120.0
144.0
243.9
390.24
```

3. Agora, teste suas classes no próprio módulo `conta.py`. Acrescente a condição quando o módulo for igual a `__main__` para executarmos no console no VScode. Instancie essas classes, atualize-as e veja o resultado:

```
85  if __name__ == '__main__':
86      c = Conta('123-4', 'Joao', 1000.0)
87      cc = ContaCorrente('123-5', 'Jose', 1000.0)
88      cp = ContaPoupanca('123-6', 'Maria', 1000.0)
89      c.atualiza(0.01)
90      cc.atualiza(0.01)
91      cp.atualiza(0.01)
92      print(c.saldo)
93      print(cc.saldo)
94      print(cp.saldo)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-Python313> python.exe c:/Users/vini/OneDrive/Documentos/GitHub/Atividades-de-Python313/conta.py
1010.0
1020.0
1030.0
```

4. Implemente o método `__str__()` na classe `Conta`. Faça com que ele imprima uma representação mais amigável de um `Conta` contendo todos os seus atributos.

```
74  def __str__(self):
75      return f"Dados da Conta: \nNumero: {self._numero} \nTitular: {self._cliente} \nSaldo: {self._limite} \nLimite:{self._saldo}"
76
```

You, 15 minutes ago | 1 author (You)

```
77  > class ContaCorrente(Conta): ...
83
```

You, 6 minutes ago | 1 author (You)

```
84  > class ContaPoupanca(Conta): ...
87
```

```
88  if __name__ == '__main__':
89      c = Conta('123-4', 'Joao', 1000.0)
90      cc = ContaCorrente('123-5', 'Jose', 1000.0)
91      cp = ContaPoupanca('123-6', 'Maria', 1000.0)
92      c.atualiza(0.01)
93      cc.atualiza(0.01)
94      cp.atualiza(0.01)
95      print(c.saldo)
96      print(cc.saldo)
97      print(cp.saldo)
98  print(cc)
```

You, 1 minute ago • Uncommitted changes

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vini/AppData/Local/Programs/Python/Python313/python.exe c:/Users/vini/OneDrive\Documentos\GitHub\Atividades-de-poo/00/conta.py
1010.0
1020.0
1030.0
Dados da Conta:
Numero: 123-5
Titular: Jose
Saldo: 2000.0
Limite:1020.0
```

5. Vamos criar uma classe que seja responsável por fazer a atualização de todas as contas bancárias e gerar um relatório com o saldo anterior e saldo novo de cada uma das contas. Na pasta src , crie a classe AtualizadorDeContas :

```
32     def atualiza(self, taxa):
33         self._saldo += self._saldo * taxa
34         return self._saldo
35
36     @property
37     def saldo(self): ...
38
39     @saldo.setter
40     def saldo(self, saldo): ...
41
42     def imprime_historico(self): ...
43
44     def deposita(self, valor): ...
45
46     def saca(self, valor): ...
47
48     def extrato(self): ...
49
50     def transfere_para(self, destino, valor): ...
51
52     def __str__(self): ...
53
54
55 You, 1 second ago | 1 author (You)
56 class ContaCorrente(Conta):
57     def atualiza(self, taxa):
58         self._saldo += self._saldo * taxa * 2
59         return self._saldo
60
61     def deposita(self, valor): ...
62
63 You, 1 second ago * Uncommitted changes
64 You, 1 second ago | 1 author (You)
65 class ContaPoupanca(Conta):
66     def atualiza(self, taxa):
67         self._saldo += self._saldo * taxa * 3
68         return self._saldo
69
70 You, 1 second ago | 1 author (You)
71 class AtualizadorDeContas:
72     def __init__(self, selic, saldo_total=0):
73         self._selic = selic
74         self._saldo_total = saldo_total
75         #propriedades
76
77     def roda(self, conta):
78         print("Saldo da Conta: {}".format(conta.saldo))
79         self._saldo_total += conta.atualiza(self._selic)
80         print("Saldo Final: {}".format(self._saldo_total))
81
82
83 cliente1 = Cliente("Vitor", "Torres", "111.222.333-44")
84 conta1 = Conta('123-45', cliente1, 120.0, 1000.0)
85 adc = AtualizadorDeContas(0.02)
86
87 print(conta1.saldo)
88 ContaCorrente.atualiza(conta1, 0.10)
89 print(conta1.saldo)
90 ContaCorrente.deposita(conta1,100)
91 print(conta1.saldo)
92 ContaPoupanca.atualiza(conta1, 0.20)
93 print(conta1.saldo)
94
95 adc.roda(conta1)
96
97 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
98 PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvini/
99 120.0
100 144.0
101 243.9
102 390.24
103 Saldo da Conta: 390.24
104 Saldo Final: 398.0448
105 PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

Questões opcionais:

1. Na 'main', vamos criar algumas contas e rodá-las a partir do AtualizadorDeContas :

```
91 class AtualizadorDeContas:
92     def __init__(self, selic, saldo_total=0):
93         self._selic = selic
94         self._saldo_total = saldo_total
95         #propriedades
96     def roda(self, conta):
97         print("Saldo da Conta: {}".format(conta.saldo))
98         self._saldo_total += conta.atualiza(self._selic)
99         print("Saldo Final: {}".format(self._saldo_total))
100
101 if __name__ == '__main__':
102     c = Conta('123-4', 'Vitor', 1000.0)
103     cc = ContaCorrente('123-5', 'Vinicius', 2000.0)
104     cp = ContaPoupanca('123-6', 'Torres', 3000.0)
105     adc = AtualizadorDeContas(0.02)
106     adc.roda(c)
107     adc.roda(cc)
108     adc.roda(cp)
109     print(f'Saldo total: {adc._saldo_total}') # 0 saldo final
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
os/GitHub/Atividades-de-poo/SRC/conta.py
Saldo da Conta: 1000.0
Saldo Final: 1020.0
Saldo da Conta: 2000.0
Saldo Final: 3100.0
Saldo da Conta: 3000.0
Saldo Final: 6280.0
Saldo total: 6280.0
```

2. (opcional) Se você precisasse criar uma classe ContaInvestimento , e seu método atualiza() fosse complicadíssimo, você precisaria alterar a classe AtualizadorDeContas ?
 - Não, pois se receber a taxa e retornar o self._saldo não importa a complexidade do código.

3. (opcional, Trabalhoso) Crie uma classe Banco que possui uma lista de contas. Repare que em uma lista de contas você pode colocar tanto ContaCorrente quanto ContaPoupanca . Crie um método adiciona() que adiciona uma conta na lista de contas; um método pegaConta() que devolve a conta em determinada posição da lista e outro pegaTotalDeContas() que retorna o total de contas na lista. Depois teste criando diversas contas, insira-as no Banco e depois, com um laço for , percorra todas as contas do Banco para passá-las como argumento para o método roda() do AtualizadorDeContas .

```
You, 1 hour ago | 1 author (You)
91 class AtualizadorDeContas:
92     def __init__(self, selic, saldo_total=0):
93         self._selic = selic
94         self._saldo_total = saldo_total
95         #propriedades
96     def roda(self, conta):
97         print(f'Saldo da Conta: {conta.saldo}')
98         self._saldo_total += conta.atualiza(self._selic)
99         print(f'Saldo Final: {self._saldo_total}')
100
You, 3 minutes ago | 1 author (You)
101 class Banco:
102     def __init__(self):
103         self._lista_contas = []
104
105     def adiciona(self, conta):
106         self._lista_contas.append(conta)
107
108     def pegaConta(self, posicao_conta):
109         return self._lista_contas[posicao_conta]
110
111     @property
112     def pegaTotalDeContas(self):
113         n = 0
114         for _ in self._lista_contas:
115             n += 1
116         return f'O número total de contas é: {n}'
117
You, 1 second ago • Uncommitted changes
118 if __name__ == '__main__':
119     c = Conta('123-4', 'Vitor', 1000.0)
120     cc = ContaCorrente('123-5', 'Vinicius', 2000.0)
121     cc2 = ContaCorrente('123-6', 'porangaba', 4000.0)
122     cp = ContaPoupanca('123-7', 'Torres', 3000.0)
123     adc = AtualizadorDeContas(0.02)
124     banco = Banco()
125     banco.adiciona(c)
126     banco.adiciona(cc)
127     banco.adiciona(cc2)
128     banco.adiciona(cp)
129     print(banco.pegaConta(0))
130     print(banco.pegaTotalDeContas)
131     for i in banco._lista_contas:
132         adc.roda(i)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\vvini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/U
Dados da Conta:
Numero: 123-4
Titular: Vitor
Saldo: 2000.0
Limite:1000.0
O número total de contas é: 4
Saldo da Conta: 1000.0
Saldo Final: 1020.0
Saldo da Conta: 2000.0
Saldo Final: 3100.0
Saldo da Conta: 4000.0
Saldo Final: 7260.0
Saldo da Conta: 3000.0
Saldo Final: 10440.0
```

4. (opcional) Que maneira poderíamos implementar o método `atualiza()` nas classes `ContaCorrente` e `ContaPoupanca` poupando reescrita de código?

```
20 class Conta:
32     def atualiza(self, taxa):
33         self._saldo += self._saldo * taxa
34         return self._saldo
35
36     @property
37 >     def saldo(self): ...
39
40     @saldo.setter
41 >     def saldo(self, saldo): ...
46
47 >     def imprime_historico(self): ...
49
50 >     def deposita(self, valor): ...
53
54 >     def saca(self, valor): ...
61
62 >     def extrato(self): ...
65
66 >     def transfere_para(self, destino, valor): ...
74
75 >     def __str__(self): ...
77
You, 4 minutes ago | 1 author (You)
78 class ContaCorrente(Conta):
79     def atualiza(self, taxa):
80         return super().atualiza(taxa * 2)
81
82
83 >     def deposita(self, valor): ...
85
You, 4 minutes ago | 1 author (You)
86 class ContaPoupanca(Conta):
87     def atualiza(self, taxa):
88         return super().atualiza(taxa * 3)
```

- Usando o método `super()`

5. (opcional) E se criarmos uma classe que não é filha de Conta e tentar passar uma instância no método roda de AtualizadorDeContas ? Com o que aprendemos até aqui, como podemos evitar que erros aconteçam nestes casos? -sim

```
89 class Pato:
90     def grasna():
91         print('quack')
92 class AtualizadorDeContas:
93     def __init__(self, selic, saldo_total=0):
94         self._selic = selic
95         self._saldo_total = saldo_total
96         #propriedades
97     def roda(self, conta):
98         try:
99             print(f"Saldo anterior da Conta: {conta.saldo}")
100             self._saldo_total += conta.atualiza(self._selic)
101             print(f"Saldo Final: {self._saldo_total}")
102         except AttributeError:
103             print(f'Erro: O objeto {conta} não é uma conta válida! (Não possui o método atualiza()).')
104
105 > class Banco: ...
121
122 if __name__ == '__main__':
123
124     adc = AtualizadorDeContas(0.02)
125     pato = Pato.grasna()
126     adc.roda(pato)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

PS C:\Users\vvin\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvin/AppData/Local/Programs/Python/P
quack
Erro: O objeto None não é uma conta válida! (Não possui o método atualiza()).
PS C:\Users\vvin\OneDrive\Documentos\GitHub\Atividades-de-poo>

OU

```
89 class Pato:
90     def grasna():
91         print('quack')
92 class AtualizadorDeContas:
93     def __init__(self, selic, saldo_total=0):
94         self._selic = selic
95         self._saldo_total = saldo_total
96
97     def roda(self, conta):
98         if hasattr(conta, 'atualiza'):
99             print(f"Saldo anterior da Conta: {conta.saldo}")
100             self._saldo_total += conta.atualiza(self._selic)
101             print(f"Saldo Final: {self._saldo_total}")
102         else:
103             print(f'Erro: O objeto {conta} não é uma conta válida! (Não possui o método atualiza()).')
104
105 > class Banco: ...
121
122 if __name__ == '__main__':
123
124     adc = AtualizadorDeContas(0.02)
125     pato = Pato.grasna()
126     adc.roda(pato)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

PS C:\Users\vvin\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vvin/AppData/Local/Programs/Python/P
quack
Erro: O objeto None não é uma conta válida! (Não possui o método atualiza()).
PS C:\Users\vvin\OneDrive\Documentos\GitHub\Atividades-de-poo>

11.9 EXERCÍCIOS - CLASSES ABSTRATAS

1. Torne a classe Conta abstrata:

```
1 # Vitor Vinícius Porangaba Torres - 512
2
3 import datetime
4 import abc
5 > class Historico: ...
15
16 > class Cliente: ...
21 class Conta(abc.ABC):
22     __slots__ = ['_numero', '_cliente', '_saldo', '_limite', '_historico', '_identificador']
23     identificador = 1
24     def __init__(self, numero, cliente, saldo, limite = 2000.0):
25         self._numero = numero
26         self._cliente = cliente
27         self._saldo = saldo
28         self._limite = limite
29         self._historico = Historico()
30         self._identificador = Conta.identificador
31         Conta.identificador += 1
32
```

2. Torne o método atualiza() abstrato:

```
1 # Vitor Vinícius Porangaba Torres - 512
2
3 import datetime
4 import abc
5 > class Historico: ...
15
16 > class Cliente: ...
21 class Conta(abc.ABC):
22     __slots__ = ['_numero', '_cliente', '_saldo', '_limite', '_historico', '_identificador']
23     identificador = 1
24     def __init__(self, numero, cliente, saldo, limite = 2000.0):
25         self._numero = numero
26         self._cliente = cliente
27         self._saldo = saldo
28         self._limite = limite
29         self._historico = Historico()
30         self._identificador = Conta.identificador
31         Conta.identificador += 1
32
33     @abc.abstractmethod
34     def atualiza(self, taxa):
35         self._saldo += self._saldo * taxa
36         return self._saldo
```

3. Tente instanciar uma Conta :

```
125 if __name__ == '__main__':
126     c = Conta('24', 'Vitor', 1000.0)
127
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vini/AppData/Local/Programs/Python/Python39-64/Python.exe -i

Traceback (most recent call last):
 File "c:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo\src\conta.py", line 126, in <module>
 c = Conta('24', 'Vitor', 1000.0)
TypeError: Can't instantiate abstract class Conta without an implementation for abstract method 'atualiza'

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo>

- Ocorreu um erro porque a classe Conta é abstrata.

4. Agora, instancie uma ContaCorrente e uma ContaPoupanca , e teste o código chamando o método atualiza() .

```
125 if __name__ == '__main__':
126     cc = ContaCorrente('123-5', 'Vinicius', 2000.0)
127     cp = ContaPoupanca('123-7', 'Torres', 3000.0)
128     cc.atualiza(0.01)
129     cp.atualiza(0.01)
130     print(cc.saldo)
131     print(cp.saldo)
132
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vini/AppData/Local/Programs/Python/Python39-64/Python.exe -i

2040.0
3090.0

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo>

5. Crie uma classe chamada ContaInvestimento :

```
92 class ContaInvestimento(Conta):
93     pass
```

6. Instancie uma ContaInvestimento :

```
126 if __name__ == '__main__':
127     ci = ContaInvestimento('123-4', 'Vitor', 1000.0)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> & C:/Users/vini/AppData/Local/Programs/Python/Python39-64/Python.exe -i

Traceback (most recent call last):
 File "c:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo\src\conta.py", line 127, in <module>
 ci = ContaInvestimento('123-4', 'Vitor', 1000.0)
TypeError: Can't instantiate abstract class ContaInvestimento without an implementation for abstract method 'atualiza'

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo>

7. Não conseguimos instanciar uma ContaInvestimento que herda Conta sem implementar o método abstrato atualiza(). Vamos criar uma implementação dentro da classe ContaInvestimento :

```
You, 50 seconds ago | 1 author (You)
92 class ContaInvestimento(Conta):
93     def atualiza(self, taxa):
94         return super().atualiza(taxa * 5)
95
```

8. Agora teste instanciando uma ContaInvestimento e chame o método atualiza() :

```
127 if __name__ == '__main__':
128     ci = ContaInvestimento('123-4', 'Vitor', 1000.0)
129     ci.deposita(1000)
130     ci.atualiza(0.2)
131     print(ci.saldo)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
4000.0
PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```

9. (opcional) Crie um atributo tipo nas classes ContaCorrente , ContaPoupanca e ContaInvestimento . Faça com que o tipo também seja impresso quando usamos a função print():

```
You, 57 seconds ago | 1 author (You)
80 class ContaCorrente(Conta):
81     tipo = 'Conta Corrente'
82     def atualiza(self, taxa):
83         return super().atualiza(taxa * 2)
84
85
86 > def deposita(self, valor): ...
88
You, 57 seconds ago | 1 author (You)
89 class ContaPoupanca(Conta):
90     tipo = 'Conta Poupança'
91     def atualiza(self, taxa):
92         return super().atualiza(taxa * 3)
93
You, 57 seconds ago | 1 author (You)
94 class ContaInvestimento(Conta):
95     tipo = 'Conta Investimento'
96     def atualiza(self, taxa):
97         return super().atualiza(taxa * 5)
98
You, 2 hours ago | 1 author (You)
99 > class AtualizadorDeContas: ...
You, 4 hours ago | 1 author (You)
112 > class Banco: ...
128
129
130 if __name__ == '__main__':
131     ci = ContaInvestimento('123-4', 'Vitor', 1000.0)
132     cc = ContaCorrente('123-5', 'Vinicius', 2000.0)
133     cp = ContaPoupanca('123-7', 'Torres', 3000.0)
134     print(ci.tipo)
135     print(cc.tipo)
136     print(cp.tipo)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo> &
Conta Investimento
Conta Corrente
Conta Poupança
PS C:\Users\vini\OneDrive\Documentos\GitHub\Atividades-de-poo>
```