Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Programação com Interfaces Gráficas

AP1 1^o **semestre de 2020.**

Nome -

Assinatura -

---

1. (2 pontos) O trecho de código a seguir deve imprimir o valor listado abaixo. Complete a função `__add__` de modo que isso ocorra.

```
class C(object):
    def __init__(self,a,b):
        self.a = a
        self.b = b

    def __add__(self, c):
        return C(???, self.b+c.b)

    def __repr__(self):
        return self.a+self.b

print (C("ABC", "DEF")+C("foo", "bar"))
```

rabCBADEFbar

2. (2 pontos) O trecho de código a seguir deve imprimir os valores listados abaixo. Complete a chamada recursiva g(???) da função `g(a,b)`.

```
def g (a,b):
    if len(a)<b:return [a]
    return [a[:b]]+g(???)

print (g(list(range(10)),3))
print (g("*"*10,4))
```

```
[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9]]
['****', '****', '**']
```

3. (6 pontos) Considere uma classe `Agencia`, cujos métodos estão listados a seguir. A ideia é que um objeto dessa classe represente operações efetuadas numa agência bancária. Implemente os trechos de código marcados com "...".

   Sugestão: utilize um dicionário.

```
class Agencia(object):
    """Representa contas e suas movimentações numa
       agência bancária."""

    def __init__(self): # (1 ponto)
        """Cria uma nova agência sem nenhuma conta cadastrada."""
        ...

    def cadastradas(self): # (1 ponto)
        """Retorna uma lista de todas as contas cadastradas
            nesta agência, ordenadas por número da conta."""
        ...

    def saldo(self,nConta): # (0.5 ponto)
        """Retorna o saldo da conta cujo número é nConta."""
        ...

    def limite(self,nConta): # (0.5 ponto)
        """Retorna o limite para saque a descoberto
           da conta nConta."""
        ...

    def alteraLimite(self,nConta,vLimite): # (0.5 ponto)
        """Altera o limite para saque a descoberto da conta nConta
        para vLimite, que deve ser 0 ou um número negativo."""
        ...

    def movimenta (self,nConta,valor): # (1.5 pontos)
        """Faz um depósito ou saque.
        nConta é o número da conta e valor é a quantia a ser
        depositada (se positiva) ou a ser sacada (se negativa).
        Se a conta não existe, ela é criada, desde que o valor > 0.
        Saques só são permitidos se não ultrapassarem o limite
        de saques a descoberto da conta, definido
        por default como 0.
        Se a movimentação não for permitida, causa uma exceção
        do tipo ValueError."""
        ...

    def extrato(self,nConta): # (1 ponto)
        """Retorna uma lista das movimentações da conta nConta."""
        ...
```

```
    def __str__(self):
        str = ""
        for c in self.cadastradas():
            str += "conta %d, limite = %.2f" % (c, self.limite(c))
            str += ", extrato = %s, saldo = %.2f\n" % \
                   (self.extrato(c),self.saldo(c))
        return str
```

Eis um exemplo de utilização:

```
a = Agencia()
a.movimenta(5,100)
a.movimenta(7,200)
a.movimenta(5,-50)
a.movimenta(2,400)
print(a)
try:
    a.movimenta(7,-300)
except ValueError as v:
    print (v)
try:
    a.movimenta(8,-10)
except ValueError as v:
    print (v)
a.alteraLimite(7,-400)
a.movimenta(7,-300)
print(a)
```

Eis o que é impresso pelo exemplo acima:

```
conta 2 : limite = 0.00 , extrato = [400] , saldo = 400.00
conta 5 : limite = 0.00 , extrato = [100, -50] , saldo = 50.00
conta 7 : limite = 0.00 , extrato = [200] , saldo = 200.00

Conta sem saldo suficiente p saque
Conta nao pode ser criada com mov negativa
conta 2 : limite = 0.00 , extrato = [400] , saldo = 400.00
conta 5 : limite = 0.00 , extrato = [100, -50] , saldo = 50.00
conta 7 : limite = -400.00 , extrato = [200, -300] , saldo = -100.00
```