Iniciado em	domingo, 2 jul. 2023, 15:50
Estado	Finalizada
Concluída em	domingo, 2 jul. 2023, 18:17
Tempo empregado	2 horas 26 minutos
Avaliar	**10,00** de um máximo de 10,00(**100**%)

Questão 1

Correto

Atingiu 2,00 de 2,00

Dupla

Faça a função $tuplaDupla$ que recebe duas variáveis como argumento e retorne uma tupla dessas duas variáveis. As variáveis podem ser de qualquer tipo.

Entrada

A entrada consiste nos parâmetros da função **tuplaDupla,** `que são duas variáveis de qualquer tipo.`

Saída

Sua função deve retornar a tupla.

Observações

For example:

Test	Result
`print(tuplaDupla(4,2))`	`(4, 2)`
`print(tuplaDupla(4,"bobagem, unb eh top"))`	`(4, 'bobagem, unb eh top')`
`print(tuplaDupla(True,False))`	`(True, False)`

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
def tuplaDupla(a,b):
    return a,b
```

	Test	Expected	Got	
✔	`print(tuplaDupla(4,2))`	`(4, 2)`	`(4, 2)`	✔
✔	`print(tuplaDupla(4,"bobagem, unb eh top"))`	`(4, 'bobagem, unb eh top')`	`(4, 'bobagem, unb eh top')`	✔
✔	`print(tuplaDupla(True,False))`	`(True, False)`	`(True, False)`	✔
✔	`print(tuplaDupla("COISA",False))`	`('COISA', False)`	`('COISA', False)`	✔
✔	`print(tuplaDupla(4.2,22222))`	`(4.2, 22222)`	`(4.2, 22222)`	✔

	Test	Expected	Got	
✔	`print(tuplaDupla(1237192,238799))`	`(1237192, 238799)`	`(1237192, 238799)`	✔

Passou em todos os teste! ✔

Para resolver o problema, deve-se retornar uma tupla dos elementos.

Correto

Notas para este envio: 2,00/2,00.

Questão 2

Correto

Atingiu 2,00 de 2,00

Converter tuplas

Escreva uma função chamada **convert** que receberá uma lista de tuplas com dois valores, chave e valor, e retornará um dicionário, acumulando todos os valores de chaves iguais em uma lista, como nos exemplos.

Entrada

Não há entrada de dados, o teste chama a função **convert** para uma lista de tuplas.

Saída

Não há saída explícita e a função deverá retornar a lista atualizada. Não imprima nada, apenas retorne a resposta.

Observação

- Observando as entradas do testes, implemente somente a função **convert(l)** .
- No primeiro caso de teste, a lista inicial tem um tupla com chave igual a 3, duas tuplas com chave igual a 4 e três tuplas com chave igual a 1. Portanto o resultado é um dicionário com três pares chave-valor.
- Submeta somente o que foi solicitado.

Particularidade do Tópico

Atenção, a criação de uma função com o nome determinado pelo enunciado é fundamental para a prática do aluno e o Moodle irá descontar pontos, caso a criação não tenha sido feita corretamente (sendo ***case-sensitive*** o nome da função).

For example:

Test	Result

`l = [(3, 91), (4, 69), (1, 85), (1, 96), (1, 7), (4, 94)]`
`resposta = convert(l)`
`print(resposta)`
	`{3: [91], 4: [69, 94], 1: [85, 96, 7]}`

`l = [(1, 69), (2, 94), (2, 15), (4, 59), (4, 65), (4, 17)]`
`resposta = convert(l)`
`print(resposta)`
	`{1: [69], 2: [94, 15], 4: [59, 65, 17]}`

`l = [(4, 63), (4, 60), (2, 4), (4, 18), (4, 62)]`
`resposta = convert(l)`
`print(resposta)`
	`{4: [63, 60, 18, 62], 2: [4]}`

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
def convert(l):
    dic = {}
    for tupla in l:
        if tupla[0] in dic.keys():
            dic[tupla[0]] = dic[tupla[0]] +[tupla[1]]
        else:
            dic[tupla[0]] = [tupla[1]]

    return dic

```

	Test	Expected	Got	

✔	`l = [(3, 91), (4, 69), (1, 85), (1, 96), (1, 7), (4, 94)]`
	`resposta = convert(l)`
	`print(resposta)`
		`{3: [91], 4: [69, 94], 1: [85, 96, 7]}`	`{3: [91], 4: [69, 94], 1: [85, 96, 7]}`	✔

✔	`l = [(1, 69), (2, 94), (2, 15), (4, 59), (4, 65), (4, 17)]`
	`resposta = convert(l)`
	`print(resposta)`
		`{1: [69], 2: [94, 15], 4: [59, 65, 17]}`	`{1: [69], 2: [94, 15], 4: [59, 65, 17]}`	✔

✔	`l = [(4, 63), (4, 60), (2, 4), (4, 18), (4, 62)]`
	`resposta = convert(l)`
	`print(resposta)`
		`{4: [63, 60, 18, 62], 2: [4]}`	`{4: [63, 60, 18, 62], 2: [4]}`	✔

✔	`l = [(3, 15), (1, 11), (4, 98), (2, 32), (2, 95), (3, 62), (1, 67), (3, 77), (2, 37), (2, 33)]`
	`resposta = convert(l)`
	`print(resposta)`
		`{3: [15, 62, 77], 1: [11, 67], 4: [98], 2: [32, 95, 37, 33]}`	`{3: [15, 62, 77], 1: [11, 67], 4: [98], 2: [32, 95, 37, 33]}`	✔

✔	`l = [(1, 287), (19, 236), (16, 290), (19, 511), (10, 286), (16, 957), (12, 640), (5, 937), (5, 603), (8, 131), (16, 741), (4, 455), (18, 731), (11, 400), (7, 771), (2, 843), (12, 341), (8, 79), (17, 531), (16, 278), (1, 918), (6, 385), (7, 706), (17, 659), (8, 302), (2, 991), (2, 577), (2, 906), (9, 978), (11, 674), (10, 628), (16, 563), (4, 65), (2, 37), (19, 945), (11, 989), (15, 799), (19, 590), (11, 598), (13, 270), (8, 579), (7, 637), (18, 704), (7, 820), (14, 714), (1, 373), (18, 750), (18, 751), (15, 917), (18, 416), (16, 174), (13, 379), (2, 325), (5, 832), (16, 76), (16, 440), (15, 797), (16, 954), (5, 622), (19, 897), (8, 839), (12, 751), (14, 764), (1, 654), (1, 637), (10, 51), (6, 727), (15, 783), (17, 384), (17, 961), (15, 220), (5, 97), (5, 869), (9, 386), (3, 22), (14, 397), (7, 477), (5, 841), (19, 237), (16, 138), (7, 251), (18, 462), (17, 400), (14, 680), (13, 155), (9, 828), (2, 336), (18, 143), (8, 478), (19, 337), (7, 877), (12, 736), (10, 174), (6, 317), (16, 431), (4, 622), (10, 536), (19, 88), (15, 306), (10, 4)]` `resposta = convert(l)` `print(resposta)`	`{1: [287, 918, 373, 654, 637], 19: [236, 511, 945, 590, 897, 237, 337, 88], 16: [290, 957, 741, 278, 563, 174, 76, 440, 954, 138, 431], 10: [286, 628, 51, 174, 536, 4], 12: [640, 341, 751, 736], 5: [937, 603, 832, 622, 97, 869, 841], 8: [131, 79, 302, 579, 839, 478], 4: [455, 65, 622], 18: [731, 704, 750, 751, 416, 462, 143], 11: [400, 674, 989, 598], 7: [771, 706, 637, 820, 477, 251, 877], 2: [843, 991, 577, 906, 37, 325, 336], 17: [531, 659, 384, 961, 400], 6: [385, 727, 317], 9: [978, 386, 828], 15: [799, 917, 797, 783, 220, 306], 13: [270, 379, 155], 14: [714, 764, 397, 680], 3: [22]}`	`{1: [287, 918, 373, 654, 637], 19: [236, 511, 945, 590, 897, 237, 337, 88], 16: [290, 957, 741, 278, 563, 174, 76, 440, 954, 138, 431], 10: [286, 628, 51, 174, 536, 4], 12: [640, 341, 751, 736], 5: [937, 603, 832, 622, 97, 869, 841], 8: [131, 79, 302, 579, 839, 478], 4: [455, 65, 622], 18: [731, 704, 750, 751, 416, 462, 143], 11: [400, 674, 989, 598], 7: [771, 706, 637, 820, 477, 251, 877], 2: [843, 991, 577, 906, 37, 325, 336], 17: [531, 659, 384, 961, 400], 6: [385, 727, 317], 9: [978, 386, 828], 15: [799, 917, 797, 783, 220, 306], 13: [270, 379, 155], 14: [714, 764, 397, 680], 3: [22]}`	✔

	Test	Expected	Got	
✔	`l = [(12, 303), (16, 867), (19, 8), (5, 533), (7, 842), (3, 136), (13, 961), (17, 798), (17, 488), (3, 895), (18, 647), (19, 187), (4, 657), (6, 953), (9, 199), (14, 562), (20, 237), (10, 928), (13, 718), (11, 785), (13, 317), (8, 495), (8, 269), (8, 437), (12, 65), (9, 91), (6, 888), (9, 600), (12, 723), (5, 187), (20, 424), (4, 549), (4, 677), (9, 54), (5, 293), (15, 153), (1, 112), (1, 951), (4, 993), (17, 216), (7, 212), (15, 49), (4, 639), (5, 600), (20, 383), (14, 243), (13, 811), (3, 241), (11, 927), (4, 453), (18, 776), (8, 843), (20, 276), (13, 907), (16, 727), (2, 811), (15, 205), (11, 828), (3, 510), (19, 850), (12, 209), (6, 902), (14, 343), (7, 50), (4, 786), (8, 677), (7, 768), (12, 198), (7, 474), (17, 911), (3, 300), (11, 666), (14, 910), (20, 187), (12, 747), (18, 463), (16, 905), (19, 784), (1, 882), (19, 379), (19, 880), (19, 689), (2, 950), (18, 523), (10, 458), (10, 157), (10, 393), (12, 640), (19, 212), (10, 720), (18, 570), (19, 633), (20, 244), (11, 683), (14, 808), (11, 959), (16, 766), (10, 191), (7, 341), (12, 524)]` `resposta = convert(l)` `print(resposta)`	`{12: [303, 65, 723, 209, 198, 747, 640, 524], 16: [867, 727, 905, 766], 19: [8, 187, 850, 784, 379, 880, 689, 212, 633], 5: [533, 187, 293, 600], 7: [842, 212, 50, 768, 474, 341], 3: [136, 895, 241, 510, 300], 13: [961, 718, 317, 811, 907], 17: [798, 488, 216, 911], 18: [647, 776, 463, 523, 570], 4: [657, 549, 677, 993, 639, 453, 786], 6: [953, 888, 902], 9: [199, 91, 600, 54], 14: [562, 243, 343, 910, 808], 20: [237, 424, 383, 276, 187, 244], 10: [928, 458, 157, 393, 720, 191], 11: [785, 927, 828, 666, 683, 959], 8: [495, 269, 437, 843, 677], 15: [153, 49, 205], 1: [112, 951, 882], 2: [811, 950]}`	`{12: [303, 65, 723, 209, 198, 747, 640, 524], 16: [867, 727, 905, 766], 19: [8, 187, 850, 784, 379, 880, 689, 212, 633], 5: [533, 187, 293, 600], 7: [842, 212, 50, 768, 474, 341], 3: [136, 895, 241, 510, 300], 13: [961, 718, 317, 811, 907], 17: [798, 488, 216, 911], 18: [647, 776, 463, 523, 570], 4: [657, 549, 677, 993, 639, 453, 786], 6: [953, 888, 902], 9: [199, 91, 600, 54], 14: [562, 243, 343, 910, 808], 20: [237, 424, 383, 276, 187, 244], 10: [928, 458, 157, 393, 720, 191], 11: [785, 927, 828, 666, 683, 959], 8: [495, 269, 437, 843, 677], 15: [153, 49, 205], 1: [112, 951, 882], 2: [811, 950]}`	✔

Passou em todos os teste! ✔

Uma forma para resolver a questão está em fazer uma iteração sobre a lista l recuperando cada valor da tupla separado. Em seguida pode-se percorrer o dicionário resultado para verificar se o valor de chave correspondente ao primeiro valor da tupla, já existe. Caso já exista adiciona o segundo valor da tupla no valor associado a chave, caso contrário cria-se um novo par chave-valor.

Correto

Notas para este envio: 2,00/2,00.

Questão 3

Correto

Atingiu 2,00 de 2,00

Aprendendo Zip 2

Crie um programa que lê 5 números inteiros e armazena na lista $lista1$ e depois lê mais 5 número inteiros e armazena na $lista2$.

Em seguida, crie e imprima a lista de tuplas $list\_tuple$ formada a partir dos elementos da $lista1$ e $lista2$. (Dica: utilize a função zip.)

Por fim, imprima uma lista $medias$ cujos elementos são as médias das tuplas.

Entrada

A entrada consiste em 10 números inteiros.

Saída

A saída consiste na lista de tuplas formada pelos elementos de $lista1$ e $lista2$, e na lista das médias das tuplas.

Observações

- No primeiro exemplo de teste, foram digitados os elementos da $lista1$, 14, 23, 0, 4 e 5, e os elementos da $lista2$, 93, 84, 5, 43 e 1. Retornando as listas de tuplas [(14, 93), (23, 84), (0, 5), (4, 43), (5, 1)] e de médias de cada tupla [53.5, 53.5, 2.5, 23.5, 3.0].

For example:

Input	Result
14 23 0 4 5 93 84 5 43 1	[(14, 93), (23, 84), (0, 5), (4, 43), (5, 1)] [53.5, 53.5, 2.5, 23.5, 3.0]
0 0 0 0 0 0 0 0 0 0	[(0, 0), (0, 0), (0, 0), (0, 0), (0, 0)] [0.0, 0.0, 0.0, 0.0, 0.0]
1 2 3 4 5 6 7 8 9 10	[(1, 6), (2, 7), (3, 8), (4, 9), (5, 10)] [3.5, 4.5, 5.5, 6.5, 7.5]

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
lista1 =[]
lista2 =[]
for i in range(10):
    if i > 4:
```

```
        lista2 +=[int(input())]
    else:
        lista1 +=[int(input())]
list_tuple = list(zip(lista1,lista2))
print(list_tuple)
list_media =[]
for i in range(5):
    list_media += [(list_tuple[i][0]+list_tuple[i][1])/2]
print(list_media)

```

	Input	Expected	Got	
✔	14 23 0 4 5 93 84 5 43 1	[(14, 93), (23, 84), (0, 5), (4, 43), (5, 1)] [53.5, 53.5, 2.5, 23.5, 3.0]	[(14, 93), (23, 84), (0, 5), (4, 43), (5, 1)] [53.5, 53.5, 2.5, 23.5, 3.0]	✔
✔	0 0 0 0 0 0 0 0 0 0	[(0, 0), (0, 0), (0, 0), (0, 0), (0, 0)] [0.0, 0.0, 0.0, 0.0, 0.0]	[(0, 0), (0, 0), (0, 0), (0, 0), (0, 0)] [0.0, 0.0, 0.0, 0.0, 0.0]	✔
✔	1 2 3 4 5 6 7 8 9 10	[(1, 6), (2, 7), (3, 8), (4, 9), (5, 10)] [3.5, 4.5, 5.5, 6.5, 7.5]	[(1, 6), (2, 7), (3, 8), (4, 9), (5, 10)] [3.5, 4.5, 5.5, 6.5, 7.5]	✔
✔	-21 93 -4 5 1 8992 -2 1 43 56	[(-21, 8992), (93, -2), (-4, 1), (5, 43), (1, 56)] [4485.5, 45.5, -1.5, 24.0, 28.5]	[(-21, 8992), (93, -2), (-4, 1), (5, 43), (1, 56)] [4485.5, 45.5, -1.5, 24.0, 28.5]	✔

	Input	Expected	Got	
✔	4847395 37483 128983274 387492 192838 -849283 -283 2341 371 -9382	[(4847395, -849283), (37483, -283), (128983274, 2341), (387492, 371), (192838, -9382)] [1999056.0, 18600.0, 64492807.5, 193931.5, 91728.0]	[(4847395, -849283), (37483, -283), (128983274, 2341), (387492, 371), (192838, -9382)] [1999056.0, 18600.0, 64492807.5, 193931.5, 91728.0]	✔
✔	-1 1 -2 2 -3 3 -4 4 -5 5	[(-1, 3), (1, -4), (-2, 4), (2, -5), (-3, 5)] [1.0, -1.5, 1.0, -1.5, 1.0]	[(-1, 3), (1, -4), (-2, 4), (2, -5), (-3, 5)] [1.0, -1.5, 1.0, -1.5, 1.0]	✔

Passou em todos os teste! ✔

Correto

Notas para este envio: 2,00/2,00.

Questão 4

Correto

Atingiu 2,00 de 2,00

Apagar tuplas

Escreva uma função chamada "**erase**" que receberá uma lista de tuplas, possivelmente tuplas vazias, e retornará uma lista de tuplas sem nenhuma tupla vazia.

Entrada

Essa função recebe como entrada uma lista de tuplas.

Saída

A função deverá retornar a lista atualizada.

For example:

Test	Result

`l = [(), (15,), (), (), (2, 15, 17)]`
`resposta = erase(l)`
`print(resposta)`
	`[(15,), (2, 15, 17)]`

`l = [(), (), (), (17, 4, 6, 2, 1), ()]`
`resposta = erase(l)`
`print(resposta)`
	`[(17, 4, 6, 2, 1)]`

`l = [(), (9, 16, 9), (14, 15), (), ()]`
`resposta = erase(l)`
`print(resposta)`
	`[(9, 16, 9), (14, 15)]`

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
def erase(l):
    newl =[]
    for element in l:
        if element != ():
            newl += [element]
    return newl
```

	Test	Expected	Got	

✔	`l = [(), (15,), (), (), (2, 15, 17)]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(15,), (2, 15, 17)]`	`[(15,), (2, 15, 17)]`	✔

✔	`l = [(), (), (), (17, 4, 6, 2, 1), ()]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(17, 4, 6, 2, 1)]`	`[(17, 4, 6, 2, 1)]`	✔

✔	`l = [(), (9, 16, 9), (14, 15), (), ()]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(9, 16, 9), (14, 15)]`	`[(9, 16, 9), (14, 15)]`	✔

	Test	Expected	Got	

✔	`l = [(), (9, 16, 9), (14, 15), (), ()]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(9, 16, 9), (14, 15)]`	`[(9, 16, 9), (14, 15)]`	✔

✔	`l = [(4,), (), (11, 10, 18, 19, 7, 1, 19, 15, 11, 2), (), (5, 6, 16, 6, 13, 15, 16, 20, 20)]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(4,), (11, 10, 18, 19, 7, 1, 19, 15, 11, 2), (5, 6, 16, 6, 13, 15, 16, 20, 20)]`	`[(4,), (11, 10, 18, 19, 7, 1, 19, 15, 11, 2), (5, 6, 16, 6, 13, 15, 16, 20, 20)]`	✔

✔	`l = [(), (), (), (1482, 316, 1617, 657, 1595, 45, 850, 798, 234, 1844), (), (), (1951,), (872, 1394, 1283, 487, 1320, 314), (), (1324, 850), (), (), (), (), (), (), (), (), (), (1443, 598, 573, 686, 29)]`
	`resposta = erase(l)`
	`print(resposta)`
		`[(1482, 316, 1617, 657, 1595, 45, 850, 798, 234, 1844), (1951,), (872, 1394, 1283, 487, 1320, 314), (1324, 850), (1443, 598, 573, 686, 29)]`	`[(1482, 316, 1617, 657, 1595, 45, 850, 798, 234, 1844), (1951,), (872, 1394, 1283, 487, 1320, 314), (1324, 850), (1443, 598, 573, 686, 29)]`	✔

Passou em todos os teste! ✔

Correto

Notas para este envio: 2,00/2,00.

Questão 5

Correto

Atingiu 2,00 de 2,00

Stock market

Um bloco de ações pode ser definido como um conjunto de informações a respeito de ações ou títulos. As informações são normalmente a data de compra, o preço de compra, o número de ações compradas e o símbolo da ação. Podemos registrar essas informações em uma lista de tuplas para cada bloco de ações e fazer uma série de operações simples nos blocos. Implemente a função **stockmarket** que recebe uma lista de tuplas de ações e retorna o valor total das ações sumarizadas por data na ordem fornecida em um dicionário.

Entrada

Não há entrada de dados, o teste chama a função **stockmarket** para uma lista de tuplas de ações.

Saída

Não há saída explícita: o teste apresenta o dicionário sumarizado do valor da ações por data resultante da chamada da função automaticamente.

Observação

- No primeiro caso de teste, foram realizadas compras nos dias 24, 25, 26 e 27 de outubro. Portanto, o dicionário de saída contém 4 entradas, cada uma para cada data, armazenadas como chaves. No dia 24 de outubro foram compradas 25 ações por 43.50 cada o que deu um total de 1087.5 em valores gastos com a compra.
- Submeta somente o que foi solicitado.

Particularidade do Tópico

Atenção, a criação de uma função com o nome determinado pelo enunciado é fundamental para a prática do aluno e o Moodle irá descontar pontos, caso a criação não tenha sido feita corretamente (sendo ***case-sensitive*** o nome da função).

For example:

Test	Result

`stock = [('24-Out-2020', 43.50, 25, 'CAT'),`
`('25-Out-2020', 42.80, 50, 'ITU'),`
`('26-Out-2020', 42.10, 75, 'ITU'),`
`('27-Out-2020', 37.58, 100, 'GM')]`
`print(stockmarket(stock))`
	`{'24-Out-2020': 1087.5, '25-Out-2020': 2140.0, '26-Out-2020': 3157.5, '27-Out-2020': 3758.0}`

`stock = [('24-Out-2020', 43.0, 25, 'NUB'),`
`('24-Out-2020', 20.0, 50, 'NUB'),`
`('25-Out-2020', 30.0, 75, 'ITU'),`
`('25-Out-2020', 35.0, 100, 'ITU')]`
`print(stockmarket(stock))`
	`{'24-Out-2020': 2075.0, '25-Out-2020': 5750.0}`

`stock = [('29-Out-2020', 15.10, 1000, 'BRA')]` `print(stockmarket(stock))`	`{'29-Out-2020': 15100.0}`

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
def stockmarket(stock):
    dic_stock ={}
    for operacao in stock:
        if operacao[0] in dic_stock.keys():
            dic_stock[operacao[0]] = dic_stock[operacao[0]]+ float(operacao[1]*operacao[2])
        else:
            dic_stock[operacao[0]] = float(operacao[1]*operacao[2])
    return dic_stock


```

	Test	Expected	Got	
✔	`stock = [('24-Out-2020', 43.50, 25, 'CAT'), ('25-Out-2020', 42.80, 50, 'ITU'), ('26-Out-2020', 42.10, 75, 'ITU'), ('27-Out-2020', 37.58, 100, 'GM')]` `print(stockmarket(stock))`	`{'24-Out-2020': 1087.5, '25-Out-2020': 2140.0, '26-Out-2020': 3157.5, '27-Out-2020': 3758.0}`	`{'24-Out-2020': 1087.5, '25-Out-2020': 2140.0, '26-Out-2020': 3157.5, '27-Out-2020': 3758.0}`	✔
✔	`stock = [('24-Out-2020', 43.0, 25, 'NUB'), ('24-Out-2020', 20.0, 50, 'NUB'), ('25-Out-2020', 30.0, 75, 'ITU'), ('25-Out-2020', 35.0, 100, 'ITU')]` `print(stockmarket(stock))`	`{'24-Out-2020': 2075.0, '25-Out-2020': 5750.0}`	`{'24-Out-2020': 2075.0, '25-Out-2020': 5750.0}`	✔
✔	`stock = [('29-Out-2020', 15.10, 1000, 'BRA')]` `print(stockmarket(stock))`	`{'29-Out-2020': 15100.0}`	`{'29-Out-2020': 15100.0}`	✔
✔	`stock = [('25-Out-2020', 37.58, 100, 'GM'), ('25-Out-2020', 37.58, 100, 'FIT'), ('25-Out-2020', 37.58, 100, 'FRD'), ('25-Out-2020', 37.58, 100, 'HND'), ('25-Out-2020', 37.58, 100, 'TYO'), ('25-Out-2020', 37.58, 100, 'CHV'), ('25-Out-2020', 37.58, 100, 'JEP')]` `print(stockmarket(stock))`	`{'25-Out-2020': 26306.0}`	`{'25-Out-2020': 26306.0}`	✔
✔	`stock = [('25-Out-2020', 37.58, 100, 'GM'), ('25-Out-2020', 37.58, 100, 'FIT'), ('25-Out-2020', 37.58, 100, 'FRD'), ('25-Out-2020', 37.58, 100, 'HND'), ('25-Out-2020', 37.58, 100, 'TYO'), ('25-Out-2020', 37.58, 100, 'CHV'), ('25-Out-2020', 37.58, 100, 'JEP')]` `print(stockmarket(stock))`	`{'25-Out-2020': 26306.0}`	`{'25-Out-2020': 26306.0}`	✔

	Test	Expected	Got	
✔	`stock = [('25-Out-2020', 40.0, 100, 'GM'), ('25-Out-2020', 42.0, 100, 'FIT'), ('01-Nov-2020', 36, 100, 'GM'), ('01-Nov-2020', 20, 100, 'FIT')]` `print(stockmarket(stock))`	`{'25-Out-2020': 8200.0, '01-Nov-2020': 5600.0}`	`{'25-Out-2020': 8200.0, '01-Nov-2020': 5600.0}`	✔

Passou em todos os teste! ✔

A questão pode ser resolvida realizando uma iteração sobre a lista onde a primeira posição é armazenada como chave no dicionário e o valor é calculado multiplicando a segunda pela terceira posições da tupla. Caso a data já exista como chave no dicionário, o valor deve ser atualizado com a multiplicação calculada.

Correto

Notas para este envio: 2,00/2,00.