Started on	Tuesday, 1 December 2020, 8:17 PM
State	Finished
Completed on	Thursday, 3 December 2020, 3:13 PM
Time taken	1 day 18 hours
Grade	**12.00** out of 12.00 (**100**%)

Question **1**

Correct

Mark 3.00 out of 3.00

Defina a função `pivotSoma` que recebe uma lista de inteiros e devolve o índice do elemento da lista para o qual a soma do elementos à sua esquerda é igual à soma dos elementos à sua direita (chamamos a este elemento o *pivot*).

Por exemplo, para a lista `[9, 3, 8, 1]` o resultado é 1, ou seja, o índice onde se encontra o pivot. Neste exemplo, o pivot é o elemento 3, dado que 9=8+1.

Se houver mais que um *pivot*, a função deve devolver o índice mais pequeno entre as soluções possíveis.

Se não houver solução, a função deve devolver -1.

For example:

Test	Result
`print(pivotSoma([9,3,8,1]))`	`1`
`print(pivotSoma([2,2]))`	`-1`
`print(pivotSoma([7,-1,0,-1,1,1,2,3]))`	`2`

Answer: (penalty regime: 0 %)

Reset answer

```
def pivotSoma(lista):
    somaEsq = 0
    somaDir = sum(lista)

    for i, x in enumerate(lista):
        somaDir -= x
        if somaEsq == somaDir:
            return i
        somaEsq += x
    return -1
```

	Test	Expected	Got	
✔	`print(pivotSoma([9,3,8,1]))`	`1`	`1`	✔
✔	`print(pivotSoma([2,2]))`	`-1`	`-1`	✔
✔	`print(pivotSoma([7,-1,0,-1,1,1,2,3]))`	`2`	`2`	✔

Passed all tests! ✔

Correct

Marks for this submission: 3.00/3.00.

Question **2**

Correct

Mark 3.00 out of 3.00

Defina a função `minesweeper` que recebe uma lista de listas de ***strings*** que define uma grelha como a seguinte:

```
-  -  -  #  #
-  #  -  -  -
-  -  #  -  -
-  #  #  -  -
-  -  -  -  -
```

Cada `#` representa uma mina, e cada `-` representa um espaço vazio.

A função que devem implementar calcula, para cada espaço vazio, quantas minas lhe são adjacentes (na vertical, horizontal, e na diagonal).

A função deve devolver o resultado final neste formato:

```
1  1  2  #  #
1  #  3  3  2
2  4  #  2  0
1  #  #  2  0
1  2  2  1  0
```

For example:

Test	Result

`mostrarGrelha(minesweeper(grelha))`	`1  1  2  #  #`
	`1  #  3  3  2`
	`2  4  #  2  0`
	`1  #  #  2  0`
	`1  2  2  1  0`

`mostrarGrelha(minesweeper([['#','-','-']]))`	`#  1  0`

Answer: (penalty regime: 0 %)

Reset answer

```
grelha = [
  ["-", "-", "-", "#", "#"],
  ["-", "#", "-", "-", "-"],
  ["-", "-", "#", "-", "-"],
  ["-", "#", "#", "-", "-"],
  ["-", "-", "-", "-", "-"] ]

def mostrarGrelha(grelha):
  print('\n'.join([''.join(['{:3}'.format(str(item)) for item in linha])
                   for linha in grelha]))


def inc(grelha, l, c, comp, alt):
    indices = [
        (l-1, c-1), (l-1, c), (l-1, c+1),
        (l, c-1)  ,           (l, c+1)  ,
        (l+1, c-1), (l+1, c), (l+1, c+1) ]
```

	Test	Expected	Got	

✔	`mostrarGrelha(minesweeper(grelha))`	`1  1  2  #  #`
		`1  #  3  3  2`
		`2  4  #  2  0`
		`1  #  #  2  0`
		`1  2  2  1  0`
			`1  1  2  #  #`
			`1  #  3  3  2`
			`2  4  #  2  0`
			`1  #  #  2  0`
			`1  2  2  1  0`
				✔

✔	`mostrarGrelha(minesweeper([['#','-','-']]))`	`#  1  0`	`#  1  0`	✔

Passed all tests! ✔

Correct

Marks for this submission: 3.00/3.00.

Question **3**

Correct

Mark 6.00 out of 6.00

Defina a função `contarAscendente` que recebe uma ***string*** com dígitos, e verifica se podemos interpretar esses dígitos como uma sequência crescente de valores inteiros.

Por exemplo, a ***string*** "50515253" pode ser interpretada como a sequência crescente 50, 51, 52, 53. Logo, neste caso, a função `contarAscendente` deve devolver o resultado `True`.

Já a ***string*** "5051525" não pode ser interpretada desta forma, logo o resultado seria `False`.

For example:

Test	Result
`print(contarAscendente("50515253"))`	`True`
`print(contarAscendente("501502503"))`	`True`
`print(contarAscendente("8910"))`	`True`
`print(contarAscendente("9899100101"))`	`True`
`print(contarAscendente("899"))`	`False`
`print(contarAscendente("50150250"))`	`False`
`print(contarAscendente("989910010110"))`	`False`

Answer: (penalty regime: 0 %)

Reset answer

```
s = "50515253"  # exemplo de string

def contarAscendente(s):
    compTot = len(s)
    comp = 1
    prox = int(s[0]) + 1
    i = 1
    res = False
    while i < compTot:
        lenAtu = len(str(prox))
        atu = int(s[i:i+lenAtu])

        if atu == prox:
            prox = atu + 1
            comp = lenAtu
            i += comp
            res = True
```

	Test	Expected	Got	
✔	`print(contarAscendente("50515253"))`	`True`	`True`	✔
✔	`print(contarAscendente("501502503"))`	`True`	`True`	✔
✔	`print(contarAscendente("8910"))`	`True`	`True`	✔
✔	`print(contarAscendente("9899100101"))`	`True`	`True`	✔
✔	`print(contarAscendente("899"))`	`False`	`False`	✔
✔	`print(contarAscendente("50150250"))`	`False`	`False`	✔
✔	`print(contarAscendente("989910010110"))`	`False`	`False`	✔

Passed all tests! ✔

Correct

Marks for this submission: 6.00/6.00.

« PREVIOUS ACTIVITY
TP & Lab - Ficheiros e Módulos

NEXT ACTIVITY
Comentário Avaliação Contínua 6 »