Started on	Saturday, 28 November 2020, 10:00 PM
State	Finished
Completed on	Saturday, 28 November 2020, 11:27 PM
Time taken	1 hour 27 mins
Grade	**12.00** out of 12.00 (**100**%)

Question **1**

Correct

Mark 2.00 out of 2.00

Defina a função `contarDigitos` que recebe um inteiro **n** e devolve o número de dígitos que contém.

A função não deve ser recursiva, devendo resolver o exercício com o uso de ciclos.

For example:

Test	Result
`print(contarDigitos(123))`	3
`print(contarDigitos(-1000))`	4

Answer: (penalty regime: 0 %)

Reset answer

```
def contarDigitos(n):
    if n < 0:
        n *= -1
    elif n == 0:
        return 1

    cnt = 0
    while n > 0:
        cnt += 1
        n //= 10

    return cnt
```

	Test	Expected	Got	
✔	`print(contarDigitos(123))`	3	3	✔
✔	`print(contarDigitos(-1000))`	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 2.00/2.00.

Question **2**

Correct

Mark 2.00 out of 2.00

Defina a função `idxMaxPar` que recebe uma lista de inteiros positivos e devolve o índice onde se encontra o maior número par da lista. Se a lista não contiver números pares, a função deve devolver -1.

For example:

Test	Result
`print(idxMaxPar([3, 7, 2, 1, 7, 9, 10, 13]))`	`6`
`print(idxMaxPar([1, 3, 5, 7]))`	`-1`

Answer: (penalty regime: 0 %)

Reset answer

```
def idxMaxPar(lista):
  maior = -1

  for idx in range(len(lista)):
    if lista[idx] % 2 == 0 and (maior == -1 or lista[idx] > lista[maior]):
        maior = idx

  return maior
```

	Test	Expected	Got	
✔	`print(idxMaxPar([3, 7, 2, 1, 7, 9, 10, 13]))`	`6`	`6`	✔
✔	`print(idxMaxPar([1, 3, 5, 7]))`	`-1`	`-1`	✔

Passed all tests! ✔

Correct

Marks for this submission: 2.00/2.00.

Question **3**

Correct

Mark 2.00 out of 2.00

Defina a função `neutralizarSinais` que recebe duas ***strings*** `s1` e `s2` de igual tamanho compostas por símbolos '+-'. A função deve devolver uma ***string*** do mesmo tamanho onde o i-ésimo caracter deve ser:

- '+' se s1[i] e s2[i] forem ambas '+'
- '-' se s1[i] e s2[i] forem ambas '-'
- '0' caso contrário (ou seja, os sinais opostos neutralizam-se)

For example:

Test	Result
`print(neutralizarSinais("+-+", "+--"))`	`+-0`
`print(neutralizarSinais("--++--", "++--++"))`	`000000`

Answer: (penalty regime: 0 %)

Reset answer

```
def neutralizarSinais(s1, s2):
    nova = ''
    for idx in range(len(s1)):
        nova += s1[idx] if s1[idx] == s2[idx] else '0'
    return nova
```

	Test	Expected	Got	
✔	`print(neutralizarSinais("+-+", "+--"))`	`+-0`	`+-0`	✔
✔	`print(neutralizarSinais("--++--", "++--++"))`	`000000`	`000000`	✔

Passed all tests! ✔

Correct

Marks for this submission: 2.00/2.00.

Question **4**

Correct

Mark 2.00 out of 2.00

Defina a função `somaAcc` que recebe uma lista de inteiros e devolve a lista com as somas acumuladas de ir juntando mais um elemento de cada vez, e pela ordem inicial dada.

Por exemplo, `somaAcc(1,2,3,4)` deve retornar a lista `[1,1+2,1+2+3,1+2+3+4]=[1,3,6,10]`.

For example:

Test	Result
`print(somaAcc([1,2,3,4]))`	`[1, 3, 6, 10]`
`print(somaAcc([]))`	`[]`
`print(somaAcc([4,3,2,1]))`	`[4, 7, 9, 10]`

Answer: (penalty regime: 0 %)

Reset answer

```
def somaAcc(lista):
    comp = len(lista)
    if comp == 0:
        return []

    nova = [lista[0]]
    for idx in range(1, comp):
      nova += [nova[idx-1] + lista[idx]]

    return nova
```

	Test	Expected	Got	
✔	`print(somaAcc([1,2,3,4]))`	`[1, 3, 6, 10]`	`[1, 3, 6, 10]`	✔
✔	`print(somaAcc([]))`	`[]`	`[]`	✔
✔	`print(somaAcc([4,3,2,1]))`	`[4, 7, 9, 10]`	`[4, 7, 9, 10]`	✔

Passed all tests! ✔

Correct

Marks for this submission: 2.00/2.00.

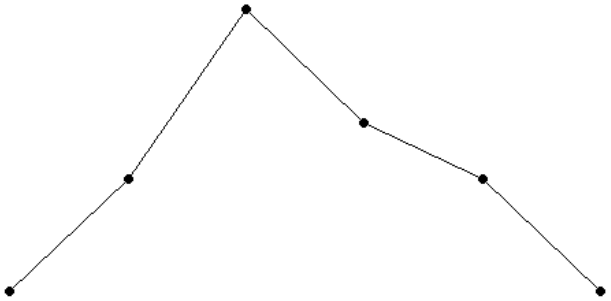
Question **5**

Correct

Mark 4.00 out of 4.00

Defina a função `classificar` que recebe uma ***string*** com a descrição de uma paisagem. Esta paisagem é composta por inteiros positivos que identificam a altura de certos picos de montanha ou vales.

Por exemplo, a lista `[1, 3, 6, 4, 3, 1]` corresponde à seguinte paisagem:



O objetivo é classificar uma dada paisagem, a função deve devolver a ***string*** de valor:

- 'montanha' se houver apenas um pico (o pico não pode ocorrer numa das extremidades)
- 'vale' se houver apenas uma depressão (a depressão não pode ocorrer numa das extremidades)
- 'nenhum' se a paisagem não for um dos casos acima

No caso da lista `[1, 3, 6, 4, 3, 1]` o resultado esperado é a ***string*** 'montanha'

For example:

Test	Result
`print( classificar([3, 4, 5, 4, 3]) )`	`montanha`
`print( classificar([9, 7, 3, 1, 2, 4]) )`	`vale`
`print( classificar([9, 7, 9, 4]) )`	`nenhum`

Answer: (penalty regime: 0 %)

Reset answer

```
def classificar(paisagem):
    pic = 0
    dep = 0

    for idx in range(1, len(paisagem)-1):
        ant = paisagem[idx-1]
        atu = paisagem[idx]
        pos = paisagem[idx+1]

        if atu > ant and atu > pos:
            pic += 1
        elif atu < ant and atu < pos:
            dep += 1

    if pic == 1 and dep == 0:
        return 'montanha'
    elif dep == 1 and pic == 0:
```

	Test	Expected	Got	
✔	`print( classificar([3, 4, 5, 4, 3]) )`	`montanha`	`montanha`	✔
✔	`print( classificar([9, 7, 3, 1, 2, 4]) )`	`vale`	`vale`	✔
✔	`print( classificar([9, 7, 9, 4]) )`	`nenhum`	`nenhum`	✔

Passed all tests! ✔

Correct

Marks for this submission: 4.00/4.00.

« PREVIOUS ACTIVITY
Comentário Avaliação Contínua 4

NEXT ACTIVITY
Projeto I »