Lista_01_Solucoes

August 27, 2020

1 Lista de Exercícios 1 (soluções propostas)

Exercícios retirados do livro: The Python Workbook - Ben Stephenson.

Não é um gabarito fechado. As soluções encontradas aqui são apenas algumas possibilidades. Usem a criatividade!

1.0.1 QUESTÃO 1:

Vamos calcular a soma dos n primeiros inteiros positivos? Podemos resolver isso de várias formas, mas por agora, a tarefa é fazer um programa que recebe um inteiro positivo n e, em seguida, calcula a soma de todos os inteiros positivos de 1 a n.

Lembram da fórmula?

$$S_n = \frac{n(n+1)}{2}$$

Inicialmente, vamos resolver por meio do `input()` e do `print()`:

[1]:
```
n=int(input('Digite um número inteiro: '))

soma=(n*(n+1))/2

print('A soma dos ' + str(n) + ' primeiros inteiros positivos é ' + str(soma))
```

```
Digite um número inteiro: 10
A soma dos 10 primeiros inteiros positivos é 55.0
```

Agora, usando a ideia de funções:

[2]:
```
def somaN(x):
    return (x*(x+1))/2
```

[3]:
```
somaN(10)
```

[3]:
```
55.0
```

Nesse caso, foi definida uma função chamada `somaN(x)`, que recebe como **parâmetro de entrada** o mesmo número n que o usuário digitaria no `input()`. Aqui, no lugar de printar o resultado, fazemos o `retunrn` e chamamos a função com o valor do x desejado.

1.0.2 QUESTÃO 2:

Uma forma de calcular a área de um triângulo é por meio de uma fórmula que depende do comprimento de todos os seus lados: s_1, s_2 e s_3. Seja

$$s = \frac{(s_1 + s_2 + s_3)}{2}$$

, podemos calcular a área da seguinte forma:

$$A = \sqrt{s \times (s - s_1) \times (s - s_2) \times (s - s_3)}$$

Crie um programa que receba os comprimentos de todos os lados de um triângulo e exiba a sua área.

Condição de existência de um triângulo (ou desigualdade traingular): para construirmos um triângulos é necessário que a medida de qualquer um dos lados seja menor que a soma das medidas dos outros dois e maior que o valor absoluto da diferença entre essas medidas.

Dado um triângulo cujos lados são a, b e c, esse triângulo somente existirá se:

$$\begin{aligned} a + b &< c \\ a + c &< b \\ b + c &< a \end{aligned}$$

Esse conjunto de inequações é conhecido como **desigualdade triangular**.

[4]:
```
import math
```

Inicialmente, vamos resolver por meio do `input()` e do `print()`:

[5]:
```
s1=float(input("Digite o lado s1 do triângulo: "))
s2=float(input("Digite o lado s2 do triângulo: "))
s3=float(input("Digite o lado s3 do triângulo: "))

s=(s1+s2+s3)/2
A=math.sqrt(s*(s-s1)*(s-s2)*(s-s3))

print("A área do triângulo é {} unidades quadradas.".format(A))
```

```
Digite o lado s1 do triângulo: 10
Digite o lado s2 do triângulo: 20
Digite o lado s3 do triângulo: 30
A área do triângulo é 0.0 unidades quadradas.
```

Mas você pode inserir números que não formem um triângulo. Não seria legal criar uma função para testar se os números passados, de fato, correspondem aos lados de um triângulo? Vamos definir a função `triangulo(a,b,c)`:

[6]:
```
def triangulo(a,b,c):
    if a<b+c and b<a+c and c<a+b:
        tri=True
    else:
        tri=False
    return tri
```

[7]:
```
triangulo(s1,s2,s3)
```

[7]:
```
False
```

[8]:
```
def areaTriangulo(s1,s2,s3):
    if triangulo(s1,s2,s3)==False:
        print('ERRO! Os números não correspondem aos lados de um triângulo!')
        return
    else:
        s=(s1+s2+s3)/2
    return math.sqrt(s*(s-s1)*(s-s2)*(s-s3))
```

[9]:
```
areaTriangulo(s1,s2,s3)
```

```
ERRO! Os números não correspondem aos lados de um triângulo!
```

[10]:
```
areaTriangulo(16,21,30)
```

[10]:
```
160.15129565507735
```

1.0.3 QUESTÃO 3:

Crie um programa que receba um número inteiro de 4 dígitos e exiba a soma desses dígitos. Por exemplo, se o número for 2020, seu programa deverá retornar $2 + 0 + 2 + 0 = 4$.

Inicialmente, vamos resolver por meio do `input()` e do `print()`:

[13]:
```
num=input("Digite um número de 4 dígitos: ")

soma=int(num[0])
soma+=int(num[1]) #soma=soma+algo
soma+=int(num[2])
soma+=int(num[3])

print("A soma dos 4 dígitos do número " + num + " é {}.".format(soma))
```

```
Digite um número de 4 dígitos: 2020
A soma dos 4 dígitos do número 2020 é 4.
```

Agora, usando a ideia de funções:

[14]:
```
def somaDigitos(num):
    soma=0
    for i in str(num):
        soma=soma+int(i)
    return soma
```

[15]:
```
somaDigitos(2020)
```

[15]: 4

[16]:
```
def somaDigitos2(num):
    lista=[int(i) for i in str(num)]
    return sum(lista)
```

[17]:
```
somaDigitos2(2020)
```

[17]: 4

1.0.4 QUESTÃO 4:

Crie um programa que recebe três números inteiros e os exiba em ordem crescente.

Dica: as funções `max()` e `min()` poderão ser úteis!

Inicialmente, vamos resolver por meio do `input()` e do `print()`:

[18]:
```
n1=int(input("Digite o primeiro número: "))
n2=int(input("Digite o segundo número: "))
n3=int(input("Digite o terceiro número: "))

menor=min(n1, n2, n3)
maior=max(n1, n2, n3)
medio=(n1+n2+n3) - menor - maior

print("Os valores em ordem crescente são: {}, {}, {}.".format(menor, medio,␣
↪maior))
```

```
Digite o primeiro número: 123
Digite o segundo número: 12
Digite o terceiro número: 190
Os valores em ordem crescente são: 12, 123, 190.
```

Agora, usando a ideia de função e de condicionais:

[19]:
```
def ordem(n1, n2, n3):
    if n1>n2:
        if n1>n3: #n1 é maior que os dois outros
            terceiro=n1
            if n2>n3: #n2 é segundo maior e n3 é o menor
```

```
                segundo=n2
                primeiro=n3
            else: #n3 é segundo maior e n2 é o menor
                segundo=n3
                primeiro=n2
        else: #n3 é o maior de todos
            terceiro=n3
            segundo=n1 #n1 é segundo maior e n2 é o menor
            primeiro=n2
    else:
        if n2>n3: #n2 é o maior de todos
            terceiro=n2
            if n1>n3: #n1 é o segundo maior e n3 é o menor
                segundo=n1
                primeiro=n3
            else: #n3 é o segundo maior e n1 é o menor
                segundo=n3
                primeiro=n1
        else: #n3 é o maior de todos
            terceiro=n3
            segundo=n2 #n2 é segundo maior e n1 é o menor
            primeiro=n1
    return (primeiro, segundo, terceiro)
```

[20]:
```
ordem(3, 5, 1)
```

[20]:
```
(1, 3, 5)
```

1.0.5 QUESTÃO 5:

É vogal ou consoante? A ideia é você criar um programa que classifica uma letra do alfabeto passada pelo usuário e devolve uma menssagem informando a sua classificação.

Inicialmente, vamos resolver por meio do `input()` e do `print()`:

[21]:
```
letra=input('Digite uma letra do alfabeto: ').lower()

if letra=='a' or letra=='e' or letra=='i' or letra=='o' or letra=='u':
    print('A letra {} é uma volgal!'.format(letra))
else:
    print('A letra {} é uma consoante!'.format(letra))
```

```
Digite uma letra do alfabeto: W
A letra w é uma consoante!
```

[22]:
```
def classificaLetra(letra):
    vogais=['a', 'e', 'i', 'o', 'u']
    if letra.lower() in vogais:
```

```
        return 'vogal'
    else:
        return 'consoante'
```

Pensar em um `return` que seja útil caso essa função seja utilizada como auxiliar de uma outra função. Retornar textos grandes não é prático para realizar comparações. Calma! Iremos trabalhar mais esse conceito.

[23]:
```
classificaLetra('W')
```

[23]:
```
'consoante'
```

No lugar de criar uma função que classifica uma letra, poderia ter sido criada uma função que testa se dada letra é uma vogal:

[24]:
```
def vogal(letra):
    vogais='aeiou'
    if letra.lower() in vogais:
        return True
    else:
        return False
```

[25]:
```
vogal('A')
```

[25]:
```
True
```

Perceba que as letras que pertencem ao conjunto de vogais foram definidas de três formas diferentes:
* strings separadas

```
if letra=='a' or letra=='e' or letra=='i' or letra=='o' or letra=='u':
```

- strings que formam uma palavra maior (que também é uma string)

```
vogais='aeiou'
if letra.lower() in vogais:
```

- uma lista (ainda iremos explorar esse tipo de dados)

```
vogais=['a', 'e', 'i', 'o', 'u']
if letra.lower() in vogais:
```

[26]:
```
vogais='aeiou'
list(vogais)
```

[26]:
```
['a', 'e', 'i', 'o', 'u']
```

1.0.6 QUESTÃO 6:

Quantos dias tem um determinado mês? Sabemos que a duração de um mês pode variar de 28 a 31 dias. A ideia é criar um programa que, de acordo com o nome do mês que receber, exibe uma

menssagem informando a quantidade de dias. Lembrando que fevereiro pode variar entre “28 ou 29 dias”.

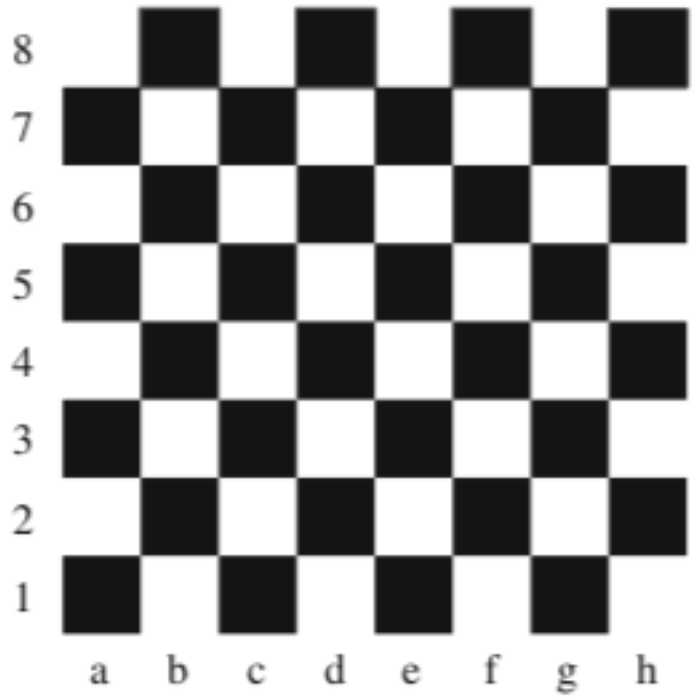
Vamos resolver por meio do `input()` e do `print()`:

[42]:
```
meses=['janeiro','fevereiro','março','abril','maio','junho','julho','agosto','setembro','outub
mes=input("Digite um mês: ").lower()

erro=True
while erro:
    if mes not in meses:
        print('ERRO! Vocês não digitou um mês válido!')
        mes = input("Digite um mês: ").lower()
    else:
        erro=False


if mes == 'abril' or mes == 'junho' or mes == 'setembro' or mes == 'novembro':
    dias=30
elif mes == "fevereiro":
    dias="28 ou 29"
else:
    dias=31

print(mes.title(), 'tem', dias, 'dias!')
```

```
Digite um mês: jan
ERRO! Vocês não digitou um mês válido!
Digite um mês: janeiro
Janeiro tem 31 dias!
```

1.0.7 QUESTÃO 7:

As posições em um tabuleiro de xadrez são identificadas por uma **letra** (coluna) e um **número** (linha), conforme mostrado abaixo:



Desenvolva um programa que receba uma posição e determina a cor do quadrado. Por exemplo, para a posição **a1**, o seu programa deverá informar que o quadrado é **preto**; para a posição **d5**, deverá informar que o quadrado é **branco**.

[44]:
```
gridPos = input("Digite uma posição do tabuleiro de xadrez: ")

col=gridPos[0].lower()
lin=int(gridPos[1])

# cor de início da coluna
if col in "aceg":
    colQuadradoPreto=True
else:
    colQuadradoPreto=False

if colQuadradoPreto:
    if lin%2==0:
        quadradoBranco=True
    else:
        quadradoBranco=False
else:
    if lin%2==0:
        quadradoBranco=False
    else:
        quadradoBranco=True
```

```
if quadradoBranco:
    print('A posição', gridPos, 'do tabuleiro possui quadrado Branco!')
else:
    print('A posição', gridPos, 'do tabuleiro possui quadrado Preto!')
```

```
Digite uma posição do tabuleiro de xadrez: d5
A posição d5 do tabuleiro possui quadrado Branco!
```

1.0.8 QUESTÃO 8:

Crie um programa que recebe um **mês** e um **dia** e, em seguida, de acordo com as classificações da tabela, determina a estação do ano correspondente.

Estação	Data de Início
Outono	20 de março
Inverno	20 de junho
Primavera	22 de setembro
Verão	21 de dezembro

[45]:
```
def estacao(dia, mes):
    mes=mes.lower()

    if mes=='janeiro' or mes=='fevereiro':
        estacao='Verão'

    elif mes=='marco':
        if dia<20:
            estacao='Verão'
        else:
            estacao='Outono'

    elif mes=='abril' or mes=='maio':
        estacao='Outono'

    elif mes=='junho':
        if dia<20:
            estacao='Outono'
        else:
            estacao='Inverno'

    elif mes=='julho' or mes=='agosto':
        estacao='Inverno'

    elif mes=='setembro':
        if dia<22:
            estacao='Inverno'
```

```
        else:
            estacao='Primavera'

    elif mes=='outubro' or mes=='novembro':
        estacao='Primavera'

    elif mes=='dezembro':
        if dia<21:
            estacao='Primavera'
        else:
            estacao='Verão'

    return estacao
```

[46]:
```
estacao(11, 'junho')
```

[46]:
```
'Outono'
```

1.0.9 QUESTÃO 9:

As regras para determinar se um ano é ou não um bissexto são as seguintes:

- Qualquer ano divisível por 400 é um ano bissexto.
- Dos anos restantes, qualquer ano divisível por 100 não é um ano bissexto.
- Dos anos restantes, qualquer ano divisível por 4 é um ano bissexto.
- Todos os outros anos não são anos bissextos.

Escreva um programa que a partir de um ano informado, determina se o ano é ou não bissexto.

[47]:
```
def bissexto(ano):
    if ano%400==0:
        anoBissexto=True
    elif ano%100==0:
        anoBissexto=True
    elif ano%4==0:
        anoBissexto=True
    else:
        anoBissexto=False

    return anoBissexto
```

[48]:
```
bissexto(2020)
```

[48]:
```
True
```

[49]:
```
# Lista com os últimas anos bissextos
for i in range(1991,2021):
    if bissexto(i):
        print(i)
```

```
1992
1996
2000
2004
2008
2012
2016
2020
```

1.0.10 QUESTÃO 10:

Um **palíndromo** é uma palavra, frase ou qualquer outra sequência de unidades que tenha a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita. Alguns exemplos: 'aibofobia', 'luz azul', 'Hanah' e '20/02/2002'.

Escreva um programa que recebe uma string e determina se é um palíndromo.

[50]:
```
def palindromo(texto):
    if texto==texto[::-1]:
        return True
    else:
        return False
```

[51]:
```
palindromo('aibofobia')
```

[51]: True

[52]:
```
palindromo('Hanah')
```

[52]: False

[53]:
```
def palindromo2(texto):
    texto=texto.lower()
    if texto==texto[::-1]:
        return True
    else:
        return False
```

[54]:
```
palindromo2('Hanah')
```

[54]: True

[55]:
```
palindromo2('luz azul')
```

[55]: False

[56]:
```
x1='luz azul'
x2='20/02/2002'
x1=x1.replace(' ','')
x2=x2.replace(' ','')
print(x1, x2)
x1=x1.replace('/','')
x2=x2.replace('/','')
print(x1, x2)
```

```
luzazul 20/02/2002
luzazul 20022002
```

[57]:
```
def palindromo3(texto):
    texto=texto.lower()
    texto=texto.replace(' ','')
    texto=texto.replace('/','')
    if texto==texto[::-1]:
        return True
    else:
        return False
```

[58]:
```
palindromo3('luz azul')
```

[58]: True

[59]:
```
palindromo3('20/02/2002')
```

[59]: True