Truques de programação II - Python

Nesta folha você deve resolver alguns exercícios de programação. Cada um deles sugere um truque que quando apreendido pode ser usado em inúmeros outros problemas parecidos ou não.

Para seu processamento, você deve ler o arquivo

`F184001.myd`

publicado no lugar usual.

Números similares Faça um programa para ler 2 números naturais de 3 algarismos e verificar se eles são similares. Para serem similares estes números precisam ter os mesmos algarismos na sua formação.

```
Exemplo de execução:
Digite 2 números: 231 312
Números similares

Digite 2 números: 452 357
Números não similares

Outro exemplo de execução:
Digite 2 números: 167 167
Números similares
```

O truque: quebrar um número inteiro qualquer em seus dígitos componentes e a seguir processar tais dígitos isoladamente é tarefa comum em programação. Primeiro, a quantidade de dígitos é obtida pela expressão $ceil(log_{10}(x))$ que só não funciona em números cheios: 1 seguido de n zeros. A separação de dígitos pode ser feita, rodando-se sucessivamente o trecho de programação (em Python)

```
x é o número que se quer quebrar
digitos[n] é um vetor de dígitos
dig=[]
x=...
while x>0:
  dig.append(x%10)
  x=x//10
dig.reverse()
print(dig)
```

Este trecho só não funciona para o número 0 que deve ser tratado à parte. Para este problema em particular, como se sabe que os números terão 3 dígitos, a conversão pode ser mais direta:

```
def f184():
 f=open("c:/p/n/184/f184001_exemplo.myd","r")
 i=0
 ct=0
 while i<500:
  x=f.readline()
  nums=x.split()
  a=int(nums[0])
  b=int(nums[1])
  d0=a//100
  d2=a%10
  d1=(a//10)%10
  e0=b//100
  e2=b%10
  e1=(b//10)%10
  if (((d0==e0) or (d0==e1) or (d0==e2)) and
   ((d1==e0) or (d1==e1) or (d1==e2)) and
   ((d2==e0) or (d2==e1) or (d2==e2))):
    if (((e0==d0) or (e0==d1) or (e0==d2)) and
     ((e1==d0) or (e1==d1) or (e1==d2)) and
     ((e2==d0) or (e2==d1) or (e2==d2))):
       ct=ct+1
  i=i+1
 print(ct)
```

Para você fazer Leia no arquivo acima descrito, 500 sequencias de 2 números de 3 dígitos e informe quantas dessas sequencias são formadas por números similares.

números similares:

Brinquedos inteligentes A empresa tem muitos brinquedos pequenos sempre guardados em caixas no formato de cubos (paralelepípedos retos com arestas iguais). A empresa comprou muitas esferas de raios 10, 20 e 30 cm. Pretende encerrar cada brinquedo dentro de uma esfera (como se fosse um kinder ovo). Você deve escrever um programa que peça e receba a aresta do cubo de um brinquedo e deve descobrir qual a menor esfera que pode escondê-lo, imprimindo o raio da esfera. Se o brinquedo não couber nem na maior esfera, o programa deve imprimir NAO CABE. O programa deve processar uma quantidade indeterminada de brinquedos, parando ao receber uma aresta negativa, que não deve ser processada e indica final dos dados. Dados: $V_{cubo} = a^3$, $V_{esfera} = 4/3\pi \times r^3$, Diagonal do cubo: $a \times \sqrt{3}$, Diâmetro da esfera: $2 \times r$

```
Dados de uma execucao real:
Informe a aresta do brinquedo 10
esfera de 10cm de raio
Informe a aresta do brinquedo 12
esfera de 20cm de raio
Informe a aresta do brinquedo 20
esfera de 20cm de raio
Informe a aresta do brinquedo 30
esfera de 30cm de raio
Informe a aresta do brinquedo -1
```

Um possível gabarito

```
a10=a20=a30=anao=0
i=0
p=[]
while i<500:
  x=f.readline()
  nums=x.split()
  a=int(nums[0])
  b=int(nums[1])
  p.append(a)
  p.append(b)
  i=i+1
for i in range(1000):
  diagonal=p[i]*np.sqrt(3)
  if diagonal<=10:
    a10=a10+1
  if diagonal>10 and diagonal <= 20:
    a20=a20+1
  if diagonal>20 and diagonal <=30:
    a30=a30+1
  if diagonal > 30:
    anao=anao+1
print("10=",a10,"20=",a20,"30=",a30,"~=",anao)
```

O truque: Processar objetos com base em suas características geométricas. Aqui, tem-se um cubo inscrito a uma esfera. Seus pontos de contato são os pares de vértices opostos do cubo, que neste ponto coincidem com um diâmetro da esfera.

No arquivo acima, há 1000 brinquedos representados por suas arestas. Processe-os e descubra quantos brinquedos cabem na esferas de 10,20 e 30 cm além dos que não cabem.

10cm	20cm	30cm	não cabem

distância cartesiana Escreva um programa que leia as coordenadas cartesianas (x,y) de 2 pontos, A e B no espaço e imprima na tela a menor distância entre os 2 pontos e também a distância do menor caminho de A até B passando por um ponto intermediário, C com coordenadas (0,0). Deve ser definida e usada a função de nome dist() que retorna a distância entre 2 pontos, cujas coordenadas são passadas como parâmetros. Obs.: Considere que a distância entre 2 pontos de coordenadas (x_a, y_a) e (x_b, y_b) é dada por $distAB = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$ A distância do caminho de A até B passando por C = (0,0) é dado por caminho AB = dist AC + dist CB .

```
Exemplo de execução:
Digite coordenada de A: 10 50
Digite coordenada de B: 70 35
Menor distância entre A e B: 61.8466
Caminho de A a B, passando por (0,0): 129.253
```

O truque: Trata-se de associar a objetos do mundo real, suas coordenadas em algum sistema cartesiano. Com a popularização do Google maps e sistemas similares, bem como o barateamento dos dispositivos GPS, é questão de tempo que praticamente todos os itens que entrem em algum meio magnético (computacional) tragam com eles suas coordenadas. Daí a precisar calcular distâncias é um passo bem pequeno.

```
import numpy as np
def dcart(p1,p2):
  return (((p1[0]-p2[0])**2)+
      ((p1[1]-p2[1])**2))**0.5
...
  i=0
  dab=dacb=0
  while i<1000:
    x=f.readline()
    nums=x.split()
    p1=[int(nums[0]),int(nums[1])]
    p2=[int(nums[2]),int(nums[3])]
    dab=dab+dcart(p1,p2)
    dacb=dacb+dcart(p1,[0,0])+dcart([0,0],p2)
    i=i+1
  print("D1 = ",dab,"  D2=",dacb)}
```

No arquivo há 1000 pares de pontos. As menores distâncias entre eles devem ser somadas, bem como os caminhos de A a B passando por C.

$\sum A-B$	$\sum A-B-C$

☞ Para você fazer

similares	10cm	20cm
30cm	+30cm	$\sum$AB
$\sum$ACB	/////// /////// /////// /////// ///////	/////// /////// /////// /////// ///////

```
===== 04/12/2019 10:46:27.4 =====E=PL184p
    1    7     167  171  209  453     50793.7   78532.7
```