_____ / _____ / _____

Truques de programação I - Python

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

☞ Para você fazer

Os seus dados estão em um arquivo publicado no lugar de sempre com o nome de

`F183001.myd`

Peso de um trem Um trem é composto por muitos vagões conectados. Neste exercício você é um despachante de trens. Uma composição pode ter dezenas de vagões e locomotivas e o trem precisa passar por pontes e viadutos que tem limites rígidos de peso suportado. Cada vagão tem um certo peso e a composição precisa ser pensada para que num determinado instante, a soma de 5 ou menos carros adjacentes (5 é um número fixo, e ele corresponde ao comprimento da maior ponte na malha ferroviária) não ultrapasse o limite de peso das estruturas pelas quais o trem passará. Os últimos valores do vetor geralmente correspondem ao peso das locomotivas, e elas devem ser tratadas como se vagões fossem. Do ponto de vista deste problema, não há diferença entre vagão e locomotiva, ambos pesam.

Nos limites inicial e final do trem, a soma do peso deve levar em consideração este fato. Assim, se apenas a locomotiva está sobre a ponte, apenas o peso dela deve ser considerado. À medida em que mais vagões vão entrando, a soma deve ser incrementada, até o limite de 5 parcelas, pois quando o sexto vagão entrar, o primeiro terá saído da ponte e deve ser desconsiderado.

Você deve escrever um programa que leia um vetor representando os pesos de cada vagão e locomotivas de um trem e um limite de peso máximo suportado naquele trajeto e deve certificar se aquele trem pode fazer ou não essa viagem.

Havendo ultrapassagem do limite proposto, o programa deve sinalizar qual o vagão central (o terceiro do conjunto de 5) que ocasionou este fato. Para esta sinalização, pode numerar os vagões a partir do zero, tal como o C++ faz.

Se ao final o trem estiver liberado o programa deve mandar a mensagem "Liberado"ou senão a mensagem "O trem nao pode passar".

A seguir, dois exemplos de execução: Para um trem formado por: 100, 3, 3, 3, 2, 5 e 200 e com um limite de 101 o programa deve informar ter ultrapassado o limite nos vagões 0, 1, 2, 4, 5 e 6, e portanto "o trem não pode passar".

Um segundo exemplo é: 10, 20, 30, 40, 50, 40, 30, 20, 10 e 160, com limite de 200. Os vagões 7 e 8 impedem a passagem. Finalmente o mesmo trem com limite de 261, permite a passagem.

No arquivo acima, há 100 trens cada um com 20 vagões/locomotivas. A 21^a coluna do arquivo é o limite de peso que as pontes aguentam. Processe os 100 trens e descubra quantos estão liberados para a viagem.

liberados

Truque: janela deslizante Trata-se de uma janela que percorre os dados e em cada momento faz algo com os dados que estão sob a janela naquele instante. A idéia é muito usada em processamento digital de imagens, para filtros, melhoradores, etc. Aqui a janela é unidimensional correspondendo a 5 vagões. As dificuldades do truque são representadas pelo tratamento dos limites ou das bordas. Uma possibilidade é acrescentar dois zeros antes e depois do vetor e começar a análise pela terceira posição, encerrando-a na antepenúltima.

```
def f183():
  f=open("c:/p/n/183/f183001_exemplo.myd","r")
  i=0
  liberado=0
  while(i<100):
    x=f.readline()
    trem = x.split()
    tr=[]
    for j in trem:
      tr.append(int(j))
    # processa tr (que é uma lista de inteiros)
    k=2
    passou=0
    while (k<18):
      if (tr[k-2]+tr[k-1]+tr[k]+tr[k+1]+tr[k+2])
                        <=tr[20]:
        passou=passou+1
      k=k+1
    if passou==16:
      liberado=liberado+1
    i=i+1
  print("Trens liberados: ",liberado)
```

Porto e Containeres Como se sabe o comércio mundial teve um enorme incremento com o surgimento da idéia de container. Trata-se de uma caixa de aço que é embalada na origem, manipulada em diversos modais (caminhão, trem e navio) de forma ágil e desembalada apenas no destino.

Imagine um megaporto com um enorme pátio de containeres. Para nossa facilidade o pátio está dividido em ruas e avenidas e entre elas existem quarteirões onde os containeres são armazenados. Nesta simulação, o pátio é representado por uma matriz retangular. Nela, cada célula contém o número de containeres que existem naquele quarteirão no pátio. Por exemplo, se na matriz, na linha 8 coluna 5 existe o valor 88, isto significa que no quarteirão da avenida 8 com a rua 5 existem 88 containeres guardados.

Escreva um programa que leia essa matriz de M linhas por N colunas (M e N definidas externamente) chamada PORTO contendo a disposição atual dos containeres no pátio e determine o a quantidade e localização do maior e do menor quarteirões e também a distância em linha reta (medida em quarteirões) entre eles. Por característica do problema nunca haverá dois ou mais valores iguais na matriz.

```
Por exemplo,
18 17 22 31 12   -> maior:90 L1, C4
16  8  2  4 90   -> menor: 1 L1, C2
32 33 34 35 36   -> dist.: 2,82
40 41  1 11 15

 1  2  3  4  5  6  -> mai: 24 L3, C5
 7  8  9 10 11 12  -> men: 1 L0, C0
13 14 15 16 17 18  -> d: 5,83
19 20 21 22 23 24
```

Truque: Teorema de Pitágoras Usado aqui para calcular a distância entre dois pontos. Basta criar um triângulo retângulo no qual a hipotenusa é o valor da distância a descobrir. Agora o problema se transfere para descobrir o valor dos catetos do triângulo. Neste caso é fácil pelo uso de um sistemas de coordenadas ortogonais.

```
i=0
while (i<4):
  patio=np.zeros((22,22),int)
  lin=0
  while lin<22:
    x=f.readline()
    pa=x.split()
    p=[]
    for j in pa:
      p.append(int(j))
    xixa=0
    while xixa<22:
      patio[lin][xixa]=p[xixa]
      xixa=xixa+1
    lin=lin+1
  c1x=np.argmax(patio)//22
  c1y=np.argmax(patio)%22
  c2x=np.argmin(patio)//22
  c2y=np.argmin(patio)%22
  print("Max-Min: ",i+1,
  (((c1x-c2x)**2)+((c1y-c2y)**2))**0.5)
  i=i+1
```

No arquivo acima há 4 configurações de pátios, cada uma com 22 x 22 containeres. Em cada uma você deve descobrir qual a distância entre o ponto de máximo e o de mínimo em cada pátio.

1	2	3	4

Avaliação de uma mão de poker Como se sabe no jogo de poker, as possibilidades de pontuação são

⋆ straight flush (5 cartas seguidas do mesmo naipe)
⋆ quadra (4 cartas de mesmo valor)
⋆ full house (3 cartas iguais entre si e outras 2 iguais entre si)
⋆ flush (5 cartas de mesmo naipe)
⋆ sequencia: (5 cartas seguidas)
⋆ trinca (3 cartas iguais)
⋆ jogos menores: 2 pares, 1 par, maior carta...

A próxima discussão é a respeito da representação das cartas. Inúmeras possibilidades ocorrem aqui, mas vai-se usar a seguinte: O naipe da carta é representado pela centena: 100=ouros, 200=espadas, 300=copas e 400=paus. A carta será representada pelas dezenas: 01=ás, 02 a 10=a própria carta, 11=dama, 12=valete e 13=rei. Por exemplo, a carta 408 é um 8 de paus, 113 é um rei de ouros e 201 é o ás de espadas.

Truque: usar e abusar de resto e de divisão inteira Usar-se-ão estas 2 operações para quebrar um número como 113 em 100 (113 *div* 100) e em 13 (113 *mod* 100).

```
def quadra(x):
  l=[0]*5
  for i in range(5):
    l[i]=x[i]%100
  l.sort()
  if ((l[0]==l[1] and l[1]==l[2] and l[2]==l[3])
   or (l[1]==l[2] and l[2]==l[3] and l[3]==l[4])):
    return 1
  else:
    return 0
def fullh(x):
  l=[0]*5
  for i in range(5):
    l[i]=x[i]%100
  l.sort()
  if ((l[0]==l[1] and l[1]==l[2] and l[3]==l[4])
   or (l[0]==l[1] and l[2]==l[3] and l[3]==l[4])):
    return 1
  else:
    return 0
def flush(x):
def seque(x):
def trinca(x):
...
  i=0 ; ctfl=ctqu=ctfh=ctse=cttr=0
  while i<200:
    x=f.readline()
    pa=x.split()
    p=[]
    for j in pa:
      p.append(int(j))
    for k in range(5):
      ctfl=ctfl+flush(p[(k*5):5+(k*5):])
      ctqu=ctqu+quadra(p[(k*5):5+(k*5):])
      ctfh=ctfh+fullh(p[(k*5):5+(k*5):])
      ctse=ctse+seque(p[(k*5):5+(k*5):])
      cttr=cttr+trinca(p[(k*5):5+(k*5):])
    i=i+1
  print("Flush = ",ctfl)
  print("Quadra = ",ctqu)
  print("Full hand = ",ctfh)
  print("Sequencia = ",ctse)
  print("Trinca = ",cttr)
```

No arquivo, há 200 rodadas, cada uma com 5 mãos de poker, totalizando 1000 mãos. Estude-as e descubra quantos flushes, quadras, full-hands, sequencias e trincas há.

fl	qu	fh	se	tr

```
===== 04/12/2019 10:42:06.3 =====E=PL183p
    1   60    10.2 18.0  3.2  5.1     0  1  2  4 27
```