Truques de programação III - 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

`F185001.myd`

publicado no lugar usual.

Números crescentes

Vamos chamar de crescente um número natural $n = d_1 d_2 ... d_k$ cujos dígitos d_i estão em ordem crescente, isto é, tal que $d_1 < d_2 < ... < d_k$. Faça um programa em C++ que leia um número inteiro e positivo n de 3 dígitos e verifique se n é crescente. Seu programa deve também verificar se n possui exatamente 3 dígitos e imprimir mensagens adequadas em cada caso.

```
Exemplo de execução:
Entre com um inteiro positivo:
3416
valor inválido.

Entre com um inteiro positivo: 152
152 não é crescente.

Entre com um inteiro positivo: 378
378 é crescente.
```

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. Neste exemplo, como a demanda é direta (3 dígitos) basta testar o número entre 101 e 999.

A separação de dígitos pode ser feita, rodando-se sucessivamente o trecho de programação (em C++)

```
digs=[] # conterá os dígitos
x=... # o valor a quebrar
while x>0:
   digs.append(x%10)
   x=x//10
digs.reverse()
print(digs)
```

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 f185():
  f=open("c:/p/n/185/f185001_exemplo.myd","r")
  i=0
  p=[]
  while i<50:
    x=f.readline()
    nums=x.split()
    for j in range(10):
      p.append(int(nums[j]))
    i=i+1
  ct=0
  for i in range(len(p)):
    x=p[i]
    dig=[]
    while x>0:
      dig.append(x%10)
      x=x//10
    sentinela=0
    for j in range(1,len(dig)):
      if dig[j]>=dig[j-1]:
        sentinela=1
    if sentinela==0:
      ct=ct+1
  print("Crescentes=",ct)
```

Para você fazer Leia no arquivo acima descrito, 500 números e indique quantos são crescentes

números crescentes:

Números perfeitos, deficientes e abundantes

Um inteiro n é dito um número perfeito se ele é igual a soma de todos os seus divisores positivos, excluindo ele mesmo. Por exemplo, 6 é um número perfeito, pois seus divisores são 1, 2 e 3 e 1+2+3 = 6. O número 28 também é perfeito, pois 1+2+4+7+14=28. Faça um programa em C++ que leia uma sequência de números inteiros $n_1, n_2, ..., n_k$ e informe ao usuário, para cada n_i lido, se ele é um número perfeito ou não. Aproveitando o desenvolvimento, se um número é menor que a soma dos divisores próprios ele é deficiente e se é maior ele é abundante. (por exemplo, 12 é abundante pois: $1+2+3+4+6 = 16 > 12$). Note que a origem dos índices é zero.

```
Exemplo de execução:
Entre com um inteiro: 2
2 não é um número perfeito.
Entre com um inteiro: 28
28 é um número perfeito.
Entre com um inteiro: 79
79 não é um número perfeito.
Entre com um inteiro: 8128
8128 é um número perfeito.
```

```
import numpy as np
def divs(x):
  soma=1
  for i in range(2,1+x//2):
    if x%i==0:
      soma=soma+i
  return soma
...
  i=0
  p=[]
  while i<50:
    x=f.readline()
    nums=x.split()
    for j in range(10):
      p.append(int(nums[j]))
    i=i+1
  cper=cabu=cdef=0
  for i in range(len(p)):
    xx=divs(p[i])
    if xx==p[i]:
      cper=cper+1
    elif xx>p[i]:
      cabu=cabu+1
    else:
      cdef=cdef+1
  print("Perf=",cper," Abu=",cabu," Def=",cdef)
```

O truque: Achar os divisores de um número inteiro. Note-se que se está no âmbito da matemática inteira, razão da divisão inteira e da operação resto.

No arquivo acima, há 500 valores e você deve contar quantos são perfeitos, abundantes e deficientes

perfeitos	abundantes	deficientes

Localização de mercado

Uma empresa de supermercados pretende abrir uma filial em alguma região de uma cidade absolutamente regular, já que todos os seus quarteirões são quadrados de 100x100m e as ruas sempre se interceptam em ângulos retos.

Seu programa deve ler uma matriz quadrada de ordem N (N é definido via #define no começo do programa) que contém a quantidade de habitantes da cidade em cada quarteirão. Assim, se a posição 2,2 da matriz tem 50, significa que no quarteirao da linha 3, coluna 3 há 50 clientes. Note que a numeração começa em 0, assim 2 significa linha ou coluna 3.

Deve ler também 3 valores: as coordenadas x,y da possível localização do supermercado (começando em 0) e também um parâmetro denominado aa = área de abrangência do mercado. aa é sempre um número ímpar, isto não precisa ser testado. x,y nunca vai estar nos limites da cidade, assim sempre haverá espaço para acomodar a submatriz de dimensão $aa \times aa$.

O programa deve totalizar quantos clientes serão atingidos pelo mercado naquela localização e com aquela área de abrangência, somando todas as células da matriz que estejam incluídas na submatriz quadrada centralizada em x, y e com dimensões aa.

Deve imprimir também a quantidade de clientes atingidos no maior quarteirão (aquele que tem mais clientes).

Ao final a matriz deve ser impressa.

```
Exemplos de execução:
Seja a matriz (N=6)
 1 1 3 0 5 3
 7 2 9 1 1 2
 3 4 5 6 0 6
 1 2 1 2 3 4
 5 2 2 8 9 3
 1 2 3 3 5 3
com x=3, y=3 e aa=3, R=36 e 9 respectivamente.
com x=2, y=2 e aa=5, R=83 e 9.
com x=1, y=4 e aa=1, R=1 e 1.
com x=1, y=4 e aa=3, R=24 e 6.
```

```
...
  i=tudo=tudomax=0
  while i<50:
    m=np.zeros((10,10))
    for j in range(10):
      x=f.readline()
      nums=x.split()
      for k in range(10):
        m[j,k]=int(nums[k])
    x=f.readline()
    nums=x.split()
    lin=int(nums[0])
    col=int(nums[1])
    tam=int(nums[2])
    tt=tam//2
    somsom=0
    maximo=-999999
    for ii in range(lin-tt,lin+tt+1):
      for jj in range(col-tt,col+tt+1):
        somsom=somsom+m[ii,jj]
        if m[ii,jj]>maximo:
          maximo=m[ii,jj]
    tudo=tudo+somsom
    tudomax=tudomax+maximo
    i=i+1
  print(tudo,tudomax)
```

No arquivo há 50 matrizes de 10×10. Após cada cidade, há 3 números que são as coordenadas da localização do mercado e também a sua abrangência. Calcule a soma das 10 somas de clientes e a soma dos 10 maiores quarteiros em cada caso.

$\sum clientes$	$\sum maiores$

Leitura de dados

Para ler os dados de entrada de um arquivo (ao invés do teclado) em Python deve-se: i. abrir o arquivo, ii. ler seus registros (via readline()) iii. Transformar a linha lida em um vetor de caracteres (via x.split(), iv. Transformar cada número desses em um inteiro, colocando-o em uma lista (via append.int(nums[k])). Veja no código acima como isso foi feito.

☞ Para você fazer

crescentes	perfeitos	abundantes
deficientes	$\sum$ clientes	$\sum$ maiores

```
===== 04/12/2019 10:47:50.2 =====E=PL185p
    1       4      13  206  281      604424   47469
```