www.estudar.com.vc

Lista de Exercícios

P2 2018

Programação em Python Poli

USP

1. Tema

Simule o código abaixo e selecione as opções correspondentes à saída impressa do programa:

```
def f(L,e,d):
    c = (e+d)//2
    print(L[c+1])
    g(L,e+3,c+2)

def g(L,e,d):
    c = (e+d)//2
    for i in range(c, d):
        print(L[i])

def main():
    A = [14,42,35,87,65,23]
    f(A,0,5)
    g(A,1,3)

main()
```

a. O primeiro número impresso é:

A. 65 **B.** 23 **C.** 42 **D.** 35 **E.** 87 **F.** 14

b. O segundo número impresso é:

A. 14 **B.** 65 **C.** 87 **D.** 35 **E.** 42 **F.** 23

c. O terceiro número impresso é:

A. 42 **B.** 87 **C.** 23 **D.** N/A **E.** 35 **F.** 14 **G. 65**

d. O quarto número impresso é:

A. N/A **B.** 87 **C.** 65 **D.** 23 **E.** 35 **F.** 42 **G.** 14

2. Tema

Escolha os códigos que calculam a soma dos quatro vizinhos (superior, inferior, esquerdo e direito) de um dado elemento na linha y e coluna x de uma matriz A. O valor do elemento central não deve ser incluído na soma. Assuma que $0 < x < len(A[0]) - 1$ e $0 < y < len(A) - 1$. Mais de uma alternativa pode estar correta.

A.

```
soma = 0
D = [[0,-1],[0,1],[-1,0],[1,0]]
for i in range(len(A)):
    for j in range(len(A[0])):
        if [i-y,j-x] in D:
            soma += A[i][j]
```

B.

```
soma = 0
for dy in [-1, 0, 1]:
```

```
    for dx in [-1, 0, 1]:
        if dy+dx not in [-1, 1]:
            soma += A[y+dy][x+dx]
```

C.

```
soma = A[y][x+1] + A[y][x-1]
soma += A[y-1][x] + A[y+1][x]
```

D.

```
soma = 0
for dy in range(-1,1):
    for dx in range(-1,1):
        if dy*dx == 0 and dy != dx:
            soma += A[y+dy][x+dx]
```

E.

```
soma = 0
dx = dy = [ 0, 1, 0, -1]
for k in range(len(dx)):
    j = x + dx[k]
    i = y + dy[k+1]
    soma += A[i][j]
```

F.

```
soma = 0
for i in range(y-1,y+2):
    for j in range(x-1,x+2):
        soma += A[i][j]
```

G.

```
soma = 0
for dy in range(-1,2):
    for dx in range(-1,2):
        if dy*dx == 0 and dy != dx:
            soma += A[y+dy][x+dx]
```

H.

```
soma = 0
D = [[0,-1],[0,1],[-1,0],[1,0]]
for L in D:
    x += L[0]
    y += L[1]
    soma += A[y][x]
```

I.

```
soma = 0
dx = dy = [0, 1, 0,-1]
for k in range(-1,len(dx)-1):
    j = x + dx[k]
    i = y + dy[k+1]
    soma += A[i][j]
```

J.

```
soma = 0
D = [[0,-1],[0,1],[-1,0],[1,0]]
for L in D:
    x += L[0]
    y += L[1]
    soma += A[x][y]
```

K.

```
soma = -A[y][x]
for i in range(y-1,y+2):
    for j in range(x-1,x+2):
        if i == x or j == y:
            soma += A[i][j]
```

L.

```
soma = 0
for dy in [-1, 0, 1]:
    for dx in [-1, 0, 1]:
        if dy+dx == 1 or dy+dx == -1:
            soma += A[y+dy][x+dx}
```

M.

```
soma = 0
for i in range(y-1,y+2):
    for j in range(x-1,x+2):
        if i*j == 0 and i != j:
            soma += A[i][j]
```

N.

```
soma = 0
D = [[0,-1],[0,1],[-1,0],[1,0]]
for k in range(len(D)):
    x += D[k][0]
    y += D[k][1]
    soma += A[y][x]
```

O.

```
soma = 0
dx = [ 0, 1, 0,-1]
dy = [-1, 0, 1, 0]
for k in range(len(dx)):
    j = x + dx[k]
    i = y + dy[k]
    soma += A[i][j]
```

P.

```
soma = 0
for i in range(y-1,y+2):
    for j in range(x-1,x+2):
        if i == x or j == y:
            soma += A[i][j]
```

3. TEMA

Elaboração própria

Seja A uma matriz de dimensões $p\ x\ q$ e B uma matriz de dimensões $r\ x\ s$. Então, o produto de *Kronecker* $A \otimes B$ é uma matriz de dimensões $pr\ x\ qs$ da forma:

$$\begin{pmatrix}
a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1s} & \cdots & \cdots & a_{1q}b_{11} & a_{1q}b_{12} & \cdots & a_{1q}b_{1s} \\
a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2s} & \cdots & \cdots & a_{1q}b_{21} & a_{1q}b_{22} & \cdots & a_{1q}b_{2s} \\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\
a_{11}b_{r1} & a_{11}b_{r2} & \cdots & a_{11}b_{rs} & \cdots & \cdots & a_{1q}b_{r1} & a_{1q}b_{r2} & \cdots & a_{1q}b_{rs} \\
\vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\
\vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\
a_{p1}b_{11} & a_{p1}b_{12} & \cdots & a_{p1}b_{1s} & \cdots & \cdots & a_{pq}b_{11} & a_{pq}b_{12} & \cdots & a_{pq}b_{1s} \\
a_{p1}b_{21} & a_{p1}b_{22} & \cdots & a_{p1}b_{1s} & \cdots & \cdots & a_{pq}b_{21} & a_{pq}b_{22} & \cdots & a_{pq}b_{2s} \\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\
a_{p1}b_{r1} & a_{p1}b_{r1} & \cdots & a_{p1}b_{rs} & \cdots & \cdots & a_{pq}b_{r1} & a_{pq}b_{r2} & \cdots & a_{pq}b_{rs}
\end{pmatrix}$$

Abaixo está um exemplo:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix}
1 \cdot 0 & 1 \cdot 5 & 2 \cdot 0 & 2 \cdot 5 \\
1 \cdot 6 & 1 \cdot 7 & 2 \cdot 6 & 2 \cdot 7 \\
3 \cdot 0 & 3 \cdot 5 & 4 \cdot 0 & 4 \cdot 5 \\
3 \cdot 6 & 3 \cdot 7 & 4 \cdot 6 & 4 \cdot 7
\end{pmatrix}$$

$$= \begin{pmatrix}
0 & 5 & 0 & 10 \\
6 & 7 & 12 & 14 \\
0 & 15 & 0 & 20 \\
18 & 21 & 24 & 28
\end{pmatrix}$$

Faça uma função que recebe duas matrizes reais A e B e devolve o produto de *Kronecker* $A \otimes B$ na matriz C.

```
def Kronecker(A,B):
    p,q = len(A), len(A[0])
    r,s = len(B), len(B[0])
    C = []
    L1
        linha = []
```

```
        L2
            linha.append(0)
        C.append(linha)
    L3
        L4
            L5
            L6
                L7
                L8
                    L9
                    L10
                L11
    return C
```

Para cada um dos 11 itens a seguir, correspondendo às lacunas no código acima, assinale a única resposta que torna o programa acima correto.

L1:

A. $for\ i\ in\ range(q + s)$:

B. $for\ i\ in\ range(p * s)$:

C. $for\ i\ in\ range(q * s)$:

D. $for\ i\ in\ range(p * r)$:

E. $for\ i\ in\ range(p + r)$:

L2:

A. $for\ j\ in\ range(p * r)$:

B. $for\ i\ in\ range(q + s)$:

C. $for\ i\ in\ range(p + r)$:

D. $for\ i\ in\ range(q * s)$:

E. $for\ i\ in\ range(p * s)$:

L3:

A. $for\ i\ in\ range(p + q)$:

B. $for\ i\ in\ range(r)$:

C. $for\ i\ in\ range(p * q)$:

D. $for\ i\ in\ range(q * s)$:

E. $for\ i\ in\ range(p)$:

L4:

A. $for\ i\ in\ range(p + q)$:

B. $for\ i\ in\ range(q * s)$:

C. $for\ i\ in\ range(p * q)$:

D. $for\ i\ in\ range(q)$:

E. $for\ i\ in\ range(s)$:

L5:

A. $lin = i * s$

B. $lin = i * r$

C. $col = i * p$

D. $lin = i * q$

E. $lin = i * p$

F. $col = i * q$

L6:

A. $for\ k\ in\ range(s)$:

B. $for\ k\ in\ range(p * q)$:

C. $for\ k\ in\ range(p * s)$:

D. $for\ k\ in\ range(r)$:
E. $for\ k\ in\ range(r * s)$:

L7:
A. $col = j * p$
B. $col = j * s$
C. $col = j * q$
D. $col = j + s$
E. $col = j * r$

L8:
A. $for\ l\ in\ range(p)$:
B. $for\ l\ in\ range(r * s)$:
C. $for\ l\ in\ range(q)$:
D. $for\ l\ in\ range(p * q)$:
E. $for\ l\ in\ range(s)$:

L9:
A. $C[lin][col] = A[i][j] * B[i][j]$
B. $C[lin][col] = A[i][j] * B[k][l]$
C. $C[lin][col] = A[i][l] * B[k][j]$
D. $C[lin][col] = A[k][l] * B[k][l]$
E. $C[lin][col] = A[k][l] * B[i][j]$

L10:
A. $lin = l + 1$
B. $col\ = lin + 1$
C. $col\ += 1$

D. $lin \mathrel{+}= 1$
E. $lin = col + 1$

L11:
A. $lin \mathrel{+}= 1$
B. $lin = col + 1$
C. $col \mathrel{+}= 1$
D. $col = l + 1$
E. $col = lin + 1$

4. TEMA

Preencha as lacunas no código abaixo (L1 até L10), de forma a obter um programa que recebe um *string* contendo exclusivamente palavras separadas por espaços em branco e determina a palavra mais frequente e sua frequência relativa de ocorrência de texto.

Por exemplo, para "figurinha no pacote de figurinha", a saída do programa é:

```
palavra: figurinha freq. rel.: 0.4
```

```
def separa (texto):
    L = L1
    p = '
    for c in texto:
        if c == ' ':
            L.append(p)
            p = ''
```

```
            else:
                p += c
        L2
        return L

def maxfreq (texto):
        L = L3
        pf = []
        f = L4
        for p in L:
            L5
                pf.append(p)
                f.append(1)
            L6
                for i in range(len(pf)):
                    if p == pf[i]:
                        L7
        maxpf = ''
        maxf = 0
        for i in range(len(pf)):
            if f[i] > maxf:
                L8
                L9
        Return L10

def main ():
        texto = input('Digite as palavras: ')
        p, f = maxfreq(texto)
        print ('palavra:', p, 'freq. rel.:', f)
```

```
main()
```

Para cada um dos 10 itens a seguir, correspondendo às lacunas no código acima, assinale a única resposta que torna o programa acima correto.

L1:

A. $[texto]$

B. ''

C. $texto$

D. 0

E. []

L2:

A. $L \mathrel{+}= 1$

B. $L \mathrel{+}= c$

C. $L = [L,\ p]$

D. $L = [L]$

E. $L.append(p)$

L3:

A. $[separa(texto)]$

B. $texto[i]$

C. $texto$

D. $separa(texto)$

E. []

L4:

A. ''

B. []
C. 0
D. [0]
E. L

L5:
A. $if\ p[i] == pf[i]:$
B. $while\ p\ in\ pf:$
C. $if\ p\ != f:$
D. $if\ p\ != pf:$
E. $if\ not\ p\ in\ pf:$

L6:
A. $else:$
B. $if\ f == 0:$
C. $elif\ p\ not\ in\ pf:$
D. $if\ not\ f\ in\ pf:$
E. $if\ pf[i] == f[i]:$

L7:
A. $f[i] += 1$
B. $f.append(f[i] + 1)$
C. $f[i] = 2$
D. $pf[i] = f[i]$
E. $pf.append(f)$

L8:
A. $maxf = f[i]$

B. $maxf = 0$
C. $maxf = f[len(f) - 1]$
D. $maxf = len(pf[i])$
E. $maxf = i$

L9:

A. $maxpf = separa(pf[i])[1]$
B. $maxpf = str(maxf)$
C. $maxpf = pf[i]$
D. $maxpf = ''$
E. $maxpf = separa(texto)[i]$

L10:

A. $maxf, len(L)$
B. $maxpf, maxf$
C. pf, f
D. $maxf/len(pf)\ , maxpf$
E. $maxpf, maxf/len(L)$

Gabarito

1. Alternativa D.

2. Alternativa B.

3. Alternativa A.

4. Alternativa I.