Resolução dos Exercícios da Aula 1 (Listas)

1. **Regressiva**: Faça uma função que crie e retorne uma lista com todos os números pares de 1 a 100, porém na ordem regressiva.

```
1  def pares():
2      i = 100
3      l = []
4
5      while i >= 1:
6          if i%2 == 0:
7              l.append(i)
8          i = i-1
9
10     return l
11
12 def pares2():
13     i = 100
14     l = []
15
16     while i >= 2:
17         l.append(i)
18         i = i-2
19     return l
20
21 print(pares())
22 print(pares2())
```

2. **Metade**: Faça um procedimento que leia 10 números digitados pelo usuário, armazene a metade de cada um deles numa lista, e depois imprima esta lista.

```
1  def metade():
2      l = []
3
4      for i in range(10):
5          n = float(input("Digite um numero: "))
6          l.append(n/2)
7
8      for n in l:
9          print(n)
10
11 metade()
```

3. **Leitura**: Dado um número n, faça uma função que leia n números inteiros, e retorne uma lista com esses números.

```
def n_numeros(n):
    i = 0
    l = []

    while i < n:
        x = int(input("Digite um numero: "))
        l.append(x)
        i += 1

    return l

n = int(input("Digite a quantidade de numeros: "))
print(n_numeros(n))
```

4. **Ocorrências**: Dada uma lista e um elemento, retorne o número de ocorrências do elemento na lista.

```
def ocorrencias(x, l):
    n = 0

    for elem in l:
        if elem == x:
            n += 1

    return n

l = [1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 4, 3, 3, 3, 2, 1]
print(ocorrencias(3, l))
```

5. **Máximo**: Dada uma lista de números, faça uma função que encontre e retorne o maior deles.

```
def maior(l):
    maior = l[0]

    for elem in l:
        if elem > maior:
            maior = elem

    return maior

```

```
10 l = [1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 4, 3, 3, 3, 2, 1]
11 print(maior(l))
```

6. **Posição do Máximo**: Dada uma lista de números, faça uma função que encontre e retorne o índice do maior deles.

```
1  def posMaior(l):
2      p = 0
3
4      for i in range(len(l)):
5          if l[i] > l[p]:
6              p = i
7
8      return p
9
10 l = [1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 4, 3, 3, 3, 2, 1]
11 print(posMaior(l))
```

7. **Inverter**: Dada uma lista, faça um procedimento que inverta a ordem de seus elementos.

```
1  def troca(l, i, j):
2      aux = l[i]
3      l[i] = l[j]
4      l[j] = aux
5
6  def inverte(l):
7      posFinal = len(l)-1
8
9      for i in range(len(l)//2):
10         troca(l, i, posFinal-i)
11
12 l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
13 print(l)
14 inverte(l)
15 print(l)
```

Resolução dos Exercícios da Aula 2 (Listas)

8. **Fibonacci**: Dado um número n, retorne uma lista com os n primeiros elementos da sequência de Fibonacci. Obs.: Cada elemento da sequência é obtido através da soma dos dois elementos anteriores:

$$S = \{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots\}$$

```
1  def fib(n):
2      if n == 1: return [1]
3      elif n == 2: return [1,1]
4      else:
5          l = [1, 1]
6
7          for i in range(n-2):
8              l.append(l[-1]+l[-2])
9
10         return l
11
12 print(fib(10))
```

9. **Ordenadas e abscissas**: Defina um procedimento que receba duas listas com a mesma quantidade de números inteiros. A primeira lista contém as abscissas de um conjunto de pontos, e a segunda contém as ordenadas desses mesmos pontos. Calcule o número a de abscissas que são pares e o número b de ordenadas que são ímpares. Se $a \geq b$, imprima a soma de todas as abscissas. Caso contrário, imprima o produto de todas as ordenadas.

10. **k Múltiplos**: Dadas dois números k e n como parâmetros, retorne uma lista com todos os k primeiros múltiplos de n.

Resolução dos Exercícios da Aula 3 (Listas)

11. **Média**: Faça um procedimento que leia um número n e depois a notas de n alunos ($0 \leq n \leq 100$). Em seguida, calcule e imprima a média da turma, e o número de alunos que ficaram com nota acima de 60.

12. **Temperaturas**: Faça um procedimento que leia um número n e a temperatura de n dias do ano. Em seguida, calcule a média de temperatura anual e imprima o número de dias em que a temperatura ficou abaixo da média.

```
1  def temperaturas():
2      n = int(input("Digite n: "))
3      l = []
4
5      for i in range(n):
6          t = float(input("Digite uma temperatura: "))
```

```
7           l.append(t)
8
9       media = sum(l) / n
10      d = 0
11
12      for t in l:
13          if t < media:
14              d += 1
15
16      print(d, "dias abaixo da media.")
17
18  temperaturas()
```

13. **Iguais**: Dadas duas listas $l1$ e $l2$ com a mesma quantidade de números, imprima quantos elementos aparecem exatamente na mesma posição em ambas as listas.

```
1   def iguais(l1, l2):
2       cont = 0
3
4       for i in range(len(l1)):
5           if l1[i] == l2[i]:
6               cont += 1
7
8       print("Ha", cont, "elementos equivalentes.")
9
10  l1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
11  l2 = [1, 1, 3, 3, 3, 6, 6, 8, 9, 0]
12  iguais(l1, l2)
```

14. **Salários**: Dado um número n, faça um procedimento que leia o nome e o salário de n funcionários de uma empresa e imprima o nome de todos os funcionários que ganham mais que a média dos demais.

15. **Sublista**: Dada uma lista ordenada l e dois inteiros x e y ($x < y$), retorne uma sublista contendo todos os elementos de l que estiverem entre x e y.

16. **Troca de Cartas**: *[Maratona de Programação 2009]* Alice e Beatriz colecionam cartas de Pokémon. As cartas são produzidas para um jogo que reproduz a batalha introduzida em um dos mais bem sucedidos jogos de videogame da história, mas Alice e Beatriz são muito pequenas para jogar, e estão interessadas apenas nas cartas propriamente ditas. Para facilitar, vamos considerar que cada carta possui um identificador único, que é um número inteiro.

Cada uma das duas meninas possui um conjunto de cartas e, como a maioria das garotas de sua idade, gostam de trocar entre si as cartas que têm. Elas obviamente não têm interesse em trocar cartas idênticas, que ambas possuem, e não querem receber cartas repetidas na troca. Além disso, as cartas serão trocadas em uma única operação de troca: Alice dá para Beatriz um sub-conjunto com N cartas distintas e recebe de volta um outro sub-conjunto com N cartas distintas.

As meninas querem saber qual é o número máximo de cartas que podem ser trocadas. Por exemplo, se Alice tem o conjunto de cartas $[1, 1, 2, 3, 5, 7, 8, 8, 9, 15]$ e Beatriz o conjunto $[2, 2, 2, 3, 4, 6, 10, 11, 11]$, elas podem trocar entre si no máximo quatro cartas. Escreva uma função que receba como parâmetros a lista de cartas que Alice tem e a lista de cartas que Beatriz possui, e imprima o número máximo de cartas que podem ser trocadas. As cartas de Alice e Beatriz são apresentadas em ordem não decrescente. Exemplos:

Entrada		***Saída***
Cartas de Alice	***Cartas de Beatriz***	
$[1000]$	$[1000]$	0
$[1, 3, 5]$	$[2, 4, 6, 8]$	3
$[1, 1, 2, 3, 5, 7, 8, 8, 9, 15]$	$[2, 2, 2, 3, 4, 6, 10, 11, 11]$	4

```
1  def semRepeticao(l):
2      '''Dada uma lista ordenada l, retorna os elementos
3      de l sem repeticao.
4      Entrada: Lista
5      Saida: Lista'''
6      l2 = [l[0]]
7
8      for item in l:
9          if item != l2[-1]:
10             l2.append(item)
11
12     return l2
13
14 def troca(l1, l2):
15     '''
16     Entrada: Duas listas contendo o identificador das cartas
17     de cada jogador.
18     Saida: O numero de trocas que podem ser realizadas.
19     '''
20     l1 = semRepeticao(l1)
21     l2 = semRepeticao(l2)
```

```
22
23      trocasA = 0
24      for carta in l1:
25          if carta not in l2:
26              trocasA += 1
27
28      trocasB = 0
29      for carta in l2:
30          if carta not in l1:
31              trocasB += 1
32
33      return min(trocasA, trocasB)
34
35  print(troca([1,1,2,3,5,7,8,8,9,15], [2,2,2,3,4,6,10,11,11]))
```

Resolução dos Exercícios da Aula 4 (Matrizes)

17. **Criar Matriz**: Dados m e n, crie e retorne uma matriz $M_{M \times N}$ nula.

```
Resolvido nas notas de aula
```

18. **Imprimir Matriz**: Dada uma matriz M, imprima esta matriz na tela.

```
Resolvido nas notas de aula
```

19. **Somar Matrizes**: Dadas duas matrizes A e B de mesmo tamanho, retorne uma terceira matriz com o resultado de $A + B$.

```
Resolvido nas notas de aula
```

20. **Notas**: Um professor deseja calcular o média de notas de uma turma. Faça um procedimento que leia uma matriz contendo as notas dos alunos. O procedimento começa perguntado o número m de alunos e o número n de notas, e cria uma matriz $m \times n$ que armazena as n notas de cada um dos m alunos (os valores de m, n e das notas serão lidos do teclado). A nota final de cada aluno é a média simples das suas n notas. O procedimento deve imprimir a nota de cada aluno, e no final a média geral da turma. Exemplo:

```
Aluno 1: 10.0
Aluno 2: 8.0
Aluno 3: 6.0
Média da turna: 8.0
```

```
Resolvido nas notas de aula
```

21. **Matriz identidade**: Dada uma matriz, verifique se ela é uma matriz identidade:

```
Resolvido nas notas de aula
```

22. **Determinante**: Dada uma matriz $M_{3\times3}$, calcule o determinante de M:

```
Resolvido nas notas de aula
```

23. **Triangular inferior da transposta de uma matriz**: Defina um procedimento que receba uma matriz quadrada M como parâmetro. A função deve transformar a matriz M na matriz transposta de M, em seguida transformar essa matriz transposta em uma matriz triangular inferior e, por fim, imprimir a matriz resultante. Observações: (1) A função não pode criar listas auxiliares. (2) Uma matriz é triangular inferior quando todos os seus elementos acima da diagonal principal são iguais a 0.

```
Resolvido nas notas de aula
```

24. **Cavalo**: Considere um jogo de xadrez onde peças são movimentadas em um tabuleiro dividido em 8 linhas e 8 colunas. Considere ainda os movimentos do cavalo: a partir de uma dada posição, conforme diagrama a seguir (onde cada possível movimento é designado por * e o cavalo é representado na célula em destaque). No esquema, o cavalo localizado na posição $(4, 3)$ pode fazer oito movimentos, sendo que um deles o levaria para a posição $(6, 4)$. Defina um procedimento que recebe uma matriz $M_{8\times8}$ como parâmetro, na qual todos os elementos são nulos exceto a posição em que o cavalo se encontra (representado pelo número 1). Encontre esta posição e imprima a quantidade de movimentos que este cavalo pode fazer, considerando que ele não pode se movimentar para uma posição fora do tabuleiro.

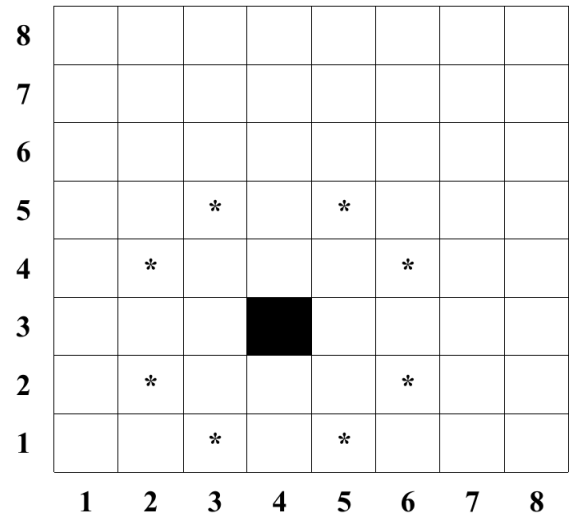


Figura 1: Movimentos possíveis do cavalo em um tabuleiro de xadrez.

```
1  def posCavalo(M):
2      ''' Achar a posicao do cavalo na matriz '''
3      for i in range(len(M)):
4          for j in range(len(M[0])):
5              if M[i][j] == 1:
6                  return i+1, j+1
7
8  def cavalo(M):
9      x, y = posCavalo(M)
10     movs = 0
11
12     if y == 8 or y == 1:
13         if x == 1 or x == 8: return 2
14         if x == 2 or x == 7: return 3
15         else: return 4
16     elif y == 2 or y == 7:
17         if x == 1 or x == 8: return 3
18         if x == 2 or x == 7: return 4
19         else: return 6
20     else:
21         if x == 1 or x == 8: return 4
22         if x == 2 or x == 7: return 6
23         else: return 8
```

25. **Robô Colecionador**: *[Maratona de Programação 2010]* Um dos esportes favoritos na Robolândia é o Rali dos Robôs. Este rali é praticado em uma arena retangular gigante de N linhas por M colunas de células quadradas. Algumas das células estão vazias, algumas contêm figurinhas da Copa (muito apreciadas pelas inteligências

artificiais da Robolândia) e algumas são ocupadas por pilastras que sustentam o teto da arena. Em seu percurso os robôs podem ocupar qualquer célula da arena, exceto as que contém pilastras, que bloqueiam o seu movimento. O percurso do robô na arena durante o rali é determinado por uma sequência de instruções. Cada instrução é representada por um dos seguintes caracteres: 'D', 'E' e 'F', significando, respectivamente, "gire 90 graus para a direita", "gire 90 graus para a esquerda" e "ande uma célula para a frente". O robô começa o rali em uma posição inicial na arena e segue fielmente a sequência de instruções dada (afinal, eles são robôs!). Sempre que o robô ocupa uma célula que contém uma figurinha da Copa ele a coleta. As figurinhas da Copa não são repostas, ou seja, cada figurinha pode ser coletada uma única vez. Quando um robô tenta andar para uma célula onde existe uma pilastra ele patina, permanecendo na célula onde estava, com a mesma orientação. O mesmo também acontece quando um robô tenta sair da arena. Dados o mapa da arena, descrevendo a posição de pilastras e figurinhas, e a sequência de instruções de um robô, você deve escrever um programa para imprimir o número de figurinhas coletadas pelo robô.

A entrada contém uma matriz representando a arena e uma lista com as instruções do robô. Cada elemento da matriz pode conter um dos seguintes caracteres:

- '.' - célula normal;
- '*' - célula que contém uma figurinha da Copa;
- '#' - célula que contém uma pilastra;
- 'N', 'S', 'L', 'O' - célula onde o robô inicia o percurso (única na arena). A letra representa a orientação inicial do robô (Norte, Sul, Leste e Oeste, respectivamente). A lista de entrada contém uma sequência de S caracteres dentre 'D', 'E' e 'F', representando as instruções do robô. Exemplos:
  - Para a arena a seguir,

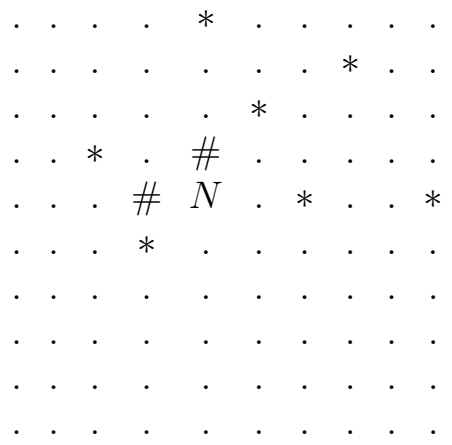
    $$\begin{matrix} * & * & * \\ * & N & * \\ * & * & * \end{matrix}$$

    e a sequência de instruções "DE", o robô coleciona 0 figurinhas.
  - Para a arena a seguir,

    $$\begin{matrix} . & . & . & \# \\ * & \# & O & . \\ * & . & * & . \\ * & . & \# & . \end{matrix}$$

    e a sequência de instruções "FFEFF", o robô coleciona 1 figurinha.

– Por fim, para a arena a seguir,



e a sequência de instruções “FDFFFFFFEEFFFFFFEFDF”, o robô coleciona 3 figurinha.