1 Exercícios - Dicionários

1. **Agenda**: Ver notas de aula
2. **Concurso**:

```
1  def classificados(candidatos, areas, codArea):
2      l = []
3      _, lista = areas[codArea]
4
5      for cod in lista:
6          nome, data, nota1, nota2 = candidatos[cod]
7
8          if nota1 >= 60:
9              l.append(cod)
10
11     return l
12
13 def imprimirClassificados(candidatos, areas, codArea):
14     l = classificados(candidatos, areas, codArea)
15
16     nomeArea, _ = areas[codArea]
17     print("Aprovados para a area {}:".format(nomeArea))
18     for cod in l:
19         nome, _, _, _ = candidatos[cod]
20         print("{} ({})".format(nome, cod))
21
22
23 def notaAprovado(candidatos, lista):
24     notaMaxima = 0
25
26     for cod in lista:
27         _, _, nota1, nota2 = candidatos[cod]
28
29         nota = nota1 + nota2
30
31         if nota > notaMaxima:
32             notaMaxima = nota
33
34     return notaMaxima
35
36 def maisVelho(data1, data2):
37     dia1, mes1, ano1 = data1
```

```
38      dia2, mes2, ano2 = data2
39
40      if ano1 < ano2: return True
41      if ano1 > ano2: return False
42      if mes1 < mes2: return True
43      if mes1 > mes2: return False
44      if dia1 < ano2: return True
45      return False
46
47  def aprovados(candidatos, areas):
48      print("Aprovados por area:")
49      for codArea in areas:
50          nomeArea, _ = areas[codArea]
51          lista = classificados(candidatos, areas, codArea)
52
53          maiorNota = notaAprovado(candidatos, lista)
54
55          codAprov = lista[0]
56          dataAprov = candidatos[codAprov][1]
57
58          for cod in lista:
59              nome, data, nota1, nota2 = candidatos[cod]
60
61              nota = nota1 + nota2
62              if nota == maiorNota and
63                 maisVelho(data, dataAprov):
64                  codAprov = cod
65                  dataAprov = data
66
67          nomeAprov = candidatos[codAprov][0]
68          print("{} - {} ({})".format(nomeArea, nomeAprov,
69                                       codAprov))
70
71  def main(argv=None):
72
73      candidatos = { "0001" : ("Joao", (28, 2, 1994), 71, 88),
74                     "0002" : ("Mara", (28, 2, 1993), 82, 77),
75                     "0003" : ("Jose", (2, 12, 2000), 59, 0),
76                     "0004" : ("Ana", (4, 12, 1981), 100, 40),
77                     "0005" : ("Beth", (14, 1, 1991), 60, 81)}
78
79      areas = { "001" : ("Fisica", ["0001", "0002", "0003"]),
```

```
                "002" : ("Artes", ["0004", "0005"]) }

    for area in areas:
        imprimirClassificados(candidatos, areas, area)

    aprovados(candidatos, areas)

if __name__ == '__main__':
    main()
```

3. Bolão - Dicionários:

```
import os
import random
import pickle

def limpaTela():
    if os.name == "nt":
        os.system("cls")
    else:
        os.system("clear")

def cadastrarJogador(jogadores):
    nome = input("Digite o nome do jogador: ")
    cpf = input("Digite o cpf do jogador: ")

    if cpf not in jogadores:
        jogadores[cpf] = nome
        print("Pessoa cadastrada com sucesso! :-)")
    else:
        print("Erro! Pessoa ja cadastrada no sistema.")

def visualizarJogadores(jogadores):
    if len(jogadores) == 0:
        print("Nenhuma pessoa cadastrada.")
    else:
        print("Pessoas cadastradas no sistema:")
        for cpf in jogadores:
            print("{} - CPF: {}".format(jogadores[cpf],cpf))

def lerNumApostadores(jogadores):
    qtd = int(input("Digite o numero de apostadores: "))

```

```
32      while qtd < 1 or qtd > len(jogadores):
33          print("Erro: Apenas", len(jogadores), "jogadores.")
34          qtd = int(input("Digite o numero de apostadores: "))
35
36      return qtd
37
38
39  def leituraCPFs(jogadores):
40      qtd = lerNumApostadores(jogadores)
41
42      cpfs = []
43      while len(cpfs) < qtd:
44          visualizarJogadores(jogadores)
45          cpf = input("Digite o CPF ou 0 para sair: ")
46
47          while cpf in cpfs:
48              print('Erro: CPF ja inserido neste bilhete.')
49              cpf = input("Tente outro CPF: ")
50          limpaTela()
51          if cpf == "0": return []
52
53          if cpf in jogadores:
54              cpfs.append(cpf)
55              print(jogadores[cpf], "cadastradx com sucesso")
56          else:
57              print("Erro! Pessoa nao cadastrada.")
58
59      return cpfs
60
61  def lerQtdNumeros():
62      n = int(input("Quantos numeros no bilhete? (6 a 15): "))
63
64      while n < 6 or n > 15:
65          n = int(input("Informe um numero de 6 a 15: "))
66
67      return n
68
69  def lerNumeros(n):
70      numeros = []
71      while len(numeros) < n:
72          num = int(input("Digite um numero apostado: "))
73
```

```
74          while num < 1 or num > 60 or num in numeros:
75              if num < 1 or num > 60:
76                  print("Este numero nao existe num cartao.")
77              else:
78                  print("Erro! Numero ja cadastrado.")
79                  print("Cadastrados ate agora:", numeros)
80
81              num = int(input("Digite um numero apostado: "))
82
83          numeros.append(num)
84      return numeros
85
86  def numerosDoBilhete():
87      n = lerQtdNumeros()
88
89      menu = '''Digite 1 para informar os numeros ou
90  outra tecla para que sejam escolhidos aleatoriamente: '''
91      opcao = input(menu)
92
93      if opcao == "1":
94          numeros = lerNumeros(n)
95      else:
96          numeros = random.sample(list(range(1,61)), n)
97
98      numeros.sort()
99      return numeros
100
101 def cadastrarAposta(jogadores, apostas):
102     cpfs = leituraCPFs(jogadores)
103     if cpfs != []:
104         numeros = numerosDoBilhete()
105         apostas.append( (cpfs, numeros) )
106         print("Aposta cadastrada com sucesso.")
107
108 def visualizarApostas(jogadores, apostas):
109     i = 1
110
111     limpaTela()
112     for cpfs, numeros in apostas:
113         print("\nAPOSTA", i)
114
115         print("Numeros:", end=" ")
```

```
116         for num in numeros:
117             print(num, end=" ")
118 
119         print("\nJogadores:")
120         for cpf in cpfs:
121             print(jogadores[cpf], "- CPF:", cpf)
122 
123         i = i+1
124 
125 def numerosSorteados():
126     l = []
127     while len(l) < 6:
128         x = int(input("Digite um dos numeros sorteados: "))
129         if x not in l and x > 0 and x <= 60:
130             l.append(x)
131         else:
132             print("Erro. Numero ja inserido ou invalido.")
133 
134     return l
135 
136 
137 def contida(l1, l2):
138     for elem in l1:
139         if elem not in l2:
140             return False
141 
142     return True
143 
144 def bilhetesPremiados(apostas, numeros):
145     l = []
146     for i in range(len(apostas)):
147         (_, nums) = apostas[i]
148         if contida(numeros, nums):
149             l.append(i)
150 
151     return l
152 
153 def listarVencedores(jogadores, apostas, posVencedores,
154                      premioPorBilhete):
155     limpaTela()
156     print("Vencedores:")
157 
```

```
158     for pos in posVencedores:
159         print( "BILHETE:", pos+1)
160         (cpfs, _) = apostas[pos]
161         for cpf in cpfs:
162             print(jogadores[cpf], "- CPF:", cpf, "- R$",
163                   premioPorBilhete/len(cpfs))
164 
165 def insereSorteio(jogadores, apostas):
166     nums = numerosSorteados()
167     premio = float(input("Digite o valor total do premio:"))
168 
169     ganhadores = bilhetesPremiados(apostas, nums)
170     nBilhetesVencedores = len(ganhadores)
171     premioPorBilhete = premio / nBilhetesVencedores
172 
173     if nBilhetesVencedores > 0:
174         listarVencedores(jogadores, apostas, ganhadores,
175                          premioPorBilhete)
176     else:
177         print("Nao houve vencedorxs.")
178 
179 def carregarDados():
180     if os.path.isfile("bolao.bin"):
181         with open("bolao.bin", "rb") as f:
182             jogadores = pickle.load(f)
183             apostas = pickle.load(f)
184     else:
185         jogadores = {}
186         apostas = []
187 
188     return jogadores, apostas
189 
190 def salvarDados(jogadores, apostas):
191     with open("bolao.bin", "wb") as f:
192         pickle.dump(jogadores, f)
193         pickle.dump(apostas, f)
194 
195 def main(args):
196 
197     print("Bem vindo")
198 
199     menu = '''
```

```
Escolha uma opcao:

1) Cadastrar novo jogador
2) Visualizar jogadores cadastrados
3) Cadastrar aposta
4) Visualizar apostas
5) Inserir sorteio e listar vencedores
0) Sair

'''

    jogadores, apostas = carregarDados()

    opcao = input(menu)
    while opcao != "0":
        limpaTela()
        if opcao == "1":
            cadastrarJogador(jogadores)
            salvarDados(jogadores, apostas)
        elif opcao == "2":
            visualizarJogadores(jogadores)
        elif opcao == "3":
            cadastrarAposta(jogadores, apostas)
            salvarDados(jogadores, apostas)
        elif opcao == "4":
            visualizarApostas(jogadores, apostas)
        elif opcao == "5":
            insereSorteio(jogadores, apostas)
        elif opcao != 0:
            print("Opcao invalida.")
        opcao = input(menu)

    print("Tchau!")


if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

4. **Multas**:

```
def anterior(d1, d2):
    dia1, mes1, ano1 = d1
```

```
3       dia2, mes2, ano2 = d2
4
5       if ano1 < ano2: return True
6       if ano1 > ano2: return False
7       if mes1 < mes2: return True
8       if mes1 > mes2: return False
9       if dia1 < dia2: return True
10      return False
11  def removerAntigas(infracoes, data):
12
13      dia, mes, ano = data
14      ano = ano-1
15
16      recentes = []
17      for infracao in infracoes:
18          if anterior( (dia, mes, ano), infracao[1] ):
19              recentes.append(infracao)
20
21      return recentes
22
23  def pontosCNH(cnh, infracoes, veiculos, motoristas,
24                    naturezas):
25      pontos = 0
26      for infracao in infracoes:
27          placa = infracao[2]
28          nat = infracao[3]
29          proprietario = veiculos[placa][0]
30
31          if proprietario == cnh:
32              pontos += naturezas[nat]
33
34      return pontos
35
36  def blitz(cnh, placa, dataAtual, infracoes, veiculos,
37                    motoristas, naturezas):
38      dataMotorista = motoristas[cnh][1]
39      pontosMotorista = pontosCNH(cnh, infracoes, veiculos,
40                    motoristas, naturezas)
41
42      if placa not in veiculos:
43          print("ATENCAO: Placa", placa ,"nao encontrada!")
44
```

```
45      elif anterior(dataMotorista, dataAtual):
46          print("ATENCAO: Motorista", motoristas[cnh][0],
47                      "com CNH vencida!")
48
49      elif pontosMotorista >= 20:
50          print("ATENCAO: Motorista", motoristas[cnh][0],
51                      "com", pontosMotorista, "pontos!")
52
53      else:
54          cnhDono, modelo, cor = veiculos[placa]
55          print("Modelo:", modelo)
56          print("Cor:", cor)
57          print("Proprietario:", motoristas[cnhDono][0])
58          print("Motorista:", motoristas[cnh][0])
59          print("Pontos do Motorista:", pontosMotorista)
60
61      print()
```

5. **Academia**:

```
1  def getGrupos(treinos, login):
2      grupos = []
3      for (_, _, _, g) in treinos[login]:
4          if g not in grupos:
5              grupos.append(g)
6      return grupos
7
8  def validate(exercicios, alunos, treinos):
9      login = input("Digite seu usuario: ")
10
11     if login not in alunos:
12         print("Usuario inexistente.")
13         return None
14     else:
15         senha = input("Digite seu usuario: ")
16         if senha != alunos[login][1]:
17             print("Senha incorreta.")
18             return None
19
20     return login
21
22 def auth(exercicios, alunos, treinos):
23     login = validate(exercicios, alunos, treinos)
```

```
24
25      if login is not None:
26          grupos = getGrupos(treinos, login)
27          grupo = input("Digite o grupo de hoje: ")
28          if grupo not in grupos:
29              print("Grupo inexistente.")
30          else:
31              treinoDoDia(exercicios, alunos, treinos, login,
32                          grupo)
33
34  def treinoDoDia(exercicios, alunos, treinos, login, grupo):
35      print("Aluno: {}".format(alunos[login][0]))
36      print("Grupo: {}\n".format(grupo))
37
38      for (cod, nSeries, nRep, g) in treinos[login]:
39          if g == grupo:
40              print("{} - {} de {}".format(exercicios[cod],
41                                           nSeries, nRep))
42      print("\n**************************************\n")
43
44  def naoFaz(exercicios, alunos, treinos, login):
45      faz = []
46      for (cod, _, _, _) in treinos[login]:
47          if cod not in faz:
48              faz.append(cod)
49
50      print("Exercicios que {} nao faz:".format(
51                                          alunos[login][0]))
52      for cod in exercicios:
53          if cod not in faz:
54              print(exercicios[cod])
55      print("\n**************************************\n")
```

6. **Produtos**

```
1   def emEstoque(p, clientes, produtos, pedidos):
2       qtd = 0
3
4       for codPed in pedidos:
5           codCli, entregue, lista = pedidos[codPed]
6           if not entregue:
7               for codProd, qtdProd in lista:
8                   if codProd == p:
```

```
9                       qtd += qtdProd
10
11      if qtd > produtos[p][2]:
12          print("Quantidade insuficiente de", produtos[p][1])
13
14
15  def imprimirPedido(p, clientes, produtos, pedidos):
16      codCli, entregue, lista = pedidos[p]
17      print("Pedido #" + str(p) + ":")
18      for codProd, qtdProd in lista:
19          valor = produtos[codProd][0]
20          nome = produtos[codProd][1]
21          qtd = produtos[codProd][2]
22          print("-", nome)
23          print("  Qtd:", qtdProd)
24          print(" Valor unitario: R$", valor)
25          print(" Valor total: R$", qtdProd*valor)
26          if entregue:
27              print("Status: Entregue")
28          else:
29              print("Status: Em aberto")
30
31
32  def totalPedido(p, clientes, produtos, pedidos):
33      codCli, entregue, lista = pedidos[p]
34      valorTotal = 0
35      for codProd, qtdProd in lista:
36          valor = produtos[codProd][0]
37          valorTotal += valor*qtdProd
38
39      return valorTotal
40
41
42  def totalPorCliente(clientes, produtos, pedidos):
43      for codCli in clientes:
44          totalCli = 0
45          for codPed in pedidos:
46              cliPed = pedidos[codPed][0]
47              if codCli == cliPed:
48                  totalCli += totalPedido(codPed, clientes,
49                                          produtos, pedidos)
50
```

```
51          print(clientes[codCli][0], totalCli)
52
53
54  def main(args):
55      clientes = {"123.456.789-00" :
56                      ("Maria da Silva", "maria@gmail.com"),
57                  "234.567.890-00" :
58                      ("Joao da Cruz", "joao@gmail.com"),
59                  "345.678.900-00" :
60                      ("Jose de Souza", "zezinho@gmail.com") }
61
62      produtos = {"A0001" : (1.20, "Pera", 1),
63                  "A0002" : (3.40, "Uva", 1),
64                  "A0003" : (1.00, "Maca", 1),
65                  "A0004" : (10.00, "Salada de frutas", 1),
66                  "A0005" : (12.00, "Acai medio", 1),
67                  "A0006" : (3.00, "Granola", 1),
68                  "A0007" : (5.00, "Suco 300ml", 1) }
69
70      pedidos = {"345" : ("123.456.789-00", True,
71                      [("A0001",3), ("A0002",1),
72                      ("A0003",5), ("A0004",1)]),
73                  "123" : ("234.567.890-00", False,
74                      [("A0005",2), ("A0006",1)]) }
75
76      for p in produtos:
77          emEstoque(p, clientes, produtos, pedidos)
78
79      for p in pedidos:
80          imprimirPedido(p, clientes, produtos, pedidos)
81
82      for p in pedidos:
83          print(totalPedido(p, clientes, produtos, pedidos))
84
85      totalPorCliente(clientes, produtos, pedidos)
86
87  if __name__ == '__main__':
88      import sys
89      sys.exit(main(sys.argv))
```

7. Voos

```
1  def questao1(x, voos):
2      cia, escalas = voos[x]       # acessando os dados do voo
3
4      cidade1, horario1 = escalas[0]  # horario de partida
5      cidade2, horario2 = escalas[-1] # horario de chegada
6
7      (h1, m1) = horario1          # hora e minuto de partida
8      (h2, m2) = horario2          # hora e minuto de chegada
9
10     # Tempo decorrido em minutos desde o inicio do dia:
11     tempo1 = h1*60 + m1
12     tempo2 = h2*60 + m2
13
14     diferenca = tempo2 - tempo1
15     if diferenca > 0:   # caso normal
16         return diferenca
17     else:           # comecou num dia e terminou no outro
18         return 24*60 + diferenca
19
20
21 def questao2(voos):
22     for numVoo in voos:
23         cia, escalas = voos[numVoo]
24
25         nomeOrigem, _ = escalas[0]  # nome da cidade origem
26         nomeFinal, _ = escalas[-1]  # nome da cidade final
27
28         print("Numero do voo:" + numVoo)
29         print("Cidade origem:" + nomeOrigem)
30         print("Destino final:" + nomeFinal)
31
32 def questao3(a, b, voos):
33     print("Saem de {} e passam por {}: ".format(a, b))
34     for numVoo in voos:
35         cia, escalas = voos[numVoo]
36
37         origem, _ = escalas[0]       # nome da cidade origem
38         for cid, _ in escalas[1:]:  # demais cidades
39             if origem == a and cid == b:
40                 print(numVoo, end=" ")
41
42     print("\n")
```

```
43
44
45 def questao4(a, b, voos):
46     # Salvando o numero de todos os voos entre a e b:
47     meusVoos = []
48     for numVoo in voos:
49         cia, escalas = voos[numVoo]
50
51         nomeOrigem, _ = escalas[0]  # nome da cidade origem
52         nomeFinal, _ = escalas[-1]  # nome da cidade final
53
54         if nomeOrigem == a and nomeFinal == b:
55             meusVoos.append( numVoo )
56
57     # Encontrando o voo mais curto:
58     if len(meusVoos) == 0:
59         print("Nao foram encontrados voos disponiveis")
60     else:
61         melhorVoo = meusVoos[0]
62         tempoMelhorVoo = questao1(melhorVoo, voos)
63
64         for numVoo in meusVoos:
65             tempoVoo = questao1(numVoo, voos)
66
67             if tempoVoo < tempoMelhorVoo:
68                 melhorVoo = numVoo
69
70         print("O voo mais curto entre" + a + "e" + b + ":")
71         print("{} ({})".format(melhorVoo, voos[melhorVoo][0]))
```

8. **Séries**:

```
1 import os
2 import pickle
3
4 def limpaTela():
5     if os.name == "nt":
6         os.system("cls")
7     else:
8         os.system("clear")
9
10 def novaSerie(series):
11     nome = input("Digite o nome da serie: ")
```

```
12
13      if nome in series:
14          print("Esta serie ja foi inserida anteriormente")
15      else:
16          series[nome] = []
17          print("Serie cadastrada com sucesso")
18
19  def apagaSerie(series):
20      if series == {}:
21          print("Nenhuma serie cadastrada")
22          return
23      else:
24          print("Series ja cadastradas:")
25          for serie in series:
26              print(serie, " ")
27          print()
28
29      nome = input("Digite o nome da serie: ")
30
31      if nome in series:
32          del series[nome]
33          print("Serie apagada com sucesso")
34      else:
35          print("Serie inexistente")
36
37  def imprimeSerie(series, nome):
38          print("*", nome, "*")
39          print()
40          temporadas = series[nome]
41          if len(temporadas) == 0:
42              print(" Nenhuma temporada cadastrada")
43              print()
44          else:
45              i = 0
46              for temporada in temporadas:
47                  print(" Temporada", i+1, end=": ")
48                  j = 0
49                  for episodio in temporada:
50                      msg = "T" + str(i+1) + "E" + str(j+1)
51                      if episodio:
52                          print("+"+msg, end=" ")
53                      else:
```

```
54                          print("-"+msg, end=" ")
55                      j+=1
56                  print()
57                  i+=1
58              print()
59
60  def imprimeSeries(series):
61      if len(series) == 0:
62          print("Nenhuma serie cadastrada")
63      else:
64          for nome in series:
65              imprimeSerie(series, nome)
66
67  def cadastraTemp(series, nome):
68      temporadas = series[nome]
69      print("Numero de temporadas ja cadastradas:",
70            len(temporadas))
71      msg = "Digite a quantidade de episodios da temporada " +
72            str(len(temporadas)+1) + " ou 0 para desistir: "
73      n = int(input(msg))
74
75      if n != 0:
76          series[nome].append(n*[False])
77          print("Temporada cadastrada com sucesso")
78      else:
79          print("Nenhuma nova temporada foi cadastrada")
80
81  def novaTemp(series):
82      if series == {}:
83          print("Nenhuma series cadastrada")
84          return
85      else:
86          print("Series ja cadastradas:")
87          for serie in series:
88              print(serie, " ")
89          print()
90
91      nome = input("Escolha uma serie: ")
92
93      if nome not in series:
94          print("Esta serie ainda nao foi cadastrada.")
95      else:
```

```
96          cadastraTemp(series, nome)
97
98  def lerSeries():
99      if not os.path.isfile("series.bin"):
100         return {}
101     else:
102         with open("series.bin", "rb") as f:
103             series = pickle.load(f)
104             return series
105
106 def salvarSeries(series):
107     with open("series.bin", "wb") as f:
108         pickle.dump(series, f)
109
110 def marcaEpisodio(series):
111     imprimeSeries(series)
112
113     nome = input("Escolha uma serie: ")
114     if nome not in series:
115         print("Serie inexistente")
116         return
117
118     temporada = int(input("Digite a temporada: "))
119     if temporada > len(series[nome]):
120         print("Temporada inexistente")
121         return
122
123     episodio = int(input("Digite o episodio: "))
124     if episodio > len(series[nome][temporada-1]):
125         print("Episodio inexistente")
126         return
127
128     series[nome][temporada-1][episodio-1] = True
129     print("Episodio T"+str(temporada)+"E"+str(episodio)+
130           " marcado como visto")
131
132
133 def marcaTemp(series):
134     imprimeSeries(series)
135
136     nome = input("Escolha uma serie: ")
137     if nome not in series:
```

```
138         print("Serie inexistente")
139         return
140 
141     temporada = int(input("Digite a temporada: "))
142     if temporada > len(series[nome]):
143         print("Temporada inexistente")
144         return
145 
146     for i in range(len(series[nome][temporada-1])):
147         series[nome][temporada-1][i] = True
148 
149     print("Temporada", temporada, "marcada como vista")
150 
151 
152 def marcaSerie(series):
153     imprimeSeries(series)
154 
155     nome = input("Escolha uma serie: ")
156     if nome not in series:
157         print("Serie inexistente")
158         return
159 
160     for i in range(len(series[nome])):
161         for j in range(len(series[nome][i])):
162             series[nome][i][j] = True
163 
164     print("Serie", nome, "marcada como vista")
165 
166 
167 def listaPendentes(series, nome):
168     l = []
169     i = 0
170     for temporada in series[nome]:
171         j = 0
172         for episodio in temporada:
173             if not episodio:
174                 msg = "T"+str(i+1)+"E"+str(j+1)
175                 l.append(msg)
176             j+=1
177         i+=1
178 
179     return l
```

```
180
181 def qtdEpisodios(series, nome):
182     n = 0
183     for temporada in series[nome]:
184         n += len(temporada)
185
186     return n
187
188 def atrasadas(series, f):
189     f.write("Episodios atrasados:\n")
190
191     for nome in series:
192         naoAssistidos = listaPendentes(series, nome)
193         if len(naoAssistidos) > 0:
194             f.write("* " + nome + ": ")
195             for episodio in  naoAssistidos:
196                 f.write(episodio + " ")
197             f.write("\n")
198
199     f.write("\n")
200
201 def emDia(series, f):
202     f.write("Series em dia:\n")
203
204     for nome in series:
205         naoAssistidos = len(listaPendentes(series, nome))
206         if naoAssistidos == 0:
207             f.write("* " + nome + "\n")
208
209     f.write("\n")
210
211
212 def totalAssistidos(series, f):
213     naoAssistidos = 0
214     total = 0
215     for nome in series:
216         total += qtdEpisodios(series, nome)
217         naoAssistidos += len(listaPendentes(series, nome))
218
219     f.write("Quantidade total de episodios cadastrados: " +
220             str(total) + "\n")
221     f.write("Quantidade total de episodios ja assistidos: "+
```

```
222             str(total-naoAssistidos) + "\n")
223
224 def estatisticas(series):
225     with open("estatisticas.txt", "w") as f:
226         f.write("Quantidade de series cadastradas: " +
227                 str(len(series)) + "\n\n")
228
229         emDia(series, f)
230         atrasadas(series, f)
231         totalAssistidos(series, f)
232
233         print("Estatisticas salvas.")
234
235
236 def main(args):
237
238     print("*** Bem vindo ao Maratonando ***")
239
240     menu = '''
241 Escolha uma opcao:
242 1) Cadastrar nova serie
243 2) Apagar serie
244 3) Cadastrar nova temporada
245 4) Exibir series
246 5) Marcar episodio como visto
247 6) Marcar temporada como vista
248 7) Marcar serie como vista
249 8) Salvar estatisticas
250 0) Sair
251 '''
252
253     series = lerSeries()
254
255     x = int(input(menu))
256     while x != 0:
257         limpaTela()
258
259         #print(series)
260
261         if x == 1:
262             novaSerie(series)
263             salvarSeries(series)
```

```
264             elif x == 2:
265                 apagaSerie(series)
266                 salvarSeries(series)
267             elif x == 3:
268                 novaTemp(series)
269                 salvarSeries(series)
270             elif x == 4:
271                 imprimeSeries(series)
272             elif x == 5:
273                 marcaEpisodio(series)
274                 salvarSeries(series)
275             elif x == 6:
276                 marcaTemp(series)
277                 salvarSeries(series)
278             elif x == 7:
279                 marcaSerie(series)
280                 salvarSeries(series)
281             elif x == 8:
282                 estatisticas(series)
283             else:
284                 print("Opcao invalida")
285 
286             x = int(input(menu))
287 
288         print("Ate breve...")
289         return 0
290 
291 
292 
293 if __name__ == '__main__':
294         import sys
295         sys.exit(main(sys.argv))
```

9. **Perícia Criminal**:

```
1 from random import randint, choice
2 
3 def lerClientes():
4         clientes = {}
5         with open("input/clientes.txt", "r") as f:
6             arq = f.read()
7             linhas = arq.splitlines()
8 
```

```
9           n = int(linhas[0])
10          k = 0
11          for i in range(n):
12              cpf = linhas[1+2*i]
13              nome = linhas[1+2*i+1]
14
15              clientes[cpf] = nome
16
17      return clientes
18
19  def lerContas():
20      contas = {}
21      with open("input/contas.txt", "r") as f:
22          arq = f.read()
23          linhas = arq.splitlines()
24
25          n = int(linhas[0])
26          k = 0
27          for i in range(n):
28              numero = linhas[1+3*i]
29              saldo = int(linhas[1+3*i+1])
30              cpf = linhas[1+3*i+2]
31
32              contas[numero] = (saldo, cpf)
33
34      return contas
35
36  def lerHistorico():
37      historico = {}
38      with open("input/historico.txt", "r") as f:
39          arq = f.read()
40          linhas = arq.splitlines()
41
42          n = int(linhas[0])
43          k = 0
44          for i in range(n):
45              ident = linhas[1+4*i]
46              valor = int(linhas[1+4*i+1])
47              num = linhas[1+4*i+2]
48              tipo = linhas[1+4*i+3]
49
50              historico[ident] = (valor, num, tipo)
```

```
51
52      return historico
53
54  def saldosIncorretosPrejuizo(contas, clientes, saldos):
55      invadidas = []
56      prejuizo = 0
57
58      print("- Contas invadidas:")
59      for conta in saldos:
60          if saldos[conta] != contas[conta][0]:
61              prejuizo += (saldos[conta] - contas[conta][0])
62              print("{} - Titular: {} - Saldo correto: R$ {} -
63                     Saldo atual: R$ {}".format(conta,
64                     clientes[contas[conta][1]], saldos[conta],
65                     contas[conta][0]))
66              invadidas.append(conta)
67
68      print("- Prejuizo do banco: R$", prejuizo)
69      return invadidas
70
71  def agenciasInvadidas(agencias, contas, contasInvadidas):
72      agenciasInvadidas = []
73      for invadida in contasInvadidas:
74          numAgencia = contas[invadida][2]
75          if numAgencia not in agenciasInvadidas:
76              agenciasInvadidas.append(numAgencia)
77
78      print("- Agencias a serem investigadas:")
79      for agencia in agenciasInvadidas:
80          nome, cidade, uf = agencias[agencia]
81          print("{} - {} - {}/{}".format(agencia, nome,
82                                         cidade, uf))
83
84
85  def saldosCorretos(contas, historico):
86      saldos = {}
87
88      for idConta in contas:
89          saldos[idConta] = 0
90
91      for idHistorico in historico:
92          valor, numConta, tipo = historico[idHistorico]
```

```
93
94          if tipo == "0":
95              saldos[numConta] -= valor
96          elif tipo == "1":
97              saldos[numConta] += valor
98
99      return saldos
100
101 def hackeadas(contas, saldos):
102     invadidas = {}
103     for numConta in contas:
104         saldoConta, cpf = contas[numConta]
105         saldoCorreto = saldos[numConta]
106
107         if saldoConta != saldoCorreto:
108             invadidas[numConta] = (saldoConta, saldoCorreto)
109
110     return invadidas
111
112 def prejuizoBanco(invadidas):
113     preju = 0
114     for numConta in invadidas:
115         saldoConta, saldoCorreto = invadidas[numConta]
116         preju += (saldoConta - saldoCorreto)
117
118     print("Prejuizo do banco:", preju)
119
120
121 def vermelho(invadidas, contas, clientes):
122     for numConta in invadidas:
123         saldoConta, saldoCorreto = invadidas[numConta]
124         _, cpf = contas[numConta]
125         nome = clientes[cpf]
126
127         if saldoConta < 0:
128             print("Numero da conta:", numConta)
129             print("CPF:", cpf)
130             print("Nome:", nome)
131
132
133 def main(argv=None):
134
```

```
135     clientes = lerClientes()
136     contas = lerContas()
137     historico = lerHistorico()
138 
139     print(clientes)
140     print(contas)
141     print(historico)
142 
143     saldos = saldosCorretos(contas, historico)
144     print(saldos)
145 
146     invadidas = hackeadas(contas, saldos)
147     print(invadidas)
148 
149     prejuizoBanco(invadidas)
150 
151     vermelho(invadidas, contas, clientes)
152 
153 
154 
155 if __name__ == '__main__':
156     main()
```