1 Teste

Uma empresa deseja saber se seus produtos estão dando lucro. Para isso, crie um procedimento que receba duas listas como parâmetro: a primeira contém o custo (número real) de cada mercadoria e a segunda contém o preço de venda de cada uma delas. Cada posição k da primeira lista se refere ao mesmo produto que está na posição k da segunda. O procedimento deve imprimir: (1) a quantidade de produtos que tiveram menos de 20% de lucro e (2) a porcentagem (utilizando duas casas decimais) de produtos com lucro superior a 25%.

Solução:

```
1  def verificaLucro(custos, precos):
2      menos20 = 0
3      mais25 = 0.0
4
5      for i in range(len(custos)):
6          custo = custos[i]
7          preco = precos[i]
8          if preco < 1.2*custo:
9              menos20 += 1
10         if preco > 1.25*custo:
11             mais25 += 1
12
13     print("a) {}".format(menos20))
14     print("b) {:.2f}%".format(100*mais25/len(custos)))
```

2 Tuplas

Uma **tupla** em Python pode armazenar um conjunto de valores e seus elementos são indexados a partir de zero, assim como nas listas. Entretanto, ao contrários das listas, os elementos de uma tupla não podem ser alterados. Uma vez criada, não é possível modificar cada elemento de uma tupla, apenas acessá-los. Um ponto cartesiano, por exemplo, possui dois valores (ordenada e abscissa) que não se modificam ao longo de um programa:

```
1  p1 = (3.0, 0.0)
2  p2 = 0.0, 4.0
3  dist = ( (p1[0] - p2[0])**2 + (p1[1] - p2[1])**2 ) ** 0.5
4  print(dist)     # Resultado: 5
```

Como vemos no exemplo acima, tuplas possuem um conjunto de elementos separados por vírgula, podendo ou não estar entre parênteses. Assim como nas listas, também podemos acessar um intervalo de valores em uma tupla:

```
1 tupla = ('a', 'b', 'c', 'd', 'e', 'f', 'g')
2 print(tupla[1:5])
```

É muito comum precisarmos trocar o valor de duas variáveis, ou até mesmo dos elementos em duas posições de uma lista. Geralmente, utilizamos uma variável auxiliar para isso:

```
1 a = 5
2 b = 7
3
4 temp = a
5 a = b
6 b = temp
```

Usando tuplas, podemos fazer isso de forma mais simples e elegante:

```
1 a, b = b, a
```

Operações comuns a listas e tuplas:

- `x in s`: Retorna verdadeiro se o elemento x pertence à tupla/lista s e falso caso contrário.
- `x not in s`: Retorna verdadeiro se o elemento x não pertence à tupla/lista s e falso caso contrário.
- `s + t`: Concatena os elementos da tupla/lista s com os elementos da tupla/lista t.
- `s*n` ou `n*s`: Concatena os elementos da tupla/lista s à ela mesma, n vezes.
- `len(s)`: Retorna á quantidade de elementos da tupla/lista s.

Também é possível utilizar um **for** para iterar entre os elementos de uma tupla da mesma forma que fazemos com listas. Por serem imutáveis, tuplas são estruturas mais rápidas que listas e por isso são indicadas quando os valores dos elementos não serão alterados ao longo do código. Implicitamente, também temos uma garantia de que aqueles dados não serão alterados incorretamente. Além disso, tuplas podem ser usadas como chaves de um dicionário, e listas não (veremos dicionários mais adiante).

Como exemplo, vamos imaginar que cada funcionário de uma empresa possui um nome, seu tempo de casa (em meses) e seu salário. A empresa armazena os dados de cada funcionário utilizando tuplas, e possui uma lista de tuplas com os dados de todos os funcionários:

```
1 l = [("Valentina", 4, 1500), ("Enzo", 33, 1200),
2      ("Anna Julia", 22, 3000), ("Simaria",  33, 1400)]
```

O segundo elemento da lista l, por exemplo, indica que o funcionário "Enzo" foi contratado há 33 meses e recebe R$1.200,00 por mês. A empresa pediu para você implementar uma sub-rotina que descubra qual é o funcionário com mais tempo de casa para lhe dar uma bonificação de 10% este mês (caso haja mais de um funcionário com este mesmo tempo de casa, todos eles receberão a bonificação). O procedimento a seguir recebe a lista de funcionários como parâmetro e imprime o nome do(s) funcionário(s) mais antigo(s) com seu salário diferenciado neste mês.

```
1  def bonus(l):
2      maiorTempo = 0
3
4      for (nome, meses, salario) in l:
5          if meses > maiorTempo:
6              maiorTempo = meses
7
8
9      for (nome, meses, salario) in l:
10         if meses == maiorTempo:
11             print(nome, 1.1*salario)
```

Agora, a mesma empresa deseja saber se seus produtos estão dando lucro. Para isso, pediu para você criar uma sub-rotina que recebe uma lista de tuplas contendo o preço de custo e o preço de venda de cada mercadoria e imprima:

- a quantidade de produtos com menos de 20% de lucro
- a porcentagem de produtos com lucro superior a 25%

```
1  def verificaLucro(l):
2      menos20 = 0
3      mais25 = 0.0
4
5      for (custo, venda) in l:
6          if venda < 1.2*custo:
7              menos20 += 1
8          if venda > 1.25*custo:
9              mais25 += 1
10
11     print("a)", menos20)
12     print("b)", 100*mais25/len(l))
```

Para verificar se três pontos $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$ no plano cartesiano estão alinhados, podemos verificar se o determinante da matriz a seguir é igual a zero:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0$$

A função a seguir recebe três tuplas como parâmetros (os três pontos), cria a matriz e utiliza a função criada anteriormente para calcular o determinante da matriz.

```
1 def alinhados(p1, p2, p3):
2     x1, y1 = p1
3     x2, y2 = p2
4     x3, y3 = p3
5
6     M = [ [x1, y1, 1], [x2, y2, 1], [x3, y3, 1] ]
7
8     return det(M) == 0
```

Uma agência de turismo possui armazenados os voos realizados por diversas companhias aéreas. Cada voo é representado como uma tupla com as seguintes informações:

- Número do voo
- Companhia que realizou o voo (String).
- Lista de escalas (cada elemento da lista é o nome de uma cidade, na ordem em que foram visitadas).

Exemplo: voos = [(1024, "TAM", ["ES", "RJ", "SP", "NY"]), (1025, "GOL", ["ES", "SP"])]

Crie funções para:

1. Dada a lista de voos, uma cidade origem a e uma cidade destino b, imprima o número e a companhia de todos os voos que se iniciem em a e cujo destino final seja b.

2. Dada a lista de voos, uma cidade origem a e uma cidade destino b, imprima quantos voos se iniciem em a e que façam alguma escala em b.

3. Dada a lista de voos, uma cidade origem a e uma cidade destino b, verifique se há algum voo direto de a para b, mesmo que a e b não sejam os destinos iniciais e finais do voo.

```
1 def voos1(l, a, b):
2     for (num, cia, escalas) in l:
3         if escalas[0] == a and escalas[-1]==b:
```

```
4               print(num, cia)
5
6   def voos2(l, a, b):
7       nVoos = 0
8       for (num, cia, escalas) in l:
9           if escalas[0] == a and b in escalas:
10              nVoos += 1
11      print(nVoos)
12
13  def voos3(l, a, b):
14      nVoos = 0
15      for (num, cia, escalas) in l:
16          for i in range(len(escalas)-1):
17              if escalas[i] == a and escalas[i+1] == b:
18                  return True
19      return False
```

2.1 Projeto: Bolão

Em um Bolão, vários apostadores se juntam para adquirir uma série de cartões de apostas. Pode-se optar também por uma única aposta, mas com uma quantidade maior de números. Por exemplo, escolhem-se 8 números para apostar na Mega Sena, sendo que apenas 6 números serão sorteados nesta loteria. Se 6 dentre estes 8 números forem sorteados, a aposta é considerada vencedora. As chances de acerto aumentam, mas é bem mais caro apostar mais de 6 números em um mesmo cartão.

Você decidiu criar um sistema para administrar bolões. Cada cartão apostado poderá ter uma quantidade qualquer de pessoas participando, e ao menos 6 números. Seu sistema deve ser capaz de armazenar nome e CPF de cada jogador, além de várias apostas diferentes. A interface do sistema, por enquanto, será no terminal e deve permitir, ao menos:

- Inserir novo jogador
- Visualizar jogadores cadastrados
- Inserir nova aposta
- Visualizar apostas cadastradas
- Inserir resultado e listar apostas vencedoras

Estruturas: Cada jogador será representado por uma tupla contendo nome e CPF. Exemplos:

```
("Enzo", "123-456-789-00")
("Valentina", "234.567-890-00")
```

Cada aposta será representada por uma tupla contendo os CPFs dos apostadores e os números sorteados. Exemplos:

```
( ["123-456-789-00", "234.567-890-00"], [12, 23, 35, 39, 41, 46, 52, 55] )
( ["123-456-789-00"], [14, 25, 31, 33, 41, 46, 53] )
```

Repare que Enzo participou de duas apostas. Na primeira, junto da Valentina, eles apostaram 8 números. Na segunda, ele apostou 7 números sozinho. Seu programa, portanto, irá armazenar duas listas (uma lista de jogadores, e uma lista de apostas). Exemplo:

```
1 jogadores = [ ("Enzo", "123-456-789-00"),
2               ("Valentina", "234.567-890-00") ]
3 apostas   = [ ( ["123-456-789-00", "234.567-890-00"],
4                 [12, 23, 35, 39, 41, 46, 52, 55] ) ,
5               ( ["123-456-789-00"],
6                 [14, 25, 31, 33, 41, 46, 53] ) ]
```

Exercícios:

1. Defina um menu para o programa, a ser exibido no console, com as seguintes opções:

   - Cadastrar novo jogador
   - Visualizar jogadores cadastrados
   - Inserir nova aposta
   - Visualizar apostas cadastradas
   - Inserir resultados e listar vencedores
   - Sair do programa

   Utilize uma função auxiliar para limpar a tela do programa cada vez que uma opção for escolhida.

```
def main(args):
    print("Bem vindo")

    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 = []

    opcao = input(menu)
    while opcao != "0":
        limpaTela()
        if opcao == "1":
            cadastrarJogador(jogadores)
        elif opcao == "2":
            visualizarJogadores(jogadores)
        elif opcao == "3":
            cadastrarAposta(jogadores, apostas)
        elif opcao == "4":
            visualizarApostas(jogadores, apostas)
        elif opcao == "5":
            opcao5(jogadores, apostas)
        elif opcao != 0:
            print("Opcao invalida.")
        opcao = input(menu)

    print("Ate breve...")
    return 0
```

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

2. Crie uma função para a primeira opção do menu (cadastrar novo jogador). A função deve fazer a leitura do nome e CPF de um jogador e inseri-lo no sistema. Caso um jogador com o mesmo CPF já tenha sido inserido, deve exibir uma mensagem de erro e volta para o menu sem que ele tenha sido inserido. Utilize uma função auxiliar para verificar se algum funcionário com aquele CPF já foi inserido no sistema.

```
1  def existe(cpf, jogadores):
2      for n, c in jogadores:
3          if c == cpf:
4              return True
5
6      return False
7
8  def cadastrarJogador(jogadores):
9      nome = input("Digite o nome do jogador: ")
10     cpf = input("Digite o cpf do jogador: ")
11
12     if not existe(cpf, jogadores):
13         jogadores.append( (nome,cpf) )
14         print("Pessoa cadastrada com sucesso! :-)")
15     else:
16         print("Erro! Pessoa ja cadastrada no sistema.")
```

3. Crie uma função para a segunda opção do menu (visualizar jogadores cadastrados). A função deve imprimir os dados dos jogadores no seguinte formato:

```
Enzo - CPF: 123.456.789-00
Valentina - CPF: 234.567.890-00
```

```
1  def visualizarJogadores(jogadores):
2      if len(jogadores) == 0:
3          print("Nenhuma pessoa cadastrada.")
4      else:
5          print("Pessoas cadastradas no sistema:")
6          for nome, cpf in jogadores:
7              print("{} - CPF: {}".format(nome, cpf))
```

4. Crie uma função para cadastrar uma nova aposta. Modularize a função, de forma que ela utilize duas sub-rotinas auxiliares:

   - Função que faz a leitura da quantidade de jogadores que irão dividir o bilhete e dos CPFs dos jogadores participantes da aposta (sempre verificando se o

CPF pertence a algum jogador já cadastrado no sistema). Quando um CPF não cadastrado for digitado, permanece solicitando um novo CPF até que o usuário digite um CPF correto, ou digite 0 para desistir do cadastro da aposta e voltar ao menu.

- Função que faz a leitura da quantidade de números a serem apostados no bilhete e dos números apostados. Caso o usuário digite um número já inserido, deve permanecer solicitando outro número até que seja digitado um número válido.

```
1  def nomeJogador(cpf, jogadores):
2      for n, c in jogadores:
3          if c == cpf:
4              return n
5
6      return None
7
8  def lerNumApostadores(jogadores):
9      '''Leitura do numero de jogadores no bilhete'''
10     qtd = int(input("Digite o numero de apostadores: "))
11
12     while qtd < 1 or qtd > len(jogadores):
13         print("Erro. Apenas", len(jogadores),
14               "jogadores foram cadastrados.")
15         qtd = int(input("Digite o numero de apostadores: "))
16
17     return qtd
18
19
20 def leituraCPFs(jogadores):
21     qtd = lerNumApostadores(jogadores)
22
23     cpfs = []
24     while len(cpfs) < qtd:
25         visualizarJogadores(jogadores)
26         cpf = input("Digite o CPF ou 0 para sair: ")
27
28         while cpf in cpfs:
29             print('Erro: CPF ja inserido neste bilhete.')
30             cpf = input("Tente outro CPF: ")
31         limpaTela()
32         if cpf == "0": return []
33
```

```
34          nome = nomeJogador(cpf, jogadores)
35
36          if nome is not None:
37              cpfs.append(cpf)
38              print(nome, "cadastradx com sucesso")
39          else:
40              print("Erro! Pessoa nao cadastrada.")
41
42      return cpfs
43
44  def lerQtdNumeros():
45      n = int(input("Quantos numeros o bilhete contem? "))
46
47      while n < 6 or n > 15:
48          n = int(input("Informe um numero de 6 a 15: "))
49
50      return n
51
52  def lerNumeros(n):
53      numeros = []
54      while len(numeros) < n:
55          num = int(input("Digite um numero apostado: "))
56
57          while num < 1 or num > 60 or num in numeros:
58              if num < 1 or num > 60:
59                  print("Este numero nao existe num cartao.")
60              else:
61                  print("Erro! Numero ja cadastrado.")
62                  print("Cadastrados ate agora:", numeros)
63
64              num = int(input("Digite um numero apostado: "))
65
66          numeros.append(num)
67      return numeros
68
69  def numerosDoBilhete():
70      '''Leitura dos numeros a serem apostados.'''
71      n = lerQtdNumeros():
72
73      menu = '''Digite 1 para informar os numeros ou
74  outra tecla para que sejam escolhidos aleatoriamente: '''
75      opcao = input(menu)
```

```
76
77      if opcao == "1":
78          numeros = lerNumeros(n)
79      else:
80          numeros = random.sample(list(range(1,61)), n)
81
82      numeros.sort()
83      return numeros
84
85  def cadastrarAposta(jogadores, apostas):
86      cpfs = leituraCPFs(jogadores)
87      if cpfs != []:
88          numeros = numerosDoBilhete()
89          apostas.append( (cpfs, numeros) )
90          print("Aposta cadastrada com sucesso.")
```

5. Crie uma função para visualizar as apostas cadastradas no sistema. A função deve imprimir os dados das apostas no seguinte formato:

```
APOSTA 1
Numeros: 12 23 35 39 41 46 52 55
Jogadores:
Enzo - CPF: 123.456.789-00
Valentina - CPF: 234.567.890-00
```

```
1  def getNome(cpf, jogadores):
2      for n, c in jogadores:
3          if c == cpf:
4              return n
5
6  def visualizaApostas(apostas, jogadores):
7      i = 0
8      while i < len(apostas):
9          numeros, cpfs = apostas[i]
10
11         print("\nAPOSTA", i+1)
12         print("Numeros:", end=" ")
13
14         for numero in numeros:
15             print(numero, end=" ")
16
```

```
17          print("\nJogadores: ")
18
19          for cpf in cpfs:
20              nome = getNome(cpf, jogadores)
21              print(nome, "- CPF:", cpf)
22
23          i = i+1
```

6. Crie uma função para inserir um sorteio, verificar as apostas vencedoras, dividir o prêmio entre os bilhetes vencedores, e listar o prêmio que cada participantes irá receber. Modularize a função utilizando ao menos as seguintes sub-rotinas:

   - Função que faz a leitura dos 6 números sorteados, e retorna uma lista com esses números.
   - Função que verifica se uma lista l_1 está contida na lista l_2. Utilizada para conferir se todos os números sorteados fazem parte dos números apostas em um bilhete.
   - Função que conta quais bilhetes foram premiados.
   - Função que lista todos os vencedores de cada bilhete premiado, além do prêmio que cada apostador receberá por ele. Recebe como parâmetro os índices dos bilhetes premiados na lista de apostas, e o prêmio daquele bilhete, já dividido entre a quantidade de bilhetes que foram premiados. Para identificar cada bilhete, enumere-os a partir de 1. Exemplo:

     ```
     Vencedores no bilhete 1
     Enzo - CPF: 123.456.789-00 - R$ 50000.0
     Valentina - CPF: 234.567.890-00 - R$ 50000.0
     ```

```
1  def numerosSorteados():
2      '''Leitura dos 6 numeros sorteados.'''
3      l = []
4      while len(l) < 6:
5          x = int(input("Digite um dos numeros sorteados: "))
6          if x not in l and x > 0 and x <= 60:
7              l.append(x)
8          else:
9              print("Erro. Numero ja inserido ou invalido.")
10
11     return l
12
13
```

```
14 def contida(l1, l2):
15     '''Verifica se a lista l1 esta contida na lista l2.'''
16     for elem in l1:
17         if elem not in l2:
18             return False
19 
20     return True
21 
22 def bilhetesPpremiados(apostas, numeros):
23     '''Retorna os indices dos bilhetes premiados.'''
24     l = []
25     for i in range(len(apostas)):
26         (_, nums) = apostas[i]
27         if contida(numeros, nums):
28             l.append(i)
29 
30     return l
31 
32 def listarVencedores(jogadores, apostas, posVencedores,
33                      premioPorBilhete):
34     limpaTela()
35     print("Vencedores:")
36 
37     for pos in posVencedores:
38         print( "BILHETE:", pos+1)
39         (cpfs, _) = apostas[pos]
40         for cpf in cpfs:
41             print(nomeJogador(cpf, jogadores), "- CPF:",
42                   cpf, "- RS", premioPorBilhete/len(cpfs))
43 
44 def insereSorteio(jogadores, apostas):
45     nums = numerosSorteados()
46     premio = float(input("Digite o valor do premio: "))
47 
48     ganhadores = bilhetesPremiados(apostas, nums)
49     nBilhetesVencedores = len(ganhadores)
50     premioPorBilhete = premio / nBilhetesVencedores
51 
52     if nBilhetesVencedores > 0:
53         listarVencedores(jogadores, apostas, ganhadores,
54                          premioPorBilhete)
55     else:
```

```
56          print("Nao houve vencedorxs.")
```

3 Números aleatórios

Ao lançarmos uma moeda ou um dado, nunca temos certeza do resultado final. Essa imprevisibilidade tem muitas aplicações. Por exemplo, a escolha dos vencedores em sorteios, a geração de casos de teste para experimentos de algoritmos, criação de inimigos em um jogo, embaralhamento de elementos numa lista, etc. Python contém módulo `random`, que permite trabalharmos com números aleatórios (também conhecidos como números randômicos).

O módulo possui duas funções para gerar inteiros aleatórios:

- `random.randrange`: Se chamarmos a função com apenas um parâmetro a, iremos gerar um número inteiro aleatório k tal que $0 \leq k < a$. É equivalente a escolher um número aleatoriamente dentre os elementos presentes em `range(a)`, por isso o nome *randrange*. De forma análoga, se passarmos dois parâmetros `randrange(a,b)`, geramos um número inteiro aleatório k tal que $a \leq k < b$, ou seja, um número presente em `range(a,b)`. Se passarmos três parâmetros, escolhemos um número em `range(a,b,c)`.
- `random.randint`: Mais intuitiva, esta função recebe dois inteiros a e b como parâmetros, e gera um número aleatório entre a e b (incluindo b). É como se chamássemos `randrange(a,b+1)`.

Também podemos escolher elementos aleatórios dentre de uma lista (ou tupla) não vazia:

- `random.choice(l)`: Escolhe um único elemento de l.
- `random.sample(l, k)`: Retorna uma lista com k elementos de l.
- `random.shuffle(l)`: Embaralha os elementos de l.

Na função que cadastrava os números de uma aposta no Bolão, já havíamos utilizado a função `random.sample(l, k)` para marcar números aleatórios num bilhete. Agora, para testar a geração de números aleatórios, vamos implementar um jogo semelhante ao Genius:

Figura 1: Jogo Genius.

Muito popular na década de 1980, o brinquedo buscava estimular a memorização de cores. Com um formato semelhante a um OVNI, possuía botões coloridos que emitiam sons harmônicos e se iluminavam em sequência. Cabia aos jogadores repetir a ordem das cores sem errar. Ao invés de cores, utilizaremos números. A cada iteração, nosso programa irá sortear um algarismo aleatório, que será exibido para o usuário. Caberá ao usuário memorizar e digitar corretamente toda a sequência de caracteres exibidos:

```
1  sorteado = randint(0,9)
2  correta = str(sorteado)
3
4  print("O primeiro numero sorteado foi:", sorteado)
5  x = input("Digite a sequencia completa: ")
6
7  while x == correta:
8      sorteado = randint(0,9)
9      correta = correta + str(sorteado)
10
11     limpaTela()
12     print("O novo numero eh:", sorteado)
13     x = input("Digite a sequencia completa: ")
14
15 print("Errou! Voce acertou", len(correta)-1, "numeros.")
```