Exercícios resolvidos sobre funções e processamento de arquivos texto

Exercício 1 - Crie uma função que receba como entrada um valor de temperatura em graus Celsius (C) e retorne o valor equivalente em graus Fahrenheit (F) através da seguinte fórmula: F = 1,8C + 32.

```
# Função que converte de Celsius para Fahrenheit
# Parâmetros: C = temperatura em graus Celsius
def conv_f(C):
   return 1.8 * C + 32;

# programa principal – testando a função
print('0C = ', conv_f(0), 'F');
print('10C = ', conv_f(10), 'F');
print('37C = ', conv_f(37), 'F');
```

Saída:

```
0C =  32.0 F
10C =  50.0 F
37C =  98.60000000000001 F
```

Exercício 2 - Escreva uma função que retorne o fatorial de um número *n*

```
#   definição   de   uma   função   que   calcula   o   fatorial   de   "n"
# (número inteiro passado como entrada)
def fatorial(n):

    # se o número for negativo, retorna None
    if n < 0: return None

    # senão, prossegue calculando e retornando o fatorial
    f = 1
    for i in range(2,n+1): f = f * i

    return f # essa linha é o retorno da função



# programa principal
a = fatorial(5) # a recebe 5!
print(a);

print(fatorial(8)) # imprime o valor de 8!
```

Saída:

```
120
40320
```

Neste exemplo, a criação da função começa com a seguinte linha: **`def`** `fatorial(n):`. Com isto estamos informando ao Python que será criada uma função denominada **`fatorial`**. Esta função possui um parâmetro de entrada, "n", que só fará sentido se for do tipo inteiro. Analisando o corpo da função **`fatorial`** vemos que a primeira linha testará se "n" possui valor negativo e, se isso ocorrer, a função retornará o valor None de imediato (uma vez que não é possível calcular o fatorial de um negativo).

A linhas seguintes dentro do código da função são responsáveis por obter o fatorial de "n" e armazenar na variável local "f". Veja que basta inicializar f com 1 e depois ir atualizando o seu

valor dentro de um laço que gera a sequência {2, ..., n}. A cada iteração do laço, o valor de "f" é multiplicado por um dos valores da sequência.

É importante agora analisar última linha dentro do código da função: `return f`. Essa linha **atribui o valor de "f" como valor de retorno da função**.

Exercício 3 - Escrever uma função chamada "mensagem_alien" com 2 parâmetros:

- **"pessoa": nome de uma pessoa para a qual o alienígena dirá uma frase**
- **"tipo_alien": 'B' = alienígena bonzinho, 'M' = alienígena malvado**

A função deverá imprimir uma das duas seguintes frases de acordo com o tipo do alien:

- **Para o tipo 'B', retornar "*pessoa*, eu vim em missão de paz!"**
- **Para o tipo 'M', retornar "*pessoa*, eu vou te abduzir e escravizar!"**

```
# Função que gera a mensagem do alienígena!
# Parâmetros:
# - "pessoa": nome de uma pessoa
# - "tipo_alien": 'B' = alien bom; 'M' = alien mau
# Se o tipo não for 'B' ou 'M' a função não vai fazer nada (nenhuma # mensagem será impressa

def mensagem_alien(pessoa, tipo_alien):
   if (tipo_alien == 'B') or (tipo_alien == 'b'):
       print(pessoa, ', eu vim em missão de paz!')

   elif (tipo_alien == 'M') or (tipo_alien == 'm'):
       print(pessoa, ', eu vim te abduzir e escravizar!')

# programa principal – testando a função
mensagem_alien('Bruno Mars', 'B')
print()
mensagem_alien('Kanye West', 'M')
```

Saída:

```
Bruno Mars, eu vim em missão de paz!

Kanye West, eu vim te abduzir e escravizar!
```

Exercício 4 - Sejam P(x1,y1) e Q(x2,y2) dois pontos quaisquer do plano. A distância entre eles é dada por: d = raiz($(x2 - x1)^2 + (y2 - y1)^2$).
• Crie uma função que receba como entrada os valores de x1, y1, x2 e y2, representando as coordenadas de dois pontos, e que compute como saída a distância entre estes pontos.

```
def dist_euclid(x1, y1, x2, y2):
    return ((x1 - x2)**2 + (y1 - y2)**2)**0.5

# programa principal – testando a função
P = (-3, -11);
Q = (2, 1)

print(dist_euclid(P[0], P[1], Q[0], Q[1]))
```

Saída:

```
13.0
```

<u>Exercício 5</u> - A Sequência de Fibonacci tem como primeiros termos os números 0 e 1 e, a seguir, cada termo subsequente é obtido pela soma dos dois termos predecessores:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

• Crie uma função "Fibonacci(n)" que retorne como saída uma lista contendo os n primeiros números da sequência de Fibonacci. Utilize essa função em um programa em que o usuário digite o valor de n para, em seguida, ver a sequência. O programa deve ser mantido em execução enquanto o usuário desejar.

```
def fibonacci(n):
    if n <= 0: return None
    elif n == 1: return [0]
    elif n == 2: return [0, 1]
    else:
        fib = [0, 1]
        for i in range(n-2):
            fib.append(fib[i] + fib[i+1])
        return fib

# testando a função
for i in range(1, 14): print(i, fibonacci(i))
```

Saída:

```
1 [0]
2 [0, 1]
3 [0, 1, 1]
4 [0, 1, 1, 2]
5 [0, 1, 1, 2, 3]
6 [0, 1, 1, 2, 3, 5]
7 [0, 1, 1, 2, 3, 5, 8]
8 [0, 1, 1, 2, 3, 5, 8, 13]
9 [0, 1, 1, 2, 3, 5, 8, 13, 21]
10 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
11 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
12 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
13 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```

<u>Exercício 6</u> – Para ser válida, uma sequência de DNA não pode ser vazia e deve possuir apenas as seguintes letras:

- **A - Adenina;**
- **T - Timina;**
- **C - Citosina;**
- **G – Guanina.**

Um exemplo é "TCGCGCAACGTCGCCTACATCTCAAGATTCA"

(a) Crie uma função chamada "seq_dna_valida" que receba como entrada uma sequência de DNA em uma string. Como saída, a função deverá retornar:

- **True, se a sequência for válida ou False, se a sequência for inválida.**

(b) Crie uma função chamada "percentual_gc" que receba como entrada uma sequência de DNA (string) e calcule número de ocorrências das letras G e C em relação ao número total de letras da sequência, retornando o valor obtido.

- **Exemplo: na sequencia "GAGCT", o valor retornado seria 3/5 = 0.6, pois das 5 letras, 3 delas são G ou C.**

- **A função “percentual_gc” deverá usar a função “seq_dna_valida” para verificar se a sequência recebida é válida ou não. Caso seja inválida, o valor None deverá ser retornado.**

```
def seq_dna_valida(sequencia):
    s = sequencia.upper()
    if  (len(s) > 0 and
         len(s) == s.count("A") + s.count("T") + s.count("C") + s.count("G")):
        return True
    else:
        return False


def percentual_gc(sequencia):
    if not seq_dna_valida(sequencia):
        return None
    else:
        s = sequencia.upper()
        return (s.count("C") + s.count("G")) / len(s)


# teste 1: sequência válida
s1 = "GAGCT"
print(f"'{s1}' é válida? -> {seq_dna_valida(s1)}")
print(f" %gc = {percentual_gc(s1)}\n")

# teste 2: outra sequência válida
s2 = "TCGCGCaaCGTCGCCTACATCTcAAGATTCA"
print(f"'{s2}' é válida? -> {seq_dna_valida(s2)}")
print(f" %gc = {percentual_gc(s2)}\n")

# teste 3: sequência inválida
s3 = "GAXCT"
print(f"'{s3}' é válida? -> {seq_dna_valida(s3)}")
print(f" %gc = {percentual_gc(s3)}\n")

# teste 4: sequência vazia
s4 = ""
print(f"'{s4}' é válida? -> {seq_dna_valida(s4)}")
print(f" %gc = {percentual_gc(s4)}\n")
```

Saída:

```
'GAGCT' é válida? -> True
 %gc = 0.6

'TCGCGCaaCGTCGCCTACATCTcAAGATTCA' é válida? -> True
 %gc = 0.5161290322580645

'GAXCT' é válida? -> False
 %gc = None

'' é válida? -> False
 %gc = None
```

<u>Exercício 7</u> – O módulo de um vetor *n*-dimensional *v* pode ser computado por:

$$||v|| = \sqrt{{v_{[1]}}^2 + {v_{[2]}}^2 + \cdots + {v_{[n]}}^2}$$

Crie uma função que receba como entrada um vetor numérico armazenado em uma tupla e retorne o seu módulo. Caso seja passada uma tupla vazia, a função deverá retornar None.

```
def f_modulo(v):
    if not v: return None

    n = len(v)
    soma_quadrados = 0
    for i in range(n):
        soma_quadrados += v[i]**2

    return soma_quadrados**0.5


# testando a função
v1 = (3, -4)                    # módulo = 5.0
print('módulo de', v1,'=',f_modulo(v1))

v2 = (0, 2, 1, 2, 0)            # módulo = 3.0
print('módulo de', v2,'=',f_modulo(v2))

v3 = (1, 2, 1, 1)               # módulo = 2.646
print('módulo de', v3,'=',f_modulo(v3))

v4 = (0, 0, 0, 0, 0, 0, 0, 0)   # módulo = 0.0
print('módulo de', v4,'=',f_modulo(v4))

v5 = tuple()                    # módulo = 0.0
print('módulo de', v5,'=',f_modulo(v5))
```

<u>Exercício 8</u> – Escreva uma função chamada "ComparaDics", que receba dois dicionários d1 e d2 como entrada. Como saída a função deverá retornar uma tupla com 2 elementos, onde cada um deles é uma lista:

- **1º elemento: lista contendo as chaves que estão presentes nos dois dicionários (ou lista vazia caso não exista nenhuma chave em comum).**
- **2º elemento: lista contendo os valores que estão presentes nos dois dicionários (ou lista vazia caso não exista nenhum valor em comum).**

```
def ComparaDics(d1, d2):
    lst_chaves = []
    lst_valores = []
    for chave, valor in d1.items():
        if chave in d2:
            lst_chaves.append(chave)

        if valor in d2.values() and valor not in lst_valores:
            lst_valores.append(valor)

    return lst_chaves, lst_valores


# testando a função
d1 = {'C1': 101, 'C2': -15, 'C5': -2}
d2 = {'C2': 47, 'C7': -2, 'C1': 0, "C4": -18}

c =  ComparaDics(d1, d2)
print(c)    # (['C1', 'C2'], [-2])
```

Exercício 9 - O arquivo "gols.csv" armazena o número de gols marcados pelo time REAL MYDREADS na copa de Jamaica. O arquivo armazena a data e o número de gols marcados em cada jogo. Os dados estão separados por tabulação (tecla tab do teclado, Unicode 9) e o arquivo está no formato ANSI. Faça um programa que compute o total de gols marcado pela equipe e a média de gols por jogo. Suponha que o arquivo está armazenado na pasta "C:/bases"

```
05/06/2019 1
09/06/2019 0
16/06/2019 5
19/06/2019 2
23/06/2019 1
27/06/2019 3
30/06/2019 0
```

Resolução: nesse caso temos um arquivo CSV onde os dados estão separados por **tabulação**, que corresponde ao caractere especial '\t'. Como o formato do arquivo é ANSI, não é preciso especificar o encoding na função open. Observe ainda que o arquivo não tem cabeçalho, então não é preciso pular a linha de cabeçalho antes de processá-lo.

```
arq = open('C:/bases/gols.csv')

soma_gols = tot_jogos = 0

for linha in arq:
    linha = linha.rstrip()
    lstDados = linha.split("\t")  # quebra a linha em uma lista


    soma_gols += int(lstDados[1])      # atualiza o total de gols
    tot_jogos += 1                     # atualiza o total de jogos

arq.close() # fecha o arquivo, já que ele foi todo processado


# imprime as estatísticas do time
media_gols = soma_gols / tot_jogos
print('O Real MyDreads marcou {} gols no torneio!'.format(soma_gols))
print('Média de {:.2f} gols por jogo.'.format(media_gols))
```

Exercício 10 - Considere um arquivo CSV chamado "paises.csv" no formato UTF-8, na pasta "c:/dados/estatisticas_paises", em que cada registro armazena os dados de um país. Cada país é descrito por 5 variáveis:

- **sigla**
- **nome**
- **continente**
- **população**
- **área em km²**

Abaixo um exemplo que apresenta a linha de cabeçalho e os dados dos 5 primeiros países do arquivo (neste arquivo, os países não estão ordenados por nenhum critério específico).

```
sigla,nome,continente,população,area
BRA,Brasil,América,212559409,8510345
AUS,Austrália,Oceania,25499881,7741220
POR,Portugal,Europa,10196707,92090
CRI,Costa Rica,América,5094114,51100
KOR,Coréia do Sul,Ásia,51269183,99720
...
```

Faça um programa que seja capaz de processar o arquivo para exibir na tela os seguintes resultados:

(a) Em que linha do arquivo (informe o número), o país com a sigla "NZL" aparece. Por exemplo, no arquivo acima BRA aparece na linha 1, AUS na linha 2, etc. (veja que a linha de cabeçalho é desconsiderada)

(b) O total de países do continente 'Oceania'

(c) A densidade populacional do continente 'Oceania'. A densidade populacional da Oceania é calculada através da soma da população de todos os países que pertencem a este continente, dividida pela soma da área ocupada por estes mesmos países.

Resolução: nesse caso temos um arquivo CSV com os dados separados por vírgula. O enunciado indicou que o arquivo está no formato UTF-8, logo é preciso indicar esse encoding na função open

```
f = open("c:/dados/estatisticas_paises/paises.csv", encoding='utf-8')

f.readline() # pula o cabeçalho

linha_nzl = None          # terá a linha da nova zelandia
soma_pop_oceania = 0      # soma da população da oceania
soma_area_oceania = 0     # soma da área da oceania
tot_paises_oceania = 0    # total de países da oceania

N = 0                     # armazena o número da linha a cada iteração

# for para processar o arquivo sequencialmente
for linha in f:
    linha = linha.strip()
    dados = linha.split(",")

    N += 1

    # se cheguei na linha da Nova Zelândia, pego ela
    if dados[0] == 'NZL':
        linha_nzl = N

    # atualiza as varíaveis dos países da oceania
    if dados[2] == 'Oceania':
        tot_paises_oceania += 1
        soma_pop_oceania += int(dados[3])
        soma_area_oceania += int(dados[4])


# agora é só exibir os resultados
dens_oceania = soma_pop_oceania / soma_area_oceania

print('Linha da Nova Zelândia:', linha_nzl)
print('Total de países da oceania:', tot_paises_oceania)
print(f'Densidade populacional da oceania: {dens_oceania:.2f}')

f.close()
```