FOLHA DE RESPOSTAS

RICHARD ARAUJO SMARSI BENTO

EXERCÍCIO 1:

```
'''
    Crie rotinas para conversão de temperaturas da escala
Celsius (°C) para a escala
    Fahrenheit (°F) e vice-versa.

'''

def celsius_fahrenheit(temperatura):
    resultado = (( temperatura * 9) / 5) + 32
    return ("%.2f" % resultado)

def fahrenheit_celsius(temperatura):
    resultado = ((temperatura - 32 ) * 5 ) / 9
    return ("%.2f" % resultado)


#Casos de teste:
print(celsius_fahrenheit(26)) #resultado esperado : 78,8
print(celsius_fahrenheit(35)) #resultado esperado : 95
print(fahrenheit_celsius(80)) #resultado esperado : 26,67
print(fahrenheit_celsius(77)) #resultado esperado : 25
```

Para a resolução deste exercício foram criadas 2 funções. Veja abaixo o que cada uma faz:

- **Função celsius_fahrenheit()**: Recebe uma temperatura em celcius e retorna em fahrenheit.

- **Função fahrenheit_celsius()**: Recebe uma remperatura em fahrenheit e retorna em celcius.

Veja abaixo um exemplo de entrada/saida do programa.

```
C:\Users\Richard\Documents\PROJETOS\Entrevistas\Argotechno\ex1>py ex1.py
78.80
95.00
26.67
25.00
```

EXERCÍCIO 2:

```
'''
Crie uma rotina para a conversão de uma lista de
temperaturas. Utilize a lista abaixo
como exemplo de argumento de entrada para a rotina. Para
o último item da lista, crie a
rotina para a conversão da escala Kelvin para Celsius

'''

# declaração das funções usadas para os cálculos --------
------------

def celsius_fahrenheit(temperatura):
    resultado = (( float(temperatura) * 9) / 5) + 32
    return ("%.2f" % resultado)

def fahrenheit_celsius(temperatura):
    resultado = ((float(temperatura) - 32 ) * 5 ) / 9
    return ("%.2f" % resultado)

def kelvin_celcius(temperatura):
    resultado = float(temperatura) - 273.15
    return ("%.2f" % resultado)

def celcius_kelvin(temperatura):
    resultado = float(temperatura) + 273.15
    return ("%.2f" % resultado)

def kelvin_fahrenheit(temperatura):
    em_celcius = kelvin_celcius(float(temperatura))
    resultado = celsius_fahrenheit(float(em_celcius))
    return ("%.2f" % float(resultado))

def fahrenheit_kelvin(temperatura):
    em_celcius = fahrenheit_celsius(float(temperatura))
    resultado = celcius_kelvin(float(em_celcius))
    return ("%.2f" % float(resultado))
```

```
# Fim das declarações de função ------------------------
----------------------

quantidade = int(input("Quantas temperaturas você irá
converter? "))
contador = 1
valores = {}
while contador <= quantidade:
    print("")
    print("")
    print("")
    print("Vamos cadastrar o item "+str(contador)+" de
"+str(quantidade))
    valores[str(contador)] = {}

    valores[str(contador)]['temperatura'] =
float(input("Digite a temperatura: "))
    valores[str(contador)]['escalaOrigem'] =
input("Digite a Escala de origem (C , F ou K): ")
    valores[str(contador)]['escalaDestino'] =
input("Digite a Escala de Destino (C , F ou K): ")

    if (valores[str(contador)]['escalaOrigem'] ==
valores[str(contador)]['escalaDestino']):
        print("ERRO: A escala de origem é igual a escala
de destino! Por favor digite novamente ou aperte 'CTRL +
C' para interromper o programa.")
    else:
        contador = contador + 1

print('-----------------------------------------------
-----------------------------------')
print('   Temperatura  |    Escala de
Origem    |    Escala de Destino    |    Resultado     ')
#Cabeçalho da tabela

for i in valores:
```

```
    origem = valores[i]['escalaOrigem']
    destino = valores[i]['escalaDestino']
    temperatura = valores[i]['temperatura']

    # Verificação para transformação do valor (chamando
as funções específicas de cada transformação)
    if origem == 'c' or origem == 'C':
        if destino == 'f' or destino == 'F':
            resultado_final =
celsius_fahrenheit(temperatura)
        if destino == 'k' or destino == 'K':
            resultado_final = celcius_kelvin(temperatura)

    if origem == 'f' or origem == 'F':
        if destino == 'c' or destino == 'C':
            resultado_final =
fahrenheit_celsius(temperatura)
        if destino == 'k' or destino == 'K':
            resultado_final =
fahrenheit_kelvin(temperatura)

    if origem == 'k' or origem == 'K':
        if destino == 'c' or destino == 'C':
            resultado_final = kelvin_celcius(temperatura)
        if destino == 'f' or destino == 'F':
            resultado_final =
kelvin_fahrenheit(temperatura)

    # Exibindo os itens da tabela final (com os
resultados)
    print('     {}        |           {}           |
    {}                |        {}       '.format(temperat
ura, origem, destino, resultado_final))

print('----------------------------------------------------
-----------------------------------')
```

Veja abaixo a descrição do programa:

- **Linha 8 até 36**: Declaração das funções usadas nas conversões de temperatura (cada função recebe a temperatura em uma escala e retorna em outra).

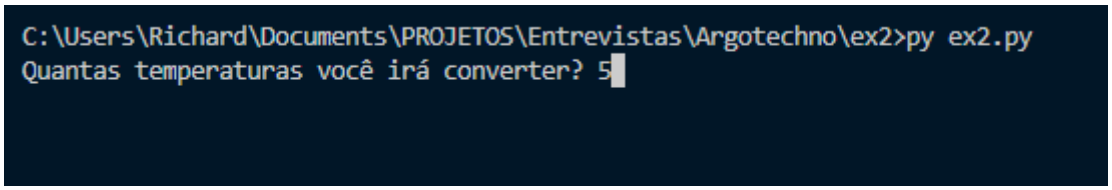
- **Linha 38 até 55**: Cuida das entradas para o programa funcionar. Neste caso por um motivo mais "didáticos" escolhi criar um laço que perdura até informarmos os dados para a quantidade de temperaturas que escolhemos "cadastrar". Basicamente o programa pergunta quantas temperaturas queremos converter e em seguida pergunta a temperatura e as escalas de origem e destino. Com esses dados montamos um dicionario para converter-los.

- **Linha 57, 58 e 87**: Monta uma espécie de "Tabela" no console do sistema.

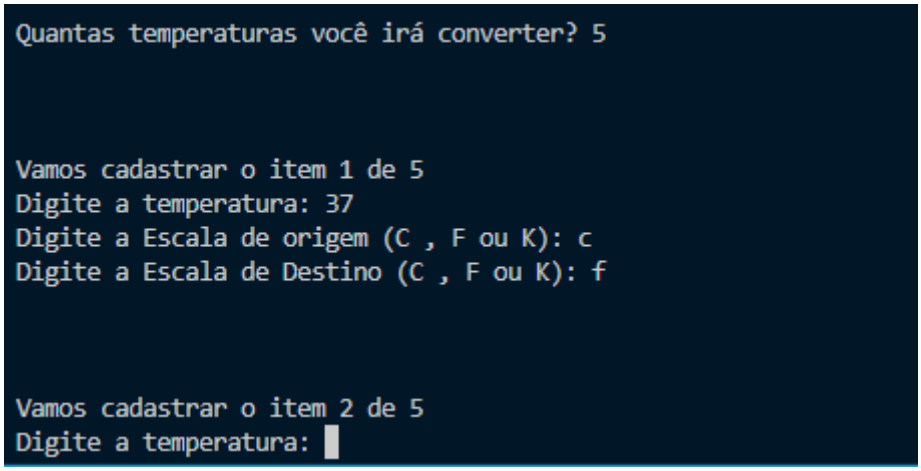
- **Linha 60 até 85**: Trata todos os dados fazendo as condicionais e chamando as funções específicas de conversão. A linha 85 exibe esses dados de acordo com a tabela formada.

Veja abaixo um exemplo completo de execução do programa:

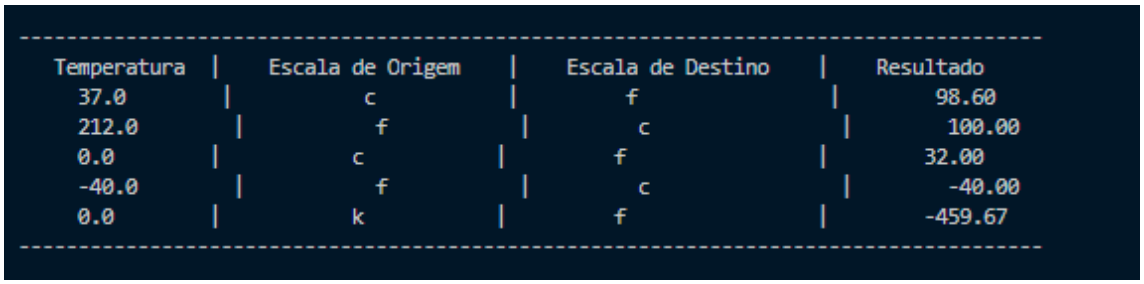
Ao executar o programa escolha quantas temperaturas quer converter (no caso do exercício é pedido uma tabela com 5 itens).



Agora iremos cadastrar a temperatura e a escala de origem para cada item (é esperado c, f ou k como parametros de escala).



Ao terminar o programa irá converter as temperaturas e imprimir uma espécie de tabela no console (não gastei muito tempo tentando estilizar este passo... afinal não faz parte do objetivo aqui.O intuíto era tornal mais 'visual').



EXERCÍCIO 3:

```
'''
Assumindo um edifício com 10 andares e 4 apartamentos por
andar, crie uma rotina
que apresente o número que deve ser colocado na porta de
cada apartamento do edifício.
Considere que esta rotina também será utilizada em
prédios com diferentes números de
andares e apartamentos por andar.

'''

# ATENÇÃO: PARA EXECUÇÃO DESTE EXERCÍCIO ESTOU
DESCONSIDERANDO O TÉRREO (CONTABILIZANDO APARTAMENTOS
APENAS A PARTIR DO 1º ANDAR)

def numeros_apartamentos(andares, apt_por_andar): #recebe
o numero de andares e o numero de apartamentos por andar
respectivamente

    if(apt_por_andar < 10):
        multiplicador = 10
    if(apt_por_andar >= 10):
        multiplicador = 100
    if(apt_por_andar > 100):
        multiplicador = 1000
    if(apt_por_andar > 1000):
        multiplicador = 10000


    resultados = {}
    contador = 1
    while contador <= andares:
        c = 1
        apts = [(contador * multiplicador)+1]
        while c < apt_por_andar:
            last_apt = apts[-1]
            apts.append(last_apt+1)
```

```
            c = c + 1
        resultados[str(contador)+'º Andar'] = apts

        contador = contador + 1

    return resultados



print(numeros_apartamentos(10, 4))
print(numeros_apartamentos(10, 10))
#print(numeros_apartamentos(60, 30))
```

Para a resolução deste exercício foi criada 1 função. Veja abaixo o que ela faz:

- **Função numeros_apartamentos()**: Recebe o numero de andares e o numero de apartamentos por andar. Realiza o calculo do numero de cada apartamento multiplicando o numero do andar por 10 (dez), 100 (cem) ou 1000 (mil) dependendo do número de apartamentos por andar.Essa função retorna um dicionário python com os andares e uma lista de seus respectivos apartamentos.Confira abaixo uma imagem de exemplo da saída deste programa, ou se preferir acesse o arquivo "resultado.txt" que se encontra neste repositório.

```
C:\Users\Richard\Documents\PROJETOS\Entrevistas\Argotechno\ex3>py ex3.py
{'1º Andar': [11, 12, 13, 14], '2º Andar': [21, 22, 23, 24], '3º Andar': [31, 32, 33, 34], '4º Andar': [41, 42, 43, 44], '5º Andar': [51, 52, 53, 54], '6º Andar': [61, 62, 63, 64],
'7º Andar': [71, 72, 73, 74], '8º Andar': [81, 82, 83, 84], '9º Andar': [91, 92, 93, 94], '10º Andar': [101, 102, 103, 104]}
{'1º Andar': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110], '2º Andar': [201, 202, 203, 204, 205, 206, 207, 208, 209, 210], '3º Andar': [301, 302, 303, 304, 305, 306, 307, 308,
 309, 310], '4º Andar': [401, 402, 403, 404, 405, 406, 407, 408, 409, 410], '5º Andar': [501, 502, 503, 504, 505, 506, 507, 508, 509, 510], '6º Andar': [601, 602, 603, 604, 605, 606
, 607, 608, 609, 610], '7º Andar': [701, 702, 703, 704, 705, 706, 707, 708, 709, 710], '8º Andar': [801, 802, 803, 804, 805, 806, 807, 808, 809, 810], '9º Andar': [901, 902, 903, 90
4, 905, 906, 907, 908, 909, 910], '10º Andar': [1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010]}

C:\Users\Richard\Documents\PROJETOS\Entrevistas\Argotechno\ex3>
```

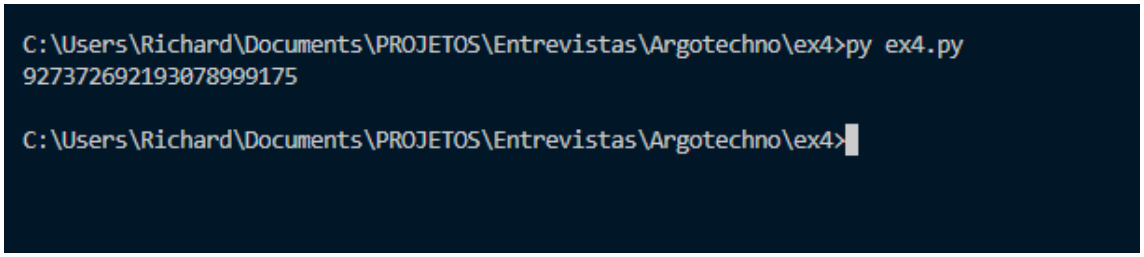
As ultimas linhas do programa estão com os dados (colocados manualmente no código), porém, em um contexto sistemico esses dados deveriam ser recebidos de algum lugar ou programa "automaticamente".

EXERCÍCIO 4:

```
'''
    Crie uma rotina que calcule a soma dos 100 primeiros
elementos da séria de
    Fibonacci (0, 1, 1, 2, 3, 5, 8, ...).

'''


sequencia = [] ##Iniciando a sequencia de Fibonacci
ultimo = 1
penultimo = 1
sequencia.append(penultimo)
sequencia.append(ultimo)

contador = 3

while contador <= 100:
    numero = ultimo + penultimo
    sequencia.append(numero)
    penultimo = ultimo
    ultimo = numero

    contador = contador + 1

print(sum(sequencia))
```

Para este exercício foi utilizado um laço While para formar a sequencia de Fibonacci com seus primeiros 100 Elementos dentro de uma lista python.

O programa retorna a soma de todos os elementos da lista através da função sum().



EXERCÍCIO 5:

```
import re

def Verifica_Validade(data):
    data_atual = 2011 #poderia pegar a data atual ou
receber como parametro da função
    data = int(data)

    if data >= data_atual:
        return "Valido"
    else:
        return "Vencido"

def Identificador(numero_identificador):
    bancos = [['2', 'kappa'], ['85', '86', 'omicron'],
['88', 'iota'], ['72', '76', 'kappa'], ['75', 'sigma']]

    for banco in bancos:
        tamanho_verificador = len(banco[0])

        if((re.findall(r"\d{"+str(tamanho_verificador)+"}
", numero_identificador)[0]) == banco[0] or
(re.findall(r"\d{"+str(tamanho_verificador)+"}",
numero_identificador)[0]) == banco[1]):
            return banco[-1]


def Tratamento_Cartao(serial):

    quebra = serial.split("=")
    data_expircacao = re.findall(r"\d{4}", quebra[1])[0]
    pan = quebra[0]
    identificador_banco = re.findall(r"\d{6}",
(pan.split(";")[1])  )[0]

    nome_banco = Identificador(identificador_banco)
    valido = Verifica_Validade(data_expircacao)
```

```
    dados_cartao = {
        "cartao"+ serial: {
            "nome_banco": nome_banco,
            "identificador_banco": identificador_banco,
            "data_expiracao": data_expircacao,
            "valido?": valido,
            "codigo_pan": pan
        }
    }

    return (dados_cartao)



cartao1 = ";854922420655947=20087011490683843?"
cartao2 = ";722792821249=190220153666234?"
cartao3 = ";8657607910110=2209701507597562?"
cartao4 = ";6034523459017=24032012187993726?"
cartao5 = ";83407977524115=2010701164703842?"
cartao6 = ";22554987787910=1903201221224791?"
cartao7 = ";7621767951747=21112018460506111?"
cartao8 = ";24478927568913=230470179307259?"
cartao9 = ";88674481889649=19062014166695784?"
cartao10 = ";76985229117350=1805701127970335?"
cartao11 = ";2147686439882=2712701977197697?"
cartao12 = ";86392841965929=2108201878359745?"

print(Tratamento_Cartao(cartao1))
print(Tratamento_Cartao(cartao2))
print(Tratamento_Cartao(cartao3))
print(Tratamento_Cartao(cartao4))
print(Tratamento_Cartao(cartao5))
print(Tratamento_Cartao(cartao6))
print(Tratamento_Cartao(cartao7))
print(Tratamento_Cartao(cartao8))
print(Tratamento_Cartao(cartao9))
print(Tratamento_Cartao(cartao10))
print(Tratamento_Cartao(cartao11))
print(Tratamento_Cartao(cartao12))
```

Para a resolução deste exercício foram criadas 3 funções. Veja abaixo o que cada uma faz:

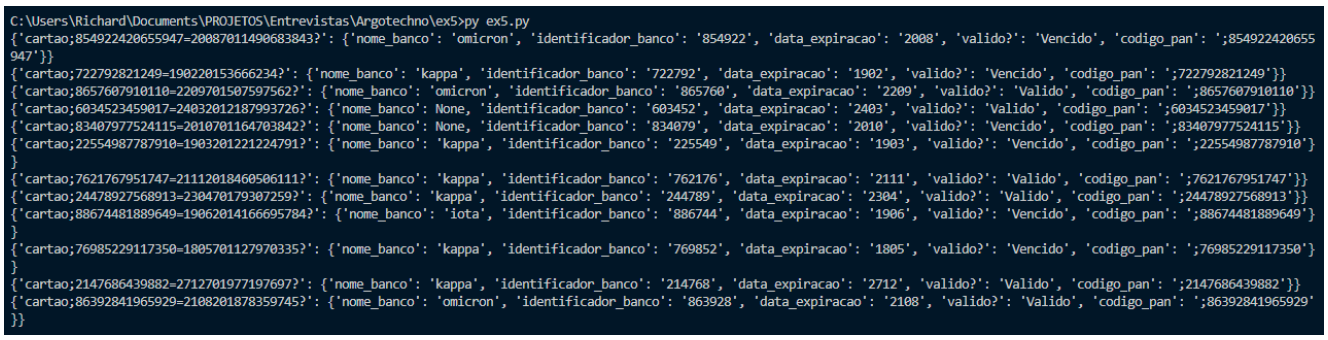
- **Função Verifica_Validade()**: Recebe uma data (retirada da trilha do cartão) e compara com a data atual. Retorna se o cartão é válido ou está expirado.

- **Função Identificador()**: Recebe os numeros identificadores retirados da trilha do cartão. Dentro desta função temos uma lista de bancos e seus códigos identificadores (considerei que nessa lista o último elemento sempre é o nome do banco). A função conta quantos digitos são necessários para fazer a comparação e percorre a lista verificando se os identificadores do cartão são compatíveis com os de algum banco. A função retorna o nome do banco.

- **Função Tratamento_Cartao()**: Recebe a trilha do cartão e destrincha todos os dados. Essa é a função principal do programa e está chamando dentro dela as demais funções. Retorna um objeto (json) com todos os dados do cartão descritos. O resultado é igual a imagem abaixo. Também é possível ver um arquivo de resultados dentro do repositório (resultado.json)

As ultimas linhas do programa estão com os dados do cartão (colocados manualmente no código), porém, em um contexto sistemico esses dados seriam recebidos por meio de uma rotina "automatizada".

Utilizei a biblioteca "re" do python para poder encontrar os digitos na quantidade desejada.



RESULTADO FINAL EX5:


```
{
    'cartao;854922420655947=20087011490683843?': {
      'nome_banco': 'omicron',
      'identificador_banco': '854922',
      'data_expiracao': '2008',
      'valido?': 'Vencido',
      'codigo_pan': ';854922420655947'
    }
  }{
    'cartao;722792821249=190220153666234?': {
      'nome_banco': 'kappa',
      'identificador_banco': '722792',
      'data_expiracao': '1902',
      'valido?': 'Vencido',
      'codigo_pan': ';722792821249'
    }
  }{
    'cartao;8657607910110=2209701507597562?': {
      'nome_banco': 'omicron',
      'identificador_banco': '865760',
      'data_expiracao': '2209',
      'valido?': 'Valido',
      'codigo_pan': ';8657607910110'
    }
  }{
    'cartao;6034523459017=24032012187993726?': {
      'nome_banco': None,
      'identificador_banco': '603452',
      'data_expiracao': '2403',
      'valido?': 'Valido',
      'codigo_pan': ';6034523459017'
    }
  }{
    'cartao;83407977524115=201070116470384 2?': {
      'nome_banco': None,
      'identificador_banco': '834079',
```

```
    'data_expiracao': '2010',
    'valido?': 'Vencido',
    'codigo_pan': ';83407977524115'
  }
}{
  'cartao;22554987787910=1903201221224791?': {
    'nome_banco': 'kappa',
    'identificador_banco': '225549',
    'data_expiracao': '1903',
    'valido?': 'Vencido',
    'codigo_pan': ';22554987787910'
  }
}{
  'cartao;7621767951747=21112018460506111?': {
    'nome_banco': 'kappa',
    'identificador_banco': '762176',
    'data_expiracao': '2111',
    'valido?': 'Valido',
    'codigo_pan': ';7621767951747'
  }
}{
  'cartao;24478927568913=230470179307259?': {
    'nome_banco': 'kappa',
    'identificador_banco': '244789',
    'data_expiracao': '2304',
    'valido?': 'Valido',
    'codigo_pan': ';24478927568913'
  }
}{
  'cartao;88674481889649=19062014166695784?': {
    'nome_banco': 'iota',
    'identificador_banco': '886744',
    'data_expiracao': '1906',
    'valido?': 'Vencido',
    'codigo_pan': ';88674481889649'
  }
}{
  'cartao;76985229117350=1805701127970335?': {
```

```
    'nome_banco': 'kappa',
    'identificador_banco': '769852',
    'data_expiracao': '1805',
    'valido?': 'Vencido',
    'codigo_pan': ';76985229117350'
  }
}{
  'cartao;2147686439882=2712701977197697?': {
    'nome_banco': 'kappa',
    'identificador_banco': '214768',
    'data_expiracao': '2712',
    'valido?': 'Valido',
    'codigo_pan': ';2147686439882'
  }
}{
  'cartao;86392841965929=2108201878359745?': {
    'nome_banco': 'omicron',
    'identificador_banco': '863928',
    'data_expiracao': '2108',
    'valido?': 'Valido',
    'codigo_pan': ';86392841965929'
  }
}
```