```
# ----------------------------------------------------Enquadramento-----------------------------------------------------
#
# Trabalho da UC de Programação da Licenciatura de Ciência de Dados - Ano letivo 2020/2021- 1º Semestre
# Realizado por Sebastião Manuel Inácio Rosalino - n.º 98437 - Turma CDA1

# Este programa suporta o funcionamento de uma plataforma (eBikeIUL) destinada à gestão de um parque de bicicletas,
# tanto manuais como elétricas, espalhadas pelo campus do Iscte, com o propósito de permitir tanto a deslocação de
# pessoas em pequenos percursos, como também em atividades de lazer.

# Funcionalmente, os utilizadores registam-se na aplicação eBikeIUL, escolhendo um nickname e
# uma password, que depois usarão para poder reservar e usar as eBikeIUL espalhadas pelo campus.
# Assim que o utilizador chega junto da viatura poderá iniciar o aluguer, o que fará destrancar as
# suas rodas e permitir que o utilizador possa usufruir dela, ficando a viatura marcada como ocupada.
# Quando terminar a sua atividade, o utilizador usa novamente a aplicação para terminar o aluguer,
# provocando a operação oposta, tranca as rodas da viatura e marca a mesma como disponível.

# O programa abarca os seguintes conceitos/funcionalidades:
# Viatura: meio de transporte (bicicleta elétrica ou manual) utilizado pelo utilizador. Todas as viaturas têm um nome,
#         que corresponde em geral ao nome de um animal.
# Utilizador: pessoa que pode alugar uma eBikeIUL, de momento restrita ao universo dos estudantes do Iscte.
# Aluguer: registo que contém informação sobre um determinado aluguer.
# Gestor: plataforma eBikeIUL, que permite gerir as viaturas, utilizadores e alugueres.


# Bibliotecas importadas pelo programa: ------------------------------------------------------------------------------

import time
import numpy
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt

# T1 --------------------------- Classes de objetos Viatura e Utilizador ----------------------------------------------
# Uma Viatura é caracterizada pelo seguinte conjunto de atributos: nome (identificador único), modelo,
# tipo eletrica (sim ou não) e preço base (euros por hora).
# Um Utilizador tem um determinado perfil, que inclui nickname, password, nome, e-mail, curso e saldo (em Euros).


class Viatura:
    def __init__(self, nome, modelo, eletrica, preco):
        self.__nome = nome
        self.__modelo = modelo
        self.__eletrica = eletrica
        self.preco = preco

    @property
    def nome(self):
        return self.__nome

    @property
    def modelo(self):
        return self.__modelo
```

```
    @property
    def eletrica(self):
        return self.__eletrica

    def alterar_preco(self):
        nome_viatura = input('Introduza o nome da viatura à qual pretende alterar o preço base: ')
        for u in g.lists_viaturas():
            if nome_viatura == u.nome:
                novo_preco = int(input('Introduza o novo preço desta viatura: '))
                self.preco = novo_preco
                print('Preço alterado com sucesso.')
                return
        print('Viatura não encontrada.')

    def __str__(self):
        return 'Nome: {}, Modelo: {}, Elétrica: {}, Preço: {}'.format(self.nome, self.__modelo, self.__eletrica, self.preco)

    def __eq__(self, other):
        return self.eletrica == other.eletrica and self.preco == other.preco

# Viaturas criados no sistema para efeitos de teste do programa:

v1 = Viatura('cao', 'ccc', 'sim', 1.50)
v2 = Viatura('sapo', 'xyw', 'sim', 1.50)
v3 = Viatura('gato', 'xyz', 'não', 0.5)
v4 = Viatura('foca', 'www', 'sim', 1.50)
v5 = Viatura('pardal', 'uuu', 'não', 0.5)
v6 = Viatura('vaca', 'jjj', 'não', 0.5)
v7 = Viatura('rato', 'rrr', 'não', 0.5)
v8 = Viatura('bode', 'bbb', 'não', 0.5)


class Utilizador:
    def __init__(self, nickname, password, nome, email, curso, saldo):
        self.__nickname = nickname
        self.password = password
        self.__nome = nome
        self.email = email
        self.curso = curso
        self.saldo = saldo

    @property
    def nickname(self):
        return self.__nickname

    @property
    def nome(self):
        return self.__nome

    def __str__(self):
        return "Nickname: {} - Nome: {} - E-mail: {} -  Curso: {} - Saldo: {} €".format(self.__nickname, self.__nome, self.email,
self.curso, self.saldo)
```

```
    def test_password(self):
        if self.password:
            return True
        return False

    def alterar_email(self):
        nickname = input('Diga o seu nickname: ')
        if nickname in g.lista_nicks_de_todos_os_users():
            novo_email = (input('Introduza o seu novo e-mail: '))
            self.email = novo_email
            print('Email alterado com sucesso.')
            return
        else:
            print('Nickname inválido.')

    def alterar_curso(self):
        nickname = input('Diga o seu nickname: ')
        if nickname in g.lista_nicks_de_todos_os_users():
            novo_curso = (input('Introduza o seu novo curso: '))
            self.curso = novo_curso
            print('Curso alterado com sucesso.')
            return
        else:
            print('Nickname inválido.')

    def altera_password(self):
        nickname = input('Introduza o seu nickname: ')
        if nickname not in g.lista_nicks_de_todos_os_users():
            print('Utilizador não encontrado.')
        else:
            if self.password:
                oldpass = input('Introduza a password antiga: ')
                if oldpass == self.password:
                    newpass = input('Introduza a nova password: ')
                    self.password = newpass
                else:
                    print('Password antiga incorreta')


# Utilizadores criados no sistema para efeitos de teste do programa:

u1 = Utilizador('smiro', '7777', 'Sebastião', 'smiro@iscte-iul.pt', 'Ciencia de Dados', 30)
u2 = Utilizador('hrosa', '6666', 'Helder', 'hmsro@iscte-iul.pt', 'Gestao', 25)
u3 = Utilizador('tete', '8888', 'Teresa', 'tete@iscte-iul.pt', 'Sociologia', 50)
u4 = Utilizador('xuxa', '3333', 'Xavier', 'xavier@iscte-iul.pt', 'Arquitetura', 14)
u5 = Utilizador('bigone', '6768', 'Alberto', 'bigone@iscte-iul.pt', 'Ciencia Politica', 5)
u6 = Utilizador('likas', '8435', 'Luis', 'likas@iscte-iul.pt', 'Economia', 18)
u7 = Utilizador('lexa', '9757', 'Leonor', 'lexa@iscte-iul.pt', 'Informatica', 15)
u8 = Utilizador('robben', '0987', 'Rui', 'robben@iscte-iul.pt', 'Gestao', 16)
```

```
# T2 ----------Classe de objetos Aluguer ----------
# Quando o utilizador aluga uma viatura é criado um objeto do tipo Aluguer, que é armazenado em memória enquanto o
# aluguer se encontrar ativo. Um objeto deste tipo tem os seguintes atributos: início do aluguer, nickname do utilizador,
# nome da viatura e preço por hora.


class Aluguer:
    def __init__(self, nickname, nomeviatura, preco):
        self.inicio = time.time()
        self.nickname = nickname
        self.nomeviatura = nomeviatura
        self.preco = preco
        self.tempodecorrido = 0
        self.valor_total = 0

    def contar_tempo(self):
        now = time.time()
        tempo_final = now - self.inicio
        return tempo_final

    def update(self):
        self.tempodecorrido = self.contar_tempo()
        self.valor_total = (self.tempodecorrido / 3600) * self.preco

    def __str__(self):
        return "Nickname: {} - Nome da Viatura: {} - Preço por hora: {} €".format(self.nickname, self.nomeviatura, self.preco)

    def formato_ficheiro(self):
        return "{}, {}, {}, {}, {}, {}\n".format(int(self.inicio), self.nickname, self.nomeviatura, self.preco,
                                                 int(self.tempodecorrido), self.valor_total)


# Alugueres criados no sistema para efeitos de teste do programa:

a1 = Aluguer('smiro', 'cao', 1.50)
a2 = Aluguer('hrosa', 'sapo', 1.50)
a3 = Aluguer('tete', 'gato', 0.50)
a4 = Aluguer('xuxa', 'foca', 1.50)
a5 = Aluguer('bigone', 'pardal', 0.5)


# T3 ----------Classe de objetos Gestor ----------
# O sistema de gestão da plataforma eBikeIUL, é um objeto da classe Gestor e tem informação sobre todas as viaturas,
# utilizadores e alugueres ativos do sistema.
# O gestor faz: Gestão de viaturas; Gestão de utilizadores; Gestão de alugueres e produção de informação de gestão.

class Gestor:
    def __init__(self):
        self.viaturas = []
        self.utilizadores = []
        self.alugueres_ativos = []
```

```
def exists_viatura(self, a):
    for c in self.viaturas:
        if c.nome == a:
            return True
    return False

def show_price(self, a):
    for c in self.viaturas:
        if a == c.nome:
            return c.preco

def add_viatura(self, viatura):
    for c in self.viaturas:
        if c.nome == viatura.nome:
            return
    self.viaturas.append(viatura)

def lists_viaturas(self):
    lista_viaturas = []
    for c in self.viaturas:
        lista_viaturas.append(c)
    return lista_viaturas

def exists_user(self, a):
    for c in self.utilizadores:
        if c.nickname == a:
            return True
    return False

def add_user(self, utilizador):
    for c in self.utilizadores:
        if c.nickname == utilizador.nickname:
            return
    self.utilizadores.append(utilizador)

def lists_users(self):
    for c in self.utilizadores:
        print(c)

def add_saldo(self, nickname, saldo_a):
    for u in self.utilizadores:
        if nickname == u.nickname:
            u.saldo += int(saldo_a)
```

```
def change_password(self, nickname):
    for u in self.utilizadores:
        if nickname == u.nickname:
            if Utilizador.test_password(u):
                oldpass = input('Introduza a password antiga: ')
                if oldpass == u.password:
                    newpass = input('Introduza a nova password: ')
                    u.password = newpass
                    print('Password alterada com sucesso.')
                    return
                else:
                    print('Password antiga incorreta.')

def listar_users_para_info(self):
    lista_utilizadores = []
    for c in self.utilizadores:
        lista_utilizadores.append(c)
    return lista_utilizadores

def lista_nicks_de_todos_os_users(self):
    a1 = []
    for u in self.utilizadores:
        a1.append(u.nickname)
    return a1

def lista_nicks_utilizadores_com_alugueres_ativos(self):
    lista_nicks = []
    for n in self.alugueres_ativos:
        lista_nicks.append(n.nickname)
    return lista_nicks

def lista_cursos_utilizadores(self):
    lista_cursos = []
    for u in self.utilizadores:
        lista_cursos.append(u.curso)
    return lista_cursos

def viatura_disponivel(self, n):
    for c in self.alugueres_ativos:
        if c.nomeviatura == n:
            return False
    return True

def listar_viaturas_disponiveis(self):
    lista_nomes_dos_aa = []
    for a in self.alugueres_ativos:
        lista_nomes_dos_aa.append(a.nomeviatura)
    for v in self.viaturas:
        if v.nome not in lista_nomes_dos_aa:
            print(v)
```

```
def lista_nomes_viaturas(self):
    lista_vit_nomes = []
    for s in self.viaturas:
        lista_vit_nomes.append(s.nome)
    return lista_vit_nomes

def novo_aluguer(self, nickname, nomedaviatura):
    preco = g.show_price(nomedaviatura)
    a = Aluguer(nickname, nomedaviatura, preco)
    self.alugueres_ativos.append(a)

def objeto_aluguer(self, nickname):
    for a in self.alugueres_ativos:
        if nickname == a.nickname:
            return a
    return None

def listar_alugueres_ativos(self):
    for b in self.alugueres_ativos:
        print(b)

def concluir_aluguer(self, nickname):
    a = self.objeto_aluguer(nickname)
    if nickname not in self.lista_nicks_utilizadores_com_alugueres_ativos():
        print('Não é possível concluir um aluguer que não está ativo.')
    else:
        a.update()
        f = open("historico.csv", "a")
        f.write(a.formato_ficheiro())
        f.close()
        self.remove_aluguer(a)
        u = self.objeto_utilizador(nickname)
        u.saldo -= a.valor_total
        print('Aluguer concluido com sucesso.')

def objeto_utilizador(self, nickname):
    for u in self.utilizadores:
        if nickname == u.nickname:
            return u

def add_aluguer(self, aluguer):
    self.alugueres_ativos.append(aluguer)
```

```
def iniciar_aluguer(self, nickname):
    for j in g.lista_nicks_de_todos_os_users():
        if nickname == j:
            nomeviatura = input("Introduza o nome da viatura a utilizar: ")
            if nomeviatura not in g.lista_nomes_viaturas():
                print('Esta viatura não existe.')
                return
            elif not self.viatura_disponivel(nomeviatura):
                print('Não é possível começar um aluguer com uma viatura já em uso.')
                return
            else:
                g.viatura_disponivel(nomeviatura)
                g.novo_aluguer(nickname, nomeviatura)
                print('Aluguer iniciado com sucesso.')
                return
    print('Viatura atualmente em uso.')

def remove_aluguer(self, aluguer):
    self.alugueres_ativos.remove(aluguer)

def ler_ficheiro(self):
    f = open("historico.csv", "r")
    for l in f:
        linha_clean = l.strip()
        print(linha_clean)
    f.close()

def soma_saldos(self):
    soma = 0
    for u in self.utilizadores:
        soma += u.saldo
    return soma

def _5_utilizadores_com_mais_saldo(self):
    lista_5_maiores_saldos = []
    lista_5_utilizadores = []
    for u in self.utilizadores:
        lista_5_maiores_saldos.append(u.saldo)
    b = sorted(lista_5_maiores_saldos)
    c = b[-5:]
    for a in c:
        for u in self.utilizadores:
            if a == u.saldo:
                lista_5_utilizadores.append(u.nome)
    return lista_5_utilizadores
```

```
    def listar_cursos_n_alunos(self):
        lista = {}
        for u in self.utilizadores:
            if u.curso not in lista:
                lista[u.curso] = 1
            else:
                lista[u.curso] += 1
        return lista

# Processo que adiciona à plataforma viaturas, utilizadores e alugueres (pelo gestor) para efeitos de teste do programa:

g = Gestor()
g.add_viatura(v1)
g.add_viatura(v2)
g.add_viatura(v3)
g.add_viatura(v4)
g.add_viatura(v5)
g.add_viatura(v6)
g.add_viatura(v7)
g.add_viatura(v8)
g.add_user(u1)
g.add_user(u2)
g.add_user(u3)
g.add_user(u4)
g.add_user(u5)
g.add_user(u6)
g.add_user(u7)
g.add_user(u8)
g.add_aluguer(a1)
g.add_aluguer(a2)
g.add_aluguer(a3)
g.add_aluguer(a4)
g.add_aluguer(a5)


# T5 ---------------------------------------Actividade por hora e dia da semana-------------------------------------------
# Módulo para produção de informação sobre a atividade da plataforma:
#   a) Número de horas ocupadas por cada cada dia da semana;
#   b) Média de alugueres por cada hora;
#   c) Hora do dia com maior número de alugueres.
# E produção de gráfico que ilustre a atividade por hora e dia da semana.

def AtividadeHD():
    f = open('historico.csv', 'r')
    linhas = f.readlines()
    resposta = []
    for linha in linhas:
        linha_clean = linha.strip()
        lst = linha_clean.split(",")
        t = int(lst[0]) + int(lst[4])
        resposta.append((time.ctime(int(lst[0])), time.ctime(t)))
    f.close()
    return resposta
```

```
def matriz():
    m = numpy.zeros([24, 7])
    resultado = AtividadeHD()
    grafico = {'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3, 'Thu': 4, 'Fri': 5, 'Sat': 6}
    for f in resultado:
        ti = int(f[0][11:13])
        tf = int(f[1][11:13])
        day = grafico[f[0][0:3]]
        for i in range(ti, tf+1):
            m[i][day] += 1
    return m


def numero_horas_ocupadas():
    m = matriz()
    dias_semana = {'Sun': 0, 'Mon': 0, 'Tue': 0, 'Wed': 0, 'Thu': 0, 'Fri': 0, 'Sat': 0}
    for i in m[:, 0]:
        if i != 0:
            dias_semana['Sun'] += 1
    for i in m[:, 1]:
        if i != 0:
            dias_semana['Mon'] += 1
    for i in m[:, 2]:
        if i != 0:
            dias_semana['Tue'] += 1
    for i in m[:, 3]:
        if i != 0:
            dias_semana['Wed'] += 1
    for i in m[:, 4]:
        if i != 0:
            dias_semana['Thu'] += 1
    for i in m[:, 5]:
        if i != 0:
            dias_semana['Fri'] += 1
    for i in m[:, 6]:
        if i != 0:
            dias_semana['Sat'] += 1
    return dias_semana


def media_alugueres_hora():
    soma_elementos = []
    new_list = []
    m = matriz()
    for linha in m:
        soma_elementos.append(sum(linha))
    for e in soma_elementos:
        new_list.append(e/7)
    horas = {'0h': new_list[0], '1h': new_list[1], '2h': new_list[2], '3h': new_list[3], '4h': new_list[4], '5h': new_list[5], '6h':
new_list[6], '7h': new_list[7], '8h': new_list[8], '9h': new_list[9], '10h': new_list[10],
             '11h': new_list[11], '12h': new_list[12], '13h': new_list[13], '14h': new_list[14], '15h': new_list[15], '16h':
```

```
new_list[16], '17h': new_list[17], '18h': new_list[18], '19h': new_list[19], '20h': new_list[20], '21h': new_list[21], '22h':
new_list[22], '23h': new_list[23]}
    return horas


def hora_com_maior_numero_alugueres():
    m = matriz()
    soma_horas = []
    for linha in m:
        soma_horas.append(sum(linha))
    max = soma_horas[0]
    index = 0
    for i in range(len(soma_horas)):
        if soma_horas[i] > max:
            max = soma_horas[i]
            index = i
    return index


def mostrar_grafico():
    mydata = matriz()
    plt.subplots(figsize=(5, 10))
    ax = sb.heatmap(mydata, annot=True, linewidths=.5)
    plt.xticks(np.arange(7) + 0.5, labels=["Seg", "Ter", "Qua", "Qui", "Sex", "Sab", "Dom"])
    ax.figure.show()
    ax.figure.savefig("activity.png")


# T4 ----------------------------------Menu para gestão da Plafaforma eBikeIUL----------------------------------------

# Para a gestão da gestão da plataforma eBikeIUL, o programa deverá apresentar um menu com as seguintes opções:
# 1. Listar utilizadores
# 2. Adicionar saldo
# 3. Alterar password
# 4. Iniciar aluguer
# 5. Concluir aluguer
# 6. Listar alugueres ativos
# 7. Listar viaturas disponíveis
# 8. Informações do sistema
# 9. Atividade por hora e dia da semana
# 0. Sair da aplicação


# ------------------------------------------------------------------------------------------------------------------
```

```
opcao = ""
while opcao != "0":
    print("\n=== Bem-vindo à plataforma eBikeIUL ===\n")
    print("1. Listar utilizadores")
    print("2. Adicionar saldo")
    print("3. Alterar password")
    print("4. Iniciar aluguer")
    print("5. Concluir aluguer")
    print("6. Listar alugueres ativos")
    print("7. Listar viaturas disponíveis")
    print("8. Informações do sistema")
    print("9. Atividade por hora e dia da semana")
    print("0. Sair da aplicação\n")
    opcao = input("Escolha a opção: ")

    if opcao == "1":
        g.lists_users()

    elif opcao == "2":
        nickname = input("Diga o seu nickname: ")
        if not g.exists_user(nickname):
            print('Utilizador não encontrado.')
        else:
            saldo_a = input("Indique o valor a adicionar: ")
            g.add_saldo(nickname, saldo_a)
            print('Saldo adicionado com sucesso.')

    elif opcao == "3":
        nickname = input("Diga o seu nickname: ")
        if nickname not in g.lista_nicks_de_todos_os_users():
            print('Utilizador não encontrado.')
        else:
            g.change_password(nickname)

    elif opcao == "4":
        nickname = input("Introduza o seu nickname: ")
        if nickname not in g.lista_nicks_de_todos_os_users():
            print('Utilizador não encontrado.')
        else:
            g.iniciar_aluguer(nickname)

    elif opcao == '5':
        nickname = input("Introduza o seu nickname: ")
        g.concluir_aluguer(nickname)

    elif opcao == "6":
        g.listar_alugueres_ativos()

    elif opcao == "7":
        g.listar_viaturas_disponiveis()
```

```
    elif opcao == '8':
        print("\nNúmero de viaturas registadas na plataforma: {}\n"
              "Número de utilizadores registados na plataforma: {}\n"
              "Soma do saldo de todos os utilizadores: {} euros\n"
              "Top 5 utilizadores com mais saldo: {}\n"
              "Lista de todos os cursos envolvidos e o respetivo número de utilizadores: {}\n"
              "Número de alugueres ativos: {}".format(len(g.lists_viaturas()), len(g.listar_users_para_info()),
                                                      g.soma_saldos(), g._5_utilizadores_com_mais_saldo(),
                                                      g.listar_cursos_n_alunos(), len(g.alugueres_ativos)))

    elif opcao == '9':
        escolher_opcao = int(input('Pressione 1 para aceder às informações da atividade semanal ou 2 para ver o gráfico: '))
        if escolher_opcao == 1:
            print("\nNúmero de horas ocupadas por cada cada dia da semana: {}\n"
                  "Média de alugueres por cada hora: {}\n"
                  "Hora do dia com maior número de alugueres: {}h".format(numero_horas_ocupadas(), media_alugueres_hora(),
hora_com_maior_numero_alugueres()))
        elif escolher_opcao == 2:
            mostrar_grafico()

    elif opcao == "0":
        print("Obrigado.")
    else:
        print('Opção invalida.')


# ----------------------------------------------------Fim do Programa-------------------------------------------------------
# Sebastião Rosalino
# ISCTE-IUL
```