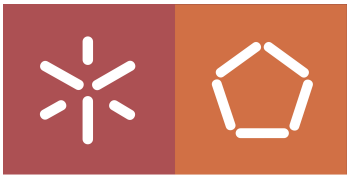

Universidade do Minho
Escola de Engenharia

INTELIGÊNCIA AMBIENTE:
TECNOLOGIAS E APLICAÇÕES

QUESTÃO DE AULA 1

RULE-BASED AUTOMOTIVE CONTROL SYSTEM

JOANA AFONSO GOMES **A84912**

15 de dezembro de 2020

ESCOLHAS INICIAS

Para a realização deste trabalho, que tem como objetivo a a conceção e aplicação de sistemas baseados em regras (SBR), com base num *dataset* definido, escolhi usar a linguagem **Python**, pela conveniência da mesma perante as funcionalidades que são pedidas.

FICHEIRO SEM DUPLICADOS

Após denotar a existência de linhas duplicadas no ficheiro `anexo.csv` (*dataset* fornecido), estabeleci um método para criar um ficheiro .csv sem repetidos. Assim, ao correr o script, vai ser originado o ficheiro `anexo_norep.csv`, que será o usado no resto do programa para os procedimentos necessários.

```
with open('anexo.csv','r') as infile,open('anexo_norep.csv','w') as outfile:
    seen = set()
    for line in infile:
        if line in seen: continue

        seen.add(line)
        outfile.write(line)
```

Listing 1: Criar ficheiro csv sem linhas duplicadas

DEFINIR OUTPUT NO FICHEIRO LOG

Com o objetivo de o output do meu programa ser sempre escrito no ficheiro `result.log`, elaborei o código apresentado de seguida. O *mode* `w` permite que seja criado um ficheiro *log* novo caso não exista, e ainda o permanente *overwrite* dos dados aquando da execução do programa.

```
logger = open ("result.log","w")
sys.stdout = logger
```

Listing 2: Dirigir resultados para o ficheiro `result.log`

ESTABELECER PARÂMETROS BÁSICOS

Com fim a que o ACS (Rule-based Automotive Control System) possa adaptar o comportamento dos dispositivos e sub-sistemas do veículo dependendo das características dos diversos momentos anuais, defini as datas de início das

estações do ano, que iram depois ser utilizadas para limitar os ajustes possíveis no espaço temporal que lhes é correspondente.

```
spring = datetime.strftime((datetime.strptime('03-20','%m-%d')),'%m-%d')
summer = datetime.strftime((datetime.strptime('06-20','%m-%d')),'%m-%d')
fall = datetime.strftime((datetime.strptime('09-22','%m-%d')),'%m-%d')
winter = datetime.strftime((datetime.strptime('12-21','%m-%d')),'%m-%d')
```

Listing 3: Definir datas para o início das estações

Assim sendo, as condições definidas para a **primavera** serão postas em prática de 20 de março a 19 de junho, as do **verão** de 20 de junho a 21 de setembro, de **outono** de 22 de setembro a 20 de dezembro, e de **inverno** de 21 de de dezembro a 19 de março.

ANÁLISE DO PROGRAMA DE LEITURA

Nesta fase de elaboração do programa, utilizo a função `open` para abrir o ficheiro já sem duplicados `anexo_norep.csv` no modo de leitura (`r`) e de seguida o `csv.reader`, separando os diferentes campos pela vírgula. De notar que é impressa uma linha em branco antes de inicializar o ciclo for, para garantir que cada linha do ficheiro log corresponde ao número da linha no ficheiro .csv. Nesta lógica, o ciclo for não começa na linha 0, mas na linha 1, dado que a linha inicial do ficheiro .csv não têm conteúdo a ser analisado pelo programa.

```
with open('anexo_norep.csv', 'r') as file:

    reader = csv.reader(file, delimiter = ',')

    listRead = list(reader)

    print()

    for row in listRead[1:]:
        temp = row[4]
        humidity = row[5]
        dateandhour = datetime.strptime(row[0], '%Y-%m-%d %H:%M:%S')
        day_month = datetime.strftime(dateandhour, '%m-%d')
```

Listing 4: Análise linha a linha do ficheiro `anexo_norep.csv`

Focando-me agora na parte inicial do ciclo, a temperatura em cada uma das linhas é definida pela variável `temp` e a humidade por `humidity`, associando cada uma delas ao índice correto após a separação dos campos (previamente efetuada). Observando o *dataset*, podemos observar que a temperatura corresponde nas várias entries corresponde à quinta coluna (índice 4) e a humidade

à sexta coluna (índice 5). É também, em cada entrada, guardada a data correspondente (da primeira coluna) para identificar posteriormente em que estação do ano se enquadra.

→ TEMPERATURA

No que toca à temperatura, defini para cada estação do ano as **temperaturas de conforto** que se seguem, baseando-me em pesquisa de informação da área e nos valores de referência do enunciado.

Primavera **23º** | Verão **25º** | Outono **20º** | Inverno **17º**

→ HUMIDADE

A partir da análise do *dataset*, decidi também incluir no ficheiro log o comportamento possível de um humidificador. Achei importante fazer o controlo da humidade dado que níveis não ideais e não controlados de humidade podem trazer diversas complicações ao utilizador do veículo.

Se a humidade relativa atingir valores iguais ou superiores a 60%, torna-se um meio propenso à formação de ferrugem, mofo e outras bactérias e fungos, levando o utilizador do veículo a ter problemas de saúde de foro respiratório, e uma maior vulnerabilidade, por exemplo, a gripes e constipações.

Assim, após pesquisa, cheguei a conclusão que a faixa ótima de humidade relativa se encontrará entre os 35% e os 55%. Porém, a humidade é relativamente mais baixa no inverno que no verão, dado que o ar frio contém menos água. Como níveis de humidade adaptados à estação do ano contribuem para nos mantermos-nos frescos no verão e quentes no inverno, procurei ainda informação para distinguir os níveis ótimos de humidade relativamente a cada fase do ano. Cheguei assim à conclusão que no verão deve ser mantido um nível relativo de humidade dentro do veículo inferior a 50%, e no inverno inferior a 40%. Deste modo, estimei os seguintes níveis de humidade relativa por estação:

Primavera **40%** | Verão **45%** | Outono **40%** | Inverno **35%**

Com estes parâmetros definidos (relativos à temperatura e à humidade), há que analisar para cada linha a que estação do ano se refere, comparando a temperatura e a humidade naquela hora com as ideais para essa estação, podendo assim gerir os comportamentos do ar condicionado e o humidificador.

Usando como exemplo o caso de a data da linha em questão pertencer à primavera, demonstrarei de seguida as condições dentro do ciclo que permitem um correto *output* no *log file*.

```
for row in listRead[1:]:
    (...)
    #if spring
    if (day_month >= spring) and (day_month<summer):
        #Temp
        difTemp = round((23.0 - float(temp)),2)
        if difTemp>0 :
            ac = "airconditioning+" + str(difTemp) + "oC"
        else:
            ac = "airconditioning-" + str(-difTemp) + "oC"
        #Humidity
        difHum = 40 - int(humidity)
        if(difHum > 0) :
            hum = "humidifier+" + str(difHum) + "%"
        else:
            hum = "humidifier-" + str(-difHum) + "%"
    (...)
```

Listing 5: Comparações da humidade e temperatura no caso da primavera

Se a data em questão estiver compreendida entre o ínicio da primavera e o início do verão, então pertencerá à primavera. Após entrar nesta condição, a temperatura dessa *entry* será subtraída com a temperatura de referência para a primavera (como indicado previamente, 23°). Se for mais baixa que essa temperatura de referência, por exemplo, 20°, será calculada a `difTemp` sugere o que há que somar à temperatura do *airconditioner* para atingir a de conforto. A variável `ac`, que vai ser parte do *output* para o ficheiro log, será neste caso `airconditioning+3.0°C`. Se a temperatura fosse maior do que a de conforto, em vez de um + estaria um - e a respetiva diferença de temperatura.

A linha de raciocínio é a mesma para o caso da humidade, sendo o *output*, por exemplo, `humidifier+30%` ou `humidifier-30%`.

O procedimento repete-se de igual forma para as outras estações do ano e as linhas do *dataset* que lhe são correspondentes.

FICHEIRO LOG

Ao invés de criar dois ficheiros log distintos para a temperatura e para a humidade, tenho apenas um ficheiro log (`result.log`) em que os *outputs* de ambas as partes estão agrupados em duas colunas.

Para garantir isto, existe, no fim do ciclo `for` (fora das condições) o `print` que se segue, sendo as variáveis `ac` e `hum` explicadas nos verbatins anteriores.

```
1 print("%-25s %-4s %s" %(ac, "|" ,hum))
```

Listing 6: Print dos dados em colunas no log

Deste modo, o ficheiro log obterá uma forma como o excerto da figura seguinte.

```
5   airconditioning+6.98ºC    |    humidifier-55%
6   airconditioning+7.17ºC    |    humidifier-55%
7   airconditioning+7.25ºC    |    humidifier-54%
8   airconditioning+7.48ºC    |    humidifier-56%
9   airconditioning+7.41ºC    |    humidifier-55%
10  airconditioning+7.53ºC    |    humidifier-54%
11  airconditioning+7.15ºC    |    humidifier-53%
12  airconditioning+6.47ºC    |    humidifier-49%
13  airconditioning+5.41ºC    |    humidifier-45%
14  airconditioning+4.93ºC    |    humidifier-43%
15  airconditioning+4.68ºC    |    humidifier-37%
16  airconditioning+4.64ºC    |    humidifier-35%
17  airconditioning+4.55ºC    |    humidifier-36%
18  airconditioning+4.86ºC    |    humidifier-39%
19  airconditioning+5.61ºC    |    humidifier-53%
20  airconditioning+5.66ºC    |    humidifier-55%
21  airconditioning+5.71ºC    |    humidifier-57%
22  airconditioning+5.72ºC    |    humidifier-56%
23  airconditioning+5.49ºC    |    humidifier-55%
24  airconditioning+5.93ºC    |    humidifier-56%
25  airconditioning+5.6ºC     |    humidifier-54%
26  airconditioning+5.56ºC    |    humidifier-45%
27  airconditioning+5.66ºC    |    humidifier-38%
28  airconditioning+5.85ºC    |    humidifier-35%
29  airconditioning+6.27ºC    |    humidifier-38%
30  airconditioning+6.34ºC    |    humidifier-26%
```

Figura 1: Parte do ficheiro *log*