Programação II

Formatos de dados

Hugo Pacheco

DCC/FCUP 20/21

Dados

- Nos ficheiros que temos visto até agora (extratos de livros), os dados estão em texto livre
- Mas têm sempre alguma estrutura, por exemplo, hierarquia de cantos e estrofes nos lusíadas

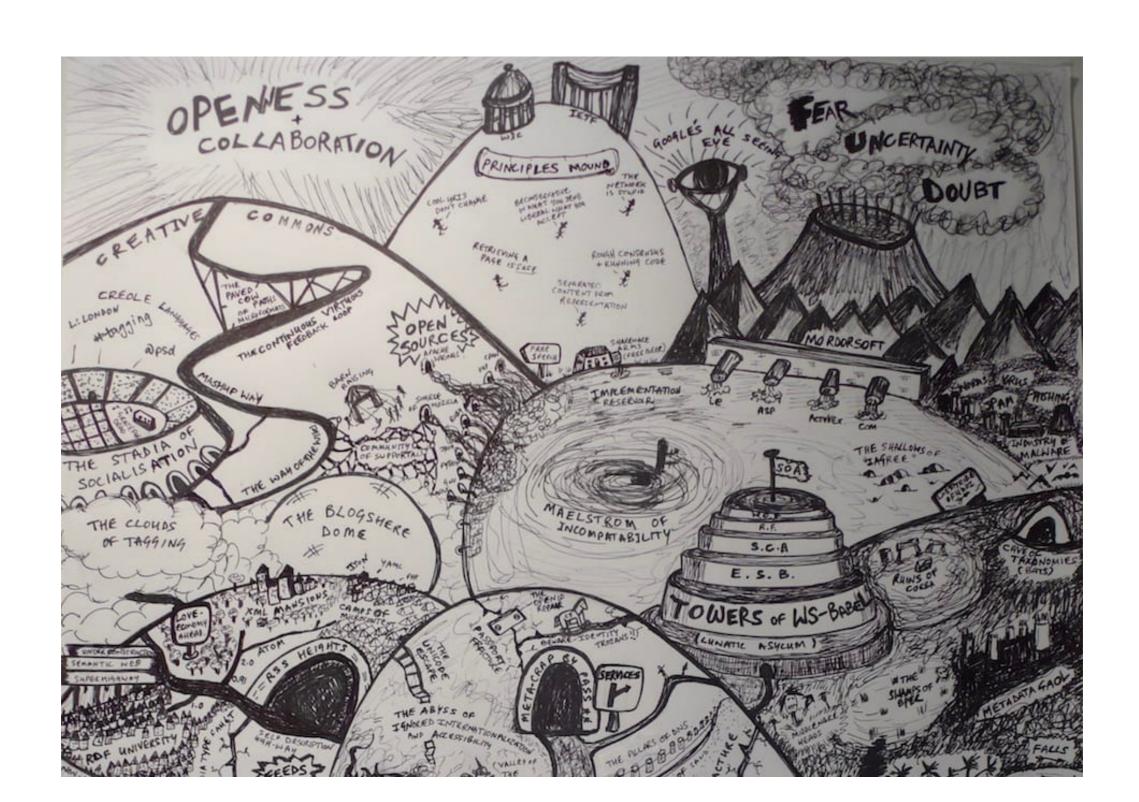
"Canto Primeiro

As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;
[...]"

Dados

- Para ler os dados temos que explorar a sua estrutura.
- Mas se estão em texto livre...
 - pode ser difícil encontrar padrões
 - cada caso é um caso
- Idealmente, diferentes conjuntos de dados devem seguir uma estrutura comum:
 - facilitar a partilha
 - garantir preservação e suporte de longo prazo
 - melhores ferramentas

"The Web is Agreement"



Formatos de dados

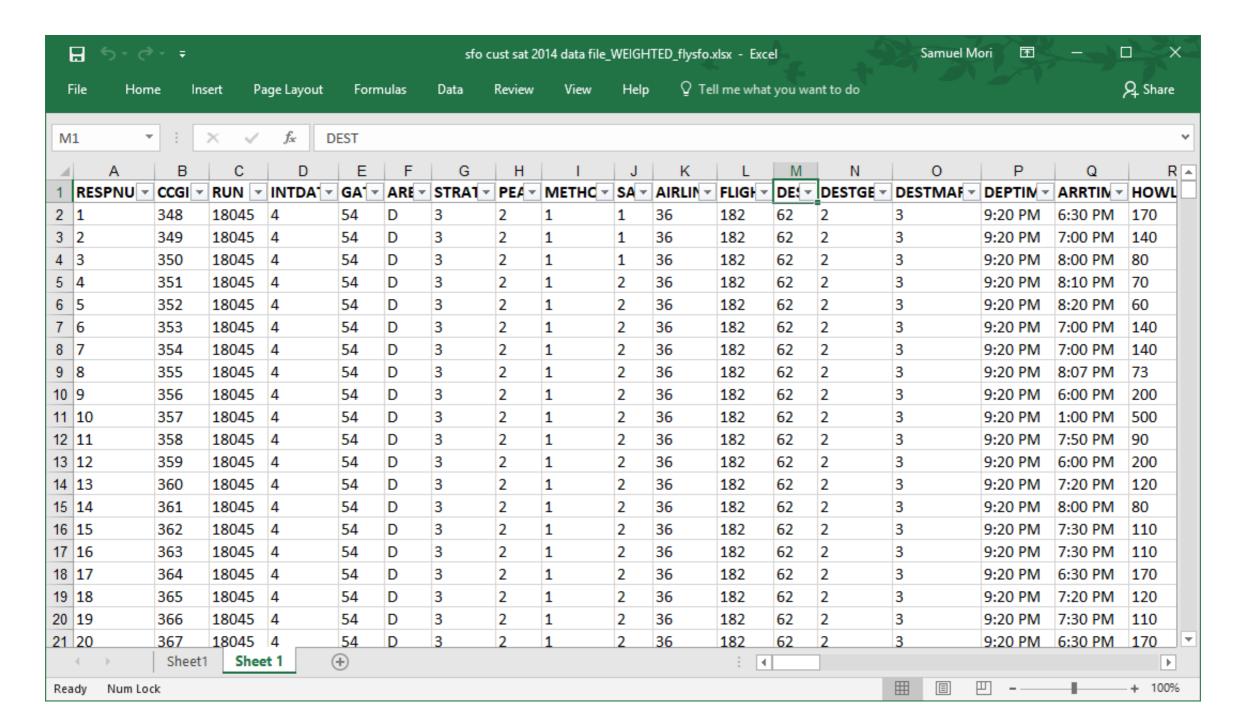
- Alguns dos formatos de dados mais comuns:
 - Comma Separated Value (CSV)
 - Excel (XLS)
 - JavaScript Object Notation (JSON)
 - eXtended Markup Language (XML)
 - Structured Query Language (SQL)
 - ...
- Fáceis de ler em Python!

- ficheiros de texto em que se usa vírgula para separar valores
- formato "standard" para comunicar dados tabulares

```
RESPNUM, CCGID, RUNID, INTDATE, AIRLINE, FLIGHT, DESTINATION, DESTGEO, DESTMARK, GATE, BAREA, STRATA, PEAK, DEPTIME, ARRTIME, HOWLONG, METHOD, Q2 PURP1, Q2 PURP1, Q3GETTO1, Q3GETTO2, Q3GETTO3, Q3PARK, Q4BAGS, Q4STORE, Q4FOOD, Q4WIFI, Q5TIMESFLOWN, Q5FI
RSTTIME, Q6LONGUSE, SAQ, Q7ART, Q7FOOD, Q7STORE, Q7SIGN, Q7WALKWAYS, Q7SCREENS, Q7INFODOWN, Q7INFOUP, Q7WIFI, Q7ROADS, Q7PARK, Q7AIRTRAIN, Q7LTPARKING, Q7CHTAL, Q8COM1, Q8COM2, Q8COM3, Q9BOARDING, Q9AIRTRAIN, Q9RENTAL, Q9FOOD, Q9RESTROOM, Q9ALL, Q9C
OMI, Q9COM2, Q9COM3, Q10SAFE, Q10COM1, Q10COM2, Q10COM3, Q11TSAPRE, Q12FRECHEKCRATE, Q12COM1, Q12COM2, Q13GETRATE, Q14FIND, Q14PASSTHRU, Q15PROBLEM, Q15COM1, Q15COM2, Q15COM3, Q16LIVE, HOME, Q17CITY, Q17STATE, Q17ZIP, Q17COUNTRY, Q13GETRATE, Q14FIND, Q14PASSTHRU, Q15PROBLEM, Q15COM1, Q15COM2, Q15COM3, Q16LIVE, HOME, Q17CITY, Q17STATE, Q17ZIP, Q17COUNTRY, Q13GETRATE, Q14FIND, Q14PASSTHRU, Q15PROBLEM, Q15COM1, Q15COM2, Q15COM3, Q16LIVE, HOME, Q17CITY, Q17STATE, Q17ZIP, Q17COUNTRY, Q13GETRATE, Q14FIND, Q14PASSTHRU, Q15PROBLEM, Q15COM1, Q15COM2, Q15COM3, Q16LIVE, HOME, Q17CITY, Q17STATE, Q17ZIP, Q17COUNTRY, Q13GETRATE, Q14FIND, Q14PASSTHRU, Q15PROBLEM, Q15COM1, Q15COM2, Q15COM3, Q16LIVE, HOME, Q17CITY, Q17STATE, Q17ZIP, Q17COUNTRY, Q13GETRATE, Q17COUNTRY, 
,Q19GENDER,Q20INCOME,Q21FLY,Q22SJC,Q22OAK,LANG,WEIGHT
3,460,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:00 AM,80,1,2,,,2,,,1,2,2,2,2,1,4,1,5,5,5,4,3,3,3,6,4,5,3,6,6,4,20,,,5,6,6,4,4,4,4,,,,5,8,,1,5,2,,,1,3,5,5,2,,,1,3,0AKLAND,CA,94619,US,6,0,1,2,2,2,1,0.720538
4,554,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:00 AM,120,1,3,,,7,,,,2,2,1,1,1,1,3,1,6,4,6,4,5,4,6,6,6,6,6,6,6,6,6,6,6,6,5,5,,,5,5,,,2,0,,,5,3,5,3,2,,,3,13,VANCOUVER,BC,,CANADA,6,2,4,1,2,2,1,0.669478
5,555,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:15 AM,105,1,2,,,6,,,1,2,2,2,1,1,2,2,4,4,4,5,5,5,5,6,3,6,6,6,6,6,6,5,5,999,,,5,6,6,5,5,5,,,,5,36,,,2,0,,,6,5,5,5,2,,,,3,13,,AB,,CANADA,2,1,3,2,2,2,1,0.669478
6,461,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,7:00 AM,20,1,3,,,2,,,2,2,1,2,1,2,1,6,6,6,5,5,5,3,3,4,4,6,6,6,6,4,,,,5,6,6,6,6,5,,,,5,21,,,2,0,,,7,4,5,5,2,,,,3,11,CHAMPAIGN,IL,61801,US,2,2,1,1,2,2,1,0.720538
10,558,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,5:45 AM,135,1,1,,,4,,,,2,2,1,5,1,2,2,3,3,3,4,3,4,4,4,4,4,35,311,,5,4,4,4,4,4,4,5,,1,1,2,F0STER CITY,CA,94404,US,4,1,4,3,1,2,1,0.669478
11,462,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,5:50 AM,90,1,1,,7,,,,2,2,1,2,3,1,4,2,3,2,3,3,3,2,6,6,6,3,4,6,4,6,2,103,202,,3,6,6,3,3,3,,,,4,,,,2,0,,,,9,5,3,3,2,,,,1,7,SANTA ROSA,CA,95404,US,5,0,0,2,2,2,1,0.720538
12,559,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,5:40 AM,140,1,3,,,1,,,1,2,2,1,1,3,1,4,2,5,4,4,5,3,4,6,4,4,4,,,,5,5,5,5,5,5,5,5,5,5,5,1,1,4,2,1,1,4,1,1,2,1,0.669478
14,560,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,N,N,1,0,,,7,,,,1,2,1,1,1,2,1,2,0,2,2,2,3,4,4,4,4,3,3,3,3,3,705,,,5,4,4,4,4,4,,,,5,,,,2,0,,,5,3,2,0,1,9,,,3,13,WHISTLER,BC,,CANADA,2,2,1,2,2,2,1,0.669478
16,561,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:00 AM,120,1,3,,,6,,,,1,1,2,2,1,2,0,2,0,0,0,0,5,4,0,0,0,4,0,0,4,,,,5,0,0,4,4,4,,,,5,,,2,0,,,1,5,5,5,2,,,,3,13,0TTAWA,ON,,CANADA,6,2,0,2,2,2,1,0.669478
17,562,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:00 AM,120,1,0,,,6,,,,1,2,2,2,1,2,1,2,3,2,2,3,5,4,2,2,5,5,6,6,6,6,4,,,,5,6,6,6,5,5,,,,4,9,,,1,5,,,5,5,4,5,2,,,,3,13,TORONTO,ON,,CANADA,3,1,3,2,2,2,1,0.669478
18,465,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:00 AM,80,1,1,,,9,,,1,2,1,2,3,1,2,2,4,4,3,3,3,3,5,3,6,4,6,3,4,,,,4,3,2,4,4,4,,,,1,3,,,5,5,4,4,2,,,,3,11,CHICAGO,IL,60642,US,3,1,3,2,1,0,1,0.720538
23,467,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:00 AM,80,1,3,,1,,,4,1,2,1,2,3,1,4,2,4,4,4,4,3,4,6,6,6,4,4,4,6,6,4,,,,4,3,6,4,3,3,,,,4,,,1,5,,,,2,4,5,5,2,,,,1,5,ALAMO,CA,94507,US,6,2,4,2,2,1,1,0.720538
24,566,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,5:00 AM,180,1,2,,,2,,,1,1,4,2,5,0,0,0,0,5,5,0,5,0,0,0,0,5,5,,,3,,,2,0,,,1,5,3,0,1,21,,,1,3,SAN LEANDRO,CA,,US,7,2,0,2,2,1,1,0.669478
26,567,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,5:40 AM,140,1,2,,,2,,,1,1,0,3,2,1,3,2,4,2,6,4,4,4,6,6,6,4,6,6,6,3,,,,5,6,6,2,3,3,,,,3,9,,,2,0,,,6,5,5,4,1,1,,,1,2,REDWOOD CITY,CA,94061,US,3,1,1,3,2,3,2,0.669478
27,568,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:00 AM,120,1,2,3,,2,,,1,2,2,1,2,1,2,1,2,1,3,4,4,4,6,6,4,4,,,5,6,6,6,4,4,,,,3,5,,,2,0,,,1,5,2,5,2,,,,1,3,0AKLAND,CA,94621,US,2,2,1,2,1,0,1,0.669478
30,469,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,5:30 AM,110,1,1,,1,,3,2,1,1,1,4,1,2,2,4,5,4,4,4,5,5,5,5,3,4,5,5,6,4,.,,5,5,6,5,4,5,.,,4,.,2,0,.,7,3,4,4,2,.,,1,4,SAN JOSE,CA,95112,US,5,1,3,2,1,2,1,0.720538
33,470,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:05 AM,75,1,3,,,2,,,,2,1,2,1,3,1,3,2,4,6,6,4,4,6,6,4,8,,,4,4,6,6,4,4,,,,5,5,,,1,5,1,,,7,5,4,5,2,,,,2,10,SANTA CRUZ,CA,95062,US,7,2,1,2,1,2,1,2,1,0.720538
34,573,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:00 AM,120,1,2,,,2,,,1,0,1,1,2,1,2,1,2,4,5,4,5,5,5,4,4,4,4,4,4,208,506,,0,0,0,3,0,0,,,2,41,,,2,0,,,0,5,0,2,,,,3,13,EDMONTON,AB,,CANADA,4,1,2,1,0,0,1,0.669478
35,574,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:30 AM,90,1,2,3,,2,,,,1,1,1,1,3,1,3,2,5,3,3,4,5,5,2,2,5,5,3,5,5,5,4,102,,5,3,3,3,2,3,,,3,98,,,1,3,7,,1,3,4,2,2,,,3,13,VANCOUVER,BC,,CANADA,3,1,2,1,2,1,1,0.669478
36,471,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:15 AM,65,1,2,3,,2,,,,2,2,1,2,1,2,5,4,4,4,4,4,4,4,4,4,6,5,6,6,6,5,5,5,,,,5,,,,1,5,1,,,3,3,5,5,2,,,,3,11,KINGSTON,IL,60145,US,6,2,2,2,2,2,1,0.720538
37,575,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:50 AM,70,1,6,,,2,,,2,2,1,2,4,1,4,2,3,2,2,4,4,4,4,6,3,3,3,4,6,4,,,,5,6,6,4,4,4,,,,5,6,6,4,4,4,,,,7,3,4,4,2,,,7,1,4,5UNNYVALE,CA,94087,US,4,1,3,2,1,2,1,0.669478
38,576,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,6:45 AM,75,1,3,,,6,,,2,1,2,0,3,1,1,2,3,3,3,4,4,6,6,1,6,6,2,6,6,3,507,,,5,5,6,5,4,4,,,,5,5,,,2,0,,,1,3,3,5,1,9,,,3,13,VANCOUVER,BC,,CANADA,3,2,2,3,0,1,1,0.669478
40,577,15083,12,38,1777,99,4,2,1,A,1,3,8:00 AM,4:08 AM,232,1,0,,,2,,,1,1,1,1,3,2,6,3,2,3,4,5,2,2,2,5,5,6,6,6,3,704,701,705,5,6,6,4,5,5,,,,5,5,,,2,0,,,1,5,5,5,2,,,3,13,TORONTO,ON,,CANADA,4,2,0,3,1,2,1,0.669478
41,473,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,5:45 AM,95,1,6,,,2,,,,2,2,1,1,2,1,4,2,3,4,4,4,3,6,6,4,5,5,6,6,4,4,,,,4,6,3,4,4,4,,,,1,5,2,,,5,4,5,5,2,,,3,12,,NY,13601,US,4,1,3,2,2,2,1,0.720538
42,474,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:30 AM,50,1,2,,,2,,,2,2,1,3,1,3,2,5,5,5,5,5,5,5,5,5,5,5,5,6,6,6,5,5,,,,5,,,2,0,,,1,4,5,0,2,,,1,90,,CA,,US,5,2,4,2,2,1,1,0.720538
44,475,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:00 AM,80,1,1,,,4,,,,2,1,1,2,1,1,2,2,3,4,5,5,5,5,6,6,6,6,6,6,6,6,4,,,,4,6,6,5,5,5,,,,5,,,,2,0,,,5,5,5,4,2,,,,3,12,ROCHESTER,NY,14586,US,4,1,1,2,2,2,1,0.720538
45,514,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,8:30 AM,143,1,4,,,2,,,,1,1,1,1,4,1,4,1,4,4,4,3,4,4,3,3,5,4,3,3,3,4,113,,,4,3,3,4,4,4,,,,2,41,,2,0,,,17,2,4,1,2,,,2,10,SANTA CRUZ,CA,95060,US,3,1,4,1,1,1,1,0.720538
47,515,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,9:00 AM,113,1,3,,,6,,,,2,1,1,3,2,6,5,5,5,5,5,6,6,5,5,6,6,6,5,5,5,5,,,,5,,,,2,0,,,1,5,5,5,2,,,,1,3,0AKLAND,CA,91786,US,3,2,1,2,1,0,1,0.720538
48,477,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:30 AM,50,1,1,,1,,0,2,0,0,0,3,1,4,2,4,5,4,4,4,4,4,4,3,3,3,2,6,4,,,,5,4,6,3,3,4,,,,4,,,,0,0,,,,3,4,5,5,2,,,,1,6,KENTFIELD,CA,94904,US,4,2,4,2,2,1,1,0.720538
49,516,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,9:30 AM,83,1,1,,,5,,,,2,2,2,2,4,1,4,2,5,4,4,5,5,4,6,6,4,5,3,4,6,6,5,205,,,5,5,5,5,5,5,5,5,5,1,5,1,1,1,2,5,5,2,,,1,3,ALAMEDA,CA,94502,US,5,2,4,1,2,1,1,0.720538
50,517,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,9:15 AM,98,1,1,,12,,,,2,1,4,2,3,3,3,3,4,6,5,6,4,4,6,6,6,4,.,,4,6,6,4,4,4,.,,5,6,27,,2,0,.,5,3,4,5,2,.,,3,10,RESEDA,CA,91335,US,4,2,4,2,2,2,1,0.720538
51,518,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,9:10 AM,103,1,4,,,13,,,,2,1,2,2,3,1,3,2,5,4,4,5,5,5,6,6,5,5,5,6,6,5,5,5,5,,,,5,,,,2,0,,3,5,5,5,2,,,,3,10,LOS ANGELES,CA,91307,US,2,2,2,2,2,2,1,0.720538
55,479,15076,13,8,242,18,3,4,58,D,1,2,7:20 AM,6:00 AM,80,1,2,,,6,,,0,0,1,0,1,1,4,2,3,4,3,4,4,4,6,6,6,6,6,6,4,,,,4,4,4,,,,5,,,,1,5,,,5,5,5,5,2,,,,3,12,BOSTON,MA,1945,US,7,2,2,2,2,2,1,0.720538
56,521,15079,12,8,5988,50,1,4,58,D,1,1,10:53 AM,9:30 AM,83,1,3,,,2,,,2,2,2,3,1,2,2,4,4,4,4,5,4,4,5,5,6,5,6,4,5,,,4,5,1,3,,,1,5,,,5,4,4,5,1,3,,,1,1,5AN FRANCISC,CA,,US,7,2,0,2,2,2,1,0.720538
```

Excel

O mesmo exemplo em folhas de cálculo



- Ler um ficheiro CSV em Python
- E.g., índice de secura mensal para o Porto publicado pelo IPMA aqui

```
import csv
with open('mpdsi-1312-porto.csv','r') as f:
    table = csv.reader(f)
    data = [ row for row in table ]
print(data)
```

- Primeira linha é o cabeçalho que define o tipo de cada coluna
- Cada linha é uma lista de comprimento igual

```
['date', 'minimum', 'maximum', 'range', 'mean', 'std']
['2020-03-01', '1.17446672916', '1.3303924799',
'0.155925750732', '1.24881755738', '0.0448042640559']
```

Converter matriz num dicionário de classificações

```
cabecalho=data[0]
meses = data[1:]
def classifica(n):
    if n>=4: return 'chuva extrema'
    elif n>=3: return 'chuva severa'
    elif n>=2: return 'chuva moderada'
    elif n>=1: return 'chuva fraca'
    elif n>-1: return 'normal'
    elif n>-2: return 'seca fraca'
    elif n>-3: return 'seca moderada'
    elif n>-4: return 'seca severa'
    else : return 'seca extrema'
meses_cs = { mes[0] : [classifica(float(n)) for n in mes[1:]]\
          for mes in meses }
```

Selecionar a previsão mais recorrente por mês

Retornar um par com o mês mais seco de 2020 e a sua previsão

```
import dateutil.parser as date
meses date = { date.parse(mes) : c for mes,c in meses c.items() }
meses 2020 = { mes.month : meses_date[mes] for mes in meses_date\
               if mes.year == 2020 }
def desclassifica(s):
    if s=='chuva extrema' : return 4
    elif s=='chuva severa' : return 3
    elif s=='chuva moderada' : return 2
    elif s=='chuva fraca' : return 1
   elif s=='normal' : return 0
elif s=='seca fraca' : return -1
    elif s=='seca moderada' : return -2
    elif s=='seca severa' : return -3
    elif s=='seca extrema' : return -4
    else
                             : return None
def chave(mes): return desclassifica(meses 2020[mes])
mes = min (meses 2020, key=chave)
(mes, meses 2020[mes])
```

- Escrever num ficheiro CSV em Python
- E.g., guardar os índices de secura para 2020

- ficheiros de texto key-value hierárquicos
- formato "standard" para troca de dados semi-estruturados entre aplicações

JSON XML

```
<?xml version="1.0" encoding="UTF-8" ?>
1 - {
        "sessionStart": "16-03-18-12-33-09",
                                                          2 - (root)
                                                                  <sessionStart>16-03-18-12-33-09</sessionStart>
        "sessionEnd": "16-03-18-12-33-12",
                                                                  <sessionEnd>16-03-18-12-33-12</sessionEnd>
        "mapName": "TestMap",
        "logSections": [{
                                                                  <mapName>TestMap</mapName>
6 .
            "sector": {
                                                                  <logSections>
                 "x": 2.0,
                                                                      <sector>
8
                 "y": -1.0,
                                                                          <x>2</x>
9
                                                                          <y>-1</y>
10
                                                                          <z>0</z>
1 -
             "logLines": [{
                                                         11
                                                                      </sector>
12
                 "time": 37.84491729736328.
                                                         12 -
                                                                      <logLines>
13
                "state": 0,
                                                                          <time>37.84491729736328</time>
                                                         13
.4
                "action": 1,
                                                         14
                                                                          <state>0</state>
15
                "playerPosition": {
                                                                          <action>1</action>
                     "x": 24.560218811035158,
16
                                                         16 -
                                                                          <playerPosition>
                     "y": -8.940696716308594e-8,
17
                                                         17
                                                                              <x>24.560218811035156</x>
18
                     "z": 3.3498525619506838
                                                         18
                                                                               <y>-8.940696716308594e-8
19
                                                         19
                                                                              <z>3.3498525619506836</z>
10
                 "cameraRotation": {
                                                         20
                                                                          </playerPosition>
21
                     "x": 0.24549755454063416.
                                                         21 -
                                                                          <cameraRotation>
                     "y": 0.017123013734817506,
22
                                                         22
                                                                              <x>0.24549755454063416</x>
23
                     "z": 0.031348951160907748,
                                                         23
                                                                              <y>0.017123013734817505</y>
24
                     "w": -0.9687389135360718
                                                         24
                                                                              <z>0.031348951160907745</z>
25
                                                         25
                                                                               <w>-0.9687389135360718</w>
26
                                                                          </cameraRotation>
                                                         26
```

- Ler um ficheiro JSON em Python
- E.g., previsão metereológica de 5 dias para o Porto publicada pelo IPMA aqui
- JSON ≃ estruturas de dados Python

```
"owner": "IPMA",
with open('1131200.json','r') as f:
    dict = json.load(f)
print(dict)

"owner": "IPMA",
"country": "PT",
"data": [{
        "precipitaProb": "100.0",
        "tMin": "7.5",
        "tMax": "14.5",
        ...}]
...
```

- JSON = dicionários e listas aninhados uns nos outros, cujas folhas são strings, números ou booleanos
 - hierárquico: estrutura aninhada
 - <u>semi-estruturado</u>: dicionários podem ter qualquer chave/ valor; strings podem representar números, datas, etc

```
json ::= dict
dict ::= { string : value, ...}
value ::= dict | sequence | basic
sequence ::= [ value, ... ]
basic ::= string | number | true | false | null
```

• Obter previsão para o dia mais próximo de hoje

- Converter código de previsão numa descrição textual
- Descrições fornecidas pelo IPMA <u>aqui</u>

- Escrever num ficheiro JSON em Python
- Permite guardar grande parte dos objetos Python em ficheiro (serialização/deserialização)
- E.g., um dicionário dia : previsão textual, com uma formatação especial do dia

```
import calendar
def day_month(d):
    return str(d.day)+' '+calendar.month_abbr[d.month]
weather_dif = { day_month(d) : classe[w] for d,w in
weather.items() }
print(weather_dif)

with open("test.json","w") as f:
    json.dump(weather_dif,f)
with open("test.json","r") as f:
    weather_dif2 = json.load(f)
print(weather_dif2 == weather_dif)
```