

Programação II

+

Estruturas de Dados para Bioinformática

Hugo Pacheco

DCC/FCUP
23/24

Compreensões

Listas por compreensão

- É muito comum construir uma lista partindo de uma outra:
 - selecionando elementos usando uma condição
 - aplicando uma transformação a cada elemento
- E.g., calcular quadrados de números de 1 a 9

```
# utilizando ciclo
sqrs = []
for x in range(1, 10):
    sqrs.append(x**2)
```

```
# utilizando ordem superior
sqrs = list(map(lambda x:x**2, range(1,10)))
```

```
# utilizando compreensão
sqrs = [x**2 for x in range(1,10)]
```

Listas por compreensão

- Outros exemplos

```
>>> [i**2 for i in [2, 3, 5, 7]]  
[4, 9, 25, 49]  
>>> [1+x/2 for x in [0, 1, 2]]  
[1.0, 1.5, 2.0]  
>>> [ord(c) for c in "ABCDEF"]  
[65, 66, 67, 68, 69, 70]  
>>> [len(s) for s in "As armas e os barões  
assinalados".split()]  

```

Listas por compreensão

- Outros exemplos com uma condição

```
#quadrados dos números divisores de 3 inferiores a 10
```

```
>>> [x**2 for x in range(10) if x%3==0]  
[0, 9, 36, 81]
```

```
# utilizando ordem superior
```

```
list(map(lambda x:x**2 \  
, filter(lambda x:x%3==0, range(10)) \  
))
```

```
>>> [(x,x) for x in range(1,7) if x+x not in {2, 12}]  
[(2, 2), (3, 3), (4, 4), (5, 5)]
```

Listas por compreensão

- Testar se um número é primo, i.e., a lista de todos os seus divisores próprios é vazia

```
def primo(n):  
    # lista dos divisores próprios  
    divs = [d for d in range(2, n) if n%d==0]  
    # não é primo se e só se a lista for vazia  
return len(divs)==0
```

Listas por compreensão

- Podemos utilizar compreensões aninhadas, e/ou utilizar várias listas
- E.g., para criar a matriz (lista de listas) da multiplicação

```
>>> [[(i,j) for j in range(1,6)] for i in range(1,6)]
[[[1, 1), (1, 2), (1, 3), (1, 4), (1, 5)
, ...
[(5, 1), (5, 2), (5, 3), (5, 4), (5, 5)]]
```

- E.g., para calcular o produto cartesiano de dois conjuntos (como listas) (ordem influencia resultado)

```
>>> [(i,j) for i in range(1,3) for j in "ABCDE"]
[(1, 'A'), (1, 'B'), ..., (2, 'D'), (2, 'E')]
>>> [(i,j) for j in "ABCDE" for i in range(1,3) ]
[(1, 'A'), (2, 'A'), ..., (1, 'E'), (2, 'E')]
```

Compreensão

- A sintaxe por compreensão funciona para qualquer coleção (listas, dicionários, sets, etc); inspirada na teoria de conjuntos

$$\{x^2 : x \in \{1, \dots, 9\}\}$$

$$\{x^2 : x \in \{1, \dots, 9\}, 2|x\}$$

- Utilizando sets em Python (ordem não interessa)

```
>>> {x**2 for x in range(1,10) }
{64, 1, 4, 36, 9, 16, 49, 81, 25}
>>> {x**2 for x in range(1,10) if x%2==0 }
{16, 64, 4, 36}
```

Compreensão

- No fundo só muda o delimitador
- Quando a ordem interessa ⇒ listas

```
>>> [x**2 for x in range(1, 6)]
[1, 4, 9, 16, 25]
>>> [(x, x**2) for x in range(1, 6)]
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

- Quando não há repetidos e a ordem não interessa ⇒ sets

```
>>> {x**2 for x in range(1, 6)}
{1, 4, 9, 16, 25}
>>> {(x, x**2) for x in range(1, 6)}
{(2, 4), (4, 16), (1, 1), (3, 9), (5, 25)}
```

- Quando as chaves são únicas e a ordem não interessa ⇒ dicionários

```
>>> {x : x**2 for x in range(1, 6)}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Compreensões

- Quando há repetidos, o tipo de coleção é mais pertinente

```
>>> [x*y for x in [1,2,3] for y in [1,2,3]]  
[1, 2, 3, 2, 4, 6, 3, 6, 9]
```

elimina repetidos

```
>>> {x*y for x in [1,2,3] for y in [1,2,3]}  
{1, 2, 3, 4, 6, 9}
```

elimina repetidos, não os conta

```
>>> {x*y : 1 for x in [1,2,3] for y in [1,2,3]}  
{1:1, 2:1, 3:1, 4:1, 6:1, 9:1}
```

contar repetidos

```
>>> d = {}  
>>> for xy in [x*y for x in [1,2,3] for y in [1,2,3]]:  
>>>     d[xy] = 1 + d.get(xy, 0)  
{1: 1, 2: 2, 3: 2, 4: 1, 6: 2, 9: 1}
```

- Conseguimos usar apenas comprehensões para o último exemplo?

Sets por compreensão

```
fruits = {'apple', 'mango', 'banana', 'cherry'}
```

#map em sets funcional

```
>>> set(map(len, fruits))  
{5, 6}
```

map em sets por compreensão

```
>>> {len(f) for f in fruits}  
{5, 6}
```

#filter em sets funcional

```
>>> set(filter(lambda f: 'a' in f, fruits))  
{'mango', 'apple', 'banana'}
```

#filter em sets por compreensão

```
>>> {f for f in fruits if 'a' in f}  
{'apple', 'banana', 'mango'}
```

set do alfabeto

```
{chr(i) for i in range(ord('a'), ord('z'))}
```

união de listas de sets

```
>>> l = [{ 'a', 'b', 'c' }, { 'c', 'd', 'e' }, { 'b', 'k' }]  
>>> {c for s in l for c in s}  
{'b', 'd', 'e', 'a', 'k', 'c'}
```

Dicionários por compreensão

```
>>> fruits = ['apple', 'mango', 'banana', 'cherry']

# valor = comprimento da palavra
>>> dict = {f:len(f) for f in fruits}
{'apple': 5, 'mango': 5, 'banana': 6, 'cherry': 6}

# valor = índice na lista
>>> f_dict = {f:i for i,f in enumerate(fruits)}
{'apple': 0, 'mango': 1, 'banana': 2, 'cherry': 3}

# inverter um dicionário, remove repetidos
>>> d = {v:k for k,v in dict.items()}
{5: 'mango', 6: 'cherry'}

# conversão de moeda
>>> price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}
>>> {item: value*0.76 for (item, value) in price.items()}
{'milk': 0.7752, 'coffee': 1.9, 'bread': 1.9}

# nome com 4 letras e idade abaixo de 40
>>> people = {'jack': 38, 'michael': 48, 'guido': 33, 'john': 57}
{ name:age for name,age in people.items() if len(name)==4 if age<40 }
{'jack': 38}
```

Exemplo 35 Sonnets

- Extrai dicionário de poemas da obra 35 Sonnets de Fernando Pessoa

```
import urllib.request
url = 'http://www.gutenberg.org/files/19978/19978-0.txt'
urllib.request.urlretrieve(url, '35sonnets.txt')
with open('35sonnets.txt', 'r') as f: ls = f.read().splitlines()

# testa se linha é numeração romana
def roman(s):
    s = s[:-1] #deixa cair último caracter
    bad = { c for c in s if not c.isupper() }
    return len(s) > 0 and len(bad) == 0

# extrai poema
def poem(lines):
    res = []
    for l in lines:
        if len(l) == 0: break
        else: res.append(l)
    return "\n".join(res)

poemas = { l[:-1] : poem(ls[i+3:]) for i,l in enumerate(ls) if roman(l) }
```

Exemplo 35 Sonnets

- Conta palavras por poema

```
# normaliza string
def trim(s):
    return "".join([c.lower() for c in s if c.isalpha()])

# conta palavras
def conta(s):
    palavras = [trim(p) for p in s.split()]
    count = {}
    for p in palavras:
        count[p] = 1 + count.get(p, 0)
    return count

palavras_por_poema = { i:conta(p) for i,p in poemas.items() }

print(palavras_por_poema)

print(palavras_por_poema)
{'I': {'whether': 1, 'we': 9, ...}, 'II': {...}}
```

Exemplo 35 Sonnets

- Conta palavras em toda a obra

```
# junta dicionários para contar palavras em toda a obra
palavras_total = {}
for ps in palavras_por_poema.values():
    for p,n in ps.items():
        palavras_total[p] = n + palavras_total.get(p, 0)

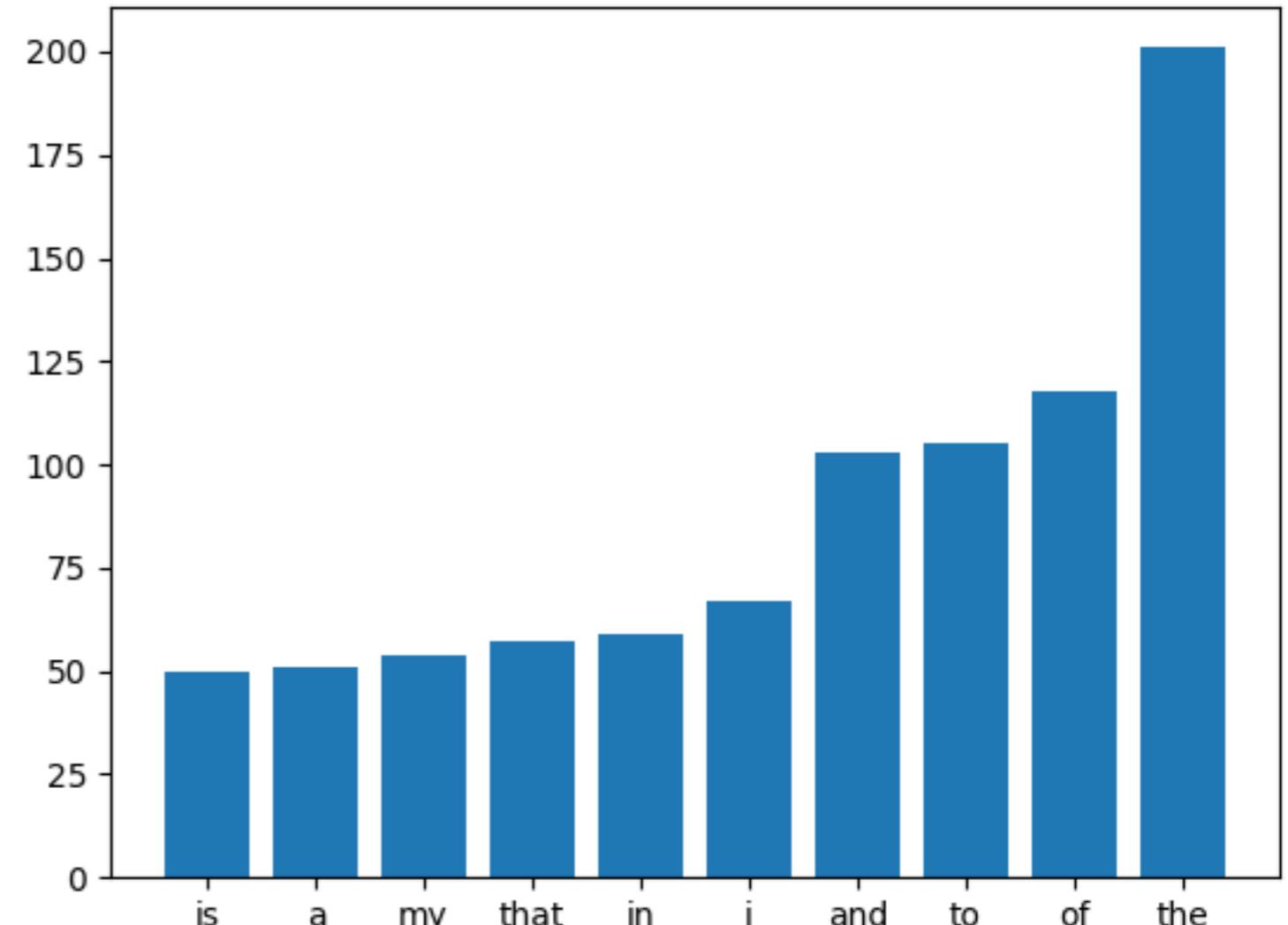
print(palavras_total)
{'whether': 1, 'we': 42, 'write': 1, 'or': 18, ...}
```

Exemplo 35 Sonnets

```
palavras_total = sorted(palavras_total.items(),key=lambda kv: kv[1])
mais_frequentes = dict(palavras_total[-10:])

import matplotlib.pyplot as plt
plt.bar(mais_frequentes.keys(), mais_frequentes.values())
plt.show()
```

- Desenhar um histograma com as 10 palavras mais frequentes (vamos estudar gráficos mais tarde)



Exemplo (Gene Ontology)

- Encontrar genes com identificadores e nome únicos

```
with open('.../.../.../dados/PZ.annot.txt', 'r') as f:  
    linhas = f.read().splitlines()  
  
pzs = {} ; gos = {} ; names = {}  
for linha in linhas:  
    pz,go,name = linha.split('\t')  
    pzs[pz] = pzs.get(pz, []) + [(go, name)]  
    gos[go] = gos.get(go, []) + [(pz, name)]  
    names[name] = names.get(name, []) + [(pz, go)]  
  
# extremamente lento  
res = { (pz,go,name) for pz,xs in pzs.items() for go,ys in  
gos.items() for name,zs in names.items() \  
       if xs==[(go, name)] and ys==[(pz, name)] and zs==[(pz, go)] }  
  
# muito mais rápido  
res = { (pz,go,name) for pz,xs in pzs.items() if len(xs)==1 for  
go,name in xs \  
       if gos[go]==[(pz, name)] and names[name]==[(pz, go)] }
```

Formatos de dados

Dados

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

“Canto Primeiro

1

*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

- Nos ficheiros que temos visto até agora, os dados estão em texto livre
- Mas têm sempre alguma estrutura, por exemplo, informação de enzimas por linha, separada por / e ‘

“REBASE version 302

staden.302

=====

*REBASE, The Restriction Enzyme Database <http://rebase.neb.com>
Copyright (c) Dr. Richard J. Roberts, 2023. All rights reserved.*

=====

Rich Roberts

Jan 27 2023

AanI/TTA'TAA//
AarI/CACCTGCNNNN'NNNN/'NNNNNNNNNGCAGGTG//
AasI/GACNNNN'NNGTC//
AatII/GACGT'C//
[...]"

Dados

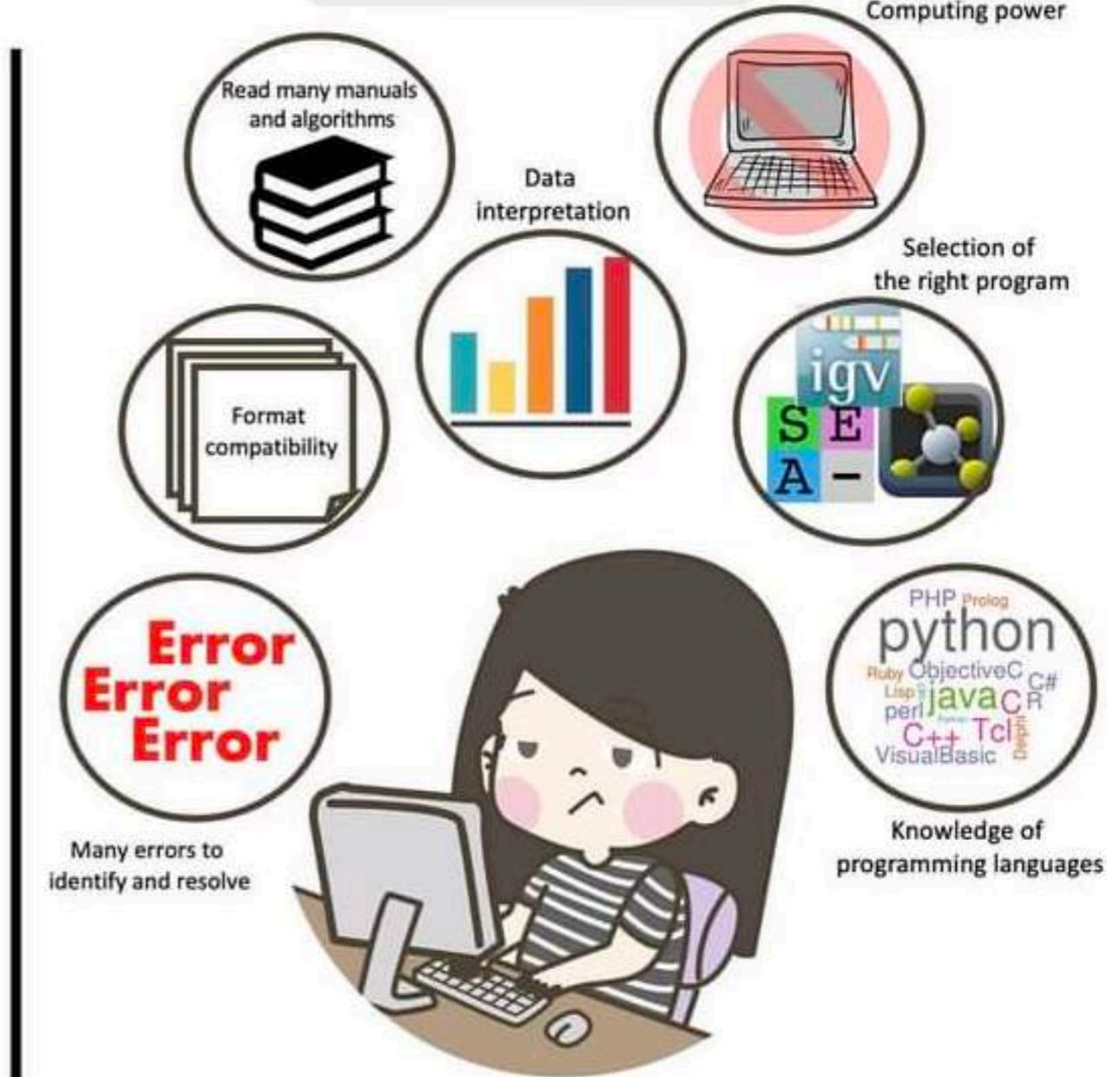
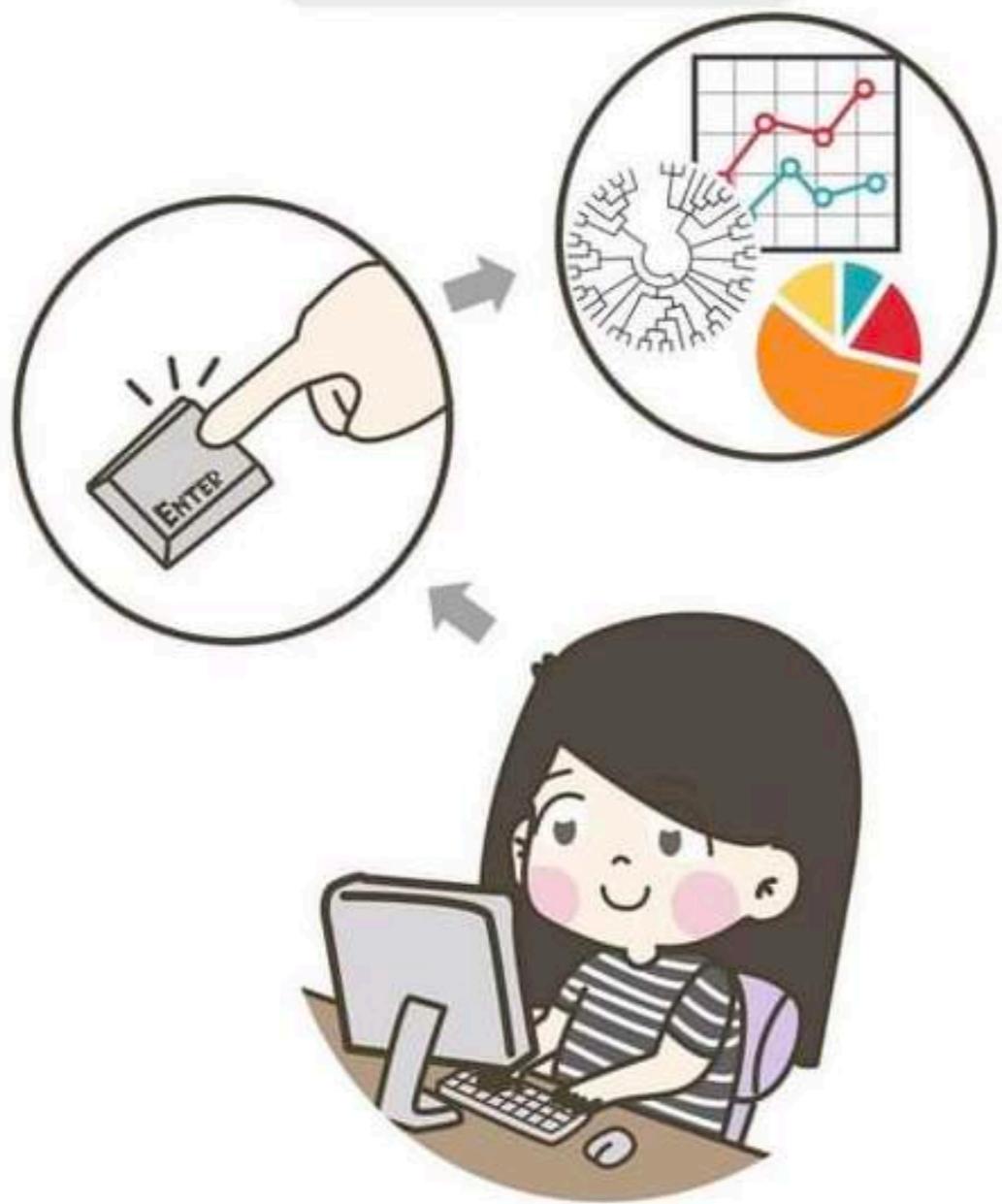
- Para ler os dados temos que explorar a sua estrutura.
- Mas se estão em texto livre...
 - pode ser difícil encontrar padrões
 - cada caso é um caso
- Idealmente, diferentes conjuntos de dados devem seguir uma estrutura comum:
 - facilitar a partilha
 - garantir preservação e suporte de longo prazo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

Dados?

Expectation

Vs

Reality



“The Web is Agreement”



[source <https://www.theweabisagreement.com/>]

Formatos de dados

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

Markdown

- ficheiros de texto em que se usa caracteres especiais para formatação
- formato “standard” para notas, blogs ou gerar HTML

```
# Projeto 1 - Análise de texto

Neste primeiro projeto vamos relembrar alguns conceitos base da programação em Python e aplicá-los no processamento de ficheiros de texto.

Aceda ao repositório [replit](https://replit.com/@up652136/Prog2-Proj1) do Projeto 1.

- Criando uma conta no [replit](https://replit.com/) e fazendo `Fork` do projeto.
- Pode consultar os ficheiros individuais na pasta [projeto1](../scripts/projeto1).
- Pode fazer download de todo o projeto como um arquivo zip [aqui](../scripts/projeto1.zip).

Neste projeto vamos processar o texto completo do *Sermão de Santo António aos Peixes do Padre António Vieira*, extrair métricas simples e reformatar o texto.

## Tarefa 1

Faça download para uma pasta local do ficheiro [sermao.txt](../scripts/projeto1/sermao.txt).

Complete a função `leTexto` que lê o conteúdo de um ficheiro de texto para uma lista de linhas.

## Tarefa 2

Analise a definição da função ``organizaSermao`` que organiza uma lista de linhas.

Por exemplo, para um sermão dado como a lista de linhas:

```python
['SERMÃO DE EXEMPLO'

```

## Projeto 1 - Análise de texto

Neste primeiro projeto vamos relembrar alguns conceitos base da programação em Python e aplicá-los no processamento de ficheiros de texto.

Aceda ao repositório [replit](#) do Projeto 1, onde pode encontrar um ficheiro `projeto1.py`:

- Criando uma conta no [replit](#) e fazendo `Fork` do projeto, pode resolver o projeto online utilizando o IDE web.
- Pode consultar os ficheiros individuais na pasta `projeto1` e fazer download dos mesmos para desenvolver o projeto no seu computador e utilizando um IDE à sua escolha.
- Pode fazer download de todo o projeto como um arquivo zip [aqui](#).

Neste projeto vamos processar o texto completo do *Sermão de Santo António aos Peixes do Padre António Vieira*, extrair métricas simples e reformatar o texto.

### Tarefa 1

Faça download para uma pasta local do ficheiro `sermao.txt`, que contém o texto integral do *Sermão de Santo António aos Peixes do Padre António Vieira*.

Complete a função `leTexto` que lê o conteúdo de um ficheiro de texto para uma lista de linhas de texto, em que cada linha de texto é uma string sem caracteres newline.

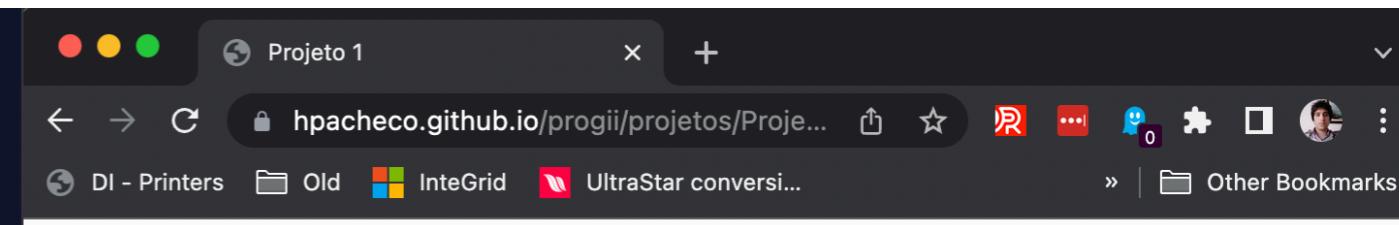
# HTML

- ficheiros de texto em que se usa a notação `<campo>texto</campo>` para definir uma estrutura hierárquica
- formato “standard” para representar páginas web

```
<body>
<header id="title-block-header">
<h1 class="title">Projeto 1</h1>
</header>
<h1 id="projeto-1---análise-de-texto">Projeto 1 - Análise de
texto</h1>
<p>Neste primeiro projeto vamos relembrar alguns conceitos base da
programação em Python e aplicá-los no processamento de ficheiros de
texto.</p>
<p>Aceda ao repositório replit do
Projeto 1, onde pode encontrar um ficheiro
<code>projeto1.py</code>:</p>

Criando uma conta no replit e
fazendo <code>Fork</code> do projeto, pode resolver o projeto online
utilizando o IDE web.
Pode consultar os ficheiros individuais na pasta projeto1 e fazer download dos mesmos
para desenvolver o projeto no seu computador e utilizando um IDE à
sua escolha.
Pode fazer download de todo o projeto como um arquivo zip aqui.

<p>Neste projeto vamos processar o texto completo do Sermão de
Santo António aos Peixes do Padre António Vieira,
extraír métricas simples e reformatar o texto.</p>
<h2 id="tarefa-1">Tarefa 1</h2>
<p>Faça download para uma pasta local do ficheiro ../scripts/projeto1/dados/sermão.txt que
```



## Projeto 1 - Análise de texto

Neste primeiro projeto vamos relembrar alguns conceitos base da programação em Python e aplicá-los no processamento de ficheiros de texto.

Aceda ao repositório replit do Projeto 1, onde pode encontrar um ficheiro projeto1.py:

- Criando uma conta no replit e fazendo Fork do projeto, pode resolver o projeto online utilizando o IDE web.
- Pode consultar os ficheiros individuais na pasta projeto1 e fazer download dos mesmos para desenvolver o projeto no seu computador e utilizando um IDE à sua escolha.
- Pode fazer download de todo o projeto como um arquivo zip aqui.

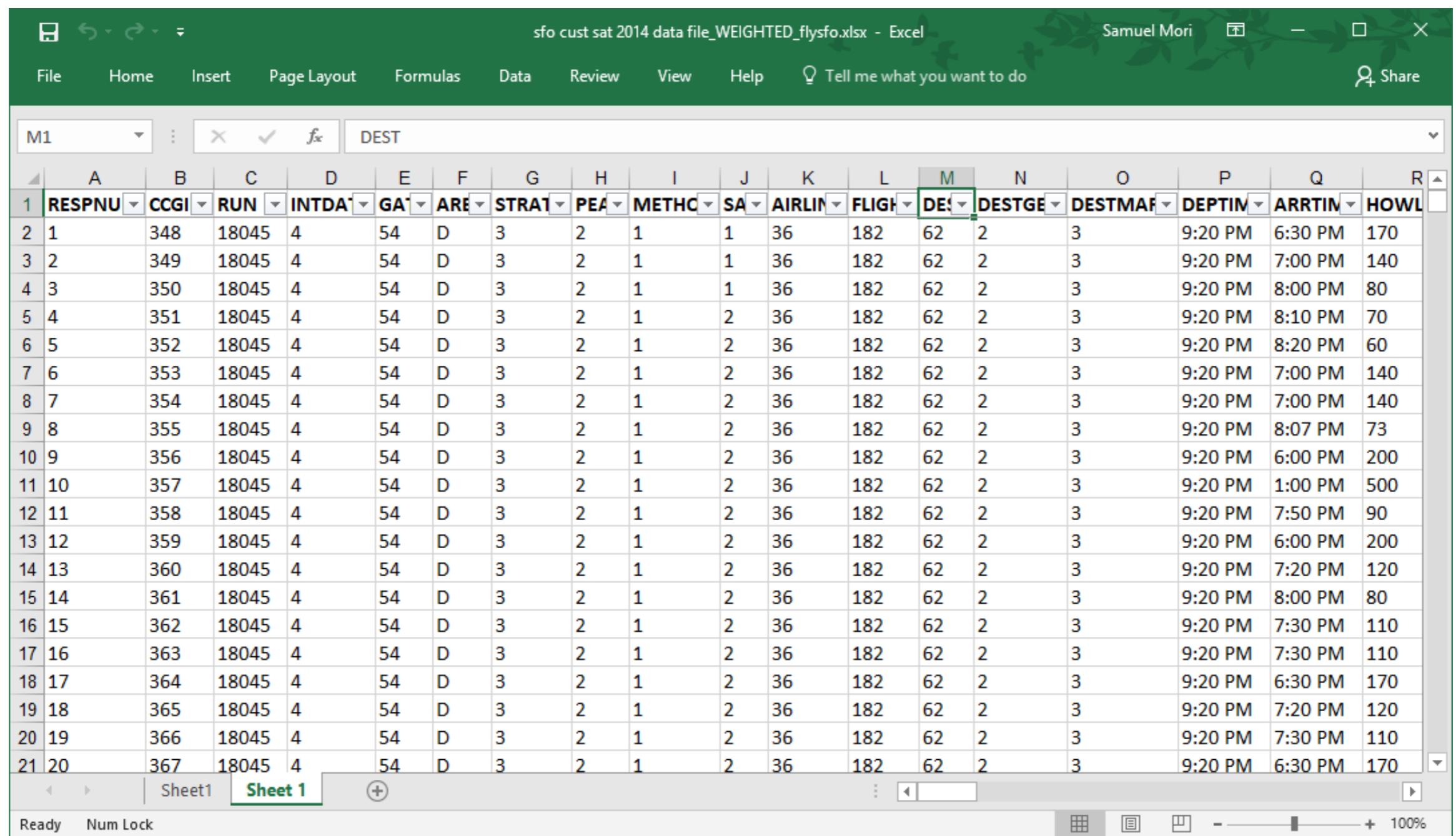
Neste projeto vamos processar o texto completo do *Sermão de Santo*

# CSV

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

# Excel

- O mesmo exemplo em folhas de cálculo



The screenshot shows a Microsoft Excel spreadsheet titled "sfo cust sat 2014 data file\_WEIGHTED\_flysfo.xlsx". The ribbon menu includes File, Home, Insert, Page Layout, Formulas, Data, Review, View, Help, and a search bar. The main area displays a data table with 21 rows and 18 columns. The columns are labeled: RESPNU, CCGI, RUN, INTDAT, GA, ARE, STRAT, PEA, METHC, SA, AIRLIN, FLIGH, DE, DESTGE, DESTMAF, DEPTIM, ARRTIN, and HOWL. Row 1 contains the column headers. The data starts from row 2, with various values such as 348, 18045, 4, 54, D, 3, etc., across the columns. The cell at row 2, column M (M2) is highlighted with a green border, indicating it is selected.

RESPNU	CCGI	RUN	INTDAT	GA	ARE	STRAT	PEA	METHC	SA	AIRLIN	FLIGH	DE	DESTGE	DESTMAF	DEPTIM	ARRTIN	HOWL
1	348	18045	4	54	D	3	2	1	1	36	182	62	2	3	9:20 PM	6:30 PM	170
2	349	18045	4	54	D	3	2	1	1	36	182	62	2	3	9:20 PM	7:00 PM	140
3	350	18045	4	54	D	3	2	1	1	36	182	62	2	3	9:20 PM	8:00 PM	80
4	351	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	8:10 PM	70
5	352	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	8:20 PM	60
6	353	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:00 PM	140
7	354	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:00 PM	140
8	355	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	8:07 PM	73
9	356	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	6:00 PM	200
10	357	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	1:00 PM	500
11	358	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:50 PM	90
12	359	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	6:00 PM	200
13	360	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:20 PM	120
14	361	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	8:00 PM	80
15	362	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:30 PM	110
16	363	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:30 PM	110
17	364	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	6:30 PM	170
18	365	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:20 PM	120
19	366	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	7:30 PM	110
20	367	18045	4	54	D	3	2	1	2	36	182	62	2	3	9:20 PM	6:30 PM	170

# CSV

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

```
import csv
with open('.../.../dados/mpdsi-1312-porto.csv', 'r') as f:
 table = csv.reader(f)
 data = list(table)
print(data)
```

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

```
['date', 'minimum', 'maximum', 'range', 'mean', 'std']
['2022-04-01', '-2.89150762558', '-2.75911641121', '0.132391214371', '-2.83662211895', '0.0348872640209']
['2022-05-01', '-3.8392894268', '-3.70847082138', '0.130818605423', '-3.7754253887', '0.0342999110149']
```

# CSV

- Criar 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 }

```

Índice PDSI (Palmer Drought Severity Index) mensal por concelho (formato CSV)

#### Invocação:

<https://api.ipma.pt/open-data/observation/climate/mpdsi/{distrito}/mpdsi-{DICO}-{concelho}.csv>

Notas: Taxa de atualização mensal. DICO: Identificador único de concelho (de acordo com a CAOP - DGT).

- *maior ou igual a 4,0* - Chuva extrema
- *3,00 a 4,0* - Chuva severa
- *2,00 a 3,99* - Chuva moderada
- *1,00 a 1,99* - Chuva fraca
- *-0,99 a 0,99* - Normal
- *-1,99 a -1,0* - Seca fraca
- *-2,99 a -2,0* - Seca moderada
- *-3,99 a -3,0* - Seca severa
- *menor ou igual a -4,00* - Seca extrema

# CSV

- Selecionar a previsão mais recorrente por mês

```
def count(xs):
 c = {}
 for x in xs: c[x] = 1 + c.get(x, 0)
 return c

def max_count(xs):
 c = count(xs)
 return max(c, key=lambda k : c[k])

meses_c = { mes : max_count(cs) \
 for mes,cs in meses_cs.items() }
```

# CSV

- Retornar um par com o mês mais seco de 2023 e a sua previsão

```

import dateutil.parser as date

meses_date = { date.parse(mes) : c for mes,c in meses_c.items() }
meses_2023 = { mes.month : meses_date[mes] for mes in meses_date \
 if mes.year == 2023 }

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

mais_seco_2023 = min(meses_2023.values(),key=desclassifica)
meses_mais_secos_2023 = { k for k,v in meses_2023.items() \
 if v==mais_seco_2023 }

```

# CSV

- Escrever num ficheiro CSV em Python
- E.g., guardar os índices de secura para 2023

```
meses_2023_tbl = [['mes', 'secura']] \n + [[m, s] for m, s in meses_2023.items()]
```

```
with open('test.csv', 'w') as f:\n writer = csv.writer(f, delimiter=',')\n writer.writerows(meses_2023_tbl)
```

# JSON

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

**JSON**

```
1 {
2 "sessionStart": "16-03-18-12-33-09",
3 "sessionEnd": "16-03-18-12-33-12",
4 "mapName": "TestMap",
5 "logSections": [
6 {"sector": {
7 "x": 2.0,
8 "y": -1.0,
9 "z": 0.0
10 }
11 },
12 {"logLines": [{
13 "time": 37.84491729736328,
14 "state": 0,
15 "action": 1,
16 "playerPosition": {
17 "x": 24.560218811035158,
18 "y": -8.940696716308594e-8,
19 "z": 3.3498525619506838
20 },
21 "cameraRotation": {
22 "x": 0.24549755454063416,
23 "y": 0.017123013734817506,
24 "z": 0.031348951160907748,
25 "w": -0.9687389135360718
26 },
27 ...
28 }
```

**XML**

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <root>
3 <sessionStart>16-03-18-12-33-09</sessionStart>
4 <sessionEnd>16-03-18-12-33-12</sessionEnd>
5 <mapName>TestMap</mapName>
6 <logSections>
7 <sector>
8 <x>2</x>
9 <y>-1</y>
10 <z>0</z>
11 </sector>
12 <logLines>
13 <time>37.84491729736328</time>
14 <state>0</state>
15 <action>1</action>
16 <playerPosition>
17 <x>24.560218811035156</x>
18 <y>-8.940696716308594e-8</y>
19 <z>3.3498525619506836</z>
20 </playerPosition>
21 <cameraRotation>
22 <x>0.24549755454063416</x>
23 <y>0.017123013734817505</y>
24 <z>0.031348951160907745</z>
25 <w>-0.9687389135360718</w>
26 </cameraRotation>
27 ...
```

# JSON

- Ler um ficheiro JSON em Python
- E.g., previsão metereológica de 5 dias para o Porto publicada pelo IPMA [aqui](#)
- JSON  $\simeq$  estruturas de dados Python

```
import json
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

- JSON = dicionários e listas aninhados uns nos outros, cujas folhas são strings, números ou booleanos
  - hierárquico: estrutura aninhada, em árvore
  - semi-estruturado: 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
```

# JSON

- Obter previsão para o dia mais próximo de hoje

```
import datetime
import dateutil.parser as date

data = dict['data']
weather = { date.parse(dia['forecastDate']) \
 : dia['idWeatherType'] for dia in data }

hoje = datetime.datetime.today()
dia = min(weather, key=lambda d : abs(hoje-d))
previsao = weather[dia]
```

# JSON

- Converter código de previsão numa descrição textual
- Descrições fornecidas pelo IPMA [aqui](#)

```
with open('weather-type-classe.json', 'r') as f:
 data = json.load(f) ['data']

classe = { d['idWeatherType'] \
 : d['descWeatherTypePT'] \
 for d in data }
tempo = classe[previsao]
```

# JSON

- 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)
```