Algoritmos para Análise de Sequências Biológicas

Ficha 4

Objetivo

- Dicionários
- Matrizes
- Exercícios sobre matrizes de pontos
- Matrizes de substituição como dicionários
- Sugestão de exercícios sobre matrizes

Dicionários

Características

- Associa chaves a valores
- Cada chave só pode existir uma vez
- Os valores podem ser qualquer tipo de dado

Métodos

get	Devolve o valor associado a uma chave ou o valor por omissão, caso exista
keys	Devolve todas as chaves
values	Devolve todos os valores
items	Devolve todos os pares chave/valor

Exemplo

```
>>> d = {'rui' : 17, 'carla' : 19}
>>> for k, v in d.items(): print(k, v)
rui 17
carla 19
>>> [chave for chave in d]
['rui', 'carla', 'ana']
>>> d['ana'] = 14
>>> [(k, v) for k, v in d.items()]
[('rui', 17), ('carla', 19), ('ana', 14)]
```

Exemplo

```
>>> d['ana']
14
>>> d['dinis']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'dinis'
>>> d.get('dinis')
>>> d.get('dinis', "não tem")
'não tem'
```

Características

- Valores podem ser qualquer coisa, incluindo dicionários e listas
- Permite construir estruturas muito elaboradas
- Aninhamento permite graus de complexidade arbitrários

Matrizes de substituição em Python

- Leia o ficheiro com o blosum62
- Crie uma função que devolva um dicionário de dicionários

Exemplo

```
>>> blos['C']
{'A': 0, 'R': -3, 'N': -3, 'D': -3, 'C': 9, 'Q': -3,
'E': -4, 'G': -3, 'H': -3, 'I': -1, 'L': -1, 'K': -3,
'M': -1, 'F': -2, 'P': -3, 'S': -1, 'T': -1, 'W': -2,
'Y': -2, 'V': -1, 'B': -3, 'Z': -3, 'X': -2, '*': -4}
>>> blos['R']['C']
-3
```

Matrizes em Python

```
>>> mat = [[1,2,3],[4,5,6],[7,8,9]]
>>> mat
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> mat[1]
[4, 5, 6]
>>> mat[2][1]
8
```

Criação de um objeto matriz

```
class Mat:
    def __init__(self, rows, cols):
        self.mat = [[0 for c in range(cols)]
                    for r in range(rows)]
    def numRows (self): return len(self.mat)
    def numCols (self): return len(self.mat[0])
    def __str__(self):
        "Devolve a matriz como uma string"
        return '\n'.join(' '.join(str(val) for val in row)
                         for row in self.mat)
    def __repr__(self):
        "Utilizado no repl quando se pede para ver a matriz"
        return str(self)
    def __getitem__ (self, n):
        "Interface para a indexação []"
        return self.mat[n]
```

Utilização

```
>>> m = Mat(3,4)
>>> m
0 0 0 0
0 0 0 0
0 0 0 0
>>> m[1][2] = 3
>>> m
0 0 0 0
0 0 3 0
0 0 0 0
```

Sugestão de exercícios sobre matrizes

1. Adicionar ou remover linhas ou colunas
2. Aplicar uma função a todos os elementos de uma matriz, e.g., logarítmo
3. Aplicar uma função a todas as linhas ou colunas de uma matriz, e.g., máximo ou soma
4. Devolver o índice do maior ou menor elemento de uma matriz
5. Devolver uma lista com os índices de uma matriz com uma dada propriedade
6. Criar uma cópia
7. Ler uma matriz a partir de um CSV
8. Somar ou multiplicar duas matrizes

Matrizes de pontos em Python

Criar uma classe **DotPlot** para representar matrizes de pontos com os seguintes atributos:

seq1, seq2	as sequências a colocar nas linhas e colunas
mat	a matriz de pontos

Métodos:

1. Faça o construtor que recebe as duas sequências
2. Faça os métodos que lhe permitam visualizar a matriz de pontos de forma agradável

Exercício

1. Implemente a matriz de pontos **DotPlot**
2. Adicione dois parâmetros opcionais ao construtor: o **window size** e o **stringency**, ambos com o valor 1 por omissão