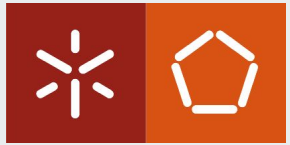
Processamento de Linguagem Natural Engenharia Biomédica

Luís Filipe Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt



Plano

- Introdução ao Python
- Unix Filters
- Expressões Regulares
- Corpora
- Gramática do Português
- Terminologia Dicionários e enciclopédias
- Word Embeddings
- Web Scraping
- Domain Specific Language
- NER / PoS / Q&A …

Ferramentas

- spaCy
- Stanza
- Flask
- BS4
- Selenium
- Gensim
- FastText
- HuggingFace
- NLTK
- Terminal

Ficheiros

- html
- xml
- json
- yaml
- latex
- tmx
- texto

Aulas

Sistema Operativo:

- Unix
- Python > 3.6

GitHub:

- https://github.com/lfcc1/plneb2223
- Bibliografia
- Aulas
- Data
- Slides
- Tpcs

Avaliação

- Testes
- Trabalhos práticos
- TPCs

- Teste - 40%
- Trabalhos práticos - 40%
- Trabalhos de casa - 20%
- Nota mínima 8

Trabalho obrigatório!

Datas: ...

TPC

- Criação de repositório Github: plneb-2223
- Criação de diretoria para cada TPC
  - TPC1
  - TPC2
  - ...

Github

Installation:

ubuntu:

```
sudo apt update
sudo apt upgrade
sudo apt install git
```

windows:
https://gitforwindows.org/

Setup:

```
echo "# Repositório PLNEB-2223" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/lfcc1/plneb-2223.git
git push -u origin master
```

Introdução ao Python

```
#!/usr/bin/env python3

print("Hello World")

editor = input("What is your favorite text editor? ")

a = [3, 6, 7, 2, 5, 9, 4, 0, 10]

for i in a:
  if i < 5:
    print(str(i) + " is lower than 5")
  elif i > 5:
    print(str(i) + " is higher than 5")
  else:
    print("FIVE!")

# comment
```

- Listas
- Dicionários
- Ficheiros
- Exercícios

Strings

Creating a String: "..." ou '...';
multiline String: """...""" ou '''...''';

```
string1 = "hello!"
string2 = 'hello'
multilineString1 = """Hello everyone!
I'm a multiline string.
"""

multilineString2 = '''Hello guys!
I'm also a multiline string!'''

string2 = 'I'm also a multiline string!'
```

Strings

* Comprimento duma string: `...len(frase)...`;
* Verificar se contém uma parte: `if "dia" in frase...`;
* Verificar se não contém uma parte: `if "dia" not in frase...`;
* Conversão para maiúsculas: `frase.upper()`;
* Conversão para minúsculas: `frase.lower()`;
* Remover espaço branco do início e do fim: `frase.strip()`;

`frase.lstrip()` `frase.rstrip()`

* Substituir partes: `frase.replace("Hoje", "Today")`;
* Partir uma string em bocados usando um separador: `frase.split("#")`;
* Concatenação de 2 strings: `s1 + s2`.

String Slicing

* Slicing a string: `frase[start:stop:step]`; # start through not past stop, by step
  - from the start: `frase[:stop]`;
  - until the end: `frase[start:]`;
  - negative indexes.
  - amount by which the index increases `frase[::step]`

```
text = "hello world"
slice = text[1:5]
slice = text[:5]
slice = text[5:]
slice = text[5:-1]
slice = text[-5:]
slice = text[:-2]
slice = text[::-1]
slice = text[::2]
slice = text[::-1]
```

Listas

```
listaA = [1,2,3]
listaB = [1,2,3,"Joana","Bruno","Filipe"]
lista = ["Joana","Bruno","Filipe"]

lista[2] # "Filipe"
"Susana" in lista # False

lista.append("Maria") # ['Joana', 'Bruno', 'Filipe', 'Maria']

lista.insert(2,"Pedro") # ['Joana', 'Bruno', 'Pedro', 'Filipe', 'Maria']

lista[3] = "João" # ['Joana', 'Bruno', 'Pedro', 'João', 'Maria']

concat = [1,2,3] + [4,5,6] # [1, 2, 3, 4, 5, 6]
```

Listas

```
lista = list(range(0,10))  #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

lista.pop(3) # [0, 1, 2, 4, 5, 6, 7, 8, 9]

lista.pop(-3) # [0, 1, 2, 4, 5, 6, 8, 9]

lista.remove(5) # [0, 1, 2, 4, 6, 8, 9]
```

Dicionários

```
dict = {}
colors = {"fcp": "blue", "slb": "red", "scp": "green"}

len(colors) # 3

"slb" in colors # True
"fcb" in colors # False

colors["fcp"] # blue
#colors["aca"] #KeyError: 'aca'

colors.get("aca") # None

colors.keys()  # ['fcp', 'slb', 'scp']
colors.values() # ['blue', 'red', 'green']
colors.items()  #[('fcp', 'blue'), ('slb', 'red'), ('scp', 'green')]
```

Listas por compreensão

```
newlist = [expression for item in iterable if condition == True]

lista = list(range(1,10))                # [1, 2, 3, 4, 5, 6, 7, 8, 9]
quadrados = [x * x for x in lista]       # [1, 4, 9, 16, 25, 36, 49, 64, 81]

pares = [x for x in lista if x % 2 == 0 ]    # [2, 4, 6, 8]

frutas = [x for x in fruits if “a” in x ]

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []
for x in fruits:
  if "a" in x:
    newlist.append(x)
```

Ordenação

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort() #['banana', 'kiwi', 'mango', 'orange', 'pineapple']
res = sorted(thislist) #['banana', 'kiwi', 'mango', 'orange', 'pineapple']

thislist.sort(reverse = True) #['pineapple', 'orange', 'mango', 'kiwi', 'banana']
res = sorted(thislist, reverse = True) #['pineapple', 'orange', 'mango', 'kiwi', 'banana']



frutas = {"orange":12, "mango":10, "kiwi":43, "pineapple":11, "banana":20}
res = sorted(frutas) #['banana', 'kiwi', 'mango', 'orange', 'pineapple']

res = sorted(frutas.items(), key=ordena) #[('mango', 10), ('pineapple', 11), ('orange', 12), ('banana',
20), ('kiwi', 43)]
```

```
def ordena(e):
    return e[1]
```

Ficheiros

```
file = open('data/exemplo', "r")
lines = file.readlines()

file1 = open('data/exemplo', "r")
string = file1.read()

file2 = open('data/exemplo', "r")
line = file2.readline(10)
```

```
['Ola! Bem vindo a Scripting no Processamento de Linguagem Natural \n', 'Este
ficheiro é apenas um exemplo.\n', 'Boa sorte!']
---
Ola! Bem vindo a Scripting no Processamento de Linguagem Natural
Este ficheiro é apenas um exemplo.
Boa sorte!
---
Ola! Bem v
```

Exercícios

1. Programa que pergunta ao utilizador o nome e imprime em maiúsculas.
2. Função que recebe array de números e imprime números pares.
3. Função que recebe nome de ficheiro e imprime linhas do ficheiro em ordem inversa.
4. Função que recebe nome de ficheiro e imprime número de ocorrências das 10 palavras mais frequentes no ficheiro.
5. Função que recebe um texto como argumento e o ”limpa”: separa palavras e pontuação com espaços, converte para minúsculas, remove acentuação de caracteres, etc.

Exercícios

Create a function that:

1. given a string “s”, reverse it.
2. given a string “s”, returns how many “a” and “A” characters are present in it.
3. given a string “s”, returns the number of vowels there are present in it.
4. given a string “s”, convert it into lowercase.
5. given a string “s”, convert it into uppercase.

Processamento de Linguagem Natural Engenharia Biomédica

Luís Filipe Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt