Python 2023 - Práctica 2

Objetivos

- Comenzar a resolver problemas un poco más complejos.
- Utilizar correctamente las estructuras de datos de Python tratando de hacer la elección más adecuada de acuerdo al problema.

Ejercicios

1. Tomando el texto del `README.md` de numpy, copiar y pegar el texto en una variable, luego imprima todas las líneas que contienen 'http' o 'https'.

2. Indique la palabra que aparece mayor cantidad de veces en el texto del `README.md` de numpy. Copie y pegue el texto en una varible.

Recordemos algunas funciones de string:

- lower
- split

Investigue el módulo Counter para simplificar la resolución.

Identificando mayúsculas, minúsculas y caracteres no letras

In [8]:
```
caracter = "T"
print(texto.split()[0].startswith(caracter))
```

```
True
```

¿Pero qué pasa si queremos saber indistintamente si la palabra comienza con dicha letra en minúscula o mayúscula?

In [9]:
```
caracter = "t"
print(texto.lower().split()[0].startswith(caracter))
```

```
True
```

¿Y si el caracter ingresado no es una letra?

In [10]:
```
import string
caracter = "?"
print(caracter in string.ascii_letters)
```

```
False
```

3. Dado el siguiente texto guardado en la varible *jupyter_info*, solicite por teclado **una letra** e imprima las palabras que comienzan con dicha letra. En caso que no se haya inrgesado un letra, indique el error. *Ver: módulo string*

```
jupyter_info = """ JupyterLab is a web-based interactive development
environment for Jupyter notebooks,
code, and data. JupyterLab is flexible: configure and arrange the user
interface to support a wide range
of workflows in data science, scientific computing, and machine learning.
JupyterLab is extensible and
modular: write plugins that add new components and integrate with existing
ones.
"""
```

4. Para la aceptación de un artículo en un congreso se definen las siguientes especificaciones que deben cumplir y recomendaciones de escritura:

- **título**: 10 palabras como máximo
- cada oración del **resumen**:
  - hasta 12 palabras: fácil de leer
  - entre 13-17 palabras: aceptable para leer
  - entre 18-25 palabras: difícil de leer
  - mas de 25 palabras: muy difícil

Dado un artículo en formato string, defina si cumple las especificaciones del título y cuántas oraciones tiene de cada categoría. El formato estándar en que recibe el string tiene la siguiente forma:

```
evaluar = """ título: Experiences in Developing a Distributed Agent-based
Modeling Toolkit with Python
resumen: Distributed agent-based modeling (ABM) on high-performance
computing resources provides the promise of capturing unprecedented details
of large-scale complex systems. However, the specialized knowledge required
for developing such ABMs creates barriers to wider adoption and utilization.
Here we present our experiences in developing an initial implementation of
Repast4Py, a Python-based distributed ABM toolkit. We build on our
experiences in developing ABM toolkits, including Repast for High
Performance Computing (Repast HPC), to identify the key elements of a useful
distributed ABM toolkit. We leverage the Numba, NumPy, and PyTorch packages
and the Python C-API to create a scalable modeling system that can exploit
the largest HPC resources and emerging computing architectures.
"""
```

En este ejemplo se debe informar:

- título: ok
- Cantidad de oraciones fáciles de leer: 0, aceptables para leer: 2, dificil de leer: 1, muy difícil de leer: 2

5. Dada una frase y un string ingresados por teclado (en ese orden), genere una lista de palabras, y sobre ella, informe la cantidad de palabras en las que se encuentra el string. No distingir entre mayúsculas y minúsculas.

Ejemplo 1

- **Para la frase**: "Tres tristes tigres, tragaban trigo en un trigal, en tres tristes trastos, tragaban trigo tres tristes tigres."
- **Palabra**: "tres"
- **Resultado**: 3

Ejemplo 2

- **Para la frase**: "Tres tristes tigres, tragaban trigo en un trigal, en tres tristes trastos, tragaban trigo tres tristes tigres."
- **Palabra**: "tigres"
- **Resultado**: 2

Ejemplo 3

- **Para la frase**: "Tres tristes tigres, tragaban trigo en un trigal, en tres tristes trastos, tragaban trigo tres tristes tigres."
- **Palabra**: "TRISTES"
- **Resultado**: 3

6. Retomamos el código visto en la teoría, que informaba si una palabra contenía la letra *a* en una palabra ingresada

In [1]:
```
palabra = input("Ingresá una palabra: ")
if "a" in palabra:
    print("Hay letras a.")
else:
    print("No hay letras a. ")
```

```
Ingresá una palabra: mundo
No hay letras a.
```

Si ahora queremos saber si contiene la letra *a* y también la letra *n*, cómo lo modificamos?

7. Dada una frase identificar mayúsculas, minúsculas y caracteres no letras y contar la cantidad de palabras sin distinguir entre mayúsculas y minúsculas, en la frase.

```
texto = """
 El salario promedio de un hombre en Argentina es de $60.000, mientras que el de una mujer es de $45.000. Además, las mujeres tienen menos posibilidades de acceder a puestos de liderazgo en las empresas.
  """
```

8. Escriba un programa que solicite que se ingrese una palabra o frase y permita identificar si la misma es un Heterograma (tenga en cuenta que el contenido del enlace es una traducción del inglés por lo cual las palabras que nombra no son heterogramas en español). Un Heterograma es una palabra o frase que no tiene ninguna letra repetida entre sus caracteres.

Tener en cuenta

- Lo que no se puede repetir en la frase son sólo aquellos caracteres que sean letras.
- No se distingue entre mayúsculas y minúsculas, es decir si en la frase o palabra tenemos la letra "T" y la letra "t" la misma NO será un Hererograma.
- Para simplificar el ejercicio vamos a tomar como que las letras con tilde y sin tilde son distintas. Ya que Python las diferencia:

```
>>> 'u' == 'ú'
False
```

Ejemplos

Entreda	¿Heterograma?
cruzamiento	Sí
centrifugados	Sí
portón	Sí
casa	No
día de sol	No
con diez uñas	Sí
no-se-duplica	Sí

9. Escriba un programa que solicite por teclado una palabra y calcule el valor de la misma dada la siguiente tabla de valores del juego Scrabble:

Letra	valor
A, E, I, O, U, L, N, R, S, T	1
D, G	2
B, C, M, P	3
F, H, V, W, Y	4
K	5
J, X	8
Q, Z	10

**Tenga en cuenta qué estructura elige para guardar estos valores en Python*

Ejemplo 1

- **Palabra**: "solo"
- **valor**: 4

Ejemplo 2

- **Palabra**: "tomate"
- **valor**: 8

10. Dada una lista de nombres de estudiantes y dos listas con sus notas en un curso, escriba un programa que manipule dichas estructuras de datos para poder resolver los siguientes puntos:
    A. Generar una estructura con todas las notas relacionando el nombre del estudiante con las notas. Utilizar esta estructura para la resolución de los siguientes items.
    B. Calcular el promedio de notas de cada estudiante.
    C. Calcular el promedio general del curso.
    D. Identificar al estudiante con la nota promedio más alta.
    E. Identificar al estudiante con la nota más baja.

Nota:

- Las 3 estructuras están ordenadas de forma que los elementos en la misma posición corresponden a un mismo alumno.
- Realizar funciones con cada item

In [1]:
```
nombres = ''' 'Agustin', 'Alan', 'Andrés', 'Ariadna', 'Bautista', 'CAROLINA', 'CESAR'
'David','Diego', 'Dolores', 'DYLAN', 'ELIANA', 'Emanuel', 'Fabián', 'Facundo',
'Francsica', 'FEDERICO', 'Fernanda', 'GONZALO', 'Gregorio', 'Ignacio', 'Jonathan',
'Joaquina', 'Jorge','JOSE', 'Javier', 'Joaquín'  , 'Julian', 'Julieta', 'Luciana',
'LAUTARO', 'Leonel', 'Luisa', 'Luis', 'Marcos', 'María', 'MATEO', 'Matias',
'Nicolás',  'Nancy', 'Noelia', 'Pablo', 'Priscila', 'Sabrina', 'Tomás', 'Ulises',
'Yanina' '''
notas_1 = [81,  60, 72, 24, 15, 91, 12, 70, 29, 42, 16, 3, 35, 67, 10, 57, 11, 69,
           12, 77, 13, 86, 48, 65, 51, 41, 87, 43, 10, 87, 91, 15, 44,
           85, 73, 37, 42, 95, 18, 7,     74, 60, 9, 65, 93, 63, 74]
notas_2 = [30, 95, 28, 84, 84, 43, 66, 51, 4, 11, 58, 10, 13, 34, 96, 71, 86, 37,
           64, 13, 8, 87, 14, 14, 49, 27, 55, 69, 77, 59, 57, 40, 96, 24, 30, 73,
           95, 19, 47, 15, 31, 39, 15, 74, 33, 57, 10]
```

Entrega 2

Pautas

- Suba la resolución total del **ejercicio 10** al repositorio individual de Github, luego elija uno de los siguientes items: A,C,D o E y realice un video explicando cómo lo resolvió y las decisiones que tomó implementando map, zip, lambda (por qué utilizó cada estructura de datos o estructura de control) y muestre la ejecución del programa en la terminal.
- **Duración máxima del video**: 5 minutos
- **Puntos**: 15.
- **Fecha límite de entrega**: Viernes, 14 de abril de 2023, 23:59
- **Modalidad de entrega**: Subir el programa al repositorio de github y copie el enlace del repositorio junto con el link del video en la resolución de la tarea de Cátedras.

In [ ]:
```
```