

# Python 2024 - Práctica 2

## Objetivos

- Comenzar a resolver problemas más complejos.
- Utilizar correctamente las estructuras de datos de Python tratando de hacer la elección más adecuada de acuerdo al problema.
- Comenzar a trabajar con Jupyter Notebook.
- Desarrollar considerando las dependencias de nuestro sistema

## Recomendación

- Realizar la practica en un entorno de **Jupyter Notebook** ya que se espera que cierta parte de la resolución del trabajo integrador se realice utilizando dicha tecnología.

## Ejercicios

1. Tomando el texto del `README.md` de [numpy](#), copiar y pegar el texto en una variable, luego imprima todas las líneas cuya *segunda* palabra comience con una vocal (A, E, I, O, U, a, e, i, o, u)
2. Indique la palabra con *más de 4 caracteres* que aparece mayor cantidad de veces en el texto del `README.md` de numpy. Copie y pegue el texto en una variable.

Recordemos algunas funciones de string:

- `lower`
- `split`

Investigue el módulo [Counter](#) para simplificar la resolución.

3. Dado el siguiente texto guardado en la variable `jupyter_info`, solicite por teclado **una letra** e imprima las palabras que contengan dicha letra. En caso que no se haya ingresado 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. Dado el siguiente código que informa si la letra `a` se encuentra en una palabra ingresada por teclado:

Código:

```
word = input("Ingresa una palabra: ")
if "a" in word:
    print("Hay al menos una letra a.")
else:
    print("No hay letras a. ")
```

Salida:

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

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

5. 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

Es importante destacar que los números no se consideran palabras al momento del análisis por su facilidad de lectura.

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:

```
article = """ título: Experiences in Developing a Distributed Agent-
based Modeling Toolkit with Python Version 3
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: not ok
- Cantidad de oraciones fáciles de leer: 1, aceptables para leer: 2, difícil de leer: 1, muy difícil de leer: 2

6. 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 distinguir 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

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

```
text = """ La brecha salarial alcanzó el 27,7%: las mujeres ocupadas
debieron trabajar 8 días y 10 horas más que los varones ocupados para
ganar lo mismo que ellos en un mes. """
```

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 Heterograma.
- Para simplificar el ejercicio vamos a tomar como que las letras con tilde y sin tilde son distintas. Ya que Python las diferencia:

### 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. Un club de hockey posee las estadísticas de todos sus jugadores de la última temporada jugada del torneo de hockey mixto. Poseen una lista con los nombres, otra

con los goles a favor, otra con los goles evitados y la última con las asistencias realizadas. El club está por otorgar los premios de cierre de temporada y necesita:

1. Generar una estructura todas las estadísticas asociadas a cada jugador o jugadora.
2. Conocer el nombre y la cantidad de goles del goleador o goleadora.
3. Conocer el nombre del jugador o jugadora más influyente, esto se consigue sumando goles a favor, goles evitados y cantidad de asistencias. La particularidad es que los goles a favor, evitados y las asistencias NO valen lo mismo (es un promedio ponderado):

	valor
goles a favor	1.5
goles en evitados	1.25
asistencias	1

4. Conocer el promedio de goles por partido del equipo en general. Dato: Se jugaron 25 partidos en la temporada.
5. Conocer el promedio de goles por partido del goleador o goleadora. Dato: Se jugaron 25 partidos en la temporada.

**Nota:**

- Las 4 estructuras están ordenadas de forma que los elementos en la misma posición corresponden a un mismo jugador o jugadora.
- Realizar funciones con cada ítem

**Código:**

```
names = "" Agustin, Yanina, Andrés, Ariadna, Bautista, CAROLINA,
CESAR, David, Diego, Dolores, DYLAN, ELIANA, Emanuel, Fabián, Noelia,
Francsica', FEDERICO, Fernanda, GONZALO, Nancy ""
goals = [0, 10, 4, 0, 5, 14, 0, 0, 7, 2, 1, 1, 1, 5, 6, 1, 1, 2, 0,
11]
goals_avoided = [0, 2, 0, 0, 5, 2, 0, 0, 1, 2, 0, 5, 5, 0, 1, 0, 2,
3, 0, 0]
assists = [0, 5, 1, 0, 5, 2, 0, 0, 1, 2, 1, 5, 5, 0, 1, 0, 2, 3, 1,
0]
```

---

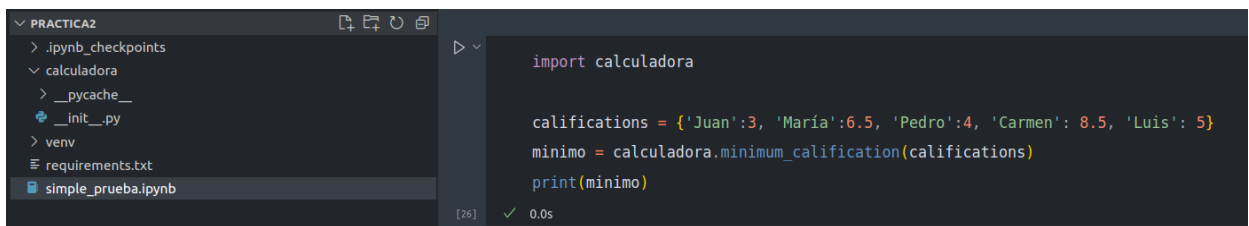
## Entrega 2

### Pautas

- Suba la resolución total del **ejercicio 10** al repositorio individual de Github, luego elija uno de los siguientes items: A,B,C o D 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.

Es **obligatorio** desarrollar un módulo en Python para la solución. Una posibilidad es que este módulo se encargue de la construcción de la estructura de datos, del cálculo necesario para el sub-inciso 2, sub-inciso 3, sub-inciso 4 y sub-inciso 5 (cada uno por separado). Una estructura de archivos podría ser la siguiente (dentro del modulo calculadora se pueden añadir más archivos si el código crece en tamaño y lo consideran pertinente).

La imagen siguiente pretende demostrar una estructura posible del modulo y su importación desde una Jupyter Notebook, se debe considerar que el ejemplo de código pertenece a la explicación práctica y no específicamente a la Practica 2.



```
import calculadora

califications = {'Juan':3, 'Maria':6.5, 'Pedro':4, 'Carmen': 8.5, 'Luis': 5}
minimo = calculadora.minimum_calification(califications)
print(minimo)
```

[26] ✓ 0.0s

- **Duración máxima del video:** 5 minutos
    - Características generales: en el video se debe poder ver claramente el código funcionando y a su vez ustedes mismos realizando la explicación. Pueden utilizar el software de grabación que deseen. En caso de no conocer ninguno [OBS](#) es una buena alternativa para investigar. En caso de que exista una **complicación insalvable** con la grabación en simultaneo del código y su imagen puede optarse por la entrega de dos videos (uno del código y otro suya explicándolo).
  - **Puntos:** 15.
  - **Fecha límite de entrega:** Viernes, 12 de abril de 2024, 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. Verifique que el repositorio sea público.
-