



PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

TAREA ONLINE PIA01

Unidad 1: Introducción a la programación de Inteligencia Artificial

INDICE

APARTADO 1: CREAR CUENTA EN GITHUB Y CREAR UN REPOSITORIO.....	3
SOLUCIÓN	3
CREAR UNA CUENTA EN GITHUB	3
CREAR UN REPOSITORIO EN GITHUB.....	3
APARTADO 2: RESOLVER CIERTOS PROBLEMAS EN PYTHON.....	4
SOLUCIÓN	5
PROBLEMA 1. PROCESAMIENTO DE UNA LISTA DE ENTEROS	5
PROBLEMA 2. FRECUENCIA DE PALABRAS EN UN TEXTO.....	6
PROBLEMA 3. TRABAJO CON CONJUNTOS	7
APARTADO 3: CONSULTAR COMPETICIÓN EN PLATAFORMA DE IA KAGGLE.....	8
SOLUCIÓN	8
CREAR UNA CUENTA EN KAGGLE.....	8
ACCEDER A UNA COMPETICIÓN ACTIVA.....	8
DESCARGAR EL DATASET.....	9
SUBIR LOS DOCUMENTOS A GITHUB.....	10

APARTADO 1: CREAR CUENTA EN GITHUB Y CREAR UN REPOSITORIO

Accede a la página web de GitHub y sigue los pasos para registrarte y crearte una cuenta.

Cuando se te pida que especifiques si eres alumno o profesor, pulsa en la opción que aparece en la parte inferior de la pantalla "skip personalization". Tras concluir el proceso de registro, crea tu primer repositorio, incluyendo, de momento, un archivo pdf en el que añadas, una breve explicación de cómo lo has hecho

SOLUCIÓN

A continuación, se describen brevemente los pasos a seguir para crear la cuenta y un repositorio en GitHub

CREAR UNA CUENTA EN GITHUB

1. Acceder a la web oficial del GitHub (<https://github.com/>) y pulsar en **Sign up**
2. Introducir email, contraseña y el nombre de usuario para identificarnos
3. Cuando se pida especificar alumno o profesor, pulsar en la opción que aparece en la parte inferior de la pantalla **Skip personalization**
4. Verificar la cuenta a través del correo electrónico

CREAR UN REPOSITORIO EN GITHUB

1. Una vez dentro de GitHub, hacemos clic en el símbolo "+" en la esquina superior derecha y seleccionamos **"New repository"**.
2. Establecemos un nombre para el repositorio, en este caso lo nombraremos siguiendo las pautas para la entrega de tareas **apellido1_apellido2_nombre_PIA01_Tarea**
3. En el apartado de visibilidad lo dejamos con la opción que viene por defecto en **Public**
4. Finalmente hacemos clic en **Create repository**

The screenshot shows the GitHub 'New repository' page. The 'General' section includes fields for 'Owner' (miigueuee), 'Repository name' (suarez_pajares_miguelangel_PIA01_Tarea), and 'Description' (Repositorio creado para subir la Tarea de la unidad 1 de PIA). The 'Configuration' section includes 'Choose visibility' (Public), 'Add README' (Off), 'Add .gitignore' (No .gitignore), and 'Add license' (No license). A 'Create repository' button is at the bottom right.

APARTADO 2: RESOLVER CIERTOS PROBLEMAS EN PYTHON

Dado que a lo largo del año vamos a tener que trabajar bastante con Python, es necesario tener cierta base sobre los aspectos básicos del lenguaje. Para ello se propone la realización de los siguientes ejercicios que deberán ser subidos al repositorio GitHub del Apartado 1.

○ Problema 1. Procesamiento de una lista de enteros.

Crea una función que reciba una lista de enteros por parámetro y devuelva otra lista, de acuerdo a las siguientes acciones:

1. Eliminar los números negativos de la lista.
2. Eliminar los valores que están repetidos, quedándonos con uno de ellos.
3. Ordenar los números resultantes de menor a mayor.

Por ejemplo, si le pasara [4, -1, 2, 4, 3, -5, 2], debería retornar [2,3,4].

○ Problema 2. Frecuencia de palabras en un texto.

Escribe una función que reciba por parámetro una lista de palabras y la ruta a un fichero de texto y devuelva un diccionario que muestre cuantas veces aparecen distintas palabras de la lista en el fichero de texto. Haz un pequeño

programa que la ponga a prueba.

Requisitos:

1. Eliminar signos de puntuación y convertir todo a minúsculas.
2. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia.
3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.

○ Problema 3. Trabajo con conjuntos

Escribe una función que reciba dos listas de enteros y devuelva un diccionario con la siguiente información **(ES**

OBLIGATORIO USAR CONJUNTOS PARA CALCULARLOS)

1. La intersección de ambos conjuntos (elementos comunes).
2. La unión de ambos conjuntos (todos los elementos sin duplicados).
3. La diferencia simétrica (elementos que están en uno u otro conjunto, pero no en ambos).

SOLUCIÓN

PROBLEMA 1. PROCESAMIENTO DE UNA LISTA DE ENTEROS

```
# Función que recibe una lista de enteros por parámetro y devuelve una lista de los
positivos sin duplicados
def procesar_lista(lista_enteros):
    # 1. Eliminar los números negativos de la lista
    lista_positivos = [num for num in lista_enteros if num >= 0]

    # 2. Eliminar los valores que están repetidos, quedándonos con uno de ellos
    lista_sin_duplicados = list(set(lista_positivos))

    # 3. Ordenar los números resultantes de menor a mayor
    lista_sin_duplicados.sort()

    return lista_sin_duplicados

# Ejemplo de prueba, pasamos [4, -1, 2, 4, 3, -5, 2], debería imprimir [2,3,4]
print(procesar_lista([4, -1, 2, 4, 3, -5, 2]))
```

PROBLEMA 2. FRECUENCIA DE PALABRAS EN UN TEXTO

```

import string

# Función que recibe por parámetro una lista de palabras y la ruta a un fichero de texto
# Devuelve un diccionario que muestra cuantas veces aparecen las distintas palabras de la
# lista en el fichero de texto
def frecuencia_palabras(lista_palabras, ruta_fichero):

    # Leer el contenido del archivo de entrada.
    with open(ruta_fichero, "r", encoding="utf-8") as archivo:
        contenido_archivo = archivo.read()

    # 1. Eliminar signos de puntuación y convertir todo a minúsculas
    texto_minusculas = contenido_archivo.lower()
    for signo in string.punctuation:
        texto_minusculas_sin_signos = texto_minusculas.replace(signo, "")

    # Dividir en palabras
    texto_en_palabras = texto_minusculas_sin_signos.split()

    # 2. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia
    # Inicializar diccionario
    frecuencias = {}
    lista = [p.lower() for p in lista_palabras]

    # Contar frecuencia
    for palabra in lista:
        frecuencias[palabra] = texto_en_palabras.count(palabra)

    # 3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra
    return dict(sorted(frecuencias.items()))

# Prueba del programa

palabras = ["inteligencia", "artificial", "python"] # Lista de palabras a buscar
ruta = "texto.txt" # Ruta al archivo de texto (ubicado en la misma carpeta)

# Debería imprimir {'artificial': 0, 'inteligencia': 1, 'python': 3}
print(frecuencia_palabras(palabras, ruta))

```

PROBLEMA 3. TRABAJO CON CONJUNTOS

```
# Función que recibe dos listas de enteros y devuelve un diccionario con la siguiente
información
# 1. La intersección de ambos conjuntos (elementos comunes).
# 2. La unión de ambos conjuntos (todos los elementos sin duplicados).
# 3. La diferencia simétrica (elementos que están en uno u otro conjunto, pero no en
ambos).

def operaciones_conjuntos(lista1, lista2):
    # Convertir listas a conjuntos
    conjunto1 = set(lista1)
    conjunto2 = set(lista2)

    # Devolver diccionario con las operaciones de conjuntos
    diccionario = {
        # 1. La intersección de ambos conjuntos (elementos comunes).
        "Intersección": conjunto1 & conjunto2,

        # 2. La unión de ambos conjuntos (todos los elementos sin duplicados)
        "Unión": conjunto1 | conjunto2,

        # 3. La diferencia simétrica
        "Diferencia simétrica": conjunto1 ^ conjunto2
    }

    return diccionario

# Ejemplo de prueba
print(operaciones_conjuntos([1,2,3], [3,4,5]))
```

APARTADO 3: CONSULTAR COMPETICIÓN EN PLATAFORMA DE IA KAGGLE

Crea una cuenta en Kaggle y haz las siguientes tareas:

- o Accede a una competición activa.
- o Descarga el dataset usado para esa competición.
- o Sube al repositorio Github del primer apartado, un documento con pantallazos de cómo has realizado el proceso y del dataset descargado.

SOLUCIÓN

CREAR UNA CUENTA EN KAGGLE

1. Vamos a [kaggle.com](https://www.kaggle.com).
2. Hacemos clic en “**Register**” y creamos una cuenta
3. Verificamos la cuenta a través del correo electrónico

ACCEDER A UNA COMPETICIÓN ACTIVA

Desde el menú lateral accedemos a la sección **Competitions**

The screenshot shows the Kaggle website's 'Competitions' section. On the left is a sidebar with navigation links: Home, Competitions (selected), Datasets, Models, Benchmarks, Game Arena, Code, Discussions, Learn, More, Your Work, Viewed, and a specific competition 'Titanic - Machine Learning from Disaster'. The main area has a 'Competitions' header with a sub-header 'Grow your data science skills by competing in our exciting competitions. Find help in the [documentation](#) or learn about [Community Competitions](#).' Below this are buttons for 'Host a Competition' and 'Your Work'. A search bar and a 'Filters' button are also present. A 'Getting Started' section highlights 'Competitions with approachable ML fundamentals.' Below this, four competition cards are displayed: 'Titanic - Machine Learning from Disaster' (16575 Teams), 'House Prices - Advanced Regression Techniques' (6048 Teams), 'Spaceship Titanic' (2578 Teams), and 'LLM Classification Finetuning' (303 Teams). Each card includes a brief description and a 'Knowledge' tag.

Disponemos de una barra de búsqueda y filtros para localizar competiciones, en este caso vamos a seleccionar la primera que aparece **Titanic - Machine Learning from Disaster**

DESCARGAR EL DATASET

Una vez dentro de la competición que hemos seleccionado, accedemos a la pestaña **Data**

Descargamos el **dataset** haciendo clic en **Download**, en este caso se trata de un zip **titanic.zip** con 3 archivos csv:

gender_submission.csv, **test.csv** y **train.csv**

The screenshot shows the Kaggle website for the 'Titanic - Machine Learning from Disaster' competition. The 'Data' tab is selected, displaying a 'Data Dictionary' and 'Variable Notes'. A red arrow points to the 'Download All' button in the 'Data Explorer' section, which is circled in red. Below the main content, a file explorer window titled 'titanic.zip' is open, showing the contents of the downloaded zip file: 'gender_submission.csv', 'test.csv', and 'train.csv'.

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

pclass: A proxy for socio-economic status (SES)
 1st = Upper
 2nd = Middle
 3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...
 Sibling = brother, sister, stepbrother, stepsister
 Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...
 Parent = mother, father
 Child = daughter, son, stepdaughter, stepson
 Some children travelled only with a nanny, therefore parch=0 for them.

Data Explorer

93.08 kB
 gender_submission.csv
 test.csv
 train.csv

Summary

3 files
 25 columns

Download All

titanic.zip

Archivo Acciones Herramientas Favoritos Opciones Ayuda

Agregar Extraer en Comprobar Ver Eliminar Buscar Asistente Información Buscar virus Comentar Autoextraíble

titanic.zip - Comprimido ZIP64, tamaño descomprimido de 93 081 bytes

Nombre	Tamaño	Comprimido	Tipo	Modificado	CRC32
..			Carpeta de archivos		
gender_submission.csv	3 258	872	Archivo de valores sep...	11/12/2019 2:17	72EBAA41
test.csv	28 629	11 171	Archivo de valores sep...	11/12/2019 2:17	36B53FB0
train.csv	61 194	22 388	Archivo de valores sep...	11/12/2019 2:17	84224229

Total 3 archivos, 93 081 bytes

SUBIR LOS DOCUMENTOS A GITHUB

Accedemos al repositorio que hemos creado en el primer apartado, hacemos clic en la opción [uploading an existing file](#), subimos los archivos arrastrándolos a la ventana de GitHub y hacemos clic en **Commit** para guardarlos

- **suarez_pajares_miguelangel_PIA01_Tarea**: Documento pdf que contiene las soluciones a los apartados.
- **Problema1-ProcesamientoLista**: Archivo que contiene el código en Python con la solución al problema 1
- **Problema2-FrecuenciaPalabras**: Archivo que contiene el código en Python con la solución al problema 2
- **Problema3-Conjuntos**: Archivo que contiene el código en Python con la solución al problema 3
- **texto**: Fichero de texto necesario para ejecutar correctamente el código del problema 2
- **titanic**: archivo zip que contiene el dataset de la competición **Titanic - Machine Learning from Disaster**

The screenshot shows the GitHub web interface for uploading files to a repository named 'suarez_pajares_miguelangel_PIA01_Tarea' under the user 'miigueueee'. The browser address bar shows the URL 'https://github.com/miigueueee/suarez_pajares_miguelangel_PIA01_Tarea/upload'. The repository navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main area displays a large box with a file icon and the text 'Drag additional files here to add them to your repository' and 'Or choose your files'. Below this, a list of files is shown, each with a file icon, name, and a close button (x):

- Problema1-ProcesamientoLista.py
- Problema2-FrecuenciaPalabras.py
- Problema3-Conjuntos.py
- texto.txt
- titanic.zip
- suarez_pajares_miguelangel_PIA01_Tarea.pdf

At the bottom, there is a 'Commit changes' section with a plus icon. It contains a text input field for 'Add files via upload' and a larger text area for 'Add an optional extended description...'. Below these fields are two buttons: 'Commit changes' (green) and 'Cancel' (white with a red border). The footer of the page shows the GitHub logo, copyright information '© 2025 GitHub, Inc.', and various links like Terms, Privacy, Security, Status, Community, Docs, Contact, Manage cookies, and Do not share my personal information. The system tray at the bottom right shows the time '0:16' and date '11/12/2025'.