TAREA DE INVESTIGACIÓN

MÉTODOS EN ARREGLOS NUMPY

QUIROGA JUAN MARCOS

E010-50

Programación Orientada a Objetos

Introducción:

Python es un lenguaje de programación de alto nivel donde una de sus ventajas es que es un lenguaje para Programación Orientada a Objetos (P.O.O.). Además. Python permite la importación de librerías que son de gran importancia para poder realizar programas, una de ellas es la librería NumPy.

NumPy es una librería de Python que provee una estructura de datos arreglo n-dimensional, esta estructura de datos nos permite trabajar de forma similar a las listas normales de Python, pero con la ventaja de que tienen una mayor velocidad y una amplia lista de métodos integrados que son muy útiles a la hora de trabajar con Clases.

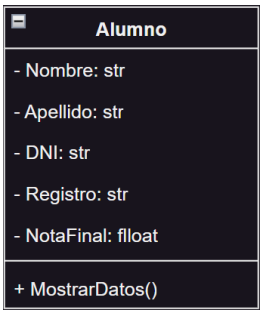
Estos arreglos nos permiten almacenar distintos tipos de datos como enteros, reales, flotantes, cadenas u objetos. En este informe se enfocará en el uso de objetos en los arreglos.

Palabras clave:

- **Arreglo:** Una variable arreglo es una estructura de datos y representa un conjunto homogéneo de datos identificados por un único nombre y almacenado en posiciones contiguas de memoria.
- **Lista:** Las listas de Python son un tipo de dato que permite almacenar datos de cualquier tipo. Son mutables y dinámicas.
- **Objeto:** Un objeto del problema es una entidad, física o conceptual, caracterizada a través de atributos y comportamiento.
- **Clase:** Una clase es una descripción de un conjunto de objetos, ya que consta de comportamientos y atributos que resumen las características comunes del conjunto.
- **Atributos de un objeto:** Los atributos describen la abstracción de características individuales que posee un objeto.

Desarrollo:

Para los ejemplos que se desarrollarán, se definió una clase Alumno que tiene como atributos: Nombre, Apellido, DNI, Registro y NotaFinal. Además cuenta con el método MostrarDatos(), que muestra los atributos mencionados anteriormente. Se desarrollará un arreglo NumPy que almacena alumnos que pertenecen a una materia en particular.



Método np.append(arreglo origen, arreglo):

Este método es esencial para los gestores de datos. El método np.append se utiliza para agregar un nuevo elemento al final del arreglo. En el siguiente ejemplo se utiliza el método para realizar una carga de alumnos.

```
def agregarAlumno(self, alumno):
    self.__listaAlumno = np.append(self.__listaAlumno, alumno)
def cargaAlumno(self):
    nombre = input("Ingrese nombre del alumno a cargar (0 para finalizar): ")
    while nombre != '0':
        apellido = input("Ingrese apellido del alumno: ")
        dni = int(input("Ingrese DNI del alumno: "))
        registro = input("Ingrese N° de registro del alumno: ")
        nota = float(input("Ingrese nota final del alumno: "))
        alumno = Alumno(nombre, apellido, dni, registro, nota)
        self.agregarAlumno(alumno)
```

nombre = input("Ingrese nombre del alumno a cargar (0 para finalizar): ")

En las líneas de código anterior se puede observar como se realiza la carga mediante un bucle *for* y el programa va solicitando al usuario que ingrese los datos del alumno, para luego crear una instancia de la clase Alumno y luego almacenarlo en el arreglo mediante el método *agregarAlumno()* donde se hace uso del método np.append(). Se hará uso de esta carga para el resto de los métodos.

```
def mostrarArreglo(self):
    i = 0
    while i < len(self.__listaAlumno):
        print(self.__listaAlumno[i])
        i += 1
```

```
Ingrese nombre del alumno a cargar (0 para finalizar): Juan Marcos
Ingrese apellido del alumno: Quiroga
Ingrese DNI del alumno: 44431847
Ingrese Nº de registro del alumno: E010-50
Ingrese nota final del alumno: 8.25
Ingrese nombre del alumno a cargar (0 para finalizar): Ignacio
Ingrese apellido del alumno: Pereyra
Ingrese DNI del alumno: 45879322
Ingrese Nº de registro del alumno: E009-34
Ingrese nota final del alumno: 9
Ingrese nombre del alumno a cargar (0 para finalizar): 0
Apellido y Nombre: Quiroga Juan Marcos   DNI: 44431847
Nº de registro: E010-50    Nota final: 8.25
Apellido y Nombre: Pereyra Ignacio   DNI: 45879322
Nº de registro: E009-34    Nota final: 9.0
```

En esta imagen se puede observar la terminal con dos instancias de ejemplo, se puede observar cómo se muestran los objetos de manera ordenada luego de realizar la carga.

Método np.sort(arreglo origen):

Este método es utilizado para poder ordenar los elementos dentro del arreglo. Para poder utilizar este método con instancias de clases, es necesario sobrecargar el operador *__gt__* o *__lt__* dependiendo el orden que desee realizar.

```
def mostrarArregloOrdenado(self):
    i = 0
    self.__listaAlumno = np.sort(self.__listaAlumno)
    while i < len(self.__listaAlumno):
        print(self.__listaAlumno[i])
        i += 1
```

```
def __lt__(self, other):
    return self.__nota < other.__nota
```

Sobrecarga del operador *__lt__* necesario para utilizar el método np.sort().

```
Ingrese nombre del alumno a cargar (0 para finalizar): Juan Marcos
Ingrese apellido del alumno: Quiroga
Ingrese DNI del alumno: 44431847
Ingrese Nº de registro del alumno: E010-50
Ingrese nota final del alumno: 7.50
Ingrese nombre del alumno a cargar (0 para finalizar): Ezequiel
Ingrese apellido del alumno: Barco
Ingrese DNI del alumno: 42768933
Ingrese Nº de registro del alumno: E009-23
Ingrese nota final del alumno: 4.75
Ingrese nombre del alumno a cargar (0 para finalizar): Claudio
Ingrese apellido del alumno: Echeverri
Ingrese DNI del alumno: 47223960
Ingrese Nº de registro del alumno: E010-78
Ingrese nota final del alumno: 9
Ingrese nombre del alumno a cargar (0 para finalizar): 0
Apellido y Nombre: Barco Ezequiel   DNI: 42768933
Nº de registro: E009-23    Nota final: 4.75
Apellido y Nombre: Quiroga Juan Marcos   DNI: 44431847
Nº de registro: E010-50    Nota final: 7.5
Apellido y Nombre: Echeverri Claudio   DNI: 47223960
Nº de registro: E010-78    Nota final: 9.0
```

En esta imagen se puede observar la terminal con tres instancias de ejemplo luego de ser ordenadas por nota final de forma ascendente.

Método copy():

Copia un arreglo para almacenar sus datos en otra variable, su formato es: *arreglo2 = arreglo1.copy()* (siendo *arreglo1* la variable destino y *arreglo2* la variable origen).

```
def copiarArreglo(self):
    arreglo = self.__listaAlumno.copy()
    for i in arreglo:
        print(i)
```

```
Ingrese nombre del alumno a cargar (0 para finalizar): Juan Marcos
Ingrese apellido del alumno: Quiroga
Ingrese DNI del alumno: 44431847
Ingrese N° de registro del alumno: E010-50
Ingrese nota final del alumno: 7.50
Ingrese nombre del alumno a cargar (0 para finalizar): Ezequiel
Ingrese apellido del alumno: Barco
Ingrese DNI del alumno: 42768933
Ingrese N° de registro del alumno: E009-23
Ingrese nota final del alumno: 4.75
Ingrese nombre del alumno a cargar (0 para finalizar): Claudio
Ingrese apellido del alumno: Echeverri
Ingrese DNI del alumno: 47223960
Ingrese N° de registro del alumno: E010-78
Ingrese nota final del alumno: 9
Ingrese nombre del alumno a cargar (0 para finalizar): 0
Apellido y Nombre: Quiroga Juan Marcos   DNI: 44431847
N° de registro: E010-50    Nota final: 7.5
Apellido y Nombre: Barco Ezequiel   DNI: 42768933
N° de registro: E009-23    Nota final: 4.75
Apellido y Nombre: Echeverri Claudio   DNI: 47223960
N° de registro: E010-78    Nota final: 9.0
```

En esta imagen se puede observar la terminal con tres instancias de ejemplo desde la variable *arreglo* luego de ser copiado de la lista alumno.

Métodos np.max(arreglo origen) y np.min(arreglo origen):

Copia la instancia de objeto que tenga el mayor o menor valor en el arreglo, dependiendo del atributo que se quiera comparar. Para el uso de este método, es necesario sobrecargar los operadores *__le__* y *__ge__* respectivamente. En el siguiente ejemplo se puede apreciar como el programa muestra los datos de los alumnos con mayor y menor nota.

```
def maximoYMinimo(self):
    print("Los datos del alumno con la nota más alta encontrada es: \n{}".format(np.max(self.__listaAlumno)))
    print("Los datos del alumno con la nota más baja encontrada es: \n{}".format(np.min(self.__listaAlumno)))
```

```
def __le__(self, other):
    return self.__nota <= other.__nota

def __ge__(self, other):
    return self.__nota >= other.__nota
```

Sobrecarga del operador *__le__* y *__ge__* necesarios para utilizar los métodos.

```
Ingrese nombre del alumno a cargar (0 para finalizar): Juan Marcos
Ingrese apellido del alumno: Quiroga
Ingrese DNI del alumno: 44431847
Ingrese N° de registro del alumno: E010-50
Ingrese nota final del alumno: 7.50
Ingrese nombre del alumno a cargar (0 para finalizar): Esequiel
Ingrese apellido del alumno: Barco
Ingrese DNI del alumno: 42768933
Ingrese N° de registro del alumno: E009-23
Ingrese nota final del alumno: 4.75
Ingrese nombre del alumno a cargar (0 para finalizar): Claudio
Ingrese apellido del alumno: Echeverri
Ingrese DNI del alumno: 47223960
Ingrese N° de registro del alumno: E010-78
Ingrese nota final del alumno: 9
Ingrese nombre del alumno a cargar (0 para finalizar): 0
Los datos del alumno con la nota más alta encontrada es:
Apellido y Nombre: Echeverri Claudio   DNI: 47223960
N° de registro: E010-78    Nota final: 9.0
Los datos del alumno con la nota más baja encontrada es:
Apellido y Nombre: Barco Esequiel   DNI: 42768933
N° de registro: E009-23    Nota final: 4.75
```

En esta imagen se puede observar la carga de tres instancias de ejemplo en la terminal y luego vemos como el método *maximoYMinimo()* solo muestra los datos de los alumnos con mayor y menor nota respectivamente.

Conclusiones:

La utilización de arreglos NumPy es muy importante a la hora de trabajar con grandes cantidades de datos, ya que esta estructura no sólo nos permite almacenar estos datos, sino también poder trabajar con ellos de una manera más simple, eficiente y, en comparación a las listas de Python, de manera más rápida. Los métodos mencionados son los más comunes para poder trabajar con instancias de clases, pero también pueden ser aplicables a otros tipos de datos (enteros, cadenas, flotantes, etc.) tanto estos métodos como muchos otros que forman parte de la librería NumPy.

Bibliografía:

https://numpy.org/doc/stable/user/quickstart.html

https://aprendeconalf.es/docencia/python/manual/numpy/

https://www.freecodecamp.org/espanol/news/la-guia-definitiva-del-paquete-numpy-para-computacion-cientifica-en-python/