Título

Comparativa entre Listas de Python y Arreglos de NumPy aplicada a objetos definidos por el usuario

Nombre y Apellido Autor/es
Enzo Atencio

Departamento de Informática /Facultad de Ciencias Exactas Físicas y Naturales / Universidad Nacional de San Juan

Av. Ignacio de la Roza 590 (O), Complejo Universitario "Islas Malvinas", Rivadavia, San Juan, Teléfonos: 4260353, 4260355 Sitio Web: http://www.exactas.unsj.edu.ar
e-mail: autor1@correo.com, autor2@correo.com

RESUMEN

La presente investigación analiza el uso de listas en Python y arreglos en NumPy como estructuras para almacenar objetos definidos por el usuario. A través de la creación y manipulación de instancias de una clase personalizada, se estudian las principales diferencias en cuanto a eficiencia, flexibilidad y consumo de memoria. Los resultados obtenidos permiten determinar en qué contextos resulta más conveniente utilizar una u otra estructura.

Palabras clave: Listas Python, Arreglos NumPy, Estructuras de datos, Programación orientada a objetos, Comparativa.

CONTEXTO

La investigación se desarrolla en el marco de la asignatura de Programación Python, correspondiente al Departamento de Informática de la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de San Juan (UNSJ). El trabajo tiene como objetivo analizar comparativamente estructuras de datos utilizadas en entornos de programación científica y académica. La experiencia práctica se realiza utilizando herramientas como Python y su biblioteca NumPy, enfocándose en la aplicación de clases creadas por el usuario.

1. INTRODUCCIÓN

Las estructuras de datos son componentes fundamentales en la programación, ya que permiten organizar y gestionar información de manera eficiente. En Python, las listas representan una estructura dinámica y flexible capaz de almacenar elementos heterogéneos, mientras que los arreglos de la biblioteca NumPy ofrecen una alternativa más eficiente para el tratamiento de grandes volúmenes de datos homogéneos. El trabajo analiza ambos enfoques, particularmente en el contexto de clases definidas por el usuario, evaluando sus características, ventajas y limitaciones.

2. DESARROLLO

Teoría: Listas de Python y Arreglos de NumPy

Las **listas en Python** son estructuras de datos dinámicas que permiten almacenar elementos de distintos tipos, como enteros, cadenas, objetos y otras listas. Su flexibilidad y facilidad de uso las convierten en una herramienta muy utilizada para resolver una amplia variedad de problemas de programación. Sin embargo,

esta flexibilidad puede implicar mayores costos en términos de eficiencia de memoria y velocidad de procesamiento, especialmente cuando se manejan grandes volúmenes de datos.

Por otro lado, los **arreglos de NumPy** están diseñados para almacenar elementos homogéneos (del mismo tipo) en una estructura contigua en memoria, optimizando así el uso de recursos y permitiendo operaciones vectorizadas que son mucho más rápidas que las realizadas sobre listas. Aunque los arreglos son menos flexibles que las listas en cuanto al tipo de datos que pueden contener, su eficiencia los hace ideales para aplicaciones científicas, procesamiento de grandes volúmenes de información y cálculos numéricos avanzados.

Ambas estructuras presentan ventajas y desventajas que deben ser consideradas al momento de diseñar una solución informática

```
class Producto:
    def __init__(self, nombre, precio):
        self.__nombre = nombre
        self.__precio = precio
    def get_nombre(self):
        return self.__nombre
    def get_precio(self):
        return self.__precio
# Crear una lista de productos
productos = []
# Agregar objetos a la lista
productos.append(Producto("Mouse", 4500))
productos.append(Producto("Teclado", 7000))
# Mostrar productos
for p in productos:
    print(f"Producto: {p.get_nombre()}, Precio: ${p.get_precio()}")
```

```
import numpy as np
```

```
class Producto:
    def __init__(self, nombre, precio):
        self.__nombre = nombre
        self.__precio = precio
    def get_nombre(self):
        return self.__nombre
    def get_precio(self):
        return self.__precio
# Crear un arreglo de objetos Producto
productos_array = np.array([
    Producto("Mouse", 4500),
    Producto("Teclado", 7000)
])
# Mostrar productos
for p in productos_array:
    print(f"Producto: {p.get_nombre()}, Precio: ${p.get_precio()}")
```

TABLA COMPARATIVA

Característica	Listas de Python	Arreglos de NumPy
Tipo de datos	Heterogéneo (diferentes tipos)	Homogéneo (mismo tipo)
Eficiencia de memoria	Menor eficiencia	Mayor eficiencia
Velocidad de acceso	Más lenta para grandes volúmenes	Más rápida mediante acceso contiguo
Flexibilidad	Alta (cualquier tipo de objeto)	Baja (mismo tipo de objeto)
Operaciones disponibles	Básicas (agregar, eliminar, buscar)	Avanzadas (operaciones matemáticas vectorizadas)
Facilidad de uso	Muy fácil, sintaxis simple	Requiere importar y conocer NumPy
Aplicaciones ideales	Programación general, objetos complejos	Cálculos numéricos, procesamiento científico

3. CONCLUSIONES

A partir del análisis realizado, se concluye que tanto las listas de Python como los arreglos de NumPy presentan ventajas particulares dependiendo del contexto de uso. Las listas ofrecen gran flexibilidad al permitir

almacenar elementos de distintos tipos, lo cual resulta ideal para proyectos que manejan estructuras heterogéneas, como clases personalizadas. Por su parte, los arreglos de NumPy brindan un mejor rendimiento en memoria y velocidad, siendo más adecuados para operaciones matemáticas sobre grandes volúmenes de datos homogéneos. La elección de una u otra estructura debe considerar las necesidades específicas de cada aplicación.

4. BIBLIOGRAFÍA

Python Software Foundation. (2024). *Python 3.12.3 Documentation*. https://docs.python.org/3/

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). *Array programming with NumPy*. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.