Métodos más destacados de las listas en Python

Universidad Nacional de San Juan

Facultad de Ciencias Exactas, Físicas y Naturales

Departamento de Informática

Programación Orientada a Objetos

Integrantes:

- Sasso Joaquín (E010-272)
- Spesia Tomas (E010-130)

1.Introduccion

Python es un lenguaje interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipado dinámico, tipos de datos de muy alto nivel y clases. Python combina un poder destacado con una sintaxis muy clara. Tiene interfaces a muchas llamadas de sistema y bibliotecas, así como a varios sistemas de ventana, y es extensible en C o C++. También es usable como un lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Por último, Python es portable: corre en muchas variantes de Unix, en Mac y en Windows 2000 y posteriores. [1]

La lista es un tipo de colección ordenada. Sería equivalente a lo que en otros lenguajes se conoce por arrays, o vectores. Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos, … y también listas. [2]

NumPy es el paquete fundamental de Python para la computación científica. Es una biblioteca que proporciona un objeto de arreglo multidimensional, varios objetos derivados (como arreglos enmascarados y matrices) y una variedad de rutinas para operaciones rápidas en arreglos, incluyendo operaciones matemáticas, lógicas, manipulación de forma, ordenamiento, selección, entrada/salida, transformadas de Fourier discretas, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más. [3]

Palabras clave: Python, arreglos y numpy

2.Desarrollo

Se utilizará la clase perro para el desarrollo de todos los ejemplos. (Se utilizaran cuadros para diferenciar los distintos modulos)

2.1 Ordenamiento

Funcion Sort:

L.sort(cmp=None, key=None, reverse=False)

Ordena la lista. Si se especifica *cmp*, este debe ser una función que tome

como parámetro dos valores x e y de la lista y devuelva -1 si x es menor

que y, 0 si son iguales y 1 si x es mayor que y.

El parámetro *reverse* es un booleano que indica si se debe ordenar

la lista de forma inversa.

Por último, si se especifica, el parámetro *key* debe ser una función que tome un elemento de la lista y devuelva una clave a utilizar a la hora de comparar, en lugar del elemento en si.

```
import numpy as np #Se importa la libreria numpy con el alias np
from clasePerro import perro as p
def main():
  perros = np.empty(4,  dtype= p)
  #Se inicializan los perro con los valores siguientes
  perros[0] = p("Coda",3)
  perros[1] = p("Firu",5)
  perros[2] = p("Tobi",7)
  perros[3] = p("Pelusa",1)
  perros.sort() #Se utiliza la funcion sort para ordenar el arreglo de perros
  #Se itera por todo el arreglo de perros para mostrar sus datos
  for perro in perros:
    print(f"Nombre: {perro.getNombre()}, edad: {perro.getEdad()}")
if __name__ == "__main__":
  main()
```

```
class perro: #Se crea la clase perro
  #Se definen sus atributos
  __nombre: str
  __edad: int
  def __init__(self, nombre, edad): #Método para inicializar el objeto
    self.__nombre = nombre
    self.__edad = edad
  def getEdad(self): #Devuelve la edad del perro
    return self.__edad
  def getNombre(self): #Devuelve el nombre del perro
    return self.__nombre
  def __lt__(self, otro):
    #Sobrecargamos el operador < comparando la edad de los perros
    return self.__edad < otro.getEdad()
```

El mensaje de salida que se vería en terminal debería tener los perros según su edad.

2.2 Selección condicional de elementos de un arreglo

Los arreglos NumPy admiten una función llamada conditional selection, que le permite generar un nuevo arreglo de valores booleanos que indican si cada elemento dentro del arreglo satisface una declaración if particular.

También puedes generar un nuevo arreglo de valores que satisfagan esta condición pasando la condición entre corchetes. [4]

```
import numpy as np

#Se importa la libreria numpy con el alias np

from clasePerro import perro as p

def main():

  perros = np.empty(4,  dtype= p)

  #Se crea el arreglo y se inicializan los perro con los valores siguientes

  perros[0] = p("Coda",3)

  perros[1] = p("Firu",5)

  perros[2] = p("Tobi",7)

  perros[3] = p("Pelusa",1)

  perrosViejos = perros[perros > 4] #Se genera un subarreglo con los perros cuya edad es mayor a 4

  for perro in perrosViejos: #Se itera por todo el arreglo de perros para mostrar sus datos

    print(f"Nombre: {perro.getNombre()}, edad: {perro.getEdad()}")

if __name__ == "__main__":

  main()
```

Además, se debe sobrecargar el operador mayor que en la clase perro

```
def __gt__(self, numero):

    return self.__edad > numero
```

3.Conclusiones

NumPy es una biblioteca esencial para trabajar con datos en Python. Facilita el manejo de arreglos multidimensionales y ofrece herramientas para realizar operaciones matemáticas y estadísticas de manera eficiente.

La función conditional select de NumPy permite seleccionar elementos de un arreglo según una condición específica, lo que simplifica el proceso de filtrado y transformación de datos basado en condiciones.

La función sort permite ordenar arreglos de manera rápida y confiable, lo que facilita la visualización y el análisis de datos. En resumen, NumPy y sus funciones, como conditional selecty funciones incluidas en Python como sort, son herramientas poderosas para el análisis y procesamiento de datos en Python.

4. Bibliografía

[1] Preguntas frecuentes generales sobre Python. (s.f). https://docs.python.org/es/3.10/faq/general.html.

[2] Raúl González Duque. (s.f). Python para todos. https://repositorio.uci.cu/bitstream/123456789/10206/1/Python_para_todos.pdf

[3] NumPy. (2023). https://wiki.python.org/moin/NumPy

[4] Fernando Cardellino. (2021). La guía definitiva del paquete NumPy para computación científica en Python. https://www.freecodecamp.org/espanol/news/la-guia-definitiva-del-paquete-numpy-para-computacion-cientifica-en-python/