ESTRUCTURA DE DATOS FUNDAMENTALES Y ALGORITMOS
GUÍA DE LABORATORIO 03 – 04
Recursividad – Arreglos y Matrices

Asignatura	Datos del alumno	Fecha y Firma
Algoritmos y solución de problemas	Apellidos:	
	Nombre:	

Instrucciones:

Desarrollar las actividades que indica el docente en base a la guía de trabajo que se presenta.

1. Objetivos:

- Escribir algoritmos y codificar haciendo uso de recursividad, arreglos y matrices.

2. Equipos, herramientas o materiales

- Computador, Software: Python, Algoritmos

3. Fundamento teórico

3.1. Conceptos Clave

3.1.1. RECURSIVIDAD

La recursividad es un concepto matemático y computacional que se refiere a la capacidad de una función o algoritmo para llamarse a sí mismo. Esto puede parecer contradictorio, pero en realidad es una herramienta muy poderosa que se puede utilizar para resolver una amplia gama de problemas.

Para entender la recursividad en Python, es importante comprender los conceptos básicos de la recursión. Hay dos partes principales de una función recursiva:

- **El caso base:** el caso base es una condición que le indica a la función que debe dejar de llamarse a sí misma y devolver un valor.
- **El caso recursivo**: el caso recursivo es la condición que le indica a la función que debe llamarse a sí misma con diferentes argumentos.

Ejemplo 01:

Aquí hay un ejemplo de una función recursiva en Python que calcula la factorial de un número:

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))
```

Esta función funciona de la siguiente manera:

El caso base es n == 0. En este caso, la función simplemente devuelve 1.

El caso recursivo es n > 0. En este caso, la función calcula la factorial de n - 1 y lo multiplica por n.

Ejemplo 03:

El cálculo del Fibonacci: se puede utilizar la recursividad para calcular los números de Fibonacci. Los números de Fibonacci son una secuencia de números en la que cada número es la suma de los dos números anteriores. La siguiente función calcula los primeros n números de Fibonacci:

```python
def fibonacci(n):
    if n == 0 or n == 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)


if __name__ == "__main__":
    altura = int(input("Introduzca la cantidad de números de Fibonacci que desea imprimir: "))
    for i in range(altura):
        print(fibonacci(i))
```

Esta implementación funciona de la siguiente manera:

En el caso base, si n es igual a 0 o 1, la función simplemente devuelve n.
En el caso recursivo, la función hace lo siguiente:

Calcula el siguiente número de Fibonacci.

Devuelve la suma de los dos últimos números de Fibonacci.

3.1.2. ARREGLOS y MATRICES

Para trabajar con matrices en Python 3.11, tienes dos opciones principales:

Usar el módulo numpy:

Instala el módulo numpy si aún no lo tienes: pip install numpy
Importa el módulo numpy en tu código:

```python
import numpy as np
```

Crea las matrices usando la función np.array():

```python
matriz_1 = np.array([[1, 2, 3], [4, 5, 6]])
matriz_2 = np.array([[7, 8, 9], [10, 11, 12]])
```

NumPy arrays: son una estructura de datos de Python que se utilizan para almacenar datos numéricos. Las matrices de NumPy son matrices multidimensionales, lo que significa que pueden tener más de una dimensión.

Creación de matrices

Las matrices de NumPy se pueden crear de varias maneras. Una forma es usar la función np.array(). La función np.array() toma una secuencia de datos como entrada y crea una matriz de NumPy de la misma forma.

Por ejemplo, el siguiente código crea una matriz de NumPy de números enteros:

```python
import numpy as np

matriz = np.array([1, 2, 3])
```

Este código crea una matriz de NumPy de una dimensión con tres elementos.

Acceso a elementos de matrices

Los elementos de las matrices de NumPy se pueden acceder mediante su índice. El índice de una matriz de NumPy es un número que identifica a un elemento específico de la matriz.

Por ejemplo, el siguiente código imprime el primer elemento de la matriz matriz:

```python
print(matriz[0])
```

Operaciones con matrices

Las matrices de NumPy se pueden manipular mediante operaciones matemáticas. Las operaciones matemáticas comunes que se pueden realizar con matrices incluyen suma, resta, multiplicación y división.

Por ejemplo, el siguiente código suma dos matrices:

```python
matriz_1 = np.array([[1, 2, 3], [4, 5, 6]])
matriz_2 = np.array([[7, 8, 9], [10, 11, 12]])

matriz_suma = matriz_1 + matriz_2

print(matriz_suma)
```

Funciones de NumPy para matrices

NumPy proporciona muchas funciones útiles para el análisis y la manipulación de matrices. Algunas de las funciones de NumPy más comunes para matrices incluyen:

np.shape(): Devuelve la forma de una matriz.
np.reshape(): Cambia la forma de una matriz.
np.transpose(): Transpone una matriz.
np.dot(): Multiplica dos matrices.
np.sum(): Suma los elementos de una matriz.
np.mean(): Calcula la media de los elementos de una matriz.
np.std(): Calcula la desviación estándar de los elementos de una matriz.

4. Desarrollo y Actividades

- **Ejercicio parte 01:**

Recursividad:

1) Ejercicio 1: Escribe una función recursiva que imprima los números pares del 1 al 100.
2) Ejercicio 2: Escribe una función recursiva que imprima la suma de los números del 1 al n.
3) Ejercicio 3: Escribe una función recursiva que imprima la pirámide de números del 1 al n.
4) Ejercicio 4: Escribe una función recursiva que imprima la pirámide de números invertidos del 1 al n.
5) Ejercicio 2: Escribe una función recursiva que imprima la tabla de multiplicar del n.

Arreglos y Matrices:

6) Crea una matriz de números reales.
7) Crea una matriz de números complejos.
8) Crea una matriz de matrices.
9) Accede al elemento central de una matriz.
10) Suma dos matrices de diferentes tamaños.
11) Multiplica una matriz por un número.
12) Calcula la media de los elementos de una matriz.

- **Ejercicio parte 01:**

Ejercicio 1:

Crea una matriz de números aleatorios de tamaño 100x100.

Ejercicio 2:

Calcula la media, la mediana y la desviación estándar de los elementos de una matriz.

Ejercicio 3:

Escribe una función que encuentre el elemento máximo de una matriz.

Ejercicio 4:

Escribe una función que encuentre la submatriz de mayor suma de una matriz.

Ejercicio 5:

Escribe una función que encuentre la matriz de covarianza de dos matrices.