FUNDACIÓN FULGOR

CURSO DE DISEÑO DIGITAL AVANZADO

PROGRAMA DE COMUNICACIONES

Trabajo Práctico N°1

OPERACIONES CON VARIABLES DE PYTHON

ALUMNO : Martinez, Luciano Micael

DOCENTES : Dr. Ing. Pola, Luis Ariel
Ing. Leguizamón, Santiago

CÓRDOBA, ARGENTINA

04 de Junio de 2024

CONTENIDO

1. Introducción

El presente informe tiene como objetivo explicar cómo se desarrolló el código del Práctico 01 de operaciones con variables en Python. El programa se ejecuta mediante el archivo `main.py` y puede ejecutarse desde la consola utilizando el comando `python3 main.py`.

2. Enunciado

El objetivo es realizar operaciones con distintos tipos de variables interactuando con el usuario por medio de la función `input()`.

2.1. Ejercicio 1

Empleando los conceptos aprendidos de manejo de variables, elaborar una calculadora la cual muestra el siguiente panel de opciones cuando se ejecuta el programa:

1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Iterativo
    a) Sumar
    b) Restar
    c) Multiplicar

El usuario debe ingresar una de las opciones que muestra el panel anterior.

- Para los casos 1-4 se deben ingresar dos números y se realiza la operación seleccionada.
- Para el caso 5 el primer número ingresado representa el paso y el segundo el número de veces que se repite el paso según el tipo de operaciones (Ej. Step:5, Iter:2, Sel. 5.a=10, 5.b=-10, 5.c=25).

2.2. Ejercicio 2

Empleando el ejercicio anterior, consultar al usuario si desea continuar realizando operaciones o desea salir.

Además, incluir la opción *producto punto* al panel anterior tal como se observa a continuación:

1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Iterativo
    a) Sumar
    b) Restar
    c) Multiplicar
6. Producto punto (*dot*)

En este caso se deberá solicitar el ingreso de dos vectores/matrices, de los cuales es necesario corroborar que tengan los mismos tamaños/dimensiones para poder realizar el producto punto. En caso contrario se debe volver a ingresar los vectores/-matrices.

3. Actividad Práctica

La ejecución de ambos ejercicios se realiza mediante el archivo `main.py`, el cual contiene el código que se muestra a continuación. En este, se solicita al usuario que ingrese el menú al que desea acceder, y se realiza una evaluación de la entrada, con el fin de asegurar un dato válido.

```
from p1Calculator import menu

flag = 0

print("Ingrese el menu' a utilizar.")
print("1. Menu' del ejercicio 1")
print("2. Menu' del ejercicio 2")

while True:
    flag = input("Opcio'n: ")
    if flag in ['1', '2']:
        break
    print("Ingrese una opcio'n va'lida")

menu(flag)
```

Listing 1: Código del menú de selección del ejercicio.

En el código se importa la función `menu` del archivo `p1Calculator.py`, la cual se explicara mas adelante, y se ejecuta enviando la opción elegida por el usuario.

3.1. Ejercicio 1

Para realizar las operaciones matemáticas especificadas en el ejercicio de la Sección 2.1, se decidió utilizar una clase llamada `Calculator`, la cual realiza las distintas operaciones mediante métodos propios de la clase.

En la siguiente sección de código se muestran los métodos constructor `__init__` y `update_values`, utilizados para definir los valores de los atributos de la clase.

```
class Calculator:
    # Constructor de la clase:
    def __init__(self, Num1 = 1.0, Num2 = 1.0, Sel = ' '):
        self.a = Num1
        self.b = Num2
        self.op = Sel
        self.result = None

    def update_values(self, Num1 = 1.0, Num2 = 1.0, Sel = ' '):
        self.a = Num1
        self.b = Num2
        self.op = Sel
        self.result = None
```

Listing 2: Código de la clase Calculator

Se definen los métodos para las cuatro funciones básicas de la calculadora por medio de los métodos `add`, `sub`, `mul` y `div`.

```
    def add(self):
        self.result = self.a + self.b

    def sub(self):
        self.result = self.a - self.b

    def mul(self):
        self.result = self.a * self.b

    def div(self):
        if self.b == 0:
            print("No se puede dividir por cero")
            self.result = 'Math. Error'
        else:
            self.result = self.a / self.b
```

Listing 3: Código de metodos basicos de la calculadora

El siguiente y último método de la clase corresponde a la función iterativa. Este requiere la definición del tipo de operación a realizar, y en función de eso, utiliza los atributos **a** y **b** como `Iter` y `Step` respectivamente.

```
    def iter(self):
        if self.op == 'c':
            self.result = 1
        else:
```

```
5               self.result = 0
6
7           for i in range(self.b):
8               if  self.op == 'a':
9                   self.result += self.a
10
11              elif self.op == 'b':
12                  self.result -= self.a
13
14              elif self.op == 'c':
15                  self.result *= self.a
```

Listing 4: Código del metodo iterativo de la clase

3.1.1. Menú principal

El menú principal del código ejecuta tanto las funciones del ejercicio 1 como del ejercicio 2, mediante la selección del usuario realizada por la bandera `flag`, la cual se evalúa en el código.

Como se ve en las líneas 21 y 22, si el usuario ingresa la opción del menú 2, se muestran las opciones de producto punto y salir del programa. Además, se realiza una evaluación de la opción ingresada mediante un bucle `while`.

```
1  from p1Class import Calculator
2  from p1Matrix import MatrixDot
3
4  #Se usa la variable flag para elegir entre:
5  #el menu del ejercicio 1 y el menu del ejercicio 2
6  def menu(flag):
7      #Se crea un objeto "obj" de tipo Calculator
8      obj = Calculator()
9
10     print("Bienvenido a la aplicacio'n de calculadora")
11     while True:
12         #Opciones del Menu 1:
13         print("Ingrese el nu'mero de la operacio'n a realizar:")
14         print("1. Sumar")
15         print("2. Restar")
16         print("3. Multiplicar")
17         print("4. Dividir")
18         print("5. Iterativo")
19
20         #Opciones adicionales del Menu 2:
21         print("6. Producto punto entre matrices")   if flag == '2'
               else None
22         print("7. Salir del programa")              if flag == '2'
               else None
23
24         #Validacio'n de la opcio'n ingresada
25         while True:
```

```
            op = input("Opcio'n: ")
            if op in ['1', '2', '3', '4', '5'] or (flag == '2' and
                op in ['6', '7']):
                break
            else:
                print("Error, ingrese una opcio'n va'lida.")
```

Listing 5: Código del menú general.

Luego, dentro del mismo bucle `while`, si se ingresa una operación simple del menú, se solicita al usuario que ingrese los números con los que desea operar. Mediante la estructura de manejo de excepciones, se evalúa si el ingreso es correcto, en este caso, correspondiente a un número real.

Una vez validados los números ingresados por el usuario, se actualizan los valores de los atributos del objeto `obj` construido anteriormente, y se opera según la elección `op` realizada anteriormente por el usuario.

```
if op in ['1', '2', '3', '4']:
    # Validacio'n de ingreso de valores
    while True:
        try:
            Num1 = float(input("Ingrese el primer nu'mero: "))
            Num2 = float(input("Ingrese el segundo nu'mero: "))
            break
        except ValueError:
            print("Error, ingrese nu'meros reales.")

    #Crea un objeto Calculator "C"
    obj.update_values(Num1, Num2)

    # Realiza las operaciones y muestra el resultado
    if op == '1':
        obj.add()
        print(f"Resultado: {Num1} + {Num2} = {obj.result}")
    elif op == '2':
        obj.sub()
        print(f"Resultado: {Num1} - {Num2} = {obj.result}")
    elif op == '3':
        obj.mul()
        print(f"Resultado: {Num1} * {Num2} = {obj.result}")
    elif op == '4':
        obj.div()
        print(f"Resultado: {Num1} / {Num2} = {obj.result}")
```

Listing 6: Código de operaciónes basicas del menú general.

Para la opción iterativa, se presenta un sub-menú que pregunta al usuario qué operación iterativa desea realizar, y se realiza una validación de la opción ingresada. Luego, mediante manejo de excepciones, se solicita al usuario que ingrese el `Step` y el número de iteraciones **Iter** que desea realizar.

Una vez realizados los ingresos de los valores mencionados, se actualizan los valores de los atributos del objeto `obj` y se muestra en pantalla el resultado del método

`iter()`.

```
# Operacio'n iterativa
elif op == '5':
    print("Seleccione una operacio'n iterativa a realizar:")
    print("a) Sumar")
    print("b) Restar")
    print("c) Multiplicacio'n")

    # Evalu'a el ingreso correcto de input()
    while True:
        op_iter = input("Ingrese el tipo de operacio'n: ")
        if op_iter in ['a', 'b', 'c']:
            break
        else:
            print("Opcio'n no va'lida.")

    # Ingreso de valores para la operacio'n iterativa
    while True:
        try:
            step = float(input("Ingrese el valor: "))
            iterations = int(input("Ingrese el nu'mero de
               iteraciones: "))
            break
        except ValueError:
            print("Error: Ingrese nu'meros va'lidos.")

    # Cambia el valor de los atributos del objeto ya creado
    obj.update_values(step, iterations, op_iter)
    # Realiza la operacio'n iterativa
    obj.iter()
    # Imprime en pantalla
    print(f"Resultado de operacio'n iterativa: {obj.result}")
```

Listing 7: Código de operaciónes iterativas del menú general.

Finalmente, si el menú seleccionado por el usuario es el número 1, se sale del bucle principal mediante un `break`, como se muestra a continuación.

```
#Si se selecciono el menu 1:
if flag == '1':
    break #Sale del bucle
```

Listing 8: Código de cierre del bucle del menú principal.

3.2. Ejercicio 2

En el caso en que el usuario seleccione el menú 2, se ejecutan las siguientes líneas correspondientes al ejercicio 2. Si se ingresa la opción 6 del menú principal, se llama a la función `MatrixDot()` del archivo `p1Matrix.py`, la cual realiza la operación del producto punto. Además, si se ingresa la opción 7, se sale del bucle mediante un `break`.

```
#Para el menu del ejercicio 2:
if flag == '2':
    #Para realizar operacio'n de producto punto
    if   op == '6':
        MatrixDot()

    #Para salir del menu' o continuar operando
    elif op == '7':
        break #Sale del bucle
```

Listing 9: Código de opciónes del ejercicio 2 del menú general.

3.2.1. Funciones para el producto punto

Para el ejercicio 2, se presenta un segundo menú que ofrece al usuario la opción de operar sobre vectores o matrices. Esta función solicita al usuario su preferencia, evalúa el dato ingresado y devuelve la selección realizada.

```
import numpy as np

#Sub-menu para operacio'n con matrices:
def SubMenu():
    print("* Elija una opcio'n.")
    print("Desea realizar el producto punto entre: ")
    print("1) Vectores.")
    print("2) Matrices.")

    #Bucle infinito
    while True:
        Sel = input("Opcio'n: ")

        #Evalu'a el ingreso del usuario
        if Sel in ['1', '2']:
            return Sel  #Sale del bucle y de la funcio'n

        #Si no es valido
        else:           #Repite el bucle
            print("Error, elija una opcio'n valida.")
```

Listing 10: Código de menú para operación de producto punto.

Se define una función `InputMatrix()` para que el usuario ingrese el contenido de la matriz o vector. Por medio del parámetro `Sel`, se selecciona entre definir una matriz o vector, y el parámetro `n` define el tamaño del mismo.

En el caso del vector, se define un vector relleno de ceros, y se ingresa el contenido mediante un bucle `for`. Dentro de este bucle, se evalúa cada ingreso del usuario mediante el manejo de excepciones.

```
#Funcio'n para ingresar el contenido del vector o matriz.
def InputMatrix(n = 1, Sel = '0'):
```

```
    #Por defecto
    if Sel == '0':
        return 0

    #Para vector
    elif Sel == '1':

        Matrix = np.zeros(n)

        #Itera en i de 0 a n, donde n es el taman~o del vector
        for i in range(n):

            #Evalu'a si las entradas del usuario
            while True:
                try:
                    Matrix[i] = float(input(f"Elemento {i+1}: "))
                    break   #Sale del while
                except ValueError:
                    print("Error, ingrese un nu'mero real.")
```

Listing 11: Código para el ingreso del contenido del vector.

En el caso de que el usuario desee operar con matrices, se define una matriz rellena de ceros, para luego ingresar cada elemento mediante un bucle similar al anterior, el cual recorre cada elemento `i, j` de la misma.

```
   #Para matriz
    elif Sel == '2':

        Matrix = np.zeros((n,n))

        #Itera en "i" de "0" a "n"
        for i in range(n):
            #Itera en "j" de "0" a "n"
            for j in range(n):

                #Evalu'a si las entradas del usuario
                while True:
                    try:
                        Matrix[i,j] = float(input(f"Elemento {i
                            +1}, {j+1}: "))
                        break   #Sale del while
                    except ValueError:
                        print("Error, ingrese un nu'mero real.")

    #Devuelve la matriz
    return Matrix
```

Listing 12: Código para el ingreso del contenido del matriz.

Finalmente, la función `MatrixDot()` llama a `SubMenu()` y mediante un bucle, solicita al usuario que ingrese el tamaño del vector o matriz. Para evitar errores, se aplica el

mismo tamaño `n` para ambos elementos a operar, y se definen los arrays rellenos con ceros mediante la función `np.zeros()` de la librería `numpy`.

Luego, se llama a la función `InputMatrix()` para realizar el ingreso de ambos arrays, y mediante la función `np.dot()` se realiza el producto punto de ambas matrices y se imprime el resultado en pantalla.

```

#Funcio'n que realiza el producto punto
def MatrixDot():

    Sel = SubMenu()

    if Sel == '1':
        Array = 'el vector'
    elif Sel == '2':
        Array = 'la matriz'

    #Evalu'a el taman~o ingresado del array
    while True:
        try:
            n = int(input(f"Ingrese el taman~o de {Array}: "))
            break
        except ValueError:
            print("Error, ingrese un nu'mero valido.")

    #Crea los arrays:
    if Sel == '1':
        Arr1 = np.zeros(n)
        Arr2 = np.zeros(n)

    elif Sel == '2':
        Arr1 = np.zeros((n,n))
        Arr2 = np.zeros((n,n))


    #Ingresa el array 1:
    print(f"Ingrese {Array} 1")
    Arr1 = InputMatrix(n, Sel)
    print(f"{Array} 2 ingresado es: \n", Arr1)

    #Ingresa el array 1:
    print(f"Ingrese {Array} 2")
    Arr2 = InputMatrix(n, Sel)
    print(f"{Array} 2 ingresado es: \n", Arr2)

    #Realiza el producto punto y lo muestra en pantalla:
    Result = np.dot(Arr1, Arr2)
    print("El producto punto es: \n", Result)
```

Listing 13: Código de función principal del producto punto.