

Práctica 2: Thrift

Funcionamiento General

La calculadora, nuevamente estará programada con una interfaz gráfica (en la terminal), donde le iremos pasando valores en torno a la operación que hayamos elegido. Esta vez, la calculadora estará compuesta por dos servidores (uno para operaciones básicas y otro para operaciones complejas con vectores) y un cliente, los tres hechos en Python. Tendremos que abrir primero los dos servidores y después el cliente en terminales diferentes.

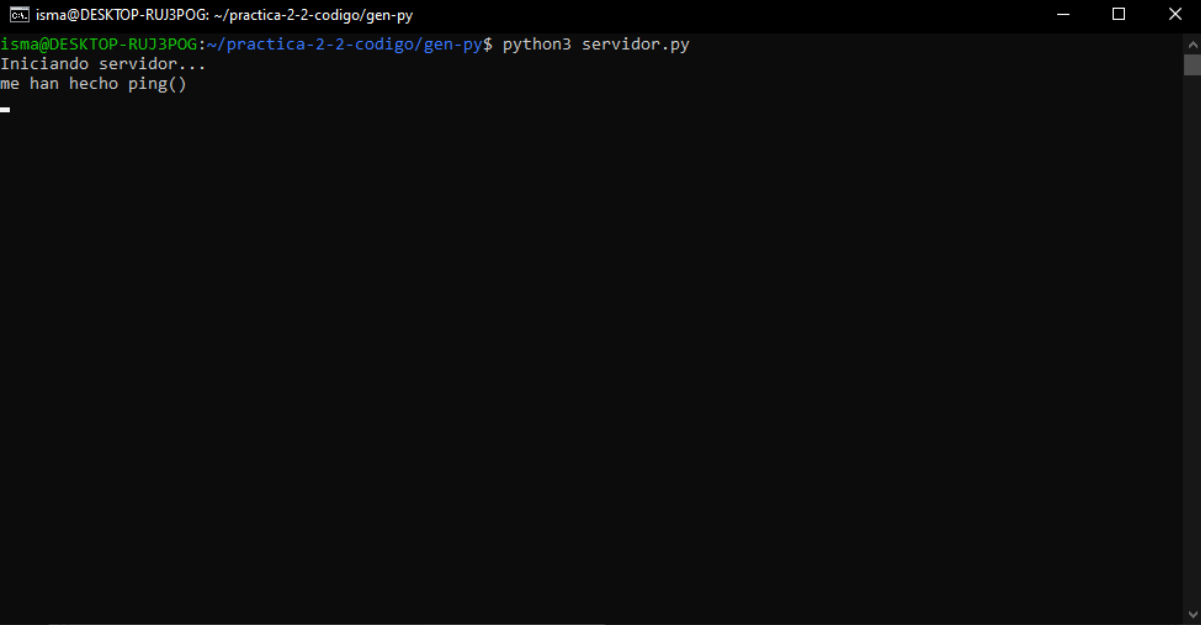
Terminal 1: `python3 ./servidor.py`

Terminal 2: `python3 ./servidor_aux.py`

Terminal 3: `python3 ./cliente.py`

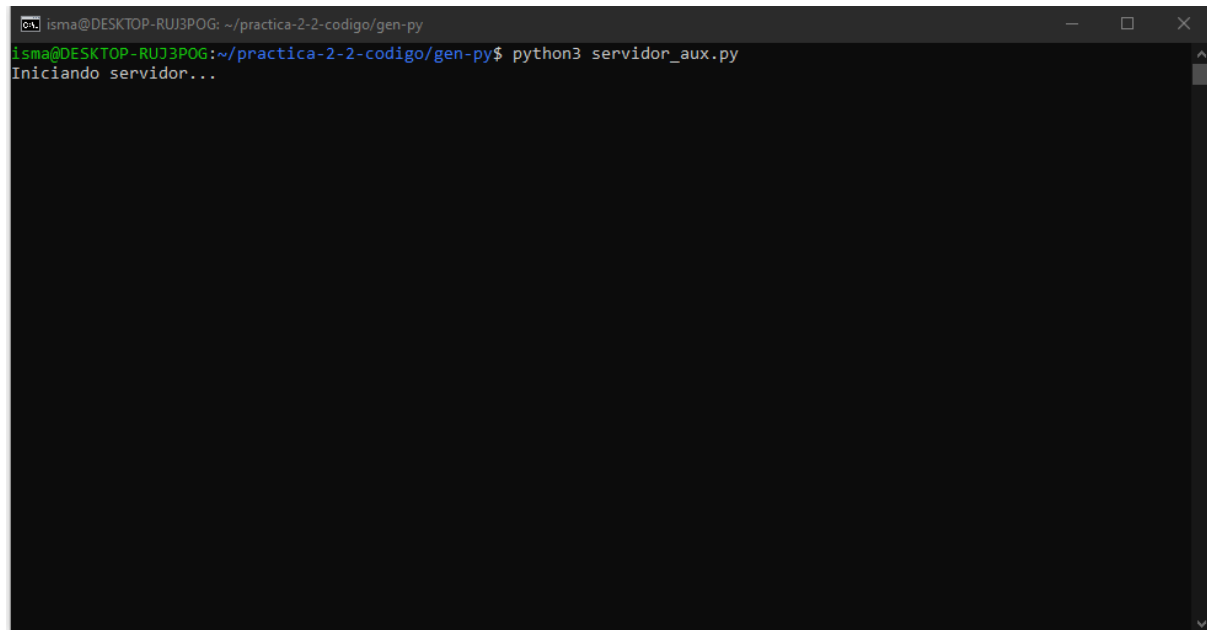
Esta vez no harán falta parámetros cuando iniciamos el cliente, caso contrario a como pasaba con RPC, donde teníamos que indicar la dirección de la red donde se encontraba el servidor.

Terminal 1:



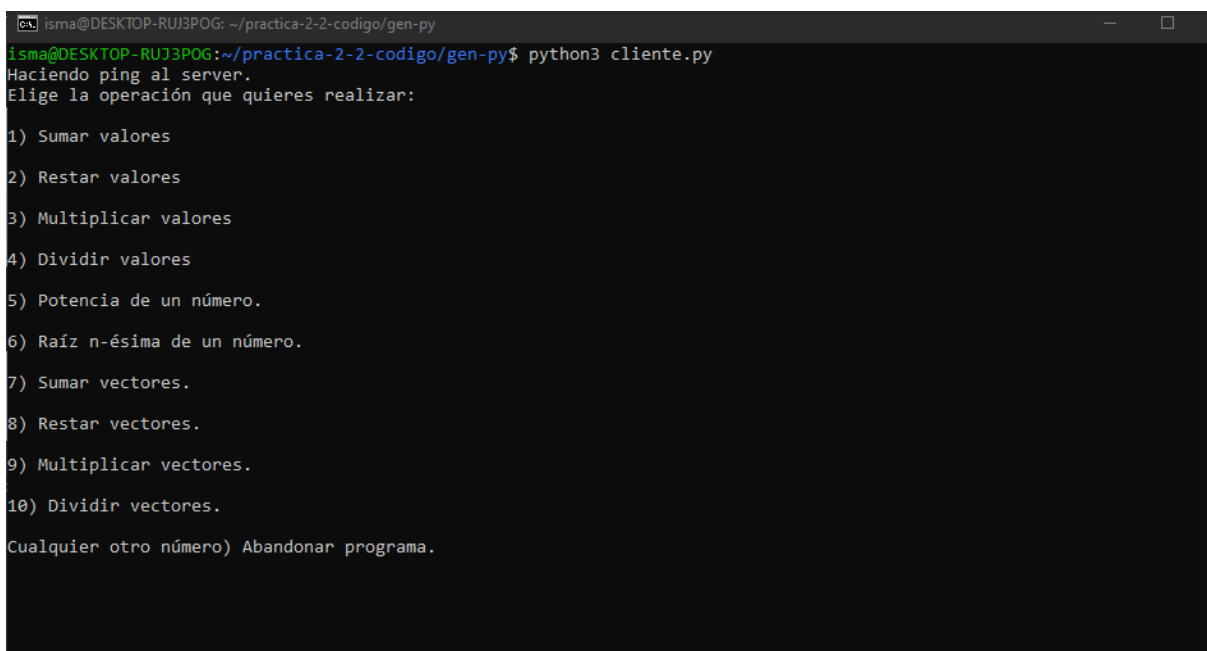
```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor.py
Iniciando servidor...
me han hecho ping()
```

Terminal 2:

A screenshot of a terminal window with a dark background. The title bar shows the user 'isma' and the path '~/practica-2-2-codigo/gen-py'. The terminal shows the command 'python3 servidor_aux.py' being executed, followed by the output 'Iniciando servidor...'.

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor_aux.py
Iniciando servidor...
```

Terminal 3:

A screenshot of a terminal window with a dark background. The title bar shows the user 'isma' and the path '~/practica-2-2-codigo/gen-py'. The terminal shows the command 'python3 cliente.py' being executed, followed by the output 'Haciendo ping al server.' and 'Elige la operación que quieres realizar:'. A list of 10 options is displayed, followed by a prompt for any other number to abandon the program.

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 cliente.py
Haciendo ping al server.
Elige la operación que quieres realizar:

1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
```

Una vez iniciada la interfaz gráfica (Terminal 3), nos dará a elegir las opciones que se muestran, siendo un total de 10.

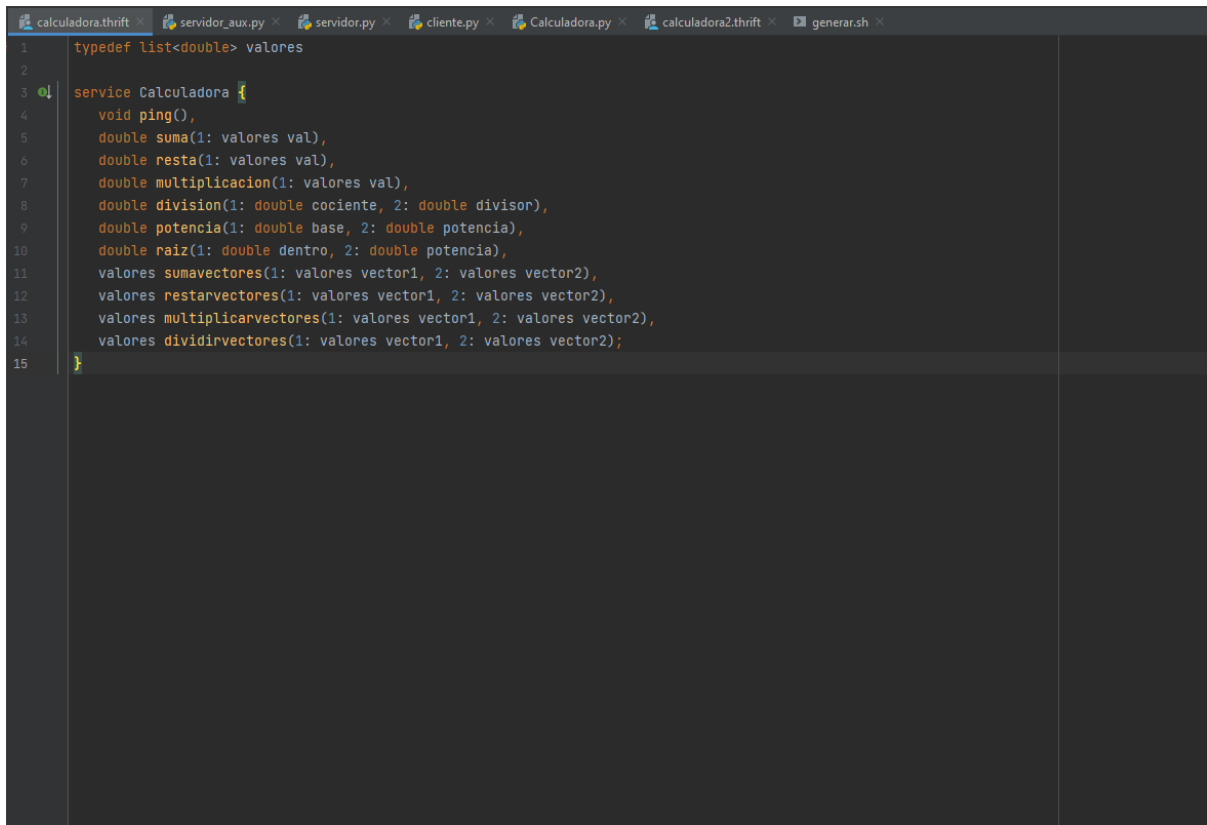
- Operaciones básicas con números reales (Suma, resta, multiplicación y división).
- Operaciones algo complejas con números reales (Potencia y raíz de un número real).
- Operaciones complejas con vectores dinámicos (Suma, resta, multiplicación y división de vectores).
 - Para la multiplicación y división, se multiplicará y dividirá cada componente de los vectores hasta alcanzar el tamaño mínimo de los vectores originales (esto último también se aplica para la suma y la resta).

Explicación del código

Existen dos ficheros .thrift, uno para cada servidor donde se indicarán las operaciones que queremos realizar en cada uno de ellos.

Nota: El cliente no se comunica directamente con el servidor de las operaciones complejas, sino que le indica al servidor básico que desea hacer una operación compleja, y este se encargará de establecer la posterior conexión con el servidor complejo.

Calculadora.thrift:



```
1  typedef list<double> valores
2
3  service Calculadora {
4      void ping(),
5      double suma(1: valores val),
6      double resta(1: valores val),
7      double multiplicacion(1: valores val),
8      double division(1: double cociente, 2: double divisor),
9      double potencia(1: double base, 2: double potencia),
10     double raiz(1: double dentro, 2: double potencia),
11     valores sumavectores(1: valores vector1, 2: valores vector2),
12     valores restarvectores(1: valores vector1, 2: valores vector2),
13     valores multiplicarvectores(1: valores vector1, 2: valores vector2),
14     valores dividirvectores(1: valores vector1, 2: valores vector2);
15 }
```

Se define una estructura “valores” para simular los vectores que se van a usar para las operaciones complejas, así como un vector con todos los números que se van a utilizar tanto para la suma, como para la resta y la multiplicación. También se usa para el otro servidor.

Para la división, podrá realizarse otra operación auxiliar (suma, resta, multiplicación o introducir el valor directamente) tanto para el cociente como para el divisor. Este último nunca podrá ser 0 y estas operaciones auxiliares realizarán también una llamada al servidor.

Llamar a las funciones “sumarvectores”, “restarvectores”, “multiplicarvectores” y “dividirvectores” implica hacer una conexión con el servidor auxiliar. En el último caso, ninguno de los valores introducidos en el segundo vector podrá ser 0.

Calculadora2.thrift:

```
calculadora.thrift x servidor_aux.py x servidor.py x cliente.py x Calculadora.py x calculadora2.thrift x generar.sh x
1 typedef list<double> valores
2
3 service Calculadora2 {
4     void ping(),
5     valores operacionvectores(1: i8 op, 2: valores vector1, 3: valores vector2);
6 }
```

El argumento “op” indicará el tipo de operación que se va a realizar sobre los vectores (este valor se lo proporcionará el servidor principal).

El fichero del servidor realizará tanto las operaciones básicas con números reales, como operaciones algo complicadas con estos mismos números. Además se encargará de establecer la conexión con el servidor auxiliar en caso de que se le indique realizar una operación compleja. Para ello he creado una función “iniciarServicio”:

```
4 usages  MixZ
def iniciarServicio(operacion, vector1, vector2):
    transport = TSocket.TSocket("localhost", 9094)
    transport = TTransport.TBufferedTransport(transport)
    protocol = TBinaryProtocol.TBinaryProtocol(transport)

    client = Calculadora2.Client(protocol)
    transport.open()

    resultado = client.operacionvectores(operacion, vector1, vector2)
    transport.close()

    return resultado
```

```
calculadora.thrift x servidor_aux.py x servidor.py x cliente.py x Calculadora.py x calculadora2.thrift x generar.sh x
58     print("\n")
59     return resultado
60
61     4 usages (4 dynamic)  MixZ
62     def division(self, cociente, divisor):
63         print("dividiendo " + str(cociente) + " con " + str(divisor))
64         return cociente / divisor
65
66     3 usages (3 dynamic)  MixZ
67     def potencia(self, base, potencia):
68         print("Calculando la potencia " + str(potencia) + "-ésima de " + str(base))
69         return base ** potencia
70
71     3 usages (3 dynamic)  MixZ
72     def raiz(self, dentro, potencia):
73         print("Calculando la raiz " + str(potencia) + "-ésima de " + str(dentro))
74         return pow(dentro, 1 / potencia)
75
76     3 usages (3 dynamic)  MixZ
77     def sumavectores(self, vector1, vector2):
78         print('OPERACIÓN COMPLEJA: Enviando vectores para sumar al servidor auxiliar...\n')
79         resultado = iniciarServicio(1, vector1, vector2)
80         print('El servidor auxiliar ha calculado con éxito los valores...\n')
81         return resultado
82
83     3 usages (3 dynamic)  MixZ
84     def restarvectores(self, vector1, vector2):
85         print('OPERACIÓN COMPLEJA: Enviando vectores para restar al servidor auxiliar...\n')
86         resultado = iniciarServicio(2, vector1, vector2)
87         print('El servidor auxiliar ha calculado con éxito los valores...\n')
88         return resultado
89
90     3 usages (3 dynamic)  MixZ
91     def multiplicarvectores(self, vector1, vector2):
92         print('OPERACIÓN COMPLEJA: Enviando vectores para multiplicar al servidor auxiliar...\n')
93         resultado = iniciarServicio(3, vector1, vector2)
94         print('El servidor auxiliar ha calculado con éxito los valores...\n')
95         return resultado
96
97     3 usages (3 dynamic)  MixZ
```

Servidor_aux.py:

```
calculadora.thrift x servidor_aux.py x servidor.py x cliente.py x Calculadora.py x calculadora2.thrift x generar.sh x
16 def __init__(self):
17     self.log = {}
18
19     ▲ MixZ
20 def ping(self):
21     print('Servidor auxiliar activo. Me han hecho ping.\n')
22
23     2 usages (2 dynamic) ▲ MixZ
24 def operacionvectores(self, op, vector1, vector2):
25     resultado = []
26     if op == 1:
27         print('Sumando vectores...\n')
28         for i in range(min(len(vector1), len(vector2))):
29             resultado.append(vector1[i] + vector2[i])
30     elif op == 2:
31         print('Restando vectores...\n')
32         for i in range(min(len(vector1), len(vector2))):
33             resultado.append(vector1[i] - vector2[i])
34     elif op == 3:
35         print('Multiplicando vectores...\n')
36         for i in range(min(len(vector1), len(vector2))):
37             resultado.append(vector1[i] * vector2[i])
38     elif op == 4:
39         print('Dividiendo vectores...\n')
40         for i in range(min(len(vector1), len(vector2))):
41             resultado.append(vector1[i] / vector2[i])
42     else:
43         resultado = "ERROR"
44     return resultado
45
46 if __name__ == "__main__":
47     handler = CalculadoraHandler()
48     processor = Calculadora2.Processor(handler)
49     transport = TSocket.TServerSocket(host="127.0.0.1", port=9094)
50     tfactory = TTransport.TBufferedTransportFactory()
51     pfactory = TBinaryProtocol.TBinaryProtocolFactory()
52     server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)
53
```

Naturalmente, se comprobará desde el cliente que no se introduzcan valores inválidos, como dividir entre 0 o hacer raíces negativas.

Del lado del cliente, tenemos tanto las comprobaciones de los números introducidos, como la llamada al servidor principal para realizar una operación u otra. Irá leyendo dato por dato dependiendo de la operación elegida:

```
while haciendo_operaciones:
    print('Elige la operación que quieres realizar: \n')
    print('1) Sumar valores\n')
    print('2) Restar valores\n')
    print('3) Multiplicar valores\n')
    print('4) Dividir valores\n')
    print('5) Potencia de un número.\n')
    print('6) Raíz n-ésima de un número.\n')
    print('7) Sumar vectores.\n')
    print('8) Restar vectores.\n')
    print('9) Multiplicar vectores.\n')
    print('10) Dividir vectores.\n')
    print('Cualquier otro número Abandonar programa.\n')
    opcion = input()
    try:
        opcion = int(opcion)
    except ValueError:
        print('No se ha podido leer el valor correctamente, por favor inténtelo de nuevo.\n')

    primer_vector = []
    segundo_vector = []

    if opcion == 1:
        valores = input('Introduzca los valores que desee sumar separados por un espacio: ')
        numeros = [float(x) for x in valores.split()]
        resultado = client.suma(numeros)
        print('Resultado: ' + str(resultado) + '\n')

    elif opcion == 2:
        valores = input('Introduzca los valores que desee restar separados por un espacio: ')
        numeros = [float(x) for x in valores.split()]
        resultado = client.resta(numeros)
        print('Resultado: ' + str(resultado) + '\n')
```

calculadora.thrift x servidor_aux.py x servidor.py x cliente.py x Calculadora.py x calculadora2.thrift x generar.sh x

```
elif opcion == 2:
    valores = input('Introduzca los valores que desee restar separados por un espacio: ')
    numeros = [float(x) for x in valores.split()]
    resultado = client.resta(numeros)
    print('Resultado: ' + str(resultado) + '\n')

elif opcion == 3:
    valores = input('Introduzca los valores que desee multiplicar separados por un espacio: ')
    numeros = [float(x) for x in valores.split()]
    resultado = client.multiplicacion(numeros)
    print('Resultado: ' + str(resultado) + '\n')

elif opcion == 4:
    valor_1 = pide_cociente()
    valor_2 = pide_divisor()
    resultado = client.division(valor_1, valor_2)
    print('Resultado: ' + str(resultado) + '\n')

elif opcion == 5:
    base = float(input('Introduzca la base de la potencia: \n'))
    potencia = float(input('Introduzca la potencia a la que desea elevar la base: \n'))
    resultado = client.potencia(base, potencia)
    print('Resultado: ' + str(resultado) + '\n')

elif opcion == 6:
    dentro = -1
    potencia = -1
    while dentro < 0 or potencia <= 0:
        print('El radicando y el indice deben ser positivos.\n')
        dentro = float(input('Introduzca el radicando: \n'))
        potencia = float(input('Introduzca el indice de la raiz: \n'))

    resultado = client.raiz(dentro, potencia)
    print('Resultado: ' + str(resultado) + '\n')

elif opcion == 7:
    advertencia_vectores()
    primer_vector = pedir_vector_1()
    segundo_vector = pedir_vector_2(1)
```


Ejemplos

1. Sumar

```

C:\Users\almaR2>ubuntu
C:\Users\almaR2>cd practica-2-2-codigo/gen-py/
C:\Users\almaR2>python3 servidor.py
Iniciando servidor...
Se han hecho ping()
Sumando los valores introducidos:
1.0
2.0
3.0
4.0
5.0
-

^
C:\Users\almaR2>python3 cliente.py
Haciendo ping al server.
Elige la operación que quieres realizar:
1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
1
Introduzca los valores que desee sumar separados por un espacio: 1 2 3 4 5
Resultado: 15.0

```

2. Restar.

```

C:\Users\almaR2>ubuntu
C:\Users\almaR2>cd practica-2-2-codigo/gen-py/
C:\Users\almaR2>python3 servidor.py
Iniciando servidor...
Se han hecho ping()
Sumando los valores introducidos:
1.0
2.0
3.0
4.0
5.0
-

^
C:\Users\almaR2>python3 cliente.py
Resultado: 15.0
Elige la operación que quieres realizar:
1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
2
Introduzca los valores que desee restar separados por un espacio: 3 2 19
Resultado: -18.0

```

3. Multiplicar.

```

C:\Users\almaR2>ubuntu
C:\Users\almaR2>cd practica-2-2-codigo/gen-py/
C:\Users\almaR2>python3 servidor.py
Iniciando servidor...
Se han hecho ping()
Sumando los valores introducidos:
1.0
2.0
3.0
4.0
5.0
-

^
C:\Users\almaR2>python3 cliente.py
Resultado: -18.0
Elige la operación que quieres realizar:
1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
3
Introduzca los valores que desee multiplicar separados por un espacio: 1 2 3 4 10 -1
Resultado: -248.0

```

4. Dividir.

```

C:\Users\almaR2>ubuntu
C:\Users\almaR2>cd practica-2-2-codigo/gen-py/
C:\Users\almaR2>python3 servidor.py
Iniciando servidor...
Se han hecho ping()
Sumando los valores introducidos:
1.0
2.0
3.0
4.0
5.0
-

^
C:\Users\almaR2>python3 cliente.py
Resultado: -18.0
Elige la operación que quieres realizar:
1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
4
Elige qué operación desea hacer con los valores que introduzca en el cociente:
1) Sumar.
2) Restar.
3) Multiplicar.
4) Introducir valor directamente.
1
Introduzca los valores que desee sumar separados por un espacio:
1 2 3 4
Elige qué operación desea hacer con los valores que introduzca en el divisor:
1) Sumar.
2) Restar.
3) Multiplicar.
Cualquier otro número) Introducir valor directamente.
4
Introduzca el valor del divisor:
4
Resultado: 2.5

```

5. Potencia de un número.

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
2.0
3.0
4.0
5.0
Restando los valores introducidos:
3.0
2.0
19.0

Multiplicando los valores introducidos:
1.0
2.0
3.0
4.0
10.0
-1.0

Sumando los valores introducidos:
1.0
2.0
3.0
4.0
dividiendo 10.0 con 4.0
Calculando la potencia 3.0-ésima de 2.0
Calculando la potencia 2.0-ésima de 2.0
-
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Elige la operación que quieres realizar:
1) Sumar valores
2) Restar valores
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
5
Introduzca la base de la potencia:
2
Introduzca la potencia a la que desea elevar la base:
3
Resultado: 8.0
```

6. Raíz n-ésima de un número.

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
1.0
2.0
3.0
4.0
10.0
-1.0

Sumando los valores introducidos:
1.0
2.0
3.0
4.0
dividiendo 10.0 con 4.0
Calculando la potencia 2.0-ésima de 2.0
Calculando la raíz 2.0-ésima de 100.0
-
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
6
El radicando y el índice deben ser positivos.
Introduzca el radicando:
100
Introduzca el índice de la raíz:
2
Resultado: 10.0
Elige la operación que quieres realizar:
```

7. Sumar vectores

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xWaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor.py
Iniciando servidor...
Se han hecho ping()
OPERACIÓN COMPLETA: Enviando vectores para sumar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
-
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
7
Tenga en cuenta que si los vectores tienen diferentes tamaño, se mantendrá el que menor tamaño tenga.
Introduzca los valores del primer vector: 1 2 3 4
Introduzca los valores del segundo vector: -1 -2 -3 -4
Resultado:
0.0
0.0
0.0
0.0
Elige la operación que quieres realizar:
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xWaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor_aux.py
Iniciando servidor...
Sumando vectores...

Restando vectores...

Sumando vectores...

Sumando vectores...
-
```

8. Restar vectores

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Sumando los valores introducidos:
1.0
2.0
3.0
4.0
dividiendo 10.0 con 4.0
Calculando la potencia 3.0-ésima de 2.0
Calculando la raíz 2.0-ésima de 100.0
OPERACIÓN COMPLEJA: Enviando vectores para sumar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
OPERACIÓN COMPLEJA: Enviando vectores para restar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...

isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
^5) Potencia de un número.
^6) Raíz n-ésima de un número.
^7) Sumar vectores.
^8) Restar vectores.
^9) Multiplicar vectores.
^10) Dividir vectores.
Cualquier otro número) Abandonar programa.
8
Tenga en cuenta que si los vectores tienen diferentes tamaño, se mantendrá el que menor tamaño tenga.
Introduzca los valores del primer vector: 1 2 10 -1 -3
Introduzca los valores del segundo vector: -1 -3 -4 5
Resultado:
2.0
5.0
23.0
-6.0
Elige la operación que quieres realizar:
1) Sumar valores
^2) Restar valores

isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xWaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor_aux.py
Iniciando servidor...
Sumando vectores...

Restando vectores...

-
```

9. Multiplicar vectores

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xWaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor.py
Iniciando servidor...
me han hecho ping()
OPERACIÓN COMPLEJA: Enviando vectores para sumar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
OPERACIÓN COMPLEJA: Enviando vectores para multiplicar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...

isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
^1) Sumar valores
^2) Multiplicar valores
^3) Dividir valores
^4) Potencia de un número.
^5) Raíz n-ésima de un número.
^6) Sumar vectores.
^7) Restar vectores.
^8) Multiplicar vectores.
^9) Dividir vectores.
Cualquier otro número) Abandonar programa.
9
Tenga en cuenta que si los vectores tienen diferentes tamaño, se mantendrá el que menor tamaño tenga.
Introduzca los valores del primer vector: 1 2 3 4 5
Introduzca los valores del segundo vector: 2
Resultado:
2.0
Elige la operación que quieres realizar:

isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xWaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor_aux.py
Iniciando servidor...
Sumando vectores...

Restando vectores...

Sumando vectores...

Sumando vectores...

Multiplicando vectores...
```

10. Dividir vectores

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xlwaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor.py
Iniciando servidor...
me han hecho ping()
OPERACIÓN COMPLEJA: Enviando vectores para sumar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
OPERACIÓN COMPLEJA: Enviando vectores para multiplicar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
OPERACIÓN COMPLEJA: Enviando vectores para multiplicar al servidor auxiliar...
El servidor auxiliar ha calculado con éxito los valores...
-
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
1) Potencia de un número.
2) Raíz n-ésima de un número.
3) Sumar vectores.
4) Restar vectores.
5) Multiplicar vectores.
6) Dividir vectores.
7) Cualquier otro número) Abandonar programa.
8)
9)
10)
Tenga en cuenta que si los vectores tienen diferentes tamaño, se mantendrá el que menor tamaño tenga.
Introduzca los valores del primer vector: 1 2 3
Introduzca los valores del segundo vector: 0 0 0
¡No puedes introducir un 0!
Introduzca los valores del segundo vector: 1 2 3
Resultado:
1.0
1.0
1.0
Elige la operación que quieres realizar:
```

```
isma@DESKTOP-RUJ3POG: ~/practica-2-2-codigo/gen-py
Microsoft Windows [Versión 10.0.19044.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\xlwaRz>ubuntu
isma@DESKTOP-RUJ3POG:~$ cd practica-2-2-codigo/gen-py/
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$ python3 servidor_aux.py
Iniciando servidor...
Sumando vectores...

Restando vectores...

Sumando vectores...

Sumando vectores...

Multiplicando vectores...

Dividiendo vectores...
-
```

Finalizar programa.

```
3) Multiplicar valores
4) Dividir valores
5) Potencia de un número.
6) Raíz n-ésima de un número.
7) Sumar vectores.
8) Restar vectores.
9) Multiplicar vectores.
10) Dividir vectores.
Cualquier otro número) Abandonar programa.
11
Nada que operar. Cerrando programa.
isma@DESKTOP-RUJ3POG:~/practica-2-2-codigo/gen-py$
```