Programmation - Calculator - 50 points

Kévin DUVERGER

Table des matières

1 Résolution Calculator : 2

1 Résolution Calculator :

Comme dans tous les autres challenges, la première chose que l'on peut faire est de tester la connexion vers `146.59.227.136` sur le port `23405` :

```
Operation #1 :
digitSum
    19984334036843109424495663441827293436338185896966113503307332456595429771848643617090005477451788731220

Plz give me answer : NOOOOO you were too slow, how am I going to do ???
```

Il semblerait donc que l'on doive répondre le plus vite possible la réponse à des opérations.

La première étape sera donc de récupérer toutes les opérations possibles qui sont au nombre de 9 :

```
digitSum : faire la somme des chiffres du nombre
addition : l'addition
intDivision : la division entière
modulus : le modulo
modularInverse : l'inverse modulaire
tetration : la tétration (j'en reparle après)
subtraction : la soustraction
multiplication : la multiplication
modularExponentiation : l'exponentiation modulaire
factorial : la factorielle
```

Après une cinquantaine d'autres tentatives, il semble qu'il n'y a pas d'autre opération, nous allons donc pouvoir passer au code ! Pour revenir sur la tétration, lorsque le programme vous envoie `tetration 4 3`, l'idée est de calculer `4 ** (4 ** 4)` (c'est une sorte de puissance répétée qui fait grandir le résultat extrêmement vite).

Voici maintenant le code (pour chaque message on récupère l'opération et les opérandes puis on calcule et on envoie) :

```python
import math
import os, sys, socket

def factorial(inputV) :
    result = 1
    for i in range(1, inputV + 1) :
        result *= i
    return result

def digitSum(number) :
    result = 0
    numberStr = str(number)
    for i in range(0, len(numberStr)) :
        result += int(numberStr[i])
    return result

def tetration(base, power) :
    currentValue = 1
    for i in range(0, power) :
        currentValue = pow(base, currentValue)
    return currentValue

def doOperation(idVal, operandList) :
    if idVal == 0 :
        return factorial(operandList[0])
    elif idVal == 1 :
        return digitSum(operandList[0])
    elif 2 <= idVal and idVal <= 6 :
        if idVal == 2 :
            return operandList[0] + operandList[1]
        elif idVal == 3 :
            return operandList[0] - operandList[1]
        elif idVal == 4 :
            return operandList[0] * operandList[1]
        elif idVal == 5 :
            return operandList[0] // operandList[1]
        else :
            return operandList[0] % operandList[1]
    elif idVal == 7 :
        return pow(operandList[0], operandList[1], operandList[2])
    elif idVal == 8 :
        return pow(operandList[0], -1, operandList[1])
    else :
        return tetration(operandList[0], operandList[1])

```

```python
def getOperationIDFromString(name) :
    if name == "factorial" :
        return 0
    elif name == "digitSum" :
        return 1
    elif name == "addition" :
        return 2
    elif name == "subtraction" :
        return 3
    elif name == "multiplication" :
        return 4
    elif name == "intDivision" :
        return 5
    elif name == "modulus" :
        return 6
    elif name == "modularExponentiation" :
        return 7
    elif name == "modularInverse" :
        return 8
    elif name == "tetration" :
        return 9
    else :
        print("Unknown operation : " + name)

server = "146.59.227.136"
port = 23405

ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    ma_socket.connect((server, port))
except Exception as e:
    print("Problème de connexion", e.args)
    sys.exit(1)

first = True
while True :
    # Getting the data and printing
    data = ma_socket.recv(4096).decode("utf-8")
    print(data)
    # Parsing
    operationString = data.split("\n")[1 if first else 2]
    operationData = operationString.split(" ")
    operationData[0] = getOperationIDFromString(operationData[0])
    for i in range(1, len(operationData)) :
        operationData[i] = int(operationData[i])
    # Doing the operation and sending
    result = doOperation(operationData[0], operationData[1:])
    ma_socket.sendall((str(result) + "\n").encode("utf-8"))
    first = False

ma_socket.close()
```