BLOCK MAKER LIVE

Guía de ejercicios

Módulo 4

Ejercicio 1. Factorial

Escribir un programa que dado un número imprima su factorial, por ejemplo 5! (Cinco Factorial) es 120.

El factorial se calcula con la multiplicación de los números naturales anteriores o iguales.

Factorial de 0, 0! = 1

```
5! = 5*4*3*2*1 = 120
```

Pistas:

- While con el input() para solicitar numero de nuevo si este no es mayor que 0
- For con in range()

El resultado debería ser algo así:

```
Introduce número natural: 5
Factorial de 5 es 120
```

Ejercicio 2. Fibonacci

Escribir un programa para calcular la serie de Fibonacci. La serie de Fibonacci es una serie de números naturales en la que cada número se calcula como la suma de los dos anteriores. Los dos primeros valores son 1 y 1.

Así, la serie es 1, 1, 2, 3, 5, 8, 13…

Pistas:

- Solicitamos cuántos números de Fibonacci deseamos que se muestren
- Asignamos los valores iniciales 0 y 1
- For con in range()

El resultado debería ser algo así:

```
Introduce cuantos números de Fibonacci calcular: 7
1
1
2
3
5
8
13
```

Ejercicio 3. Lectura de una lista de enteros

Escribe un programa para leer una lista de números enteros del usuario. Primero ha de preguntar cuántos números a leer para luego preguntar todos los números. Los números se han de almacenar en una lista e imprimir al final.

Pistas:

- Creamos lista vacía
- For para ir añadiendo números a la lista (habiendo pedido antes cuántos números se van a añadir)
- For con in range()
- usar función o método .append de las listas para añadir elementos

El resultado debería ser algo así:

```
¿Cuantos números vas a introducir? 3
Introduce el número 1: 2
Introduce el número 2: 3
Introduce el número 3: 5
[2, 3, 5]
```

Ejercicio 4. Lectura de una lista de enteros sin tamaño predefinido

Repite el ejercicio anterior pero sin tener que preguntar cuántos números ha de introducir el usuario de antemano. Para saber cuando parar de leer números vamos a pedir al usuario que no introduzca nada cuando acabe, o sea que acabaremos al leer una cadena de caracteres vacía.

Pistas:

- Creamos lista vacía
- While True
- If dentro del while
- break dentro del if
- usar función o método .append de las listas para añadir elementos

El resultado debería ser algo así:

```
Introduce el número 1 o <enter> para finalizar: 2
Introduce el número 2 o <enter> para finalizar: 6
Introduce el número 3 o <enter> para finalizar: 8
Introduce el número 4 o <enter> para finalizar: 23
Introduce el número 5 o <enter> para finalizar:
[2, 6, 8, 23]
```

Ejercicio 5. Promedio de costes

Escribir un programa que permita calcular el promedio de una lista de números introducida. Como ejemplo podría solicitar las comisiones pagadas por varias transacciones blockchain para calcular finalmente el total de costes y el coste medio por transacción.

Pistas:

- Creamos lista vacía
- While True
- If dentro del while
- break dentro del if
- usar función o método .append de las listas para añadir elementos

El resultado debería ser algo así:

```
Introduce el coste número 1 o <enter> para finalizar: 2
Introduce el coste número 2 o <enter> para finalizar: 2.5
Introduce el coste número 3 o <enter> para finalizar: 1.3
Introduce el coste número 4 o <enter> para finalizar:
Coste total: 5.8
Coste medio: 1.9333333333333333
```

Ejercicio 6. Mayor y menor de un vector

Modificar el anterior ejercicio para mostrar también los costes mayor y menor de todas las transacciones.

El resultado debería ser algo así:

```
Introduce el coste número 1 o <enter> para finalizar: 2
Introduce el coste número 2 o <enter> para finalizar: 2.5
Introduce el coste número 3 o <enter> para finalizar: 1.34
Introduce el coste número 4 o <enter> para finalizar:
Coste total: 5.84
Coste medio: 1.9466666666666665
Max/Min: 2.5 / 1.34
```

Ejercicio 7. Extraer cuentas de lista de transacciones

Dada una lista de transacciones codificadas como diccionarios Python escribe un programa que extraiga la lista de direcciones únicas (sin repetidos) de las transacciones.

transacciones = [

{"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 500},

{"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 750},

{"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 1000},

{"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 250},

{"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 175},

{"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 300},

{"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 900},

{"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 50},

{"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 700}

]

soluciona el ejercicio a partir de aquí

El resultado debería ser algo así:

```
{'0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f', '0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D', '0x5d5e8B5D9e2d42B3C6fEb1e8DE32
aB2e3b3fD3a1', '0xCda0D449b4413B5053FA3fA58ab2EE9989c06928'}
```

Ejercicio 8. Calcular saldos de cuentas

Suponiendo que todas las cuentas tienen un saldo inicial de 1000 unidades y usando la lista de transacciones anterior, escribe un programa que calcula el saldo final de cada cuenta.

```
transacciones = [
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 500},
   {"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to":
"0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 750},
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 1000},
   {"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to":
"0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 250},
   {"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to":
"0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 175},
   {"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to":
"0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 300},
   {"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to":
"0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 900},
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 50},
   {"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 700}
]

## soluciona el ejercicio a partir de aquí
```

El resultado debería ser algo así:

```
0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1 525
0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D 875
0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f 400
0xCda0D449b4413B5053FA3fA58ab2EE9989c06928 2200
```

Ejercicio 9. Verificación de transacciones

Modifica el ejercicio anterior para verificar cada transacción antes de aplicarla. No se puede realizar una transferencia si no se tiene el saldo suficiente.

```
transacciones = [
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 500},
   {"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to":
"0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 750},
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 1000},
   {"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to":
"0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "value": 250},
   {"from": "0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "to":
"0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 175},
   {"from": "0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "to":
"0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D", "value": 300},
   {"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to":
"0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "value": 900},
   {"from": "0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 50},
   {"from": "0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f", "to":
"0xCda0D449b4413B5053FA3fA58ab2EE9989c06928", "value": 700}
]

## soluciona el ejercicio a partir de aquí
```

El resultado debería ser algo así:

```
Transacción 2 desde 0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1 no ejecutada debido a falta de saldo 1000 > 500
0x5d5e8B5D9e2d42B3C6fEb1e8DE32aB2e3b3fD3a1 1525
0x45dA78A4dFEdf4a4F4c34535204A37bF47dBe57D 875
0x92A2b0B3E3B031A14f0a9d0A3807d13b4C4D4a4f 400
0xCda0D449b4413B5053FA3fA58ab2EE9989c06928 1200
```