Argentina Programa 4.0

Universidad Nacional de San Luis

Desarrollador Python

Lenguaje de Programación Python: Soporte Funcional

Autor:

Dr. Mario Marcelo Berón

Argentina Programa 4.0
Universidad Nacional de San Luis

Práctico Nro. 7.1: *Recursividad*

Ejercicio 1: Implemente la función factorial la cual se define como sigue:

$$factorial(n) = \begin{cases} 1 & \text{si n=0} \\ \text{n* factorial(n-1)} & \text{si n >0} \end{cases}$$

Ejercicio 2: Implemente una función que permita obtener el n-ésimo número de la sucesión de fibonacci. La sucesión de fibonacci se define como sigue:

$$fibonacci(n) = \begin{cases} 0 & \text{si n=0} \\ 1 & \text{si n=1} \\ fibonacci(n-1) + fibonacci(n-2) & \text{si n >2} \end{cases}$$

Ejercicio 3: Defina una función recursiva que permita calcular el mcd de dos números.

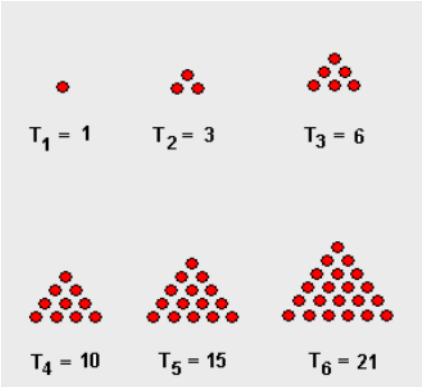
Nota: Utilice el Algoritmo de Euclides.

Ejercicio 4: Implemente la siguiente función:

$$T(n) = \begin{cases} 3 & \text{si n=0 o n=1} \\ 7+\text{T}(\frac{n}{2}) & \text{si n>1} \end{cases}$$

Ejercicio 5: Defina la función triangular(n) la cual calcula recursivamente el n-ésimo número triangular (es decir, el número 1 + 2 + 3 + ... + n).Un número triangular cuenta objetos dispuestos en un triángulo equilátero. El n-ésimo número triangular es el número de puntos en la

disposición triangular con n puntos en un lado, y es igual a la suma de los n números naturales de 1 a n, siendo por convención, el 1 el primer número triangular. Los números triangulares, junto con otros números figurados, fueron objeto de estudio por Pitágoras y los Pitagóricos, quienes consideraban sagrado el 10 escrito en forma triangular, y al que llamaban Tetraktys.



Ejercicio 6: Defina función recursiva cantidadDeDígitos(n) que recibe un número positivo, n, y devuelve la cantidad de dígitos que tiene.

Ejercicio 7: Defina la función recursiva esPotencia(n, b) la cual recibe 2 enteros, n y b, y devuelve True si n es potencia de b y False en caso contrario.

```
esPotencia(8, 2) -> True
esPotencia(64, 4) -> True
esPotencia(70, 10) -> False
esPotencia(1, 2) -> True
```

Ejercicio 8: Defina la función recursiva posiciones(a, b) que reciba como parámetros dos cadenas a y b, y devuelve una lista con las posiciones en donde se encuentra b dentro de a.

```
posiciones('Un tete a tete con Tete', 'te')
#[3, 5, 10, 12, 21]
```

Ejercicio 9: Defina dos funciones mutuamente recursivas par(n) e impar(n) que determinen la paridad del número natural dado. Tenga en cuenta que:

- 1 es impar.
- Un número mayor que uno es impar si su antecesor es par.

Ejercicio 10: Implemente la siguiente función:

$$f(n) = \begin{cases} 0 & \text{Si n=0} \\ f(\frac{1}{2}n) & \text{Si n es par y n >0} \\ 1+f(n-1) & \text{Si n es impar y n>0} \end{cases}$$

Una vez implementada la función calcule f(n) para:

1. n=1
2. n=2
3. n=3
4. n=99
5. n=100
6. n=128

Ejercicio 11: Implemente la siguiente función:

$$f(n) = \begin{cases} n & \text{Si n}\leq 1 \\ n+f(\frac{1}{2}n)) & \text{Si n es par y n >1} \\ f(\frac{1}{2}*(n+1))+f(\frac{1}{2}*(n-1)) & \text{Si n es impar y n>1} \end{cases}$$

Una vez implementada la función calcule f(n) para:

1. n=1
2. n=2
3. n=3

4. n=4
5. n=5
6. n=6

Ejercicio 12: Defina la función recursiva replicar(lista, n) la cual permite replicar n veces los elementos de una lista. Ejemplo:

```
replicar([1, 3, 3, 7], 2)
[1, 1, 3, 3, 3, 3, 7, 7])
```

Ejercicio 13: Defina función recursiva combinaciones(lista, k) la cual recibe como parámetro una lista de caracteres únicos, y un número k. La función retorna como resultado todas las posibles cadenas de longitud k formadas con los caracteres dados (se premiten caracteres repetidos).

```
>>> combinaciones(['a', 'b', 'c'], 2)
aa ab ac ba bb bc ca cb cc
```

Ejercicio 14: Defina la función recursiva búsquedaBinaria(lista, e) la cual implementa una búsqueda binaria. La función debe devolver simplemente True o False indicando si el elemento está o no en la lista.

Ejercicio 15: Realice las siguientes actividades:

1. Diseñe una representación para un árbol ternario. Un árbol ternario es aquel donde la máxima cantidad de hijos por nodo es 3.
2. Implemente los siguientes recorridos:
   a) Simétrico
   b) Post orden
   c) Pre orden