CS1100 - Introducción a Ciencia de la Computación
UTEC

UTEC

CIENCIA DE LA
COMPUTACIÓN

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- desarrollan programas en Python, utilizando funciones y recursividad.

Diseñando funciones:

Con los tipos de funciones que hemos estudiado, podemos resolver problemas como:

- realizar cálculos sucesivos. E.g. calcular la suma de cuadrados de una lista (ver ejemplo 01)
- validar o verificar. E.g. Encontrar el máximo en una lista (ver ejemplo 02)

Función que realiza cálculos (ejemplo 01)

Crear una función para encontrar la suma de los cuadrados de 1 hasta N.

```
""" codigo que realiza la suma de cuadrados de 1 a N"""
def suma_cuadrados(N):
    suma = 0
    for i in range(1, N + 1):
        suma += i * i
    return suma
N=int(input())
print(suma_cuadrados(N))
```

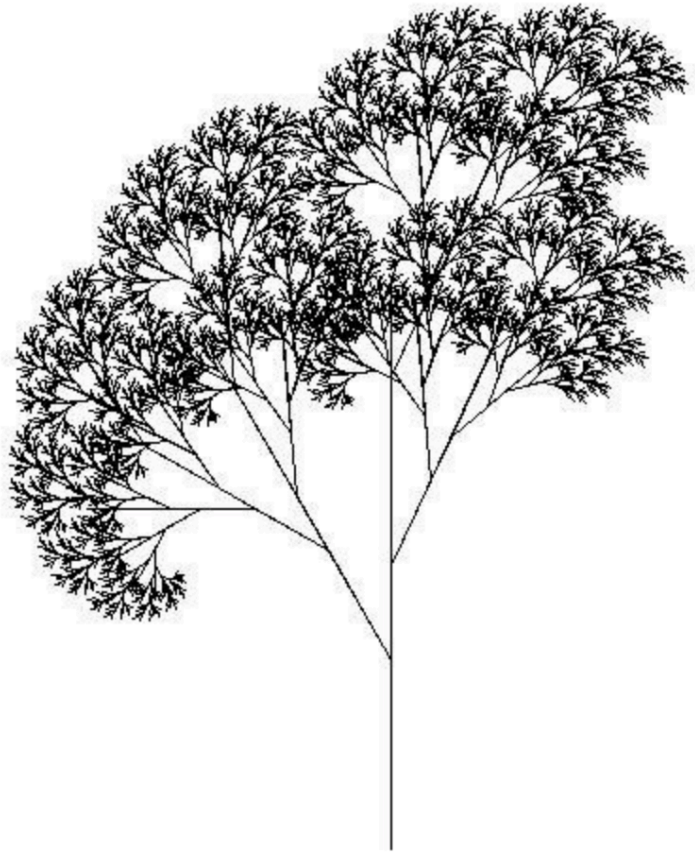
Función que realiza validaciones (ejemplo 02)

Encontrar el máximo en una lista

```
1  def Maximo(arr):
2      maximo = arr[0]
3      for item in arr:
4          if item > maximo:
5              maximo = item
6      return maximo
7
8  i=0
9  x=int(input("Ingrese un entero (termine con 0): "))
10 a=list()
11 while x != 0:
12     a.append(x)
13     i=i+1
14     x=int(input("Ingrese otro numero (termine con 0): "))
15 print("el numero maximo es: ", Maximo(a))
```

Algoritmo recursivo

Es un algoritmo que expresa la solución de un problema en términos de una llamada a si mismo (llamada recursiva o recurrente



Autoreferencia

La llamada de la función a si misma hace que se imprima el mensaje un número indeterminado de veces

```
1 def foo(s):
2     print(s)
3     foo(s)
4 foo('hola mundo')
```

El límite de recursividad previene una saturación de la memoria. Se obtiene el error:
RecursionError: maximum recursion depth exceeded while calling a Python object

Condición de salida:

La llamada de la función a si misma hace que se imprima el mensaje cada vez con un caracter menos

```
1 def foo(s):
2     if len(s) == 1:
3         print(s)
4     else:
5         print(s)
6         s = s[1:]
7         foo(s)
8 foo('hola mundo')
```

Recursividad en algoritmos: Suma de números

Forma recursiva de calcular la suma de dos números naturales a y b.

$$suma(a,b) = \begin{cases} si\ b = 0, retornar\ a \\ Retornar\ 1 + suma(a, b-1) \end{cases} \quad (1)$$

```
1 """ suma recursiva"""
2 def suma(a,b):
3     if b == 0:
4         return a
5     else:
6         return 1+suma(a,b-1)
7
8 print(suma(21,3))
```

Recursividad en algoritmos: Suma de cuadrados

Forma recursiva de calcular la suma de cuadrados de 1 a N (ver ejemplo 01)

```
1 def suma_cuadrados(N):
2     if(N==1):
3         return 1
4     else:
5         return N*N+suma_cuadrados(N-1)
6
7 N=int(input())
8 print(suma_cuadrados(N))
```

Recursividad en algoritmos: Factorial de un número

Forma recursiva de calcular el factorial de un número n.

$N! = N \times (N-1) \times (N-2) \times ... \times 2 \times 1$

$$factorial(n) = \begin{cases} si\ n = 1, retornar\ 1 \\ Retornar\ n \times factorial(n-1) \end{cases} \quad (2)$$

Ejercicio 1

Enunciado

Usando la definición anterior implementar la función factorial.

Ejercicio 2

Enunciado

Crear una función recursiva que devuelva un numero a elevado a la potencia b.

Ejercicio 3

Enunciado

Escribir una función recursiva que encuentre el mayor elemento de una lista

Evaluación

Individual Work

- www.hackerrank.com/cs1100-lab-01

Cierre

En esta sesión aprendiste:

- desarrollan programas en Python, utilizando funciones y recursividad.