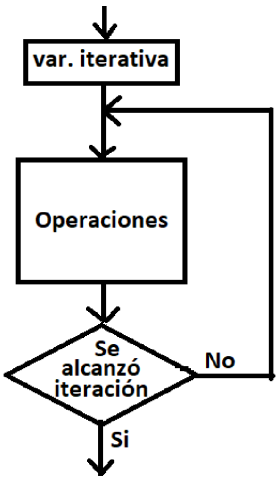
ESTRUCTURAS REPETITIVAS
Instrucción for

Cuando en un programa se requiere repetir una serie de operaciones un número determinado de veces, la instrucción *for* es la más recomendada para realizar tal acción. Su diagrama de flujo puede representarse de la siguiente forma:



Normalmente se recurre a una variable iterativa que lleve el conteo de las repeticiones; cuando se alcance el último valor de la iteración, se continua con el flujo del programa. Esta estructura también recibe el nombre de circuito, bucle o ciclo *for*.

Hay varias formas de realizar los circuitos, pero la forma más sencilla es utilizar una lista, que es una variable especial que más adelante utilizaremos, pero que por el momento generaremos con la función *range()*. Por ejemplo, la suma de los primeros *n* términos puede expresarse matemáticamente como: $\sum_{k=1}^{n} k = 1 + 2 + 3 + \ldots + n$; si utilizamos una variable auxiliar para ir acumulando la suma de los términos, llamada *suma*, referenciándole un valor inicial de cero y participando dentro del circuito, obtendremos un algoritmo para representar dicha sumatoria. Por ejemplo, si *n=10*, la suma de 1 a 10 es igual a 55. La variable *suma* recibe el nombre de <u>acumulador</u>. Si la variable se utiliza sólo para contar, como usaremos más adelante en la otra estructura repetitiva, recibe el nombre de <u>contador</u>. El programa y su corrida se muestra en la figura 8.

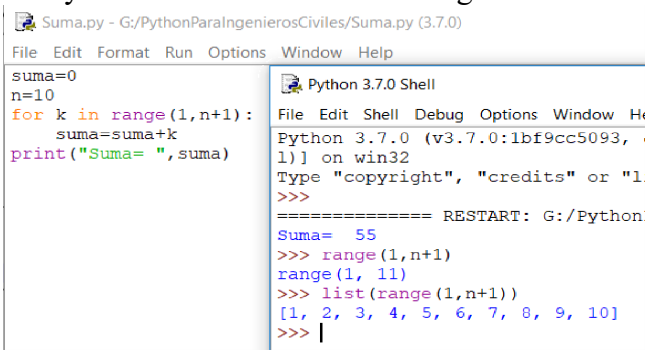


Figura 8. Programa Suma.py

Puede observarse que la función *range()* genera una lista de 1 a 10, si se incrementa el último número en 1; la variable *k* es la variable iterativa y se sustituye temporalmente por los valores de la lista (de 1 a 10); el acumulador *suma* inicia con cero y va acumulando los valores de *k*. Como recomendación, cualquier sumatoria matemática puede representarse con un acumulador, un valor final incrementado en uno y un circuito que acumule a la expresión a evaluar; en el siguiente ejemplo, pueden verse los elementos comentados arriba tanto en la expresión abreviada (sumatoria), co-

mo en su desarrollo y la representación en otras variantes, que conducen al mismo resultado, según se ve en la figura 9. Su programa y corrida se muestran en la figura 10.

$$\sum_{i=0}^{5} (1+i)^{-i} = 1.62846471$$

$$\frac{1}{(1+0)^0} + \frac{1}{(1+1)^1} + \frac{1}{(1+2)^2} + \frac{1}{(1+3)^3} + \frac{1}{(1+4)^4} + \frac{1}{(1+5)^5} = 1.62846471$$

$$1 + \frac{1}{2} + \frac{1}{9} + \frac{1}{64} + \frac{1}{625} + \frac{1}{7776} = 1.62846471$$

$$\sum_{i=0}^{5} \frac{1}{(1+i)^i} = 1.62846471$$

Figura 9. Representaciones de la sumatoria

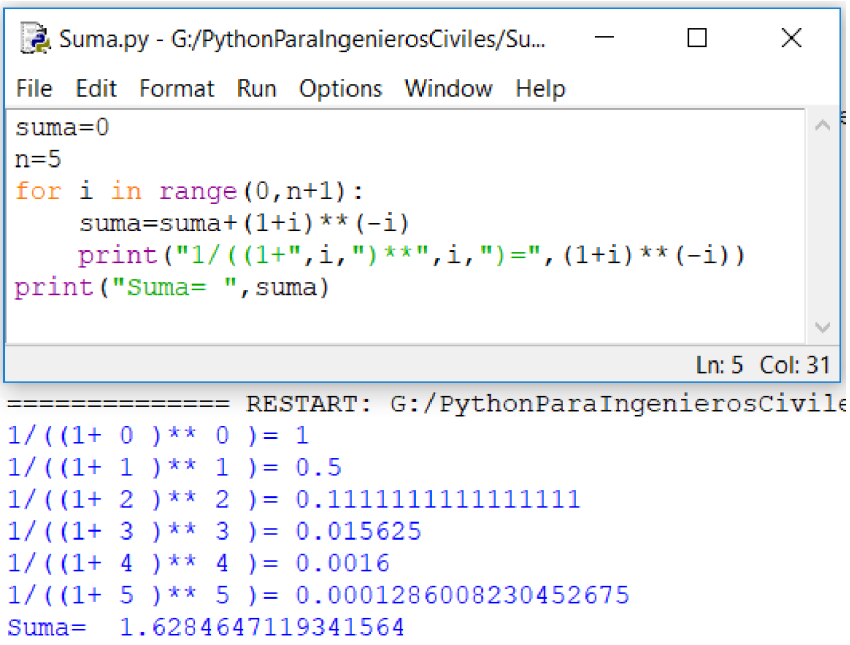


Figura 10. Programa y corrida de Suma.py

Los circuitos *for* pueden anidarse, es decir, un bucle puede estar dentro de otro u otros bucles, respetando la ejecución del más interno hacia el más externo. La siguiente expresión incorpora dos sumatorias: $\sum_{i=1}^{5} \sum_{k=1}^{i} \frac{k \cdot i}{k+i} = 20.3234127$. Su programa y ejecución se muestra en la figura 11.

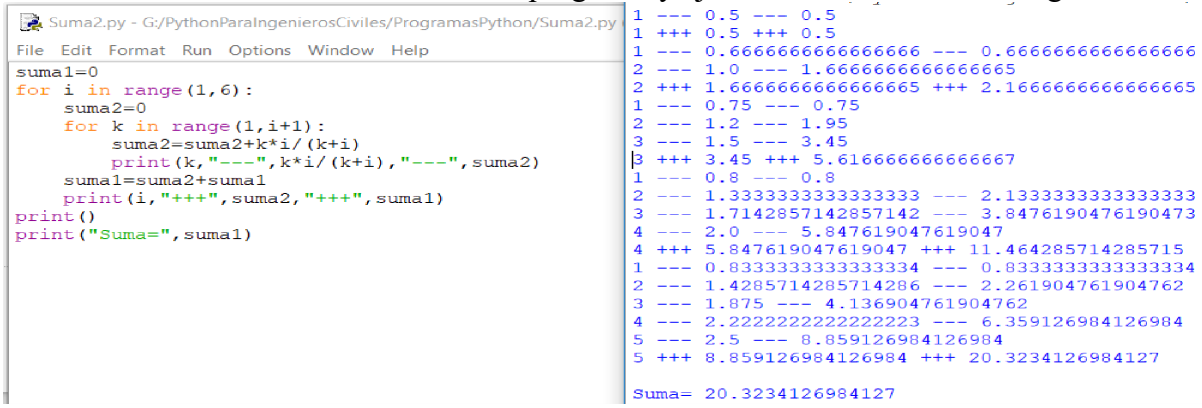


Figura 11. Programa y corrida de Suma2.py