Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

Institución Universitaria de Envigado

Diccionarios

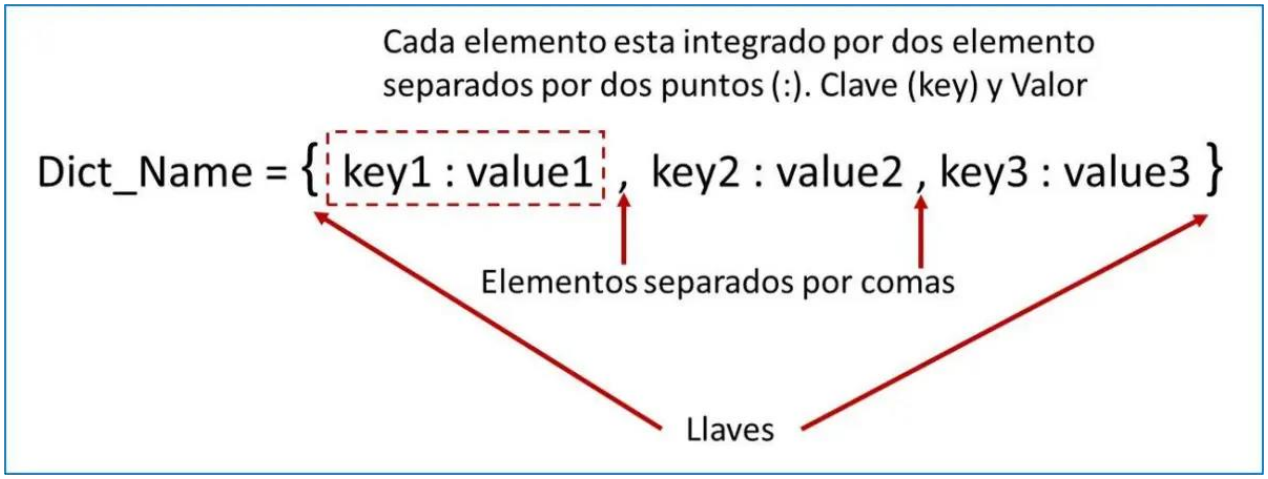
Es una estructura de datos que permite almacenar pares de elementos clave-valor. Cada elemento del diccionario está compuesto por una clave única y su respectivo valor asociado.

20_dict.py

```python
person= {
'name':'nora',
'last_name':'azua',
'langs': ['python', 'javascript'],
'age': 39
}
print(person)

person['name'] = 'maria'#reemplazar un nombre
person['age'] -= 50 #restarle a la edad 50
person['langs'].append ('rust') #añadir algo a la lista
print(person)

del person['last_name']
person.pop('age') #eliminar la llave que queremos

print(person)
```

Diccionarios

- **Clave:** Es un valor único que se utiliza para acceder al valor correspondiente en el diccionario. Las claves pueden ser de cualquier tipo inmutable, como cadenas, números o tuplas.
- **Valor:** Es el dato asociado a la clave. Puede ser cualquier tipo de dato válido en Python: números, cadenas, listas, tuplas, diccionarios, e incluso objetos personalizados.



Diccionarios

Los diccionarios en Python son mutables, lo que significa que puedes agregar, modificar o eliminar elementos después de crear el diccionario. Además, los diccionarios no mantienen ningún orden específico de los elementos, a diferencia de las listas.

```
dic1 = {  10: 7, 20: (1,2,3), 30: ['Control', 'Educación']  }
```

7	(1, 2, 3)	['Control', 'Educación']
10	20	30

```
tienda = { 'item' : ['Lápiz', 'Carpeta', 'Marcador'],
           'cantidad' : [3, 10, 5],
           'valor' : [3.50, 4.25, 7.85] }
```

item	cantidad	valor
'Lápiz'	3	3.50
'Carpeta'	10	4.25
'Marcador'	5	7.85

Diccionarios

- **Agenda:** como un diccionario puede guardar objetos o incluso otros diccionarios, es fácil pensar que se trata de una estructura de datos ideal para realizar una agenda telefónica para nuestro móvil. Se pueden almacenar todos los datos necesarios de una persona, desde su nombre o teléfono hasta su dirección de correo electrónico.
- **Control de stock:** imagina que tienes una tienda y necesitas guardar de forma ordenada todos tus productos. Un diccionario es una estructura de datos adecuada para programar una aplicación de control de stock, que permita manejar de una forma sencilla el inventario de productos de un negocio.

Diccionarios

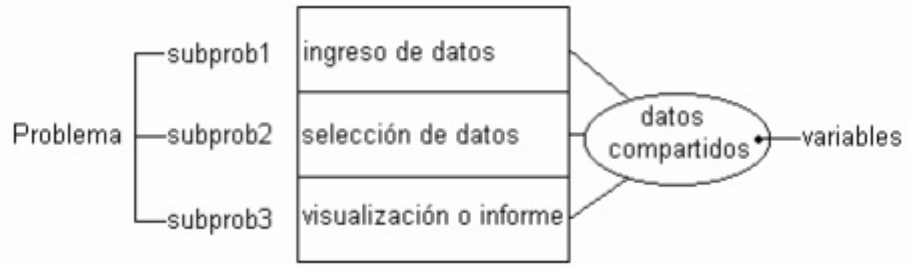
- Escribe un programa que reciba una cadena de texto como entrada y devuelva un diccionario que contenga cada palabra única de la cadena como clave y el número de veces que aparece como valor.
- Escribe un programa que solicite al usuario ingresar el nombre y la calificación de varios estudiantes y luego genere un diccionario donde las claves sean los nombres de los estudiantes y los valores sean sus calificaciones.
- Crea un programa que permita al usuario ingresar palabras en inglés y sus traducciones en otro idioma. Al final, el programa debería imprimir un diccionario con las palabras en inglés como claves y sus traducciones como valores.
- Escribe un programa que reciba una lista de palabras como entrada y genere un diccionario que cuente la cantidad de veces que aparece cada letra en todas las palabras combinadas.

Diccionarios

- Escribe un programa que reciba una cadena de texto como entrada y devuelva un diccionario que cuente la frecuencia de cada carácter en el texto, ignorando espacios en blanco y distinguiendo entre mayúsculas y minúsculas.
- Escribe una función que tome dos diccionarios como entrada y devuelva un nuevo diccionario que contenga todas las claves y valores de ambos diccionarios. Si una clave está presente en ambos diccionarios, el valor en el nuevo diccionario debería ser una lista que contenga ambos valores.
- Escribe una función que tome dos palabras como entrada y determine si son anagramas (es decir, si las letras de una palabra pueden reorganizarse para formar la otra). Utiliza un diccionario para contar la frecuencia de cada letra en ambas palabras y compara los diccionarios resultantes.
- Crea un programa que permita gestionar un inventario de productos. Debe permitir al usuario agregar nuevos productos, eliminar productos existentes, actualizar la cantidad de productos y mostrar el inventario completo.

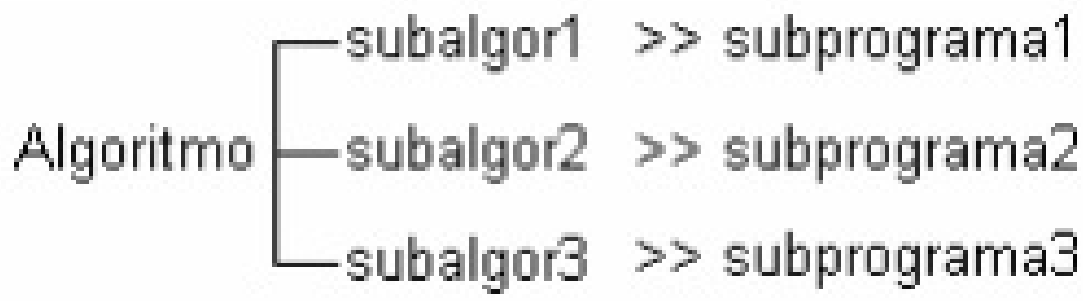
PROCEDIMIENTOS Y FUNCIONES

La resolución de problemas complejos se facilita considerablemente si se dividen en problemas más pequeños; y la resolución de estos subproblemas se realiza mediante subalgoritmos.



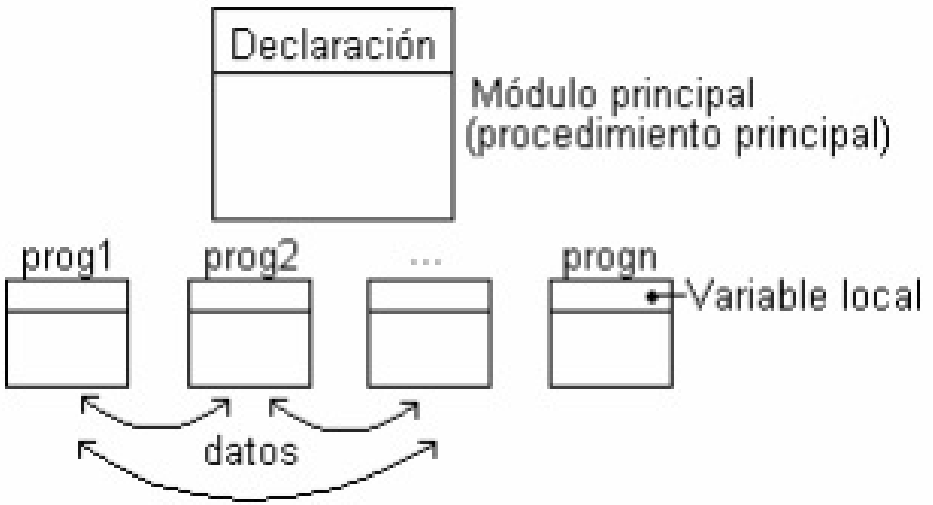
PROCEDIMIENTOS Y FUNCIONES

Los subalgoritmos son unidades de programa o módulos que están diseñados para ejecutar laguna tarea específica. Éstos, constituidos por funciones o procedimientos, se escriben solamente una vez, pero pueden ser referenciados en diferentes puntos del programa, de modo que se puede evitar la duplicación innecesaria del código.



Funciones

El módulo principal se ejecuta en una primera instancia, que da la orden de inicio de ejecución de los subprogramas. Puede ser ejecutado n veces. Es importante saber que datos se van a compartir entre los programas.



PROCEDIMIENTOS

Un procedimiento es un subprograma que ejecuta una tarea determinada. Está compuesto por un conjunto de sentencias, a las que s le asigna un nombre, o identificador. Constituyen unidades del programa, y su tarea se ejecuta siempre y cuando encuentre el nombre que se le asignó a dicho procedimiento.

Los procedimientos deben ser declarados obligatoriamente antes de que puedan ser llamados en el cuerpo del programa principal. Para ser activados o ejecutados, deben ser llamados desde el programa en que fueron declarados.

PROCEDIMIENTOS

Todo procedimiento, al igual que un programa principal, consta de una cabecera, que proporciona su nombre y sus parámetros de comunicación; de una sección de declaraciones locales y el cuerpo de sentencias ejecutables. Las ventajas más destacables de usar procedimientos son:

- Facilitan el diseño top-down.
- Se pueden ejecutar más de una vez en un programa, con solo llamarlos las veces que así desee. Con esto se ahorra tiempo de programación.
- El mismo procedimiento se puede usar en distintos programas.
- Su uso facilita la división de tareas entre los programadores de un equipo.
- Se pueden probar individualmente e incorporarlos en librerías o bibliotecas.

RECURSIÓN (recursividad)

Un subprograma que se puede llamar a sí mismo se llama recursivo. La recursión puede ser utilizada como una alternativa a la repetición o estructura repetitiva. La escritura de un procedimiento o función recursiva es similar a sus homónimos no recursivos; sin embargo, para evitar que la recursión continúe indefinidamente, es preciso incluir una condición de terminación.

RECURSIÓN (recursividad)

```
def factorial_normal(n):
    r = 1
    i = 2
    while i <= n:
        r *= i
        i += 1
    return r

factorial_normal(5)
```

```
def factorial_recursivo(n):
    if n == 1:
        return 1
    else:
        return n * factorial_recursivo(n-1)

factorial_recursivo(5)
```

RECURSIÓN (recursividad)

- Escribe una función recursiva que calcule la suma de los primeros n números naturales.
- Escribe una función recursiva que calcule el n-ésimo número de la secuencia de Fibonacci.
- Implementa una función recursiva para resolver el problema de la Torre de Hanoi con n discos.
  - n: El número de discos que deseas mover.
  - origen: El nombre de la torre de origen (A)
  - auxiliar: El nombre de la torre auxiliar (B)
  - destino: El nombre de la torre de destino (C)

RECURSIÓN (recursividad)

- Implementa una función recursiva para buscar un elemento en una lista ordenada usando búsqueda binaria.
  - lista: La lista ordenada en la que deseas buscar el elemento.
  - elemento: El elemento que deseas buscar en la lista.
  - inicio (opcional): El índice de inicio para la búsqueda en la lista. Por defecto, es 0.
  - fin (opcional): El índice de fin para la búsqueda en la lista. Por defecto, es el último índice de la lista.
- Escribe una función recursiva que genere todas las permutaciones de una lista dada.
  - lista: La lista para la cual deseas generar todas las permutaciones.