Funciones

¿Qué es una función?

Una función es un segmento de código que realiza un trabajo especifico, el cual podemos "llamar" cuando necesitemos realizar dicho trabajo u operacion matematica.

¿Para qué sirve?

Como se dijo anteriormente una función se puede invocar o llamar, cada vez que se necesite realizar un trabajo, esto nos trae como ventaja el ahorro de líneas de código y de memoria.

En el caso de la función que tenemos en la parte inferior, como ejemplo vamos a pedirle a una funcion que realice la división de dos dígitos, comprobando que no se pueda dividir en "0" ni se pueda dividir entre "0". Esto es más eficiente para el programador al momento de necesitar la función dividir, ya que solo debe llamar la función "dividir" cada vez que se necesite realizar esta operacion al escribir su código.

Sin embargo al ser un ejemplo, es una función pequeña, la ventaja no es muy notable. Pero imaginemos lo siguiente: Se requiere un algoritmo que grafique la taza de natalidad en un año por mes a mes de los últimos 5 años. y supongamos que solamente la sección del código que recibe los datos, calcula y dibuja la grafica es de 1000 líneas. Quiere decir que, si no usamos funciones, al tener que que realizar 5 graficas, 1 por año, debemos repetir la sección de 1000 linea 5 veces, solo cambiando los datos que se desean graficar. En otras palabras nuestro código será de 5000 lineas como mínimo. En este caso es mucho mas fácil y eficiente crear una función que solo reciba los datos y esta función calcule e imprima la gráfica. De esta manera solo necesitaríamos invocar la función cada vez que sea necesario realizar el gráfico, e ingresar los datos correspondiente al año, solamente haciendo esto reduciríamos 4mil líneas de códigos de nuestro programa.

Declaración de funciones

Para declarar la función debemos usar la palabra reservada o el comando "def", seguido por el nombre de nuestra función "def mi_funcion" despues de un espacio, este nombre puede ser cualquier nombre de su conveniencia, sin embargo es recomendable que el nombre esté relacionado con el trabajo que realiza la función. Seguido de este nombre se colocan paréntesis "def mi_funcion()" y dentro de estos, los parámetros que recibirá nuestra función "def mi_funcion(parametro1)", es importante que en caso que nuestra función reciba más de 1 parámetro, los parámetros estén separados por un coma "def mi_funcion(parametro1, parametro2)". Fuera de los paréntesis seguido de estos van dos puntos "def mi_funcion(parametro1):" que indican donde comienzan las líneas de código que ejecutara nuestra función. Es muy importante tener en cuenta la sangría, ya que esta es la que define donde comienza y termina el bloque de código

```
def mi_funcion(parametro1, parametro2, parametro3):
```

```
    comienzo bloque que codigo
    codigo en el bloque
    final del bloque de codigo
    return(respuesta de la funcion)
```

Se puede observar que la sangría, o espacios desde la parte izquierda de la pantalla, es igual para todos, para definir que todos hacen parte de la funcion "mi_funcion". Si la sangría varia por ejemplo:

```
def mi_funcion(parametro1, parametro2, parametro3):
    comienzo bloque que codigo
    codigo en el bloque
    final del bloque de codigo
      return(respuesta de la funcion)
```

En este segundo ejemplo, Python al momento de leer el código, interpreta lo siguiente: La función "mi_funcion" contiene un conjunto de comandos o cálculos que debo realizar, que van hasta la línea: final del bloque de código, la línea del return al estar con una sangría diferente, la lee como perteneciente a una función diferente o directamente, perteneciente a la fila: final del bloque de código. Por esto debemos tener cuidado con la sangría, si todas las líneas del código hacen parte de la función, deben tener la misma separación.

Se puede observar en otros ejemplos que a los if se le aplica una sangría que lo posiciona dentro de la función, y a su vez las líneas dentro de los if también llevan una sangría para posicionarlos dentro de ellos.

Con respecto a los parámetros de la función, como se comentó en clase, son variables que existe solamente dentro de la función, y solo se puede usar dentro de esta, para que nuestro código pueda acceder a los resultados generados dentro de nuestra función, debemos usar la instrucción "return" esta instrucción nos retorna un valor que indiquemos y nos regresa a la línea donde fue llamada la función.

¿Cómo invocar o llamar la función?

Para invocar la función solo debemos colocar su nombre y darle los parámetros que esta utilizara, ya sean variables, valores numéricos, string o booleanos, dependiendo lo que necesite la función por ejemplo:

mi_funcion(var1, 10, "homa mundo")

Nuestra función retorna un valor ya que el return se lo indicamos, esto nos permite almacenar si es necesario el resultado devuelto por nuestra función:

```
def mi_funcion(parametro1, parametro2, parametro3):
    comienzo bloque que código
    código en el bloque
    final del bloque de código
    return(respuesta de la función)
```

Aquí invocamos la función y guardamos el valor que nos retorna en una variable:

variable= mi_funcion(var1, 10, "homa mundo")

Ejercicios para practicar Semana 1

1. Leer un número entero y convertirlo a float.
2. Leer un número real y determinar su parte entera.
3. Realizar un programa que sume los valores máximo y mínimo de un grupo de datos, y que su salida sea un valor tipo str.
4. Realizar un programa que imprima hola mundo, como resultado de asignar a una variable la palabra “Hola” y a otra variable la palabra “mundo”.
5. Realizar un programa que me sume solamente los decimales de dos números.
6. Construye un programa que pregunte al usuario el radio de una circunferencia (r) y le muestre en pantalla un mensaje así: “Para una circunferencia con radio r, el perímetro es p y el área es a”. Las fórmulas para el cálculo del perímetro y el área de una circunferencia se presentan a continuación:

   Perímetro = $2*\pi*r$

   Área = $\pi*r^2$

   El programa debe definir una función para calcular el área, otra para calcular el perímetro y hacer uso de estas dos funciones.

7. Construye un programa que pregunte al usuario dos números y le muestre en pantalla un mensaje con el resultado de la suma, resta, multiplicación y división de dichos números. El programa debe definir e invocar funciones separadas para la suma, resta, multiplicación y división.
8. Construye un programa que pregunte al usuario las longitudes de los tres lados de un triángulo y arroje un mensaje informando el área del triángulo.
9. La Fórmula de Herón permite calcular el área de un triángulo cuyos lados son a, b y c:

   Área del triángulo = $\sqrt{s(s-a)(s-b)(s-c)}$

   S es el semiperímetro del triángulo, por lo tanto, se calcula así: $s = \frac{a+b+c}{2}$.

   Al igual que en los puntos anteriores, se debe construir una función que calcule el área y posteriormente hacer uso de ella dentro del programa.

10. Definir en un archivo que se llame “Practica1.py” todas las funciones que se han mencionado en el presente documento y llamarlas en un nuevo archivo que se llame “Prueba.py” con el alias “FuncionesNuevas”