Funciones y módulos

Módulo: Fundamentos de programación python para el análisis de datos

AE4: Codificar una rutina utilizando funciones preconstruidas, funciones personalizadas o de un módulo para resolver un problema de baja complejidad de acuerdo al lenguaje python.

alkemy

Introducción👩‍💻

Las funciones y módulos son herramientas fundamentales en Python que permiten estructurar, reutilizar y organizar el código de manera eficiente. Una función es un bloque de código que realiza una tarea específica y que se puede llamar en cualquier momento, lo que facilita el desarrollo de programas estructurados y reduce la repetición de código. Los módulos, por otro lado, son archivos que contienen funciones, clases y variables, y que se pueden importar en otros programas para extender su funcionalidad sin tener que escribir todo desde cero.

Python, con su amplia biblioteca estándar, incluye muchas funciones y módulos preconstruidos que permiten realizar operaciones comunes y especializadas sin esfuerzo adicional. Además, permite la creación de funciones personalizadas y módulos que se adaptan a las necesidades específicas de cada proyecto. Este manual aborda el concepto y uso de funciones y módulos, presentando las herramientas que Python ofrece para maximizar la eficiencia en la programación.

Aprendizaje esperado

Cuando finalices la lección serás capaz de:

- Comprender qué son las funciones y para qué sirven en Python.
- Utilizar funciones preconstruidas y crear funciones personalizadas en Python.
- Identificar qué es un módulo y cómo contribuye a organizar y reutilizar código.
- Conocer la biblioteca estándar de Python y cómo importar módulos.
- Utilizar el módulo `math`, el módulo `stat`, y aplicar funciones de orden superior para mejorar la eficiencia y la organización del código.

Desarrollo

Sección 1: Qué es una Función y Para Qué Sirven

Concepto de Función

Una función es un bloque de código que se define para realizar una tarea específica y que se puede ejecutar varias veces en diferentes partes del programa. En Python, las funciones se definen con la palabra clave `def`, seguida del nombre de la función, paréntesis y dos puntos. Las funciones ayudan a modularizar el código, dividiéndolo en fragmentos más pequeños y manejables que pueden usarse repetidamente.

Las funciones son esenciales para evitar la repetición de código y mejorar la legibilidad, ya que permiten encapsular una serie de instrucciones en un único nombre descriptivo. Esto facilita el mantenimiento del programa, ya que cualquier cambio en la lógica solo necesita realizarse en la función y no en cada lugar donde se utiliza la lógica. Las funciones también ayudan a organizar el código y hacen que el programa sea más claro y fácil de entender.

Ejemplo de Definición y Uso de Función

Para definir una función en Python, se utiliza la siguiente estructura:

```
def saludo():
    print("¡Hola! Bienvenido al programa.")
```

Llamando a la función `saludo()` en cualquier parte del programa, se ejecutará el bloque de código contenido en la función:

```
saludo()
# Salida: ¡Hola! Bienvenido al programa.
```

Ventajas de Utilizar Funciones

Las funciones permiten la reutilización de código, lo cual es fundamental para mejorar la eficiencia y reducir la posibilidad de errores. Por ejemplo, en lugar de escribir el mismo bloque de código en diferentes partes de un programa, se puede definir una función que encapsule ese código y llamarla en los lugares necesarios. Esto no solo reduce la cantidad de código, sino que también hace que el programa sea más fácil de depurar y modificar.

Funciones y Modularidad

Las funciones son el primer paso hacia la modularidad en la programación, ya que permiten dividir un problema en tareas más pequeñas y manejables. Esta organización facilita el trabajo en equipo y la escalabilidad del código, permitiendo que múltiples programadores trabajen en distintas funciones sin interferir en el trabajo de otros.

Documentación de Funciones

Es buena práctica documentar cada función, explicando brevemente lo que hace. En Python, se utiliza el "docstring" para añadir una descripción dentro de la función, lo cual ayuda a otros programadores (o a uno mismo en el futuro) a entender rápidamente el propósito y uso de cada función.

```
def saludo():
    """Esta función imprime un saludo de bienvenida."""
    print("¡Hola! Bienvenido al programa.")
```

Sección 2: Funciones Preconstruidas de Python

Funciones Básicas de Python

Python incluye una serie de funciones preconstruidas (o funciones integradas) que están disponibles de forma inmediata y no requieren importación de módulos adicionales. Estas funciones cubren operaciones comunes, como conversión de tipos, operaciones matemáticas básicas, manipulación de cadenas y colecciones, entre otras. Algunas de las funciones más utilizadas son `print()`, `len()`, `int()`, `float()`, `str()`, `input()` y `type()`.

Por ejemplo, la función `len()` permite conocer la longitud de una secuencia, como una lista o cadena de texto:

```
texto = "Python"
print(len(texto))  # Salida: 6
```

Función `print()` y `input()`

La función `print()` se utiliza para mostrar información en pantalla, mientras que `input()` permite capturar datos ingresados por el usuario desde la consola. Estas funciones son esenciales para interactuar con el usuario y presentar resultados.

Ejemplo de uso de `input()` y `print()` juntos:

```
nombre = input("¿Cuál es tu nombre? ")
print("Hola,", nombre)
```

Función `type()` para Determinar el Tipo de Dato

La función `type()` permite verificar el tipo de dato de una variable, lo cual es útil para asegurar que los valores sean los esperados.

```
edad = 25
print(type(edad))  # Salida: <class 'int'>
```

Conversiones de Tipo con `int()`, `float()` y `str()`

Las funciones `int()`, `float()` y `str()` convierten valores de un tipo de dato a otro, lo cual es esencial en operaciones matemáticas y de manipulación de texto.

```
numero = "10"
numero_int = int(numero)
print(numero_int + 5)  # Salida: 15
```

Ventajas de las Funciones Preconstruidas

Las funciones preconstruidas ahorran tiempo y esfuerzo, ya que permiten realizar tareas comunes de manera rápida y eficiente. Además, están optimizadas para funcionar correctamente y con buen rendimiento, por lo que es recomendable utilizarlas siempre que sea posible en lugar de implementar soluciones personalizadas.

Sección 3: Funciones Personalizadas

Definición de una Función Personalizada

Las funciones personalizadas son aquellas definidas por el usuario para realizar tareas específicas que no están cubiertas por las funciones preconstruidas. Se definen utilizando la palabra clave `def`, seguida de un nombre de función y paréntesis. Dentro del cuerpo de la función, se coloca el código que define su comportamiento.

Ejemplo de función personalizada:

```python
def cuadrado(numero):
    return numero ** 2
```

Esta función recibe un número como argumento y devuelve su cuadrado.

Parámetros y Retorno de una Función

Las funciones pueden recibir parámetros, que son valores externos que se pasan a la función para que trabaje con ellos. Los parámetros se definen entre paréntesis al declarar la función. Además, las funciones pueden retornar valores usando la palabra clave `return`.

Ejemplo de función con parámetros y retorno:

```python
def suma(a, b):
    return a + b

resultado = suma(5, 3)
print(resultado)  # Salida: 8
```

Utilización de una Función Personalizada

Para usar una función personalizada, se llama a su nombre seguido de paréntesis y, si corresponde, se pasan los valores de los parámetros. Las funciones personalizadas permiten mejorar la organización y legibilidad del código, ya que encapsulan operaciones complejas en un solo nombre.

Funciones con Parámetros Predeterminados

Python permite definir valores predeterminados para los parámetros, de modo que si no se pasan ciertos argumentos, la función utilice el valor por defecto. Esto es útil para funciones que necesitan parámetros opcionales.

Ejemplo de parámetro predeterminado:

```python
def saludo(nombre="Usuario"):
    print("Hola,", nombre)

saludo()          # Salida: Hola, Usuario
saludo("Ana")     # Salida: Hola, Ana
```

Ventajas de las Funciones Personalizadas

Las funciones personalizadas permiten crear soluciones específicas adaptadas a las necesidades de cada proyecto. Además, facilitan la reutilización del código y mejoran su modularidad, lo cual es esencial en proyectos grandes y colaborativos.

Sección 4: Qué es un Módulo y Para Qué Sirven

Concepto de Módulo

Un módulo es un archivo que contiene definiciones de funciones, variables y clases, que se pueden importar y usar en otros programas. Los módulos permiten organizar y dividir el código en partes más pequeñas y manejables, facilitando su reutilización en diferentes proyectos. En Python, cualquier archivo `.py` es un módulo que se puede importar en otros scripts.

Por ejemplo, si tenemos un archivo llamado `operaciones.py` con la siguiente función:

```python
# archivo: operaciones.py
def sumar(a, b):
    return a + b
```

Podemos importar este módulo en otro archivo y usar la función `sumar()`.

Importancia de los Módulos en la Programación

Los módulos son esenciales en la programación moderna, ya que permiten dividir el código en componentes reutilizables y organizados. Esto no solo facilita la colaboración en proyectos grandes, sino que también permite reutilizar módulos en diferentes proyectos, ahorrando tiempo y esfuerzo.

Modularidad y Mantenibilidad del Código

El uso de módulos mejora la modularidad del código, permitiendo que cada archivo se enfoque en una tarea específica. Esto facilita el mantenimiento, ya que cualquier cambio en la lógica de una función solo requiere actualizar el módulo correspondiente, sin afectar otras partes del programa.

Organización de Funcionalidades Comunes en Módulos

Los módulos permiten agrupar funcionalidades relacionadas en un solo archivo. Por ejemplo, es común crear un módulo `operaciones` para funciones matemáticas, o un módulo `archivos` para funciones de manejo de archivos.

Ejemplo de Importación de un Módulo

Para importar el módulo `operaciones.py`, basta con usar la palabra clave `import`:

```
import operaciones

print(operaciones.sumar(3, 4))  # Salida: 7
```

Sección 5: La Librería Estándar de Python

Qué es la Librería Estándar de Python

La librería estándar de Python es una colección de módulos que vienen incluidos con Python y que proporcionan funciones y herramientas listas para usar en una gran variedad de tareas. Incluye módulos para manipulación de archivos, operaciones matemáticas, gestión de fechas, acceso a internet, y más.

Ventajas de la Librería Estándar

La librería estándar facilita el desarrollo al proporcionar módulos optimizados y probados, sin necesidad de instalar herramientas adicionales. Además, estos módulos están bien documentados y ofrecen soluciones para tareas comunes.

Ejemplo de Módulo de Archivos

El módulo `os` en la librería estándar permite trabajar con el sistema de archivos. Por ejemplo:

```
import os
print(os.getcwd())  # Muestra el directorio de trabajo actual
```

Otros Módulos Útiles en la Librería Estándar

La librería estándar incluye módulos como `datetime` para manipulación de fechas, `random` para generación de números aleatorios, y `re` para expresiones regulares, todos muy útiles en distintos contextos.

Flexibilidad y Alcance de la Librería Estándar

La librería estándar es suficientemente completa para manejar gran parte de las necesidades de un proyecto sin recurrir a librerías externas, lo cual agiliza el desarrollo y simplifica la distribución del software.

Sección 6: Importación de Módulos

Importación de Módulos en Python

Para usar un módulo en Python, se utiliza la palabra clave `import`, seguida del nombre del módulo. Esto permite acceder a las funciones, variables y clases definidas en dicho módulo.

Importación con Alias

Es posible usar un alias para simplificar el nombre del módulo. Por ejemplo:

```
import math as m
print(m.sqrt(16))  # Salida: 4.0
```

Importación de Funciones Específicas

Para importar solo una función o clase de un módulo, se utiliza `from`. Por ejemplo:

```
from math import sqrt
print(sqrt(25))  # Salida: 5.0
```

Ventajas de Importar Solo lo Necesario

Importar funciones específicas mejora la legibilidad del código y reduce la memoria usada al evitar cargar funciones innecesarias.

Ejemplo Completo de Importación

```
from os import getcwd
print(getcwd())  # Muestra el directorio de trabajo actual
```

Sección 7: El Módulo `math`

Operaciones Básicas con `math`

El módulo `math` proporciona funciones para operaciones matemáticas avanzadas, como potencias, raíces y logaritmos.

Ejemplo de uso de `math`:

```
import math
print(math.sqrt(16))  # Salida: 4.0
```

Funciones Trigonométricas

`math` incluye funciones trigonométricas como `sin`, `cos`, y `tan`.

```
angulo = math.radians(45)
print(math.sin(angulo))  # Salida: 0.707...
```

Constantes de `math`

El módulo `math` incluye constantes como `pi` y `e`.

```
print(math.pi)  # Salida: 3.141592653589793
```

Operaciones de Redondeo

`math` ofrece funciones de redondeo como `ceil` (redondeo hacia arriba) y `floor` (redondeo hacia abajo).

```
print(math.ceil(2.3))  # Salida: 3
print(math.floor(2.7))  # Salida: 2
```

Aplicación en Cálculos Científicos

Las funciones del módulo `math` son útiles en aplicaciones científicas y de ingeniería, donde se requiere precisión y eficiencia en cálculos complejos.

Sección 8: El Módulo `statistics`

Funciones Básicas de Estadística

El módulo `statistics` ofrece funciones para cálculos estadísticos básicos, como media, mediana y moda.

Ejemplo:

```python
import statistics as stats
datos = [10, 20, 20, 30, 40]
print(stats.mean(datos))  # Salida: 24.0
print(stats.median(datos))  # Salida: 20
print(stats.mode(datos))  # Salida: 20
```

Medidas de Dispersión

El módulo también incluye funciones para medidas de dispersión, como desviación estándar y varianza.

```python
print(stats.stdev(datos))  # Calcula la desviación estándar
```

Cálculos Avanzados con `statistics`

Las funciones de `statistics` permiten realizar análisis de datos de manera sencilla, facilitando el desarrollo de aplicaciones de análisis de datos.

Aplicación en Ciencia de Datos

El módulo `statistics` es ideal para la ciencia de datos y la estadística, donde es necesario resumir y analizar grandes volúmenes de información.

Facilidad de Uso para Principiantes

Con el módulo `statistics`, incluso los principiantes pueden calcular medidas estadísticas sin necesidad de herramientas complejas.

Sección 9: Funciones de Orden Superior

Qué Son las Funciones de Orden Superior

Las funciones de orden superior son aquellas que aceptan otras funciones como argumentos o devuelven una función como resultado. En Python, esto permite programar de manera más flexible y reducir la repetición de código.

Funciones `map`, `filter` y `reduce`

`map` aplica una función a cada elemento de una lista; `filter` selecciona elementos que cumplen con una condición; y `reduce` combina elementos de una lista en un solo valor.

Ejemplo de `map`:

```python
numeros = [1, 2, 3]
cuadrados = list(map(lambda x: x**2, numeros))
print(cuadrados)  # Salida: [1, 4, 9]
```

Ventajas de las Funciones de Orden Superior

Las funciones de orden superior permiten construir operaciones complejas de forma compacta y legible, reduciendo la necesidad de bucles.

Uso de Expresiones Lambda

Las expresiones `lambda` son funciones anónimas que se definen en una línea. Combinadas con `map` y `filter`, permiten escribir código conciso y eficiente.

Ejemplo Completo de `filter` y `reduce`

```python
from functools import reduce
impares = list(filter(lambda x: x % 2 != 0, numeros))
suma = reduce(lambda x, y: x + y, numeros)
print(impares)  # Salida: [1, 3]
print(suma)  # Salida: 6
```

Cierre ✍️

Las funciones y módulos en Python son esenciales para desarrollar código modular, reutilizable y organizado. Con funciones, los programadores pueden encapsular lógica específica en bloques reutilizables, mientras que los módulos permiten estructurar el código en componentes bien definidos. La librería estándar de Python y sus módulos preconstruidos, como `math` y `statistics`, ofrecen herramientas potentes y listas para usar en una variedad de tareas, desde operaciones matemáticas hasta cálculos estadísticos.

Las funciones de orden superior permiten escribir código más conciso y eficiente, promoviendo un estilo de programación funcional que simplifica el manejo de datos y mejora la legibilidad. Practicar el uso de funciones y módulos permite a los desarrolladores crear soluciones más escalables y adaptables, facilitando la colaboración en proyectos grandes y mejorando la mantenibilidad del código.

El dominio de funciones y módulos es un paso esencial para avanzar hacia la programación avanzada en Python, permitiendo que los programadores desarrollen proyectos más complejos y especializados. Con estos conocimientos, es posible escribir aplicaciones que sean eficientes, organizadas y fáciles de expandir en el tiempo.

Referencias📚

Python Software Foundation. (s.f.). *Python 3 documentation*. https://docs.python.org/3/

Real Python. (s.f.). *Real Python: Learn Python programming*. https://realpython.com/

¡Muchas gracias!

Nos vemos en la próxima lección

alkemy