PROGRAMACION CON PYTHON

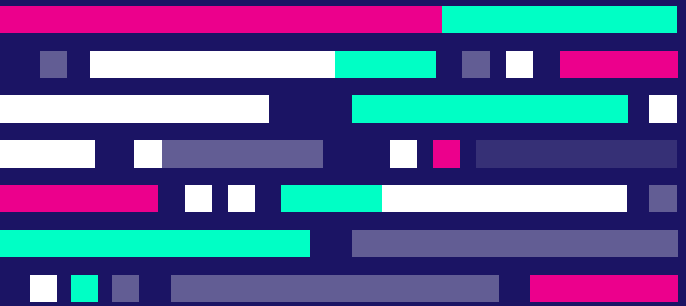
Lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código

CONTENIDO

1. Introducción a Python
2. Características de Python
3. Sintaxis y tipos de datos en Python.
4. Para qué se usa Python
5. Estructura de un programa con Python
6. IDE o métodos de compilación.
7. Bibliotecas y módulos en Python.
8. Funciones en Python.
9. Operaciones en Python
    - Operaciones aritméticas.
    - Operaciones de comparación
    - Operaciones lógicas
    - Operaciones de asignación.

INTRODUCCION

Python es un lenguaje de programación de alto nivel, interpretado, multipropósito y de código abierto que fue creado por Guido van Rossum en 1991. Es conocido por su sintaxis clara y legible, lo que lo hace ideal para principiantes, pero también es utilizado por expertos en diversas áreas de la informática, como análisis de datos, inteligencia artificial, desarrollo web, entre otros. Python es compatible con múltiples plataformas y sistemas operativos, y cuenta con una gran cantidad de bibliotecas y herramientas que facilitan el desarrollo de aplicaciones y proyectos. Además, su comunidad de usuarios es activa y diversa, lo que hace que Python sea una excelente opción para aprender a programar y crear soluciones en el mundo de la informática.



Caracteristicas de Python

1. Sintaxis clara y legible: el lenguaje está diseñado para ser fácil de leer y escribir.
2. Interpretado: el código se ejecuta directamente en un intérprete, lo que hace que el desarrollo y la depuración sean más rápidos.
3. Multipropósito: se puede utilizar para una amplia gama de tareas, desde desarrollo web hasta análisis de datos y automatización de tareas.
4. Tipado dinámico: las variables no tienen que ser declaradas con un tipo específico y pueden cambiar de tipo durante la ejecución del programa.
5. Orientado a objetos: soporta programación orientada a objetos, lo que permite la creación de clases y objetos.
6. Bibliotecas extensas: cuenta con una gran cantidad de bibliotecas y módulos que simplifican la implementación de soluciones.
7. Portabilidad: Python se ejecuta en múltiples plataformas y sistemas operativos, lo que lo hace altamente portátil.
8. Comunidad activa: cuenta con una gran comunidad de usuarios que desarrollan y mantienen bibliotecas, resuelven problemas y comparten recursos.

Sintaxis y tipos de datos

COMENTARIOS

Los comentarios comienzan con el símbolo # y se utilizan para documentar el código.

VARIABLES

Las variables se crean automáticamente cuando se les asigna un valor y no es necesario declarar el tipo de variable.

TIPOS DE DATOS

Python admite varios tipos de datos, incluidos enteros, flotantes, cadenas, booleanos y más

OPERADORES

Python admite una amplia variedad de operadores, como aritméticos, de comparación, lógicos y más

ESTRUCTURAS DE CONTROL

Python tiene varias estructuras de control de flujo, como if/else, bucles for/while y más.

SINTAXIS

Python es muy legible y fácil de aprender, lo que lo hace ideal para principiantes. Además, los tipos de datos y operadores en Python permiten una gran flexibilidad en la programación.

```
a = 5
b = “Hola”
```

—VARIABLES

```
a = 5          # Entero
b = 3.14       # Flotante
c = "Hola"     # Cadena
d = True       # Booleano
```

—Tipos de datos

PARA QUE SE USA

Desarrollo WEB

Python se utiliza ampliamente para desarrollar aplicaciones web y sitios web utilizando marcos de trabajo populares como Django.

Análisis de datos

Python es una herramienta popular para el análisis de datos y la ciencia de datos debido a sus bibliotecas y herramientas, como NumPy, Pandas, Matplotlib, y Scikit-learn.

Inteligencia artificial y aprendizaje automático

Python es una de las principales opciones para el desarrollo de modelos de aprendizaje automático e inteligencia artificial debido a sus bibliotecas y herramientas, como TensorFlow, Keras, PyTorch y más.

Automatización de tareas:

Python se utiliza a menudo para automatizar tareas como la extracción de datos, la automatización de procesos y la realización de tareas repetitivas.

Juegos

Python se utiliza para crear juegos y animaciones en 2D y 3D.

Scripting

Python se utiliza a menudo para escribir scripts para tareas específicas, como la automatización de tareas en sistemas operativos.

ESTRUCTURA

PYTHON

Línea de shebang

La primera línea de un archivo de Python puede ser una línea de shebang que indica la ubicación del intérprete de Python en el sistema. Por ejemplo, "#!/usr/bin/env python".

Comentarios

Los comentarios son líneas de texto que no se ejecutan y se utilizan para explicar el código y hacerlo más legible. En Python, los comentarios se inician con el símbolo "#" y se extienden hasta el final de la línea.

Importaciones

Las importaciones se utilizan para incluir módulos y bibliotecas en el programa. Las importaciones se colocan al comienzo del archivo Python.

Funciones

Las funciones son bloques de código que realizan una tarea específica. En Python, las funciones se definen con la palabra clave "def".

Cuerpo del programa

El cuerpo del programa es donde se escriben las líneas de código que realizan la tarea deseada. Las líneas de código se ejecutan de manera secuencial.

```
#!/usr/bin/env python

# Esto es un comentario

# Importar una biblioteca
import math

# Definir una función
def saludar(nombre):
    print("Hola, " + nombre + "!")

# Cuerpo del programa
saludar("Juan")
x = 10
y = 20
suma = x + y
print("La suma es: ", suma)
```

—Tipos de datos

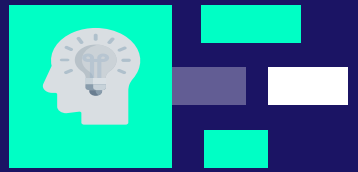
IDE o metodos de compilación

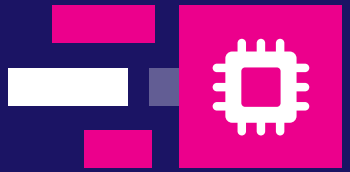
IDE (Entornos de Desarrollo Integrado)

Existen varios IDEs populares para desarrollar aplicaciones Python, que proporcionan un entorno de desarrollo integrado con muchas herramientas, incluyendo depuración, resaltado de sintaxis, autocompletado y más. Ejemplos de IDEs populares incluyen PyCharm, Visual Studio Code, Spyder, entre otros.

Compilación Just-In-Time (JIT)

Python tiene un compilador JIT incorporado llamado PyPy que puede acelerar significativamente el rendimiento de Python. PyPy compila el código a un bytecode más rápido que el intérprete predeterminado de Python y puede aumentar la velocidad de ejecución de ciertas aplicaciones.



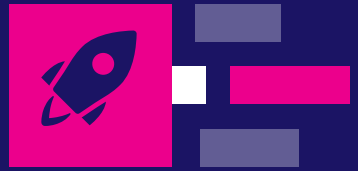


Compilación Ahead-of-Time (AOT)

Algunos proyectos como Nuitka y PyOxidizer permiten compilar código Python a un ejecutable nativo que se puede distribuir sin necesidad de tener Python instalado en el sistema. Esto es útil para distribuir aplicaciones Python como archivos independientes sin requerir la instalación de Python en la máquina destino.

Cython

Cython es un lenguaje que extiende la sintaxis de Python con construcciones de C y C++. Los archivos de Cython se compilan en código C y luego se pueden compilar en un módulo de Python. Esto puede aumentar significativamente la velocidad de ejecución de las aplicaciones Python.



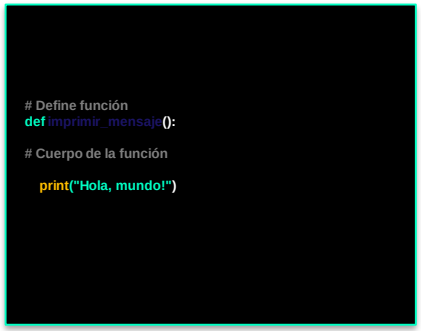
Compilador en browser, podrás ver tus compilados con una buena interfaz desde cualquier navegador.

Bibliotecas y modulos de Python

1. NumPy: Biblioteca de cálculo numérico que proporciona un conjunto de funciones matemáticas de alto nivel y una estructura de matriz N-dimensional.
2. Pandas: Biblioteca de análisis de datos que proporciona estructuras de datos de alto rendimiento y fáciles de usar, como DataFrames y Series.
3. Matplotlib: Biblioteca de trazado de gráficos 2D que proporciona una variedad de estilos de gráficos, como líneas, barras, dispersión, histogramas y más.
4. Scikit-learn: Biblioteca de aprendizaje automático que proporciona una amplia variedad de algoritmos de aprendizaje automático, como regresión lineal, clasificación, agrupamiento y más.
5. Flask: Marco web ligero que permite crear aplicaciones web rápidamente y con facilidad.
6. TensorFlow: Biblioteca de aprendizaje automático de código abierto desarrollada por Google que permite crear redes neuronales y modelos de aprendizaje profundo.
7. BeautifulSoup: Biblioteca de análisis HTML y XML que permite extraer información de sitios web.
8. Django: Marco web de alto nivel que permite crear aplicaciones web de manera rápida y eficiente, proporcionando funcionalidades como autenticación, seguridad y administración de bases de datos.
9. Pygame: Biblioteca para la creación de juegos en Python, que proporciona una gran cantidad de herramientas para el desarrollo de juegos, como el manejo de sprites, gráficos, sonidos, colisiones y más.

FUNCIONES

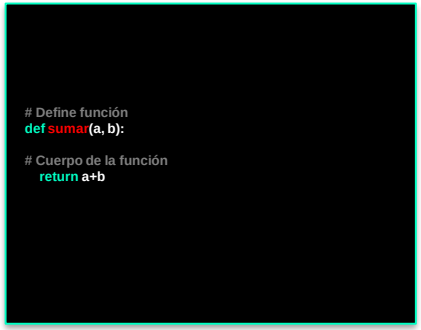
PYTHON



Una función se define mediante la palabra clave "def", seguida del nombre de la función, seguido de paréntesis y dos puntos. Por ejemplo, la definición de una función que imprime el mensaje "Hola, mundo

—Definición

—Argumentos



Una función puede recibir argumentos, que son valores que se pasan a la función para que los use en su operación. Los argumentos se especifican entre paréntesis después del nombre de la función. Por ejemplo, la siguiente función recibe dos argumentos y los suma:

—Retorno

```
# Define función
def doble(numero):

# Cuerpo de la función
    return numero * 2
```

Una función puede devolver un valor al programa que la llamó utilizando la palabra clave "return". Si una función no tiene una declaración de retorno, devuelve None por defecto. Por ejemplo, la siguiente función devuelve el doble del valor pasado como argumento:

```
# Define de constante
x = 5


# Define función
def funcion():
  # Define de constante
  x = 10
  print("Dentro de la función, x tiene el valor " ,x)

funcion():


print("Fuera de la función, x tiene el valor " ,x)
```

Las variables definidas dentro de una función tienen un ámbito local, lo que significa que solo pueden ser accedidas dentro de la función. Las variables definidas fuera de una función tienen un ámbito global y pueden ser accedidas desde cualquier parte del programa.

—Ámbito

```
# Define función
def función_exterior():

    print("Esta es la función exterior" )
        def función_interior():

            print("Esta es la función interior" )

        función_interior()
función_exterior()
```

Una función también puede contener otra función dentro de ella. Estas son conocidas como funciones anidadas o funciones internas

—ANIDADAS

```python
# Define función

cuadrado = lambda x: x ** 2

# Uso de la función lambda para calcular el cuadrado de un numero

resultado = cuadrado(5)

# Imprime
print(resultado)
```

Las funciones lambda son funciones pequeñas y anónimas que se utilizan para realizar operaciones simples. Se definen utilizando la palabra clave "lambda" seguida de los argumentos y la operación a realizar.

—LAMBDA

OPERACIONES

PYTHON

```
# Suma
resultado = 5 + 3
print(resultado) # Imprime: 8

# Suma
resultado = 10 - 4
print(resultado) # Imprime: 6

# Multiplicación
resultado = 2 * 6
print(resultado) # Imprime: 12

# División
resultado = 10 / 2
print(resultado) # Imprime: 5,0
```

En Python, se pueden realizar operaciones aritméticas básicas como suma, resta, multiplicación y división utilizando los operadores aritméticos +, -, * y /, respectivamente.

—ARITMETICAS

—COMPARACIÓN

```
# Mayor que
resultado = 5 > 3
print(resultado) # Imprime: TRUE

# Menor que
resultado = 10 < 4
print(resultado) # Imprime: False

# Mayor o igual que
resultado = 2 >= 2
print(resultado) # Imprime: True

# Mayor o igual que
resultado = 10 <= 2
print(resultado) # Imprime: False

# Igualdad
resultado = 5 == 5
print(resultado) # Imprime: True

# Desigualdad
resultado = 10 != 2
print(resultado) # Imprime: True
```

En Python, se pueden realizar operaciones de comparación entre valores utilizando operadores de comparación. Estos operadores comparan dos valores y devuelven un valor booleano True o False según si la comparación es verdadera o falsa.

```
# AND
resultado = True and False
print(resultado) # Imprime: False

# OR lógico
resultado = 2 >= 2
print(resultado) # Imprime: True

# NOT
resultado = not True
print(resultado) # Imprime: False
```

En Python, se pueden realizar operaciones lógicas entre valores booleanos utilizando operadores lógicos. Estos operadores combinan dos valores booleanos y devuelven un valor booleano según la operación realizada.

—LOGICAS

—ASIGNACION

```
# Asignación simple
resultado = 5

# Asignación compuesta (suma)
x += 3 # Equivalente a x=x+3

# Asignación compuesta (resta)
x -= 2 # Equivalente a x=x-2

# Asignación compuesta (multiplicación)
x *= 2 # Equivalente a x=x*2

# Asignación compuesta (resta)
x /= 3 # Equivalente a x=x/2
```

En Python, se pueden realizar operaciones de asignación utilizando el operador =. Este operador se utiliza para asignar un valor a una variable. Además, Python también admite operadores de asignación compuestos, que combinan una operación aritmética o lógica con la asignación en una sola instrucción.

EXERCISE

1.Escribe un programa que imprima "¡Hola Mundo!" en la pantalla.
2. Escribe un programa que tenga dos números constantes y muestre en pantalla la suma, la resta, multiplicación y división entre estos dos números.
3. Escribe un programa que convierta una temperatura en grados Celsius a grados Fahrenheit. La fórmula para la conversión es: F = (C * 1.8) + 32.
4. Crea 4 funciones por cada operación (suma,resta,multiplicación,division) , esta función debe recibir dos parámetros, pueden ser constantes , igualmente return el valor en una variable y imprimelo en pantalla.

INSTALACION

ENTORNO DE DESARROLLO

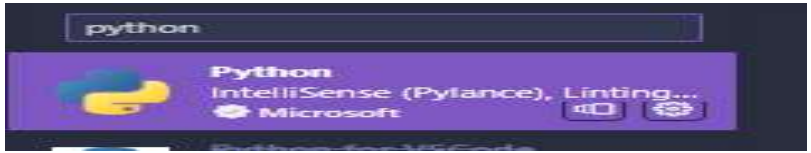
Visual Studio Code es un editor de código gratuito y multiplataforma. Ofrece características útiles para los desarrolladores y es altamente personalizable.

VISUAL STUDIO CODE

INSTALACION PYTHON

1, Realizamos la instalación de

2, Luego de instalado nos vamos a la sección de extensiones.
3. Buscamos la extensión que se llama Python
3. Instalamos la extensión.



Manejo de input

PYTHON

Esta función toma la entrada del usuario desde la consola en forma de una cadena (string).

INPUT EN PYTHON

```
nombre = input("Ingresa
tu nombre: ")

print(nombre)
#Imprime el nombre
```

Condicionales

PYTHON

Esta función toma la entrada del usuario desde la consola en forma de una cadena (string).

IF

INPUT EN PYTHON

```
edad = 18

if edad >= 18:
    print("Eres mayor de edad")
```

Se utiliza para evaluar una condición y ejecutar un bloque de código si la condición es verdadera, y otro bloque de código si la condición es falsa.

IF -ELSE

INPUT EN PYTHON

```
edad = 16

if edad >= 18:
    print("Eres mayor de edad")
else:
    print("Eres menor de edad")
```

Se utiliza para evaluar múltiples condiciones y ejecutar diferentes bloques de código en función de cada condición

IF -ELIF- ELSE

INPUT EN PYTHON

```
nota = 75

if nota >= 90:
    print("Tienes una A")
elif nota >= 80:
    print("Tienes una B")
elif nota >= 70:
    print("Tienes una C")
else:
    print("Tienes una D o una F")
```

EXERCISE

1. Calculadora básica: Crea una función que realice las operaciones básicas de suma, resta, multiplicación y división de dos números. Utiliza condicionales para asegurarte de que el usuario solo ingrese operaciones válidas.

2. Conversor de moneda: Crea una función que convierta una cantidad de dinero de una moneda a otra. Utiliza condicionales para permitir que el usuario seleccione la moneda de entrada y la moneda de salida.

3. Comprobador de números pares e impares: Crea una función que tome como entrada un número y determine si es par o impar. Utiliza condicionales y operadores para realizar la comprobación.

4. Calculadora de descuentos: Crea una función que calcule el precio final después de aplicar un descuento a un producto. Utiliza condicionales para verificar si el usuario ingresa un valor de descuento válido.

GRACIAS

Realizada por :

Carlos Alejandro Maldonado López
Jhojan Castiblanco Leon