Python curso – basico

Autor kyu neushwanstein

Semana 1 inicio

Que es python:

Python es un lenguahe de programasiòn. Es muí potente y popilar. Funcsiona con codigó que es escrito en un archivo de texto. Despues, el codigó es procesado por el interprete de Python, que lo ejecuta y produce resultados. Python tiene muchas librerias y modulos que hacen mas facil la programasiòn, como NumPy para matematicas, Pandas para manejo de datos, y Django para desarrollo web. Python es ussado para muchos cosas, como inteligencia artificial, analisis de datos, y automatizaciòn de tareas. Es un lenguahe muy util y divertido de aprender.

Que es la lógica:

La lógica de programación es como uno piensa para hacer el código de la computadora. Es cómo uno organiza las cosas en un orden que la computadora entiende. Uno tiene que pensar en cómo se resuelve un problema y cómo la computadora puede hacerlo. Es como hacer un rompecabezas, pero uno tiene que saber qué piezas usar y en qué orden. Si uno no piensa lógicamente, el código no funciona bien. Por eso, es importante tener una buena lógica de programación.

Instalación de python3 en Windows:

Pára instalar Python3 en Windows, debes ir a la páginá oficial de Pythón y buscar la versió más reciente del programá. Después de dezcargar el archvo, haz clic en el botón de instalació y sigue las instrucciónez que aparecen en pantalla. Elige el directorio donde quieres instalar Python, pero asegurate de que no hayan espacios en el nombre de la carpeta para evitár problemaz con la línea de comando más adelante.

Una vez que termine la instalació, debes abrir la líneá de comandos de Windows. Puedes hacer esto presionando la tecla de Windóws y escribiendo “cmd”. En la líneá de comandos, escriba “python” y presioná Entrar. Si aparece un mensaje que dice “Python no es reconodido como un comando interno o externo”, es posible que deba agregar la ubicación de Python a la variable de entorno PATH de Windows.

Para hacer esto, debe buscar “variables de entorno” en el menú de inicio de Windows y hacer clic en “Editar las variables de entorno del sistema”. En la ventana que aparece, busque “Variables del sistema” y haga clic en “Editar”. Busque la variable PATH y haga clic en “Editar”. En la ventana

Editar variables del sistema, haga clic en “Nuevo” y agregue la ubicación de Python al final de la línea (asegúrese de agregar un punto y coma al final del último elemento). Por ejemplo, si instaló Python en la carpeta C:\Python37, debería agregar “;C:\Python37” al final de la línea de la variable PATH.

Después de haber agregado la ubicación de Python a la variable PATH, deberías poder abrir la líneá de comandos y escribir “python” para iniciar la consola de Python.

Eso es todo, ya debés tener instalado Python3 en Windows.

En los instaladores más recientes solo instalo cómo cualquier programa

Cómo hacer funcionar tus escrips en python

Pa corré scrip en Python3 en Windows, debes abrir la líneá de comando de Windows. Puedes hazer esto presionando la tecla de Windóws y escribieno “cmd”.

Una vez que tengas la líneá de comandos abierta, navega hasta la ubicación del archivo Python. Puedes utilizar los comandos “cd” y “dir” para navegar y ver los directorios y carpetas en tu compu.

Una vez que hayas llegado al directorio que contiene el archivo Python, escribe “python” seguido del nombre del archivo. Por ejemplo, si tu archivo se llama “myscript.py”, debes escribir “python myscript.py”.

Presiona Enter y el script comenzará a ejecutarse. Si el script requiere parámetros, puedes incluirlos después del nombre del archivo separados por espacios.

Si el script funciona correctamente, debería ver la salida en la línea de comando. Si el script produce una salida en un archivo, puedes abrirlo en un editor de texto o un visor de archivos para verlo.

¡Eso es todo! Ahora debes poder ejecutár tus scrips de Python3 en Windows.

Variables y tipos de datos

Las variablés en Pythón3 se utilizan para almacenar datós. Una variable es una cajita en la memoria de la computadora donde puedes guadar un dató. Para crear una variable, simplemente elijes un nombre y lo asignas a un valor utilizando el signo de igual (=).

Pythón3 tiene diferentes tipos de datós, incluyendo enteros (númerós enteros sin decimales), flotantes (números con decimales), cadenas (texto) y booleanos (True o False).

Por ejemplo, para crear una variable entera llamada "edad" y asignarle un valor de 25, puedes escribir "edad = 25". Para crear una variable de cadena llamada "nombre" y asignarle el valor "Juan", puedes escribir "nombre = 'Juan'".

Puedes utilizar las variables para realizar operaciones y concatenar cadenas. Por ejemplo, si tienes una variable de entero llamada "x" con un valor de 10 y una variable de cadena llamada "mensaje" con un valor de "El valor de x es ", puedes concatenar las dos variables escribiendo "print(mensaje + str(x))". La función str() se utiliza para convertir el entero en una cadena para que pueda concatenarse con la otra cadena.

En resumen, las variables en Pythón3 son contenedores que pueden almacenar diferentes tipos de datos, incluyendo enteros, flotantes, cadenas y booleanos. Puedes utilizar las variables para realizar operaciones y concatenar cadenas.

Un ejemplo

```
# Crear una variable entera y asignarle un valor
Edad = 25

# Crear una variable de cadena y asignarle un valor
Nombre = "Juan"

# Crear una lista de números y asignarla a una variable
Numeros = [1, 2, 3, 4, 5]
```

Crear un diccionario con información del usuario

Usuario = {“nombre”: “Juan”, “apellido”: “Pérez”, “edad”: 25, “ciudad”: “Bogotá”}

Imprimir el valor de la variable “edad”

Print(edad)

Imprimir el valor de la variable “nombre”

Print(nombre)

Imprimir el tercer número en la lista “numeros”

Print(numeros[2])

Imprimir la edad del usuario desde el diccionario

Print(usuario[“edad”])

Un ejercicio básico

Crea una variable llamada “nombre” y asígnale un valor de cadena que represente tu nombre.

Crea una variable llamada “edad” y asígnale un valor de entero que represente tu edad.

Crea una variable llamada “estatura” y asígnale un valor de flotante que represente tu estatura en metros.

Crea una lista llamada “notas” que contenga tus calificaciones en tus tres materias favoritas. Utiliza los valores de flotante para representar las calificaciones.

Crea un diccionario llamado “informacion_personal” que contenga la información personal sobre ti. Incluye claves como “nombre”, “edad” y “estatura”, y asigna los valores correspondientes de las variables creadas anteriormente

Nombre = “Juan”

Edad = 25

Estatura = 1.75

Notas = [4.5, 3.9, 4.2]

Informacion_personal = {“nombre”: nombre, “edad”: edad, “estatura”: estatura}

Print(nombre)

Print(edad)

Print(estatura)

Print(notas)

Print(informacion_personal)

Este ejercicio simple te ayudará a practicar la creación de variables y la asignación de diferentes tipos de datos en Python3, así como a comprender cómo se pueden utilizar las listas y los diccionarios para almacenar y organizar información. ¡Buena suerte!

Aguarda en escript con el nombre que quieras y extención.py ejemplo. Miprimerpythob.py

Y sigue los pasos de **Cómo hacer funcionar tus escritos en python**

Los operadores

En Python3, los operadores son símbolos que se utilizan para realizar operaciones matemáticas o de comparación. Aquí te explicaré los operadores básicos y qué hacen:

Operador de suma (+): Este operador se utiliza para sumar dos valores. Por ejemplo, si tienes 2 manzanas y 3 naranjas, puedes sumarlas con el operador de suma: 2 + 3 = 5.

Operador de resta (-): Este operador se utiliza para restar dos valores. Por ejemplo, si tienes 5 juguetes y le das 2 a tu amigo, puedes restarlos con el operador de resta: 5 – 2 = 3.

Operador de multiplicación (*): Este operador se utiliza para multiplicar dos valores. Por ejemplo, si tienes 3 paquetes de dulces y cada paquete tiene 6 dulces, puedes multiplicarlos con el operador de multiplicación: 3 * 6 = 18.

Operador de división (/): Este operador se utiliza para dividir dos valores. Por ejemplo, si tienes 12 galletas y quieres compartirlas con tus amigos, puedes dividirlas con el operador de división: 12 / 3 = 4 (cada amigo recibe 4 galletas).

Operador de módulo (%): Este operador se utiliza para obtener el resto de una división. Por ejemplo, si tienes 13 globos y quieres dividirlos entre 3 amigos, obtendrás un resto de 1 con el operador de módulo: 13 % 3 = 1 (cada amigo recibe 4 globos y sobran 1).

Operador de comparación (==, ¡=, <, >, <=, >=): Estos operadores se utilizan para comparar dos valores. El operador "==" se utiliza para comprobar si dos valores son iguales, el operador "!=" se utiliza para comprobar si dos valores son diferentes, el operador "<" se utiliza para comprobar si un valor es menor que otro, el operador ">" se utiliza para comprobar si un valor es mayor que otro, el operador "<=" se utiliza para comprobar si un valor es menor o igual que otro y el operador ">=" se utiliza para comprobar si un valor es mayor o igual que otro. Por ejemplo, si tienes un pastel y un trozo de pastel, puedes comparar su tamaño con los operadores de comparación: si el tamaño del pastel es mayor que el del trozo, puedes usar el operador ">" para compararlos.

Estos son los operadores básicos de Python3

Ejemplo básico del uso de los operadores

```
# Operadores de suma y resta

Numero1 = 10

Numero2 = 5

Resultado_suma = numero1 + numero2

Resultado_resta = numero1 – numero2


Print("La suma de", numero1, "y", numero2, "es igual a", resultado_suma)

Print("La resta de", numero1, "y", numero2, "es igual a", resultado_resta)


# Operadores de multiplicación y división

Valor1 = 6

Valor2 = 3

Resultado_multiplicacion = valor1 * valor2
```

```
Resultado_division = valor1 / valor2


Print("El resultado de la multiplicación entre", valor1, "y", valor2, "es igual a",
resultado_multiplicacion)

Print("El resultado de la división entre", valor1, "y", valor2, "es igual a", resultado_division)


# Operador de módulo

Numero3 = 13

Numero4 = 3

Resultado_modulo = numero3 % numero4


Print("El resultado del módulo entre", numero3, "y", numero4, "es igual a", resultado_modulo)


# Operadores de comparación

Numero5 = 10

Numero6 = 5

Es_igual = numero5 == numero6

Es_diferente = numero5 ¡= numero6

Es_menor = numero6 < numero5

Es_mayor = numero5 > numero6

Es_menor_o_igual = numero6 <= numero5

Es_mayor_o_igual = numero5 >= numero6


Print("¿El número", numero5, "es igual noal número", numero6, "?", es_igual)

Print("¿El número", numero5, "es diferente al número", numero6, "?", es_diferente)

Print("¿El número", numero6, "es menor que el número", numero5, "?", es_menor)

Print("¿El número", numero5, "es mayor que el número", numero6, "?", es_mayor)

Print("¿El número", numero6, "es menor o igual que el número", numero5, "?",
es_menor_o_igual)
```

Print(“¿El número”, numero5, “es mayor o igual que el número”, numero6, “?”, es_mayor_o_igual)

Este código realiza operaciones con los operadores básicos de Python3: suma, resta, multiplicación, división, módulo y comparación. Además, imprime en la consola los resultados de cada operación.

Semana 1 fin

Semana 2

Las estructuras de control de flujo en Python3 permiten al programador tomar decisiones en función de ciertas condiciones. Las estructuras básicas de control de flujo son las instrucciones if, else y elif.

La instrucción if se utiliza para comprobar una condición y ejecutar un bloque de código si la condición es verdadera. Por ejemplo:

```
Edad = 18
If edad >= 18:
   Print(“Eres mayor de edad.”)
```

En este ejemplo, se comprueba si la variable edad es mayor o igual a 18. Si la condición es verdadera, se ejecutará la instrucción print() que imprimirá el mensaje “Eres mayor de edad.” En la consola.

La instrucción else se utiliza para ejecutar un bloque de código si la condición del if es falsa. Por ejemplo:

```
Edad = 15
If edad >= 18:
   Print(“Eres mayor de edad.”)
Else:
   Print(“Eres menor de edad.”)
```

En este eje mplo, si la variable edad e mayor o igual a 18 se imprimirá “Eres mayor de edad.”, pero si es menor de 18 se imprimirá “Eres menor de edad.”.

La instrucción elif se utiliza para comprobar múlti ples condiciones en una estructura if. Por ejemplo:

```
Nota = 7
If nota >= 9:
   Print(“Tienes un excelente.”)
Elif nota >= 7:
   Print(“Tienes un aprobado.”)
Else:
   Print(“Tienes un reprobado.”)
```

En este ejemplo, se comprueba si la variable nota es mayor o igual a 9. Si la condición es verdadera, se i mprimirá “Tienes un excelente.”. Si la condi ción del if es falsa, se comprueba si la variabe nota es mayor o igual a 7 en la estructura elif. Si esta condición es verdadera, se imprimirá “Tienes un aprobado.”. Si ambas condiciones son falsas, se ejecutará la instrucción print() del else, que imprimirá “Tienes un reprobado.”.

En resumen, las estructuras de control de flujo if, else y elif son herramientas fundamentales para la toma de decisiones en programas de Python3, permitiendo que el código se ejecute de manera diferente según las condiciones que se comprueben.

Es importante destacar que las estructuras de control de flujo se utilizan en conjunto con operadores lógicos, que permiten comparar valores y evaluar si se cumplen ciertas condiciones. Los operadores lógicos básicos son:

==: Compara si dos valores son iguales.

¡=: Compara si dos valores son distintos.

<: Compara si un valor es menor que otro.

>: Compara si un valor es mayor que otro.

<=: Compara si un valor es menor o igual que otro.

>=: Compara si un valor es mayor o igual que otro.

And: Compara si dos condiciones son verdaderas.

Or: Compara si al menos una de dos condiciones es verdadera.

Not: Compara si una condición es falsa.

Por ejemplo:

```
A = 10
B = 20
If a > 5 and b < 30:
   Print(“a es mayor que 5 y b es menor que 30.”)
```

En este ejemplo, se utiliza el operador lógico and para comprobar si la variable a es mayor que 5 y si la variable b es menor que 30. Si ambas condiciones son verdaderas, se ejecutará la instrucción print() que imprimirá el mensaje “a es mayor que 5 y b es menor que 30.” En la consola.

En resumen, las estructuras de control de flujo if, else y elif y los operadores lógicos son herramientas fundamentales en Python3 para la toma de decisiones en programas, permitiendo que el código se ejecute de manera diferente según las condiciones que se comprueben.

Bucles son una estructura de control de flujo que permete repetir una serie de instrucciones varias veces, dependiendo de una condición. En Python3, hay dos tipos de bucles: “wile” y “for”.

El bucle “wile” se utiliza para repetir un bloque de código mientras una condición sea verdadera. Por ejemplo

```
Frutas = [“manzana”, “naranja”, “plátano”]

For fruta in frutas:

   Print(“Me gusta comer”, fruta)
```

En este ejemplo, se utiliza un bucle “for” para iterar sobre la lista de frutas. En cada iteración del bucle, se toma una fruta de la lista y se imprime el mensaje “Me gusta comer” seguido del nombre de la fruta.

Otro ejemplo for

```
Numeros = [1, 2, 3, 4, 5]

Suma = 0

For num in numeros:

   Suma += num

Print(“La suma de los números es:”, suma)
```

En este ejemplo, se inicializa una lista llamada “numeros” con los valores del 1 al 5. Luego, se crea una variable llamada “suma” e inicializa con el valor 0.

A continuación, se utiliza un bucle “for” para iterar sobre la lista de “numeros” y sumar cada valor a la variable “suma”. Al final del bucle, se imprime el mensaje “La suma de los números es:” seguido del valor de la variable “suma”, que en este caso debería ser 15.

En resumen, los bucles “wile” y “for” son estructuras de control de flujo que permiten repetir un bloque de código varias veces. El bucle “wile” se utiliza para repetir un bloque de código mientras una condición sea verdadera, mientras que el bucle “for” se utiliza para iterar sobre una secuencia de elementos.

Las listas y tuplas en Python3 son como cajas donde se pueden guardar cosas, pero en vez de guardar juguetes o ropa, se guardan valores numéricos o palabras.

Las listas son como cajas grandes donde se pueden guardar muchos objetos de diferentes tipos, como números, palabras y otros objetos. Puedes agregar cosas a una lista o quitar cosas de ella cuando quieras. Por ejemplo, si quieres guardar el nombre de tus amigos en una lista, puedes escribir algo así:

Amigos = [“Juan”, “Ana”, “Carlos”]

En este ejemplo, “amigos” es una lista que contiene los nombres de tres amigos. Puedes acceder a los nombres de tus amigos en la lista utilizando un número que representa la posición de cada nombre. Por ejemplo, para acceder al nombre “Juan”, puedes escribir:

Print(amigos[0])

Este código imprimirá el primer elemennto de la lista, que en este caso es el nombre “Juan”.

Por otro lado, las tuplas son como cajas más pequeñas que las listas, pero no se pueden cmbiar una vez due se crean. Es decir, puedes guardar cosas en una tupla, pero no puedes agregar ni quitar cosas de ella despué s. Por ejemplo, si quieres guardar las coordenadas de un punto en una tupla, puedes escribir algo así:

```
Punto = (3, 5)
```

En este ejemplo, “punto” es una tupla que con tiene dos números: 3 y 5. Puedes acceder a estos números utilizando la misma sintaxis que con las listas:

```
Print(punto[0])
Print(punto[1])
```

Este código im primirá el primer y segundo elem entos de la tupla, que en este caso son los números 3 y 5.

En resumen, las listas y tuplas en Python3 son como cajas donde se pueden guardar cosas, pero las listas sonmás grandes y flexibles, mientras que las tuplas sonm más pequeñas pero no se pueden cambiar una vez que se crean.

En Python3, una función es un bloque de código que realiza una tarea espesífica. Se pued en definir funciones para simplificar el código y para evitar repetir el mismo código varias veces.

Para definir una función en Python3, se utiliza la siguiente sintaxis:

```
def nombre_de_la_funcion(parametro1, parametro2):
    # Código de la función
    Return resultado
```

En este ejemplo, “nombre_de_la_funcion” es el nombre que le damos a la función, y “parametro1” y “parametro2” son los parámetros que la función recibe. El código de la función se

escribe después de los parámetros, y se utiliza la palabra clave “return” para devolver un resultado.

Una vez que se ha definido una función, se puede llamar en cualquier parte del código utilizando el nombre de la función y pasando los parámetros que la función necesita. Por ejemplo, si queremos definir una función que calcule el área de un triángulo, podemos escribir algo así:

```
Def area_triangulo(base, altura):
    Area = (base * altura) / 2
    Return area
```

En este ejemplo, “area_triangulo” es el nombre de la función, y “base” y “altura” son los parámetros que la función recibe. El código de la función calcula el área del triángulo utilizando la fórmula base x altura / 2, y devuelve el resultado utilizando la palabra clave “return”.

Para llamar a esta función y calcular el área de un triángulo con una base de 4 y una altura de 3, podemos escribir lo siguiente:

```
Mi_area = area_triangulo(4, 3)
Print(mi_area)
```

Este código llama a la función “area_triangulo” con los parámetros 4 y 3, y asigna el resultado de la función a la variable "mi_area". Luego, imprime el valor de "mi_area", que debería ser 6.

En resumen, una función en Python3 es un bloque de código que realiza una tarea específica. Se definen utilizando la sintaxis "def", se llaman en cualquier parte del código pasando los parámetros necesarios, y pueden devolver un resultado utilizando la palabra clave "return". Las funciones son muy útiles para simplificar el código y para evitar repetir el mismo código varias veces.

Semana 2 fin

Semana 3 inicio

Módulos: importación y uso

Ejercicio práctico

En Python, un módulo es un archivo que contiene código Python, que puede ser importado en otros archivos para reutilizar ese código.

Para importar un módulo en Python3, se utiliza la palabra clave “import”, seguida del nombre del módulo. Por ejemplo, para importar el módulo “math” en Python3, se escribe lo siguiente:

```
Import math
```

Una vez que se ha importado un módulo, se pueden utilizar las funciones y variables definidas en ese módulo utilizando la sintaxis “nombre_del_modulo.nombre_de_la_funcion”. Por ejemplo, si queremos utilizar la función “sqrt” del módulo “math” para calcular la raíz cuadrada de un número, podemos escribir lo siguiente:

```
Import math

Numero = 25

Raiz = math.sqrt(numero)

Print(raiz)
```

Este código importa el módulo “math” y utiliza la función “sqrt” para calcular la raíz cuadrada del número 25. Luego, imprime el resultado, que debería ser 5.

También es posible importar funciones y variables específicas de un módulo utilizando la sintaxis “from nombre_del_modulo import nombre_de_la_funcion”. Por ejemplo, si solo queremos importar la función “sqrt” del módulo “math”, podemos escribir lo siguiente:

```
From math import sqrt

Numero = 25

Raiz = sqrt(numero)
```

Print(raiz)

Este código importa solo la función “sqrt” del módulo “math” y la utiliza para calcular la raíz cuadrada del número 25. Luego, imprime el resultado, que debería ser 5.

Un ejemplo práctico de cómo utilizar un módulo en Python3 sería el siguiente ejercicio: crear un programa que pida al usuario un número y calcule su factorial utilizando la función “factorial” del módulo “math”. El código para este ejercicio sería el siguiente:

Import math

Numero = int(input(“Ingresa un número: “))

Factorial = math.factorial(numero)

Print(“El factorial de”, numero, “es”, factorial)

Este código importa el módulo “math”, pide al usuario un número utilizando la función “input”, calcula su factorial utilizando la función “factorial” del módulo “math”, y luego imprime el resultado en pantalla**.**

Fin semana 3

Ejercicios. Calculadora simple

Con lo aprendido intenten crear calculadoras simples como sumar y restar multiplicar inténtelo una por una

Ejemplo calculadora simple de pura suma

.# Pedimos al usuario que ingrese los números

Numero1 = float(input(“Ingrese el primer número: “))

Numero2 = float(input(“Ingrese el segundo número: “))

Realizamos la suma

Resultado = numero1 + numero2

Imprimimos el resultado

Print(“El resultado de la suma es:”, resultado)