Aquí tienes los pasos para crear la carpeta, abrirla en Visual Studio Code, y el código para realizar el ejercicio:

**1. Crear la carpeta y abrir Visual Studio Code**

* En tu computadora, crea una nueva carpeta llamada Coderhouse en la ubicación que prefieras.
* Abre Visual Studio Code.
* En Visual Studio Code, haz clic en **File** (Archivo) > **Open Folder...** (Abrir carpeta...), navega hasta la carpeta Coderhouse que creaste y ábrela.

**2. Crear el archivo clase\_01.py**

* Una vez que tengas la carpeta abierta en Visual Studio Code, haz clic en el botón **New File** (Nuevo archivo) en la barra lateral o usa el atajo de teclado **Ctrl+N** (Windows) o **Cmd+N** (Mac).
* Nombra el archivo como clase\_01.py y presiona **Enter**.

**3. Escribir el código**

Ahora, copia y pega el siguiente código en el archivo clase\_01.py:

# Mensaje de bienvenida

print("¡Bienvenido a la clase de programación!")

# Solicitar nombre al usuario

nombre = input("Por favor, ingresa tu nombre: ")

# Solicitar edad al usuario

edad = input("Por favor, ingresa tu edad: ")

# Mostrar el mensaje final

print(f"{nombre} tiene {edad} años.")

=========================================================

Aquí tienes los pasos para crear la carpeta, abrirla en Visual Studio Code, y el código para realizar el ejercicio:

**1. Crear la carpeta y abrir Visual Studio Code**

* En tu computadora, crea una nueva carpeta llamada Coderhouse en la ubicación que prefieras.
* Abre Visual Studio Code.
* En Visual Studio Code, haz clic en **File** (Archivo) > **Open Folder...** (Abrir carpeta...), navega hasta la carpeta Coderhouse que creaste y ábrela.

**2. Crear el archivo clase\_01.py**

* Una vez que tengas la carpeta abierta en Visual Studio Code, haz clic en el botón **New File** (Nuevo archivo) en la barra lateral o usa el atajo de teclado **Ctrl+N** (Windows) o **Cmd+N** (Mac).
* Nombra el archivo como clase\_01.py y presiona **Enter**.

**3. Escribir el código**

Ahora, copia y pega el siguiente código en el archivo clase\_01.py:

python

Copiar código

# Mensaje de bienvenida

print("¡Bienvenido a la clase de programación!")

# Solicitar nombre al usuario

nombre = input("Por favor, ingresa tu nombre: ")

# Solicitar edad al usuario

edad = input("Por favor, ingresa tu edad: ")

# Mostrar el mensaje final

print(f"{nombre} tiene {edad} años.")

**4. Guardar y ejecutar el archivo**

* Guarda el archivo con **Ctrl+S** (Windows) o **Cmd+S** (Mac).
* Para ejecutar el archivo, abre la terminal integrada en Visual Studio Code (usa **Ctrl+`**).
* En la terminal, asegúrate de que estás en la carpeta Coderhouse, y ejecuta el archivo con el comando:

python clase\_01.py

* Visual Studio Code mostrará el mensaje de bienvenida, te pedirá tu nombre y edad, y luego mostrará el mensaje con tu nombre y edad.
* Partiendo de: cadena\_1 = "versátil" cadena\_2 = "Python" cadena\_3 = "es un lenguaje" cadena\_4 = "de programación" Crea una nueva cadena con el siguiente contenido: "Python es un lenguaje de programación versátil" Nota: Utiliza el operador + para unir las cadenas (concatenación)

Para crear la nueva cadena uniendo las partes con el operador +, sigue estos pasos:

# Definición de las cadenas

cadena\_1 = "versátil"

cadena\_2 = "Python"

cadena\_3 = "es un lenguaje"

cadena\_4 = "de programación"

# Concatenación de las cadenas

nueva\_cadena = cadena\_2 + " " + cadena\_3 + " " + cadena\_4 + " " + cadena\_1

# Mostrar el resultado

print(nueva\_cadena)

Explicación:

Concatena las cadenas cadena\_2, cadena\_3, cadena\_4, y cadena\_1 con el operador + y añade espacios " " entre ellas.

El resultado será: "Python es un lenguaje de programación versátil".

=====================================================

Descripción de la actividad. En nuestro trabajo, nos solicitan desarrollar un programa que permita calcular el promedio final de los equipos de futbol en un torneo. Para ello, debemos considerar tres aspectos: Jugaron 20 partidos durante el torneo. Los partidos poseen diferente puntaje según el resultado, los partidos ganados 3 puntos, partido empatado 1 punto, partido perdido 0 puntos. El promedio final resulta de la cantidad de puntos totales divididos la cantidad de partidos Descripción de la actividad. La cantidad de partidos que debemos considerar a un equipo para el ejemplo se detallan a continuación: partidos\_ganados 8 partido\_empatados 7 partido\_perdidos 5 Importante: Cada una de las cantidades de partidos ganados, empatados o perdidos debe solicitarse al usuario utilizando la función input().

Aquí tienes el programa para calcular el promedio final de puntos de un equipo de fútbol basándote en los resultados de sus partidos:

# Solicitar al usuario la cantidad de partidos ganados, empatados y perdidos

partidos\_ganados = int(input("Ingrese la cantidad de partidos ganados: "))

partidos\_empatados = int(input("Ingrese la cantidad de partidos empatados: "))

partidos\_perdidos = int(input("Ingrese la cantidad de partidos perdidos: "))

# Calcular la cantidad total de puntos obtenidos

puntos\_totales = (partidos\_ganados \* 3) + (partidos\_empatados \* 1) + (partidos\_perdidos \* 0)

# Calcular el promedio final de puntos

total\_partidos = partidos\_ganados + partidos\_empatados + partidos\_perdidos

promedio\_final = puntos\_totales / total\_partidos

# Mostrar el promedio final

print(f"El promedio final de puntos es: {promedio\_final}")

Explicación

El programa solicita al usuario que ingrese la cantidad de partidos ganados, empatados y perdidos.

Calcula los puntos totales obtenidos utilizando el sistema de puntaje:

Ganados: 3 puntos

Empatados: 1 punto

Perdidos: 0 puntos

Calcula el promedio final dividiendo los puntos totales por el total de partidos.

Finalmente, muestra el promedio de puntos del equipo.

Nota: Asegúrate de que la cantidad total de partidos (sumando ganados, empatados y perdidos) coincida con la cantidad de partidos que se jugaron en el torneo (20 en este caso) para obtener un resultado correcto.

===================================================

Se tiene una cadena de texto, pero al revés. Al parecer contiene el nombre de un alumno, la nota de un examen y la materia. cadena = "acitametaM ,5.8 ,otipeP ordeP" 1. Dar vuelta la cadena y asignarla a una variable llamada cadena\_volteada. Para devolver una cadena dada vuelta se usa el tercer índice negativo con slicing: cadena[::-1] 2. Extraer nombre y apellido, almacenarlo en una variable llamada nombre\_alumno 3. Extraer la nota y almacenarla en una variable llamada nota. 4. Extraer la materia y almacenarla en una variable llamada materia. 5. Mostrar por pantalla la siguiente estructura, usando la concantenación de cadenas: NOMBRE APELLIDO ha sacado un NOTA en MATERIA

Aquí tienes el programa para procesar la cadena de texto al revés y extraer la información necesaria:

# Cadena original

cadena = "acitametaM ,5.8 ,otipeP ordeP"

# 1. Dar vuelta la cadena

cadena\_volteada = cadena[::-1]

# 2. Extraer el nombre y apellido

nombre\_alumno = cadena\_volteada.split(", ")[0]

# 3. Extraer la nota

nota = cadena\_volteada.split(", ")[1]

# 4. Extraer la materia

materia = cadena\_volteada.split(", ")[2]

# 5. Mostrar el mensaje en el formato solicitado

mensaje = nombre\_alumno + " ha sacado un " + nota + " en " + materia

print(mensaje)

Explicación

La cadena se da vuelta utilizando cadena[::-1] y se guarda en cadena\_volteada.

La variable nombre\_alumno contiene el primer elemento de la cadena volteada, que se extrae con split(", ").

La variable nota almacena el segundo elemento de la cadena volteada.

La variable materia guarda el tercer elemento.

Finalmente, se muestra el mensaje concatenando las variables en el formato solicitado.

Al ejecutar el programa, el resultado en pantalla será:

Copiar código

Pedro Pepito ha sacado un 8.5 en Matematica

**1.1 ¿Qué es Python?**

**Introducción a la Programación**

**Breve Historia de la Programación**

**La historia de la programación está fuertemente vinculada a la aparición y crecimiento exponencial de los computadores. En sus inicios, programar implicaba el uso de código binario, con cadenas de 0s y 1s, el lenguaje que entiende directamente la computadora, lo cual era sumamente complejo. Más tarde, se creó el lenguaje ensamblador, que en vez de números utilizaba letras, algo más fácil de recordar.**

**Finalmente, surgieron los lenguajes de alto nivel, que suelen utilizar términos en inglés para dictar las órdenes a seguir, actuando como un intermediario entre el lenguaje de la máquina (binario) y el “código fuente” (procesado por un compilador o intérprete).**

**Lenguajes de Programación Más Conocidos**

**Lenguajes de Frontend Los lenguajes de frontend son aquellos que interactúan directamente con el usuario en la interfaz del cliente. Los principales lenguajes de frontend incluyen:**

**JavaScript**

**HTML y CSS (aunque HTML y CSS no son lenguajes de programación, son esenciales para la estructura y el estilo de los sitios web)**

**Lenguajes de Backend Los lenguajes de backend son aquellos que gestionan la lógica del servidor, bases de datos y las interacciones del servidor. Los principales lenguajes de backend incluyen:**

**Python, Java, Javascript con Node Js, Ruby, PHP**

**Diferencia entre Lenguajes de Frontend y Backend**

**Frontend El frontend se refiere a la parte de un sitio web que interactúa directamente con los usuarios, en los dispositivos que utilicen para consumir un servicio web (normalmente un navegador o app movil), también conocido como el lado del cliente. Involucra la maquetación lógica visible de un sitio web y es lo que los usuarios pueden ver e interactuar.**

**Backend El backend se refiere a la parte del sitio web que conecta con la base de datos y el servidor, también conocido como el lado del servidor. Involucra el desarrollo de APIs del lado del servidor y bases de datos, y se encarga de conectar la base de datos con la interfaz de la web o aplicación.**

**Tareas Involucradas en el Backend**

**Creación, integración y mantenimiento de bases de datos**

**Comunicación del servidor, aplicación y base de datos**

**Creación, mantenimiento, testeo y debugging del backend**

**Creación de backups de bases de datos**

**Conversión y devolución de datos**

**Mantenimiento del sistema y deploy**

**Integración con APIs**

**Seguridad**

**Introducción a Python**

**Python como Lenguaje de Backend Python es un lenguaje de programación de código abierto, orientado a objetos, simple y fácil de entender. Tiene una sintaxis sencilla y cuenta con una vasta biblioteca de herramientas, lo que lo hace un lenguaje único. Python es un lenguaje interpretado, con tipado dinámico, multiparadigma y multiplataforma. En este curso, nos enfocaremos en Python como un lenguaje de backend.**

**Ventajas y Desventajas de Python**

**Ventajas:**

**Fácil de aprender, Fácil de usar, Orientado a objetos, Software libre, Portable, Multiplataforma, Poderoso**

**Desventajas:**

**Lentitud en multithread, Lentitud de ejecución, Tipado poco estricto**

**Python es ampliamente utilizado en diversas áreas como inteligencia artificial, big data, data science, frameworks de pruebas y desarrollo web. En este curso, nos centraremos en la parte web del desarrollo.**

**Conclusión**

**La programación es un proceso fundamental para crear aplicaciones y software, utilizando diversos lenguajes que permiten materializar conceptos y proyectos. Python, como uno de los lenguajes más versátiles y accesibles, se convierte en una herramienta esencial para el desarrollo backend, ofreciendo numerosas ventajas para programadores de todos los niveles.**

**1.2 Primeros Pasos con Python y variables**

**Primeros Pasos con Python y variables**

**Se trabajara sobre cómo escribir y ejecutar su primer programa en Python utilizando Google Colab. Exploraremos conceptos básicos como la creación de variables y el uso de la función**

**print.**

**Hemos escrito y ejecutado nuestro primer programa en Python utilizando Google Colab, y ya conocen cómo crear variables y utilizar la función**

**print**

**para mostrar resultados en la pantalla. Estos son los cimientos sobre los que construiremos conocimientos más avanzados en los próximos videos. No olviden practicar lo aprendido hoy y explorar por su cuenta. Nos vemos en el próximo video, donde profundizaremos aún más en el mundo de Python.**

**Variables en Python**

**Las variables en Python son etiquetas que referencian datos almacenados en la memoria del programa. A diferencia de otros lenguajes de programación donde las variables son cajas que contienen datos, en Python, las variables son simplemente nombres asignados a objetos que contienen los datos.**

**Definición y uso de variables en Python:**

1. **Definir una variable: Para crear una variable en Python, simplemente asigna un valor a un nombre utilizando el operador de asignación (=).**

**python nombre = "Juan" edad = 25**

**En este ejemplo,**

**nombre**

**es una variable que contiene la cadena de texto "Juan", y**

**edad**

**es una variable que contiene el número entero 25.**

1. **Reglas para nombrar variables:**
   * **Los nombres de las variables deben comenzar con una letra (a-z, A-Z) o un guion bajo (\_).**
   * **El resto del nombre puede incluir letras, números (0-9) y guiones bajos.**
   * **Python distingue entre mayúsculas y minúsculas, por lo que edad y Edad son variables diferentes.**
   * **Es una buena práctica usar nombres de variables descriptivos y utilizar la convención snake\_case, donde las palabras están separadas por guiones bajos.**

**Copiar**

**fecha\_de\_nacimiento = "27 de octubre de 1997"**

1. **Asignación múltiple: Python permite asignar valores a múltiples variables en una sola línea.**

**python a, b, c = 1, 2, 3**

1. **Intercambiar valores: Puedes intercambiar los valores de dos variables sin usar una variable temporal.**

**python a, b = b, a**

1. **Entrada del usuario: Utiliza la función**

**input()**

**para obtener datos del usuario y almacenarlos en una variable.**

**python nombre = input("Ingrese su nombre: ")**

1. **Buenas prácticas para nombrar variables:**
   * **Usa nombres claros y significativos que describan el contenido o el propósito de la variable.**
   * **Evita usar palabras reservadas de Python como nombres de variables (por ejemplo, print, for, while).**

**Copiar**

**contador\_de\_usuarios = 0**

**Ejemplos prácticos:**

1. **Definir y usar una variable:**

**python mensaje = "Hola, mundo!" print(mensaje)**

1. **Actualizar el valor de una variable:**

**python x = 5 x = x + 2 print(x) # Imprime 7**

1. **Operaciones con variables:**

**python a = 10 b = 20 suma = a + b print(suma) # Imprime 30**

1. **Uso de variables con diferentes tipos de datos:**

**python nombre = "Ana" edad = 30 altura = 1.75 es\_estudiante = True**

**Resumen de buenas prácticas:**

* **Elige nombres de variables que sean descriptivos y fáciles de entender.**
* **Usa la convención snake\_case**

**para separar palabras en los nombres de variables.**

* **Evita el uso de palabras reservadas como nombres de variables.**
* **Asigna valores claros y específicos a las variables para mejorar la legibilidad del código.**

**Siguiendo estas recomendaciones, podrás definir y utilizar variables en Python de manera efectiva y clara, facilitando la escritura y el mantenimiento de tu código.**

**1.4 Introducción a Números y Cadenas de Caracteres**

**Números**

**Los números en Python se dividen en varios tipos:**

* **Enteros (int): Son números sin parte decimal, tanto positivos como negativos.**
* **Reales (float): Son números con decimales.**
* **Complejos (complex): Son números que incluyen una parte real y una imaginaria.**

**Las operaciones básicas que se pueden realizar con números incluyen:**

* **Suma (+): Adición de dos números.**
* **Resta (-): Sustracción de un número de otro.**
* **Multiplicación (\*): Producto de dos números.**
* **División (/): Cociente de dos números.**
* **División entera (//): Cociente entero de la división de dos números.**
* **Módulo (%): Resto de la división de dos números.**
* **Potenciación (): Un número elevado a la potencia de otro.**

**También se explorará la precedencia de operadores, que es el orden en que se evalúan las operaciones en una expresión matemática.**

**Cadenas de Caracteres**

**Las cadenas de caracteres son secuencias de caracteres que se utilizan para representar texto. En Python, se pueden definir utilizando comillas simples (' ') o dobles (" ").**

**Algunas operaciones básicas con cadenas incluyen:**

* **Concatenación (+): Unión de dos o más cadenas.**
* **Repetición (\*): Repetición de una cadena un número determinado de veces.**
* **Indexación: Acceso a caracteres específicos dentro de una cadena utilizando índices.**
* **Slicing: Extracción de subcadenas utilizando notaciones de inicio y fin de índices.**
* **Longitud (len): Determinación del número de caracteres en una cadena.**

**Se cubrirán también funciones integradas de Python para manipular cadenas, como:**

* **upper(): Convierte todos los caracteres de una cadena a mayúsculas.**
* **lower(): Convierte todos los caracteres de una cadena a minúsculas.**
* **replace(): Reemplaza una subcadena por otra dentro de una cadena.**
* **split(): Divide una cadena en una lista de subcadenas basadas en un delimitador.**

**En este video, vamos a dar nuestros primeros pasos con números en Python. Les mostraré cómo trabajar con diferentes tipos de números, como enteros, flotantes y complejos, y realizaremos algunas operaciones básicas con ellos. Acompáñenme en esta demostración práctica y descubran lo fácil y versátil que puede ser manejar números en Python.**

**Primeros Pasos con Números**

**Estos conocimientos son esenciales para cualquier tipo de programación que realicemos en el futuro.**

**1.5 Tipos de Número y operaciones numéricas en Python**

**Tipos de Número en Python**

**En Python, existen tres tipos principales de números que podemos utilizar en nuestras operaciones y cálculos: enteros, números reales (de punto flotante) y números complejos. A continuación, describimos cada uno de ellos con ejemplos:**

**Enteros**

**Los números enteros son aquellos que no tienen decimales. Pueden ser positivos, negativos o cero. En Python, los enteros se representan con el tipo**

**int**

**.**

**Ejemplos:**

**Copiar**

**1**

**-10**

**0**

**523**

**-817**

**Python 3 permite que los números enteros sean tan grandes como la memoria del ordenador lo permita, sin un límite máximo.**

**Números Reales (Punto Flotante)**

**Los números reales, también conocidos como números de punto flotante, son aquellos que tienen una parte decimal. En Python, se representan con el tipo**

**float**

**. Además, Python ofrece el tipo**

**Decimal**

**para representar fracciones de forma más precisa.**

**Ejemplos:**

**Copiar**

**0.5**

**-3.14**

**2.71828**

**-0.0001**

**Números Complejos**

**Los números complejos son una extensión del sistema numérico que permite la existencia de raíces cuadradas de números negativos. Un número complejo tiene una parte real y una parte imaginaria. En Python, se representan con el tipo**

**complex**

**, utilizando la letra**

**j**

**para denotar la parte imaginaria.**

**Ejemplos:**

**Copiar**

**1+2j**

**-3+4j**

**0-1j**

**5j**

**Los números complejos son útiles en diversas ramas de la matemática y la física, especialmente en la teoría de circuitos eléctricos y la mecánica cuántica.**

**Ejemplos de Uso**

**A continuación, se presentan algunos ejemplos prácticos de cómo se pueden utilizar estos tipos de números en Python:**

**Copiar**

**# Enteros**

**a = 5**

**b = -3**

**print(a + b) # Resultado: 2**

**# Números de punto flotante**

**x = 3.14**

**y = 2.71**

**print(x \* y) # Resultado: 8.5094**

**# Números complejos**

**z1 = 1 + 2j**

**z2 = 3 - 4j**

**print(z1 + z2) # Resultado: (4-2j)**

**Cada tipo de número tiene su propio conjunto de operaciones y propiedades, lo que los hace adecuados para diferentes tipos de problemas y cálculos. Con esta variedad de tipos numéricos, Python proporciona una gran flexibilidad para trabajar con números en diversas aplicaciones.**

**Operaciones Numéricas en Python**

**Python proporciona una variedad de operadores aritméticos que se utilizan para realizar cálculos matemáticos. A continuación, se detallan las operaciones aritméticas básicas y avanzadas, con ejemplos para cada una.**

**Suma**

**La suma se realiza utilizando el operador**

**+**

**. Este operador se utiliza para sumar dos números.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 5**

**resultado = a + b**

**print(resultado) # Resultado: 15**

**Resta**

**La resta se realiza utilizando el operador**

**-**

**. Este operador se utiliza para restar un número de otro.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 5**

**resultado = a - b**

**print(resultado) # Resultado: 5**

**Multiplicación**

**La multiplicación se realiza utilizando el operador**

**\***

**. Este operador se utiliza para multiplicar dos números.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 5**

**resultado = a \* b**

**print(resultado) # Resultado: 50**

**División**

**La división se realiza utilizando el operador**

**/**

**. Este operador se utiliza para dividir un número por otro.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 5**

**resultado = a / b**

**print(resultado) # Resultado: 2.0**

**División Entera**

**La división entera se realiza utilizando el operador**

**//**

**. Este operador divide un número por otro y devuelve la parte entera del resultado.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 3**

**resultado = a // b**

**print(resultado) # Resultado: 3**

**Módulo**

**El operador de módulo**

**%**

**devuelve el resto de la división de un número por otro.**

**Ejemplo:**

**Copiar**

**a = 10**

**b = 3**

**resultado = a % b**

**print(resultado) # Resultado: 1**

**Potencia**

**La potencia se realiza utilizando el operador**

**\*\***

**. Este operador se utiliza para elevar un número a la potencia de otro.**

**Ejemplo:**

**Copiar**

**a = 2**

**b = 3**

**resultado = a \*\* b**

**print(resultado) # Resultado: 8**

**Precedencia de los Operadores**

**La precedencia de los operadores determina el orden en que se evalúan las operaciones en una expresión. La siguiente es la jerarquía de precedencia de los operadores aritméticos en Python, de mayor a menor:**

1. **()**

**: Paréntesis**

1. **\***

**: Potencia**

1. **`,**

**/**

**,**

**//**

**,**

**%`: Multiplicación, División, División Entera, Módulo**

1. **+**

**, ``: Suma, Resta**

**Ejemplo:**

**Copiar**

**resultado = (2 + 3) \* 4 \*\* 2 / (5 - 1) + 6**

**print(resultado) # Resultado: 26.0**

**En este ejemplo, la expresión dentro de los paréntesis se evalúa primero, seguida de la potencia, la multiplicación/división, y finalmente la suma.**

**Ejemplos Completos**

**A continuación, se presentan algunos ejemplos completos que ilustran el uso de varias operaciones aritméticas juntas:**

**Copiar**

**# Suma, Resta, Multiplicación, y División**

**a = 10**

**b = 5**

**c = 2**

**resultado = (a + b - c) \* c / b**

**print(resultado) # Resultado: 6.0**

**# Potencia y Módulo**

**a = 4**

**b = 2**

**c = 3**

**resultado = (a \*\* b) % c**

**print(resultado) # Resultado: 1**

**Estas operaciones son fundamentales para realizar cálculos y manipular datos numéricos en Python. Con una comprensión sólida de estos operadores, puedes desarrollar programas más complejos y eficientes.**

**1.6 Cadenas de Texto en Python**

**Cadenas de Texto en Python**

**En Python, las cadenas de texto (strings) son secuencias de caracteres que se utilizan para representar texto. Las cadenas se delimitan utilizando comillas simples (**

**' '**

**) o dobles (**

**" "**

**). A continuación, se explica cómo crear, manipular y concatenar cadenas de texto en Python, con ejemplos prácticos.**

**Creación de Cadenas de Texto**

**Para crear una cadena de texto en Python, se puede utilizar comillas simples o dobles. Ambas formas son equivalentes y permiten incluir caracteres especiales dentro de la cadena.**

**Ejemplos:**

**Copiar**

**cadena1 = 'Hola, mundo!'**

**cadena2 = "Python es increíble"**

**Si se necesita incluir comillas dentro de la cadena, se pueden usar comillas opuestas o el carácter de escape (**

**\\**

**).**

**Ejemplos:**

**Copiar**

**cadena3 = "Esto es una 'cadena' con comillas simples dentro"**

**cadena4 = 'Esto es una "cadena" con comillas dobles dentro'**

**cadena5 = "Esto es una \\"cadena\\" con comillas dobles usando el carácter de escape"**

**Manipulación de Cadenas de Texto**

**Python ofrece varias formas de manipular cadenas de texto, como la indexación, el slicing y el uso de métodos de cadena.**

**Indexación**

**Cada carácter en una cadena tiene una posición específica (índice), que comienza desde 0 para el primer carácter y puede ser negativo para contar desde el final.**

**Ejemplos:**

**Copiar**

**cadena = "Python"**

**print(cadena[0]) # Resultado: 'P'**

**print(cadena[-1]) # Resultado: 'n'**

**Slicing**

**El slicing permite extraer una parte específica de una cadena utilizando la notación**

**[inicio:fin:paso]**

**.**

**Ejemplos:**

**Copiar**

**cadena = "Python"**

**print(cadena[0:2]) # Resultado: 'Py'**

**print(cadena[2:5]) # Resultado: 'tho'**

**print(cadena[::2]) # Resultado: 'Pto'**

**Métodos de Cadenas**

**Python proporciona varios métodos integrados para trabajar con cadenas. Algunos de los métodos más comunes incluyen:**

* **lower()**

**: Convierte todos los caracteres de la cadena a minúsculas.**

* **upper()**

**: Convierte todos los caracteres de la cadena a mayúsculas.**

* **replace(old, new)**

**: Reemplaza todas las apariciones de**

**old**

**con**

**new**

**.**

* **split(delimiter)**

**: Divide la cadena en una lista de subcadenas, utilizando el delimitador especificado.**

* **strip()**

**: Elimina los espacios en blanco al principio y al final de la cadena.**

**Ejemplos:**

**Copiar**

**cadena = " Python es Increíble "**

**print(cadena.lower()) # Resultado: ' python es increíble '**

**print(cadena.upper()) # Resultado: ' PYTHON ES INCREÍBLE '**

**print(cadena.replace("Increíble", "genial")) # Resultado: ' Python es genial '**

**print(cadena.split()) # Resultado: ['Python', 'es', 'Increíble']**

**print(cadena.strip()) # Resultado: 'Python es Increíble'**

**Concatenación de Cadenas**

**La concatenación de cadenas implica unir dos o más cadenas en una sola. Esto se puede hacer utilizando el operador**

**+**

**o el método**

**join()**

**.**

**Ejemplos:**

**Copiar**

**cadena1 = "Python"**

**cadena2 = "es genial"**

**resultado = cadena1 + " " + cadena2**

**print(resultado) # Resultado: 'Python es genial'**

**partes = ["Python", "es", "genial"]**

**resultado = " ".join(partes)**

**print(resultado) # Resultado: 'Python es genial'**

**Ejemplos Prácticos**

**A continuación, se presentan algunos ejemplos prácticos que ilustran cómo trabajar con cadenas de texto en Python.**

**Copiar**

**# Crear una cadena**

**saludo = "¡Hola, mundo!"**

**# Manipulación de cadenas**

**print(saludo.lower()) # Resultado: '¡hola, mundo!'**

**print(saludo.upper()) # Resultado: '¡HOLA, MUNDO!'**

**# Indexación y slicing**

**print(saludo[0]) # Resultado: '¡'**

**print(saludo[-1]) # Resultado: '!'**

**print(saludo[1:5]) # Resultado: 'Hola'**

**# Concatenación de cadenas**

**nombre = "Python"**

**frase = nombre + " es genial"**

**print(frase) # Resultado: 'Python es genial'**

**# Uso de métodos de cadena**

**frase\_modificada = frase.replace("genial", "asombroso")**

**print(frase\_modificada) # Resultado: 'Python es asombroso'**

**Estas técnicas y ejemplos proporcionan una base sólida para trabajar con cadenas de texto en Python, permitiendo realizar una amplia variedad de operaciones y manipulaciones.**

**Instalación de Python**

**¡Atención!**

**Recuerda instalar Python y tener configurado Google Colab.**

[**https://files.coderhouse.com/coder-v3/python/Tutorial*Python*-\_Google\_Colab.pdf**](https://files.coderhouse.com/coder-v3/python/Tutorial_Python_-_Google_Colab.pdf)

**======================================================================0**

**1.7 Cierre de la unidad**

**1.8 Actividad práctica**

**PARA RECORDAR**

**En un promedio pesado o ponderado no todos los valores tienen el mismo “peso” o valor.**

**El promedio entre 3 y 10 es:    (1.3 + 1.10) / 2, este es el promedio tradicional donde todos los valores tienen un peso de 1.**

**Promedio pesado entre 3 y 10 es:   (13.3 + 2.10) / 15, aquí vemos que el peso de 3 es 13, y el peso del 10 es 2, por lo que el 3 es más importante, se divide por la suma de los pesos.**

**Mi primer programa en Python**

**Consigna**

* **Trabajas en Coderhouse y te piden crear un programa que calcule la nota final de estudiantes del curso de Python. La nota final se calcula basándonos en tres notas previas de las cuales, cada una corresponde un porcentaje distinto de la nota final. Los porcentajes se detallan a continuación:**

**Los porcentajes asociados que debemos considerar de cada nota se detallan a continuación:**

* **nota\_1 cuenta como el 20% de la nota final**
* **nota\_2 cuenta como el 30% de la nota final**
* **nota\_3 cuenta como el 50% de la nota final**

**Aspectos a incluir**

* **Tener en cuenta los temas vistos en la clase 1: números, print, input, variables, operaciones matemáticas, cadena de texto.**
* **Los datos deben guardarse en variables y deben ser dinámicos por medio de input.**

**¿Quieres saber más?**

**Te compartimos el acceso al Repositorio de Github donde encontrarás todo el material complementario y scripts de la clase.**

* [**Repositorio Python**](https://github.com/CoderContenidos/Python-V4)
* [**Extreme Programming Qué es y cómo usarlo**](https://openwebinars.net/blog/extreme-programming-que-es-y-como-aplicarlo/)**| Marvin López Mendoza**
* [**Pipenv**](https://pipenv-es.readthedocs.io/es/latest/#:~:text=%C2%B6&text=Autom%C3%A1ticamente%20crea%20y%20maneja%20un,genera%20el%20m%C3%A1s%20importante%20Pipfile.)**| Ken Reitz**
* [**Variables Numéricas**](https://www.youtube.com/watch?v=h1mBPIj71J0&list=PL124RB0umOHRmFyl5Q3aVSW7TjOSNu9hl&index=2&ab_channel=Nicol%C3%A1sPerez)**| Nicolás Perez**
* [**Operaciones**](https://www.youtube.com/watch?v=XM_RK6ga2fs&list=PL124RB0umOHRmFyl5Q3aVSW7TjOSNu9hl&index=3&ab_channel=Nicol%C3%A1sPerez)**| Nicolás Perez**
* [**Variables de Texto**](https://www.youtube.com/watch?v=2NnSGwWyTxg&list=PL124RB0umOHRmFyl5Q3aVSW7TjOSNu9hl&index=4&ab_channel=Nicol%C3%A1sPerez)**| Nicolás Perez**

**Google Colab**

* [**Tutorial Google Colab**](https://docs.google.com/presentation/d/1Fhq72xjyXXO5jYP6kZIZZ8HyGgTTsiq3/edit?usp=sharing&ouid=111108720142997343839&rtpof=true&sd=true)
* [**Ejemplos de clase**](https://colab.research.google.com/drive/1mfNY8UNTnr_ImJ9xSOcAy4uwZFbRn6DV?usp=sharing)

**Te compartimos una**[**Guía de ejercicios complementarios**](https://docs.google.com/presentation/d/14L6TrRYUq1KZkV87zGtqK7XjteQdx2uL/edit?usp=sharing&ouid=111108720142997343839&rtpof=true&sd=true)**. Ten presente que estas actividades no se entregan ni tienen corrección.**