

# Entornos de desarrollo y codificación

## Breve descripción:

Este componente formativo está orientado a entregar los conceptos y conocimientos en instalación y puesta en marcha de un entorno de trabajo. También en el dar a conocer los estándares de codificación, manejo de variables, funciones, estructuras de control, clases, objetos y gestión de proyectos en Python.

---

**Mayo 2023**

## Tabla de contenido

Introducción.....	3
1. ¿Qué es un editor de código? .....	5
2. Diferentes editores de código .....	7
3. Proceso de instalación de Visual Studio Code.....	10
4. Preparar entorno de desarrollo virtual .....	12
5. Manejo y control de versiones .....	19
6. Estándares para la codificación de software.....	25
7. Creación y manejo de variables, estructuras de control, funciones, clases y objetos .....	30
8. Gestión de proyectos en Python.....	44
Síntesis .....	48
Material complementario .....	49
Glosario.....	50
Referencias bibliográficas.....	51
Créditos.....	52

## Introducción

Bienvenidos a este componente formativo en el cual se abordarán temáticas de suma importancia para el desarrollo de nuestro programa, sus contenidos están orientados a sumergirnos en un mar de conocimiento y aprendizaje sobre los conceptos y conocimientos de instalación y puesta en marcha de un entorno de trabajo en el lenguaje de programación Python. Se invita a ver el siguiente video:

### Video 1. Entornos de desarrollo y codificación



[Enlace de reproducción del video](#)

**Síntesis del video: entornos de desarrollo y codificación**

Los entornos de desarrollo y codificación son el punto de partida en la programación y en el desarrollo de aplicaciones web en Python. Para realizar correctamente el proceso, es importante seguir todos los pasos y procedimientos necesarios.

Durante este desarrollo formativo, se instalará un editor de código, el cual es indispensable para programar los distintos módulos del sistema, así como la configuración de un entorno virtual para separar el proyecto de otras dependencias en el entorno de trabajo. También es fundamental conocer el manejo y control de versiones para hacer un seguimiento específico del proyecto y mantener un estándar de codificación.

Para la redacción de código fuente, es esencial desarrollar y apropiarse de los conceptos y habilidades necesarios para programar en Python, ya que los programas escritos en este lenguaje tienen características específicas.

En este curso se abordará el manejo de variables, constantes, funciones, estructuras de control, clases y objetos, los cuales son fundamentales para la construcción de programas de calidad. Asimismo, se crearán carpetas y directorios para organizar el proyecto de manera adecuada y eficiente.

## 1. ¿Qué es un editor de código?

Un editor de código es una herramienta que permite realizar el proceso de creación de un archivo de código fuente donde se colocan todas las instrucciones necesarias para que un programa pueda ser ejecutado de manera correcta.

Los editores de código actualmente permiten la integración de diversos lenguajes de programación, lo cual facilita tanto la escritura como la modificación de código fuente. Adicionalmente, los editores de código por su poco uso de recursos de computadora son una gran alternativa para aquellas máquinas que no poseen una gran capacidad de cómputo para la ejecución de código fuente.

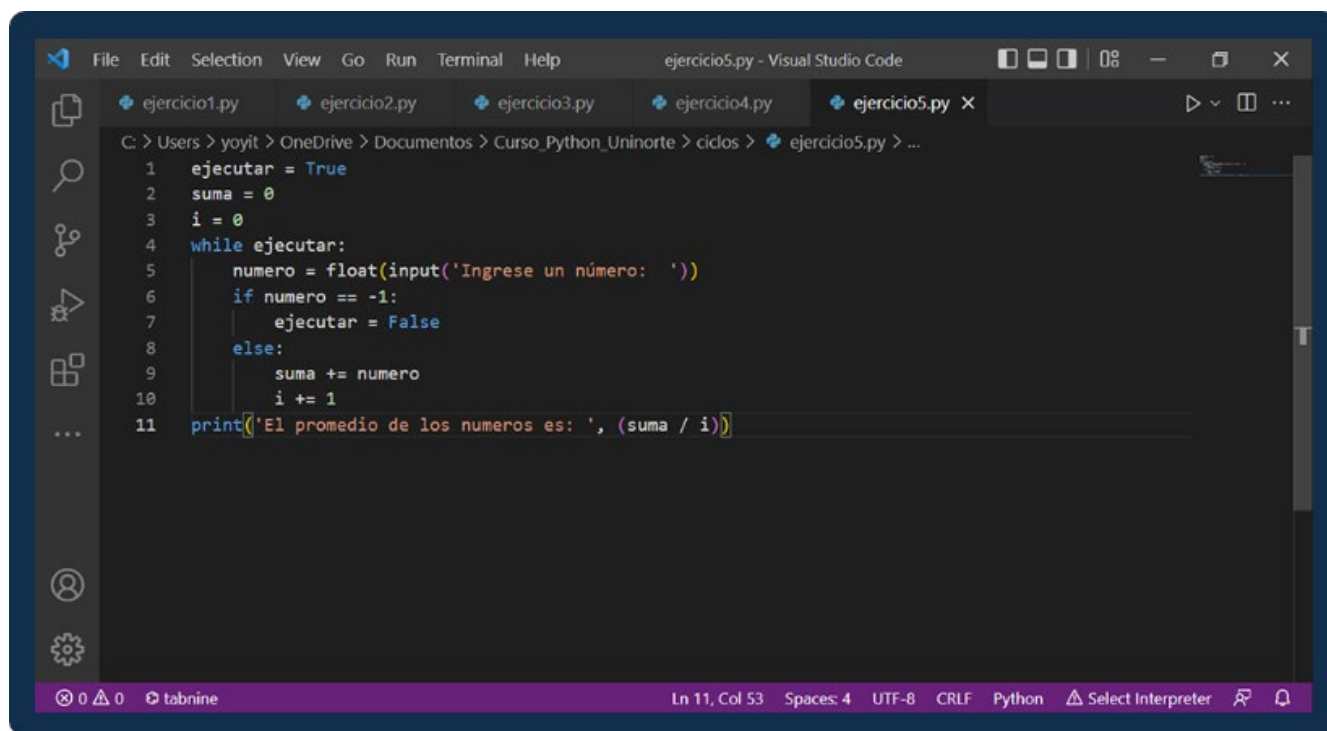
Existen muchos aspectos que se pueden rescatar de los editores de código, pero una de las particularidades más asombrosa que se puede tener en cuenta es la capacidad de adaptación a la estructura del código que se está escribiendo en el mismo. En pocas palabras, tiene la capacidad de adaptar el estilo del código de acuerdo con el lenguaje de programación que se esté utilizando.

Un ejemplo de ello es la sintaxis, por ejemplo, el lenguaje Python comparado con el lenguaje C# es totalmente distinto. En el lenguaje C# se puede utilizar llaves para abrir condicionales, ciclos y funciones, las cuales se pueden colocar en cualquier parte del código para darle un estilo de tabulación adecuado a ese diseño específico; es decir, adapta la tabulación del código de acuerdo con ese lenguaje. La sintaxis de Python, por su parte, no permite que se coloque una línea de código debajo de alguna instrucción; para evitar esto, el editor de código coloca un espacio de tabulación en el comando que se encuentra debajo de un comando “if, else, for, while” entre otros, con el fin de no generar un error por tabulación del código. Estos aspectos hacen de un

editor de código una herramienta que ayuda a organizar el código fuente de manera sencilla y eficiente al programador y es muy valioso al momento de realizar la codificación de un programa.

En la siguiente figura se evidencia la manera en cómo un editor de código, en este caso Visual Studio Code, realiza el ordenamiento de la estructura del código fuente para evitar que puedan existir errores y también ayudar en el aspecto visual al usuario para identificar las líneas de código de una manera mucho más comprensible, contribuyendo, además, a la solución de errores o problemas que se puedan presentar en el momento de realizar la codificación del programa; este es un claro ejemplo de las ventajas que ofrecen los editores de código a los desarrolladores de “software”.

**Figura 1.** Estructura de código fuente en Python



```
File Edit Selection View Go Run Terminal Help
ejercicio5.py - Visual Studio Code
ejercicio1.py ejercicio2.py ejercicio3.py ejercicio4.py ejercicio5.py x
C:\Users\yoyit> OneDrive > Documentos > Curso_Python_Uninorte > ciclos > ejercicio5.py > ...
1 ejecutar = True
2 suma = 0
3 i = 0
4 while ejecutar:
5     numero = float(input('Ingrese un número: '))
6     if numero == -1:
7         ejecutar = False
8     else:
9         suma += numero
10        i += 1
11 print('El promedio de los numeros es: ', (suma / i))
...
0 0 0 tabnine Ln 11, Col 53 Spaces: 4 UTF-8 CRLF Python Select Interpreter
```

## 2. Diferentes editores de código

En la actualidad existen muchos editores de código que se encargan de ayudar a los programadores de aplicaciones a realizar este proceso de manera organizada y estructurada. Se debe tener en cuenta que estas herramientas están con el propósito de mejorar la codificación de aplicaciones.

También, es importante conocer que muchas de estas herramientas deben ser seleccionadas por el programador, de acuerdo con las necesidades que se tengan para el desarrollo del proyecto a construir. A continuación, se presentarán algunas de ellas:

### Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web.

Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Este editor de código permite la integración de diferentes lenguajes de programación y realiza la codificación de cada uno de ellos, teniendo en cuenta su sintaxis y estructura. Actualmente, es una de las herramientas más utilizadas en el mercado para la codificación de aplicaciones informáticas con preferencia a la programación web.

### SublimeText

Es un editor de texto y editor de código fuente. Está escrito en C++ y Python para los plugin. Desarrollado originalmente como una extensión de Vim, que con el tiempo fue creando una identidad propia.

Aún conserva un modo de edición tipo vi llamado Vintage mode, el cual es muy utilizado por su gran compatibilidad con los diferentes lenguajes de programación actuales en el mercado y por tener un gran rendimiento sin utilizar gran cantidad de recursos de la máquina tales como memoria RAM y procesador.

### **NotePad++**

Es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación y soporte nativo para Microsoft Windows.

Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. Adicional a ello, se destaca por su versatilidad al momento de realizar la codificación, ya que permite, por su simple entorno gráfico, encontrar los problemas presentados en el código fuente de la aplicación, lo que a su vez hace que los desarrolladores inviertan menos tiempo en encontrar las fallas que se puedan presentar en el código.

### **Google Colab**

Es una herramienta de Google que le permite a cualquier persona escribir código fuente en el lenguaje Python utilizando solamente un navegador de Internet y por supuesto conectividad. No requiere instalar componentes adicionales en la computadora para ser utilizada.

Esta herramienta ha recibido tanto comentarios positivos y otros negativos por parte de la comunidad de desarrolladores. Se puede decir que la herramienta es una muy buena opción para aquellas personas que desean aprender este lenguaje de



programación. Sin embargo, se queda corto en algunos aspectos de integración de ciertas rutinas y manejo de procesos al momento de desarrollar.

Todas estas herramientas están a disposición para cualquier persona que desee incursionar en el mundo de la programación, son de descarga gratuita y poseen licencia “Free” (es decir, se pueden usar sin tener que pagar). Estas herramientas también permiten la integración de la mayoría de lenguajes de programación, lo que facilita la depuración y codificación de aplicaciones informáticas en la actualidad; por lo tanto, son alternativas muy utilizadas en la comunidad de programadores a nivel mundial.

Sumado a lo anterior, estas herramientas incluyen actualizaciones continuas en las que se agregan nuevos componentes tales como plugins. Esto le permite a los desarrolladores integrar tecnologías y rutinas para mejorar la forma en cómo desarrollan las aplicaciones.

Una de estas herramientas que se traen a colación es TABNINE. Es un plugin utilizado en diversos editores de código que trabaja con tecnología Deep Learning, la cual puede predecir si se utilizará una rutina específica o código. Esto sucede al llevar un registro del estilo de programación de cada persona y luego la herramienta ayuda a reutilizar dichas rutinas y a construir el código fuente de una manera mucho más sencilla y organizada.

### 3. Proceso de instalación de Visual Studio Code

Como se ha abordado hasta este momento, existen diferentes editores de código, los cuales permiten realizar el proceso de codificación de las aplicaciones informáticas. Cada uno de ellos ofrece diversas ventajas y desventajas, llegando a un punto de no poder establecer cuál de todos es el mejor, sino que de forma individual cada programador elegirá aquel con el que se sienta más cómodo y terminará convirtiéndose en su editor de preferencia.

Sabiendo esto, se presentan los pasos a seguir para realizar la instalación de la herramienta en el computador; para este caso Visual Studio Code, el editor que se utilizará para el desarrollo temático. Por favor tener en cuenta cada paso que se deberá llevar a cabo para realizar la instalación de esta herramienta de manera correcta:

- **Paso 1.** Realizar la descarga de la aplicación. Lo primero que debemos hacer es dirigirnos a la página oficial para la descarga de los instaladores que utilizaremos para hacer el proceso de instalación en nuestra computadora. Ingresar al enlace <https://code.visualstudio.com/>
- **Paso 2.** Seleccionar el sistema operativo instalado en la computadora para descargar los instaladores, este paso es importante ya que si seleccionamos el sistema operativo que no corresponde al de nuestra computadora no podremos realizar la instalación de manera correcta.
- **Pase 3.** Descargar el instalador teniendo en cuenta el sistema operativo seleccionado, localiza la descarga e inicia el archivo para proceder a la instalación. Lo primero que tendrás que hacer es aceptar los términos de la licencia y completar los primeros pasos.

- **Paso 4.** Seleccionar las opciones de “Agregar PATH”, “Registrar code como editor de tipos de datos admitidos” y marca la opción “Crear un acceso directo en el escritorio” para que se pueda acceder a la aplicación de una manera más fácil luego de realizar la instalación de la aplicación, luego vamos al siguiente paso y seleccionamos instalar.
- **Paso5.** Una vez realizada la instalación del programa debemos poder visualizar la pantalla principal de la herramienta Visual Studio Code, por lo cual nuestro proceso de instalación habrá culminado con éxito.

## 4. Preparar entorno de desarrollo virtual

A continuación, se ven los pasos a seguir para la configuración de un entorno virtual de desarrollo. Es importante seguirlos como se indica para realizar este proceso de manera correcta; para esto, se debe comenzar con una exploración de qué es un entorno de desarrollo virtual.

Un entorno de desarrollo virtual es la creación de directorios aislados, los cuales permiten realizar la configuración de los componentes que sólo requiere el proyecto sin necesidad de afectar todo el sistema en general.

Por ejemplo, al suponer que se tienen dos aplicaciones una llamada App1 y otra llamada App2.

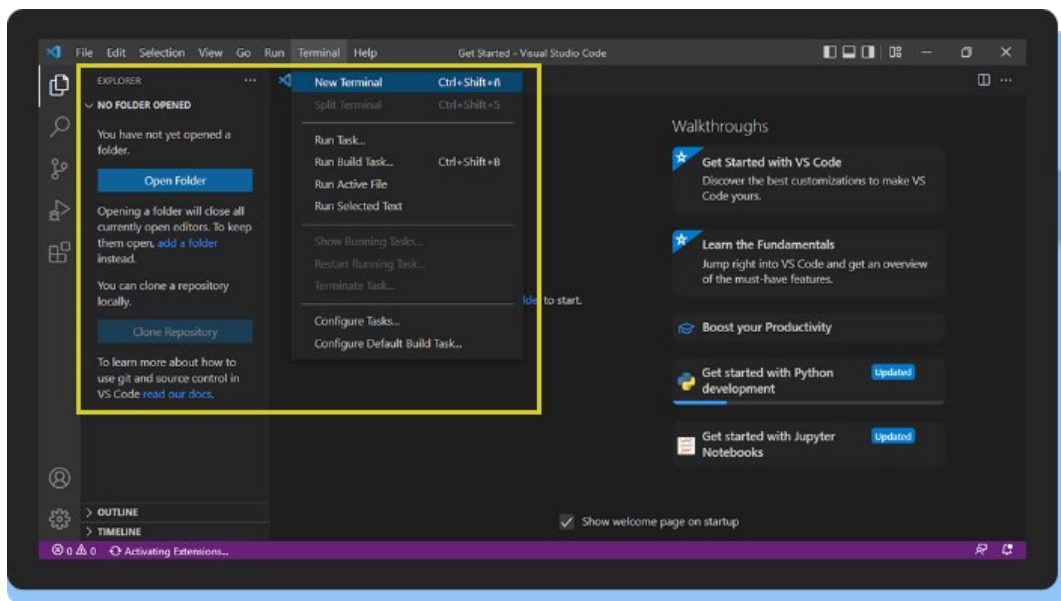
Para el primer caso se requiere la instalación de las librerías L1 y L2 y para el segundo se necesita solo instalar la librería L2.

En este caso la instalación se realizaría de manera independiente para cada aplicación, sin que todas se vean afectadas, ya que cada una tiene su propio espacio de configuración y de instalación de componentes necesarios.

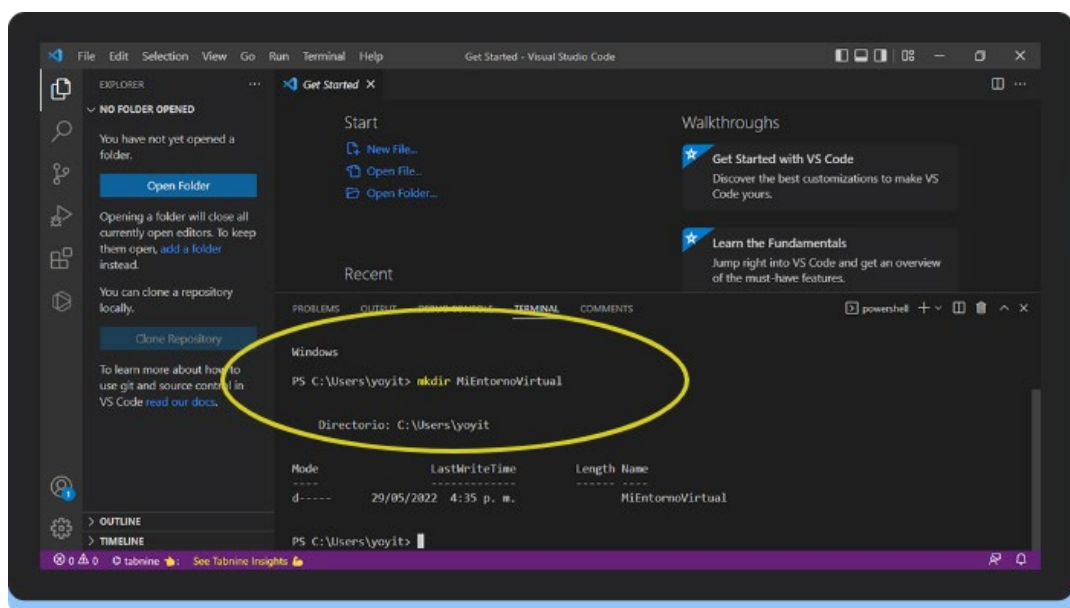
Esta es una excelente estrategia cuando se trabaja con sistemas de versionamiento, que en versiones superiores ya no se usen componentes y no se requiera instalar en las nuevas versiones.

Se pueden ver entonces los pasos para la instalación de un entorno de desarrollo virtual:

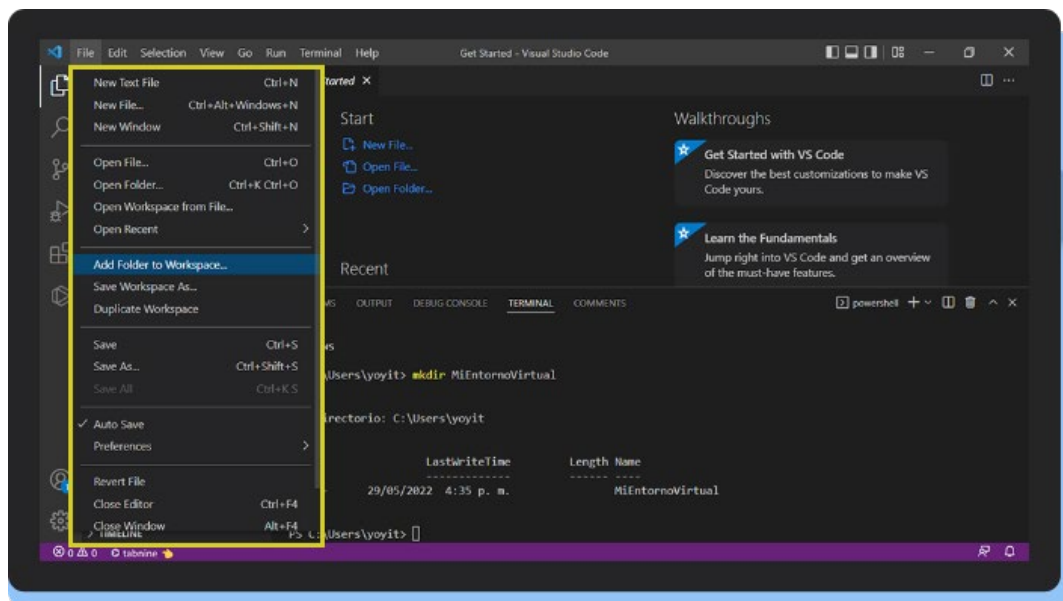
**Paso 1.** Abrir la aplicación de Visual Studio Code y hacer clic en la opción que dice “New Terminal” en la parte superior de nuestro entorno de desarrollo.



**Paso 2.** Colocar el siguiente comando en nuestra terminal: mkdir NombreDirectorio.

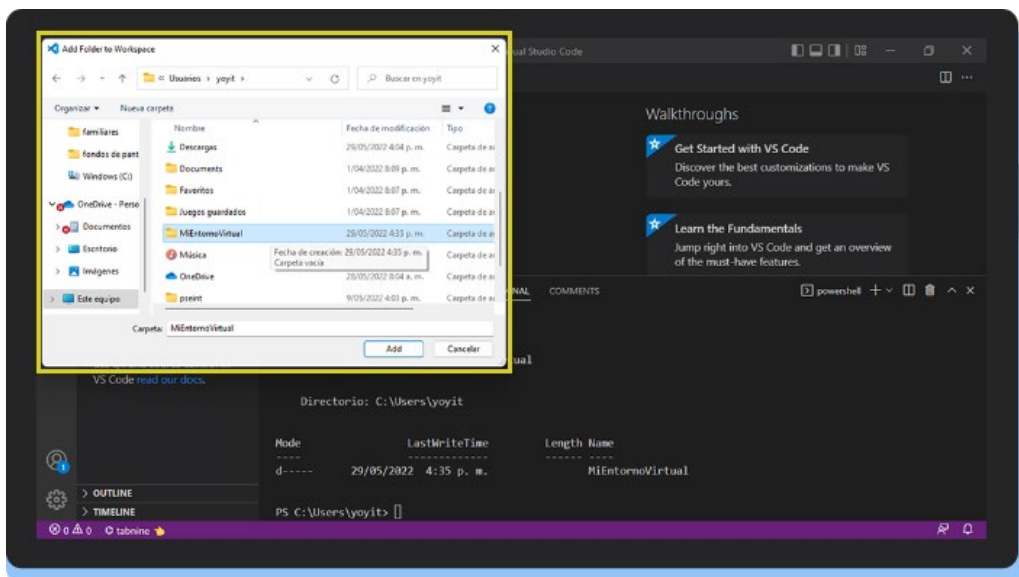


**Paso 3.** Procedemos a abrir nuestro espacio de trabajo a través de la opción “Add Folder to Workplace...”



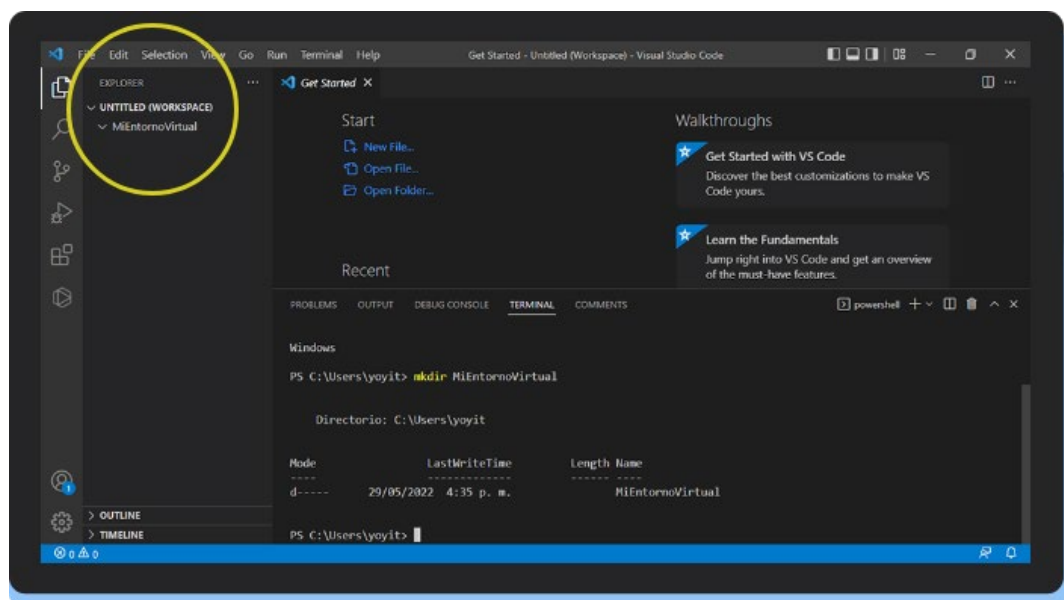
**Paso 4.** Procedemos a buscar la ubicación de nuestra carpeta y luego presionamos el botón “Add”, en este paso es importante tener en cuenta que la carpeta quedará ubicada en la posición en la que esté ubicado en la terminal es decir en un lugar diferente.

Nos aparecerá luego un mensaje donde colocaremos “Yes” en caso de confiar en los archivos contenidos en la carpeta.

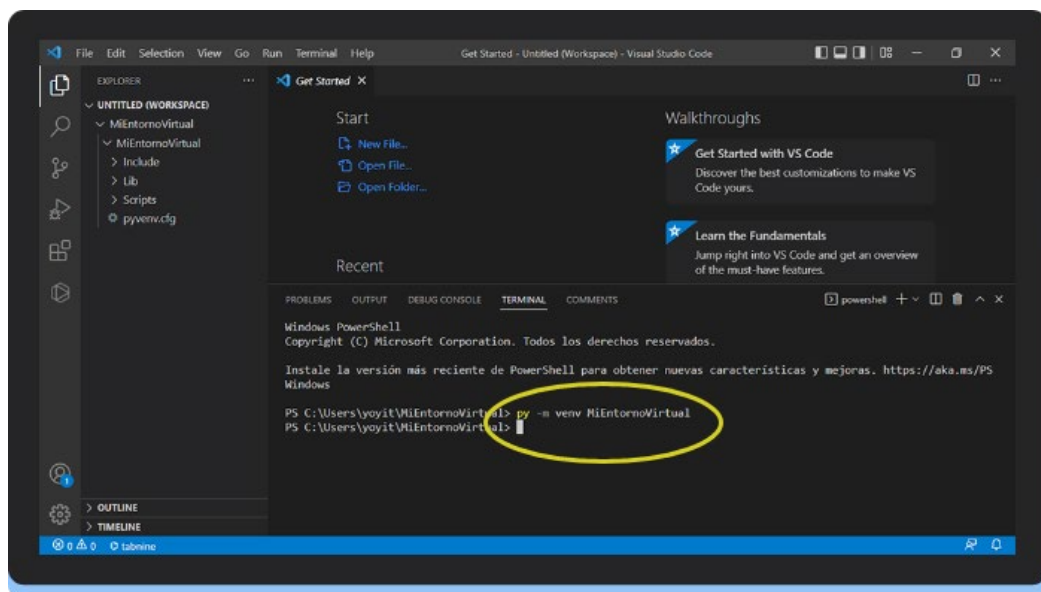


**Paso 5.** Como podemos observar en nuestro explorador a mano izquierda nos aparece el directorio que está destinado para ser nuestro entorno de desarrollo virtual.

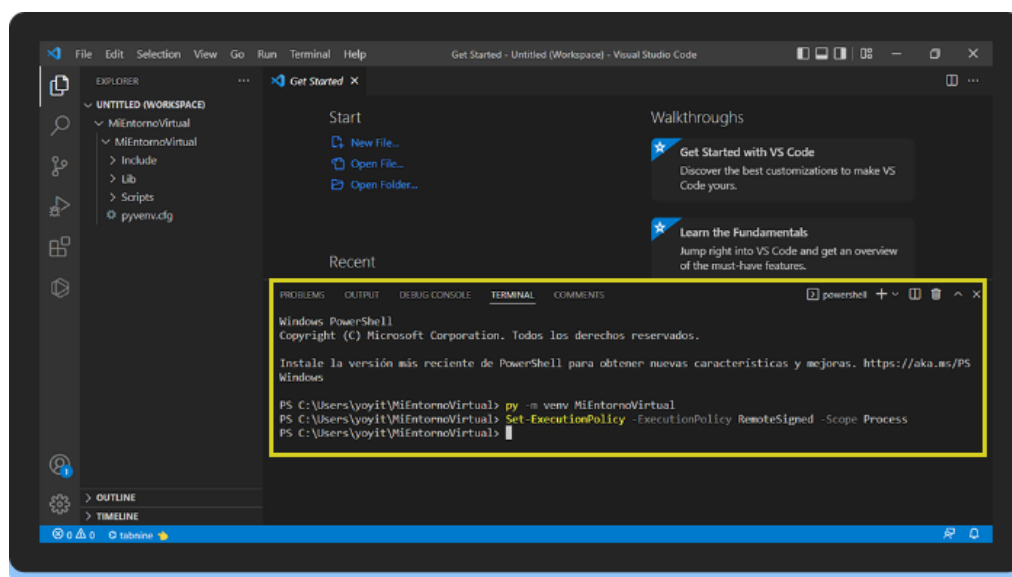
En esa misma sección hacemos clic y procedemos a abrir una nueva terminal “Open in Integrated Terminal” desde esa ubicación.



**Paso 6.** Se abrirá una nueva terminal ubicada en la dirección correspondiente al espacio creado. Allí procedemos a ejecutar el comando que permite crear nuestro entorno de desarrollo virtual con los archivos correspondientes.

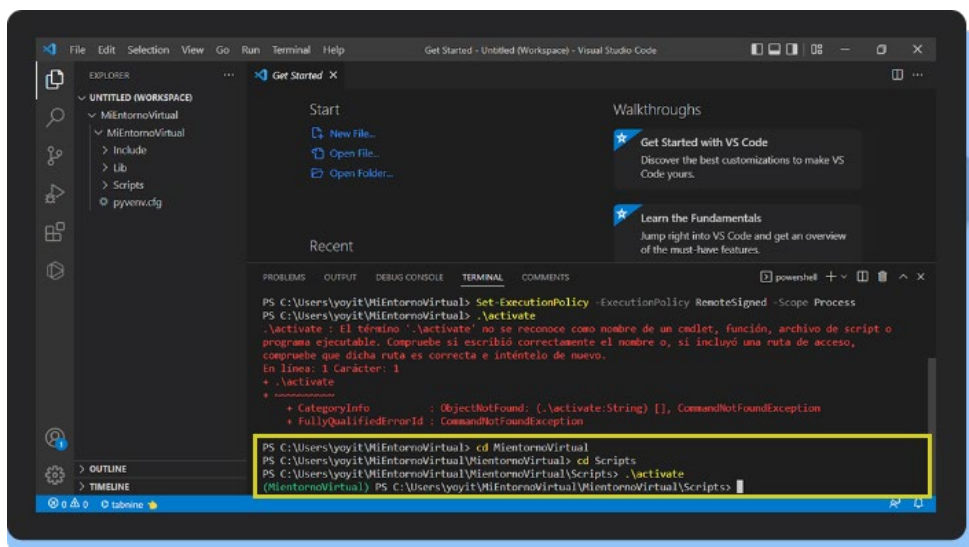


**Paso 7.** Luego procedemos a realizar las actualizaciones de políticas a utilizar en el entorno virtual utilizando el siguiente comando que se puede apreciar en la imagen.

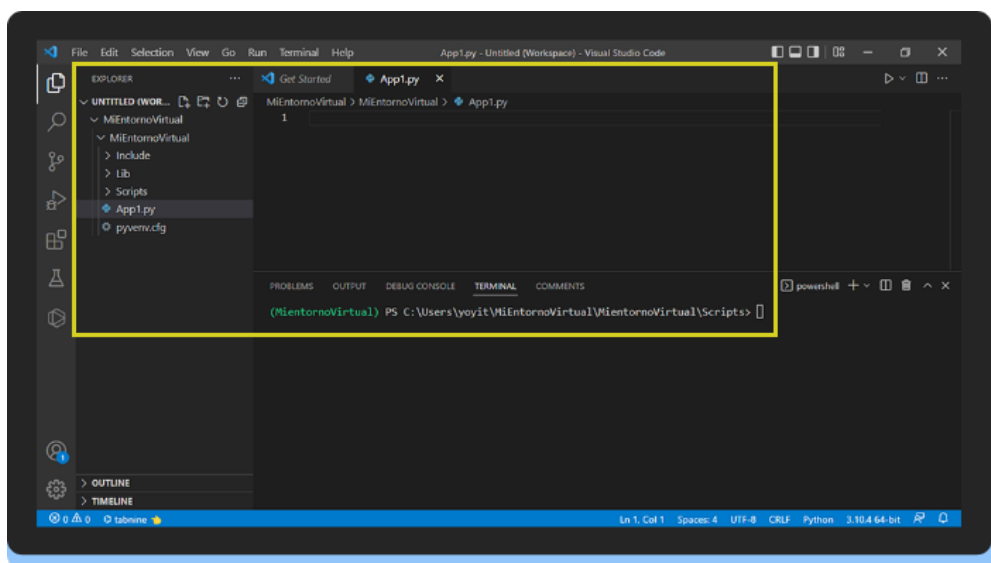




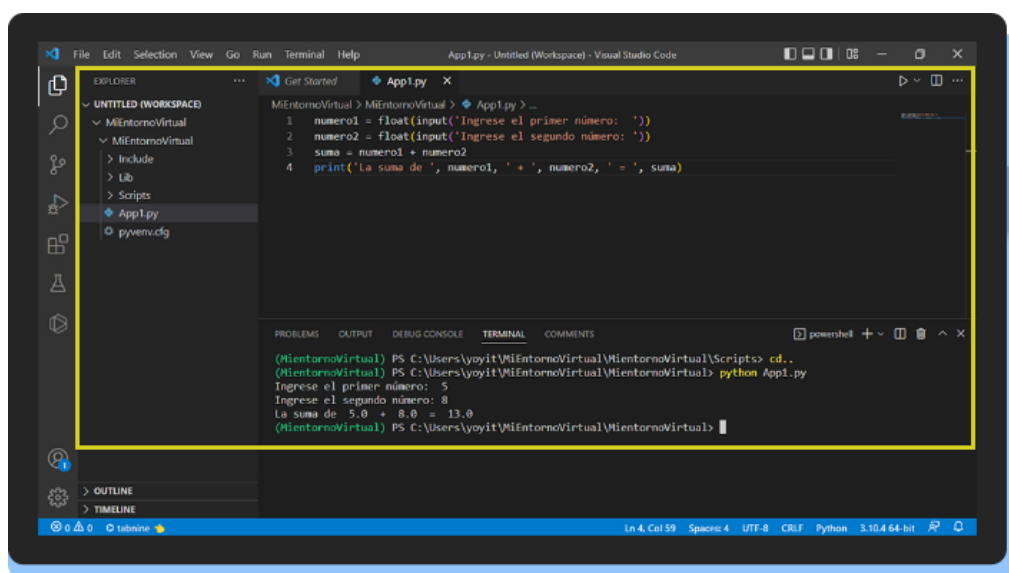
**Paso 8.** Procedemos a ingresar a la carpeta de Scripts para que finalmente procedamos a activar nuestro entorno de desarrollo, como se observa en la imagen, se coloca de color verde indicando que se encuentra activo y listo para usar.



**Paso 9.** Procedemos a realizar la creación de nuestro archivo de código fuente en Python, al dar clic derecho y señalar la opción que dice “new file” encima del entorno de desarrollo. Luego escribiremos el nombre del archivo de código fuente con su extensión la cual es .py como podemos apreciar.



**Paso 10.** En pantalla nos aparecerá el espacio donde escribiremos las líneas de código que a manera de ejemplo se llama “App1.py”, con las cuales nuestro programa realizará sus funciones establecidas. Hemos codificado nuestra aplicación para que solicite 2 números al usuario y luego realice la suma con su posterior resultado de este proceso seguido del comando para realizar la ejecución de esta.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a workspace named 'MiEntornoVirtual' containing a file 'App1.py'. The main editor displays the code for 'App1.py':

```
1 numero1 = float(input('Ingrese el primer número: '))
2 numero2 = float(input('Ingrese el segundo número: '))
3 suma = numero1 + numero2
4 print('la suma de ', numero1, ' + ', numero2, ' = ', suma)
```

Below the editor, the TERMINAL pane shows the execution of the script:

```
(MiEntornoVirtual) PS C:\Users\yoyit\MiEntornoVirtual\MiEntornoVirtual\Scripts> cd..
(MiEntornoVirtual) PS C:\Users\yoyit\MiEntornoVirtual\MiEntornoVirtual> python App1.py
Ingrese el primer número: 5
Ingrese el segundo número: 8
la suma de 5.0 + 8.0 = 13.0
(MiEntornoVirtual) PS C:\Users\yoyit\MiEntornoVirtual\MiEntornoVirtual>
```

Como se ve a través de esta ejemplificación, la creación de entornos de desarrollo virtuales ofrece grandes ventajas al momento de realizar desarrollos orientados al uso de librerías y extensiones propias de un proyecto independiente; el tema de versionamiento también toma gran fuerza, ya que se pueden establecer todos los componentes que este llevará y es mucho más fácil instalar y colocar en marcha las aplicaciones y su despliegue en servidores de producción.

## 5. Manejo y control de versiones

Para el manejo y control de versiones que se realizan a los proyectos es importante tener en cuenta que existen muchas herramientas en el mercado que se pueden utilizar para llevar a cabo este proceso.

Para este caso en particular de este componente formativo, se ha establecido trabajar con dos herramientas que son líderes a nivel mundial para realizar este proceso, por lo cual se abordará la manera en cómo estas son usadas y su importancia en los proyectos de software:

**Git:** es un programa que permite controlar las versiones que se trabajan en un software. Esto contribuye en el mejoramiento y desempeño de las aplicaciones, lo que permite realizar una actualización de los archivos de código fuente que se trabajan en el proyecto de software. También permite llevar un control detallado de todos los sucesos que ocurren al interior del código y así poder realizar cambios y que estos queden registrados para un mayor control. Esto ayuda a que se controle quién y cuándo se realiza un cambio en el código fuente de la aplicación.

Con Git es importante que se tenga en cuenta que esta herramienta tiene un entorno gráfico para realizar la administración de los procesos que se llevan a cabo con ella; entonces, lo primero que se hará es realizar su instalación. Se deja un video en el material complementario denominado “Instalación de Git en Windows paso a paso” para que se pueda observar el proceso de instalación de la herramienta Git.

**Nota:** en el video del material complementario se explica de manera detallada cómo realizar la instalación de la herramienta Git. Cuando se le pregunte el editor de código que se integrará, en ese paso se debe seleccionar la opción que dice “Visual

Studio Code” para que el editor de código quede conectado con la herramienta Git. También se puede ver que existe una versión de Git llamada GUI, la cual permite crear los repositorios de una manera mucho más cómoda, ya que tiene una interfaz gráfica muy sencilla de operar.

A continuación, se explica cómo crear un repositorio utilizando la herramienta Git y luego cómo integrarlo a Visual Studio Code; es importante estar atentos a los pasos, para lograr realizar el proceso de manera correcta:

## **Video 2. Pasos para crear un repositorio y su integración con Visual Studio Code**



[Enlace de reproducción del video](#)

**Síntesis del video: pasos para crear un repositorio y su integración con Visual Studio Code**

Pasos para crear un repositorio y su integración con Visual Studio Code. A continuación, te presentamos cómo crear e integrar un repositorio. Este proceso es muy similar según el editor de código a utilizar, que para nuestro caso será Visual Studio Code. Para esta tarea, vamos a necesitar cumplir con varios pasos que nos permitirán alcanzar nuestro objetivo, el cual es hacer de forma correcta la creación de un repositorio dentro de nuestra herramienta.

Instalamos la extensión en Visual Studio Code que permitirá la integración con Git. Primero, nos dirigimos al icono "Extensiones", el cual desplegará un buscador. En él, colocamos la palabra "Git" y podremos seleccionar e instalar la extensión "Git Extension Pack", procediendo con su instalación.

En este paso, vamos a crear nuestro espacio de trabajo realizando el proceso de creación de este. Ingresamos a la sección "File" y allí seleccionamos "Add Folder to Workspace". Seleccionamos la carpeta destinada desde nuestro computador y con eso tendremos el espacio creado para luego proceder a sincronizarlo con nuestra herramienta Git.

En este paso, procedemos con la integración de Git y Visual Studio Code. Nos dirigimos al icono llamado "Source Control" ubicado en la parte izquierda y luego damos clic en "Initialize Repository". Nos aparecerá una ventana en la parte superior donde nos muestra la ruta por defecto de la carpeta de trabajo, la cual seleccionamos para agregarla al repositorio.

En este paso, ya se habrá realizado la sincronización de nuestra carpeta de trabajo con Git y nos damos cuenta de que no tiene cambios a realizar, puesto que no hemos creado ningún archivo que requiera de su actualización.

Creamos el archivo de código fuente. El primer paso es dirigirnos al explorador, el cual se encuentra en la parte superior, y creamos el archivo. Como se muestra, habremos creado un archivo llamado "app1.py", el cual utilizaremos para realizar nuestro programa. El color verde en el área de trabajo quiere decir que ya se encuentra una actualización de este archivo en el repositorio.

Vamos a ver el espacio de trabajo de Git para ver que el archivo se encuentra para ser actualizado. El archivo de código fuente que acabamos de crear se encuentra listo para ser sincronizado con el repositorio de Git. Tener en cuenta que se debe realizar el proceso de preparar el archivo para su respectiva sincronización.

Una vez presionamos el botón "Stage Changes", una vez se realiza el proceso, nos aparece como cambio para realizar. Luego, colocamos el mensaje que corresponde y realizamos el commit. Después de colocar el mensaje y de realizar el commit, nos envía a registrarnos y autenticarnos en el repositorio de código llamado GitHub. En caso de no estar registrados, procedemos a registrarnos o autenticarnos con usuario y contraseña.

Una vez autenticados, nos solicitará que autoricemos a la extensión instalada de GitHub en Visual Studio Code a través de un botón verde. Luego, nos saldrá un cuadro de mensaje donde presionaremos "Open" para posteriormente marcar "Yes" en el cuadro inferior derecho. Se sincronizarán los repositorios en Visual Studio Code

y nos aparecerá como nos ilustra la imagen. Procedemos a ingresar a la página de GitHub y crear un repositorio para el almacenamiento de nuestro código fuente.

Estando en la página principal, presionamos el botón "New" de color verde ubicado en la parte superior izquierda del sitio. Le colocamos el nombre al repositorio con su respectiva descripción. La exposición que para este caso es pública y luego presionamos el botón verde "Create Repository" para realizar la creación del repositorio.

Procedemos a la descarga de la aplicación GitHub para administrar los repositorios de manera sencilla desde la propia aplicación de escritorio. A través de la página oficial, una vez descargada, procedemos a instalarla y ejecutarla.

Para efecto de nuestro proceso de explicación, veamos los pasos de cómo aplicar el repositorio de GitHub con Visual Studio Code. Creamos nuestro repositorio en el menú "File", opción "New Repository". Colocamos el nombre al repositorio con su descripción. Tener en cuenta que no se deben utilizar espacios en blanco para el nombre del repositorio. Presionamos el botón "Show" para crear el repositorio local desde el cual se trabajarán los archivos.

Una vez realizado el paso anterior, damos clic en el botón azul "Create Repository". Nos aparecerá la ventana del repositorio. Allí ubicaremos el botón "Publish Repository" para que nuestro repositorio local se sincronice con el que se encuentra en la nube.

En la misma ventana anterior, buscamos el botón "Open in Visual Studio Code" y damos clic, lo que hará que cargue el editor conectado con el repositorio que creamos. Allí procedemos a crear nuestro archivo de código fuente.

Ahora procedemos a verificar en nuestro GitHub que el archivo cargue en nuestro repositorio y procedemos a realizar el commit para actualizar el repositorio. Lo anterior producirá la actualización en el repositorio en la nube.

Procedemos a codificar el archivo para ver los cambios realizados y a realizar el commit. Con esto, ya los cambios quedarán realizados en el repositorio en la nube y habremos terminado todos nuestros pasos.

**Nota:** Para mayor comprensión, es importante tener en cuenta que Git y GitHub son dos entidades diferentes que ayudan a administrar y alojar archivos. En otras palabras, Git sirve para controlar las versiones de los archivos gestionando el código fuente, mientras que GitHub es una plataforma para alojar o descargar repositorios Git.

Como conclusión, se ha trabajado con los repositorios de códigos, los cuales permiten llevar un control de los cambios realizados en el código fuente y adicional a ello tener una copia en la nube para darle mejor seguridad a los archivos de código fuente de la aplicación.

Es importante tener en cuenta que este tipo de aplicaciones son muy utilizadas en las empresas y equipos de desarrollo a nivel mundial ya que proponen un nivel de seguridad alto para el código, seguimiento y la colaboración.



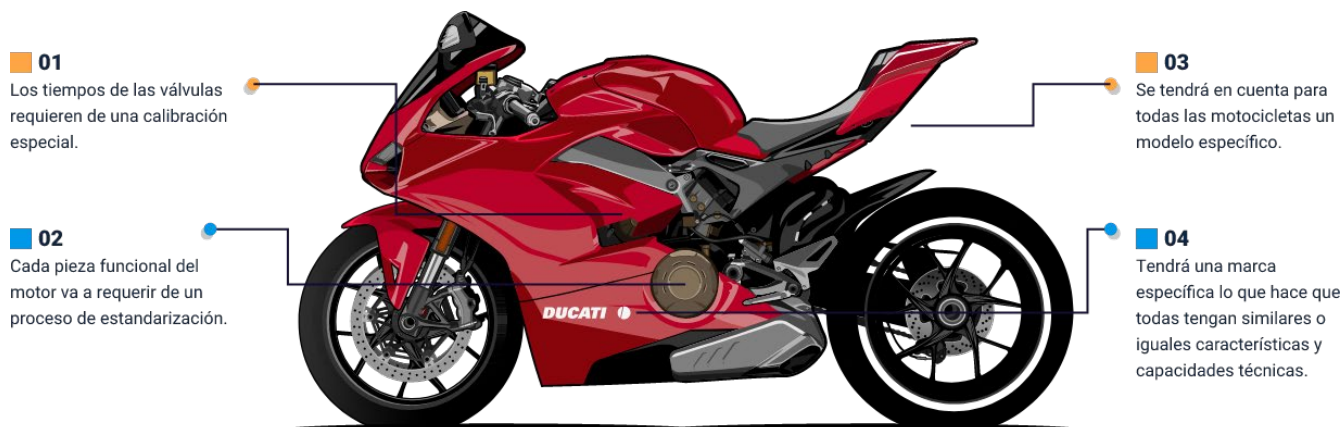
## 6. Estándares para la codificación de software

Conocer la importancia del proceso de codificación y su manejo de estándares es importante para abordar y comprender esta temática de una manera mucho más específica en cuanto a su uso y aplicación.

El concepto estándar es un modelo que se toma como referencia para construir, diseñar o fabricar un producto. Los estándares tienen en su teoría una serie de pasos y reglas que se deben cumplir para garantizar que la calidad y/o procesos se realicen de manera correcta.

Un estándar garantiza que un producto sea bueno o de calidad ya que se siguen ciertas reglas de fabricación y medición en cada fase del proceso.

Un ejemplo de lo anterior, puede ser la fabricación de una motocicleta:



Ahora bien, no solamente se hace referencia al proceso de fabricación, sino que también se debe tener en cuenta que se fabrica la motocicleta por primera vez y esta sirve de referencia para todas las demás fabricadas, ya que ha sido probada y ajustados sus valores de calibración y fondo que se tendrán en cuenta para todas las demás.

Así mismo, funciona en programación, el estándar sirve para mantener un estilo y una forma de escribir el código fuente y que este sea interpretado de una manera estandarizada.

Eso quiere decir que pueda ser entendible por otros programadores para cuando se requiera realizar una modificación o agregar nuevas funciones y se pueda realizar este proceso sin traumatismos causados por el no entendimiento. Esto incluye nombres de variables, nombre de funciones, atributos, clases, entre otros componentes.

Ahora bien, sabiendo lo que representan, es importante presentar los estándares de programación más utilizados para Python:

### **Camel Case**

Combina las palabras directamente, sin usar ningún símbolo, estableciendo que la primera letra de cada palabra esté en mayúscula a excepción de la primera palabra, estando el resto de las letras en minúsculas.

Este tipo de notación está muy extendida, siendo su uso muy común tanto en la declaración de variables como en el nombre de funciones y métodos.

Unos ejemplos de sintaxis pueden ser:

- De contar palabras, en notación Camel Case sería: contarPalabras
- De aumentar nivel dificultad, en notación Camel Case sería:  
aumentarNivelDificultad

## Pascal Case

La notación Pascal Case combina las palabras directamente, sin usar ningún símbolo, estableciendo que la primera letra de cada palabra esté en mayúscula sin excepciones, estando el resto de las letras en minúsculas.

Su uso es muy habitual en la definición de los nombres de las clases de múltiples lenguajes, como JavaScript o PHP.

Unos ejemplos de sintaxis pueden ser:

- De contar palabras, en notación Pascal Case sería: `ContarPalabras`
- De aumentar nivel dificultad, en notación Pascal Case sería:  
`AumentarNivelDificultad`

## Snake Case

La notación Snake Case combina las palabras usando un guion bajo `_` como nexos.

Existen dos versiones de esta notación, una en la que todas las letras están en minúscula y otra en la que todas las letras están en mayúscula. Esta notación, cuando se usa en mayúscula, es habitual en la declaración de constantes de lenguajes como PHP o JavaScript.

La notación Snake Case en su versión minúscula también es muy utilizada en la declaración de los nombres de los campos de las bases de datos. Además, también se utilizaba en la declaración de variables de PHP y, de hecho, aún es el estándar de muchos desarrolladores cuando programan plugin o temas para WordPress.

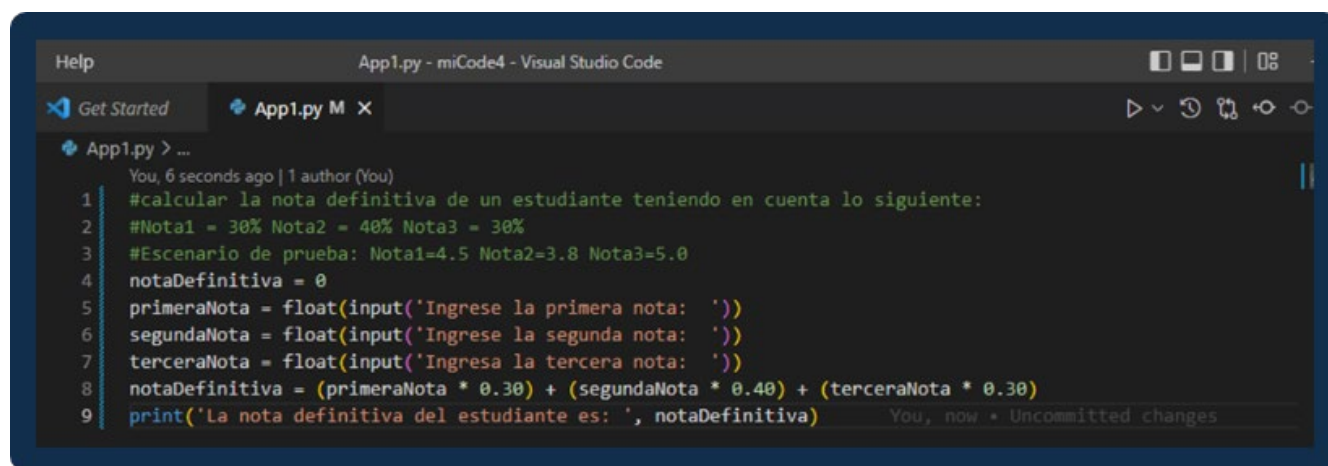
Unos ejemplos de sintaxis pueden ser:

- De contar palabras, en notación Snake Case sería: contar\_palabras
- De aumentar nivel dificultad, en notación Snake Case sería:  
aumentar\_nivel\_dificultad

**Camel Case:** Es el estándar más utilizado por la mayoría de las comunidades de desarrolladores en el mundo y se ha convertido en una de las que mayor referencia tiene en el código fuente que se ha estudiado. Es el preferido por los programadores por su sencilla manera de uso y practicidad.

Se puede ver en la siguiente figura 2 un ejemplo de Camel Case con Python:

**Figura 2.** Estándar Camel Case en Python



```

1  #calcular la nota definitiva de un estudiante teniendo en cuenta lo siguiente:
2  #Nota1 = 30% Nota2 = 40% Nota3 = 30%
3  #Escenario de prueba: Nota1=4.5 Nota2=3.8 Nota3=5.0
4  notaDefinitiva = 0
5  primeraNota = float(input('Ingrese la primera nota: '))
6  segundaNota = float(input('Ingrese la segunda nota: '))
7  terceraNota = float(input('Ingreses la tercera nota: '))
8  notaDefinitiva = (primeraNota * 0.30) + (segundaNota * 0.40) + (terceraNota * 0.30)
9  print('La nota definitiva del estudiante es: ', notaDefinitiva)
  
```

Como se puede observar se utiliza el estándar de codificación Camel Case, uno de los más utilizados y preferido por los programadores como se comentó previamente.

El proceso de construcción del código sigue un patrón específico, pero es importante tener en cuenta que la manera en cómo se estructura el código es fácil de

interpretar y de modificar las veces que sea necesarias, también se debe mencionar que este tipo de forma de codificar permite que otros programadores o miembros de un equipo de desarrollo puedan entender y trabajar de una manera mucho más fluida y ordenada.

Los estándares de desarrollo son una manera de no solo organizar el código, sino de trabajar en equipos de desarrollo.

Estos aportan calidad al producto de software que se desea construir y se debe tener en cuenta la palabra calidad, ya que el software es un producto y como tal requiere no solo de unos estándares para reflejar calidad, sino que también requiere llevar a cabo un proceso de documentación donde se consagra un documento llamado el manual del programador.

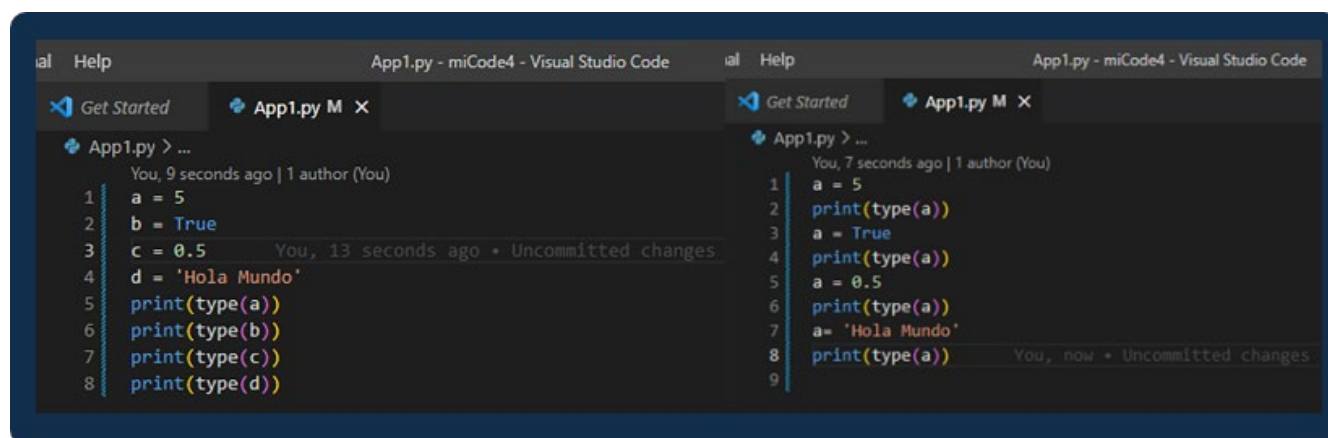
En este, se documenta el código y cada una de las funciones que realizan ciertas líneas de código y dicho documento se utiliza para generar una guía por si se requiere más adelante modificar o agregar nuevas funcionalidades al software, se tenga un punto de partida para que el proceso se lleve a cabo con los menores contratiempos esperados.

## 7. Creación y manejo de variables, estructuras de control, funciones, clases y objetos

En Python es fundamental el manejo de variables, teniendo en cuenta que este lenguaje de programación propone un uso muy particular de las mismas, permitiendo que una variable pueda recibir prácticamente cualquier tipo de información. Es decir, que en Python a diferencia de otros lenguajes de programación las variables, dependiendo del valor que se les asigne, así es su tipo de dato.

A continuación se ve un ejemplo de ello (ver figura 3):

**Figura 3.** Ejemplos de sintaxis de las variables



```

App1.py > ...
You, 9 seconds ago | 1 author (You)
1 a = 5
2 b = True
3 c = 0.5
4 d = 'Hola Mundo'
5 print(type(a))
6 print(type(b))
7 print(type(c))
8 print(type(d))

App1.py > ...
You, 7 seconds ago | 1 author (You)
1 a = 5
2 print(type(a))
3 a = True
4 print(type(a))
5 a = 0.5
6 print(type(a))
7 a = 'Hola Mundo'
8 print(type(a))
9

```

Como se puede observar en la figura las variables (a, b, c y d) tienen asignados diferentes tipos de datos y con la utilización de la función `type` permite saber qué tipo de variable es. En otros lenguajes de programación es necesario declarar el tipo de dato que soportará la variable, pero en el caso de Python una misma variable puede cambiar de tipo de dato sin tener ningún inconveniente.

Esto refleja lo poderoso y versátil que es el lenguaje Python, en este caso particular se ha utilizado la misma variable para almacenar diferentes tipos de datos.

Esta particularidad genera una ventaja al momento de realizar la codificación de un software puesto que se pueden reutilizar variables y por supuesto hacer un uso mucho más eficiente de la memoria RAM del equipo.

A continuación, se pueden ver los conceptos que junto al manejo de variables forman parte del entorno de trabajo Python. Ahora es tiempo de ver las estructuras de control, funciones, clases y objetos y conceptos claves para desarrollar en lenguaje de programación Python.

## **Estructura de control**

Para comenzar, en los lenguajes de programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Las estructuras de control permiten, a través de que se cumpla alguna condición, realizar la ejecución de una secuencia de código.

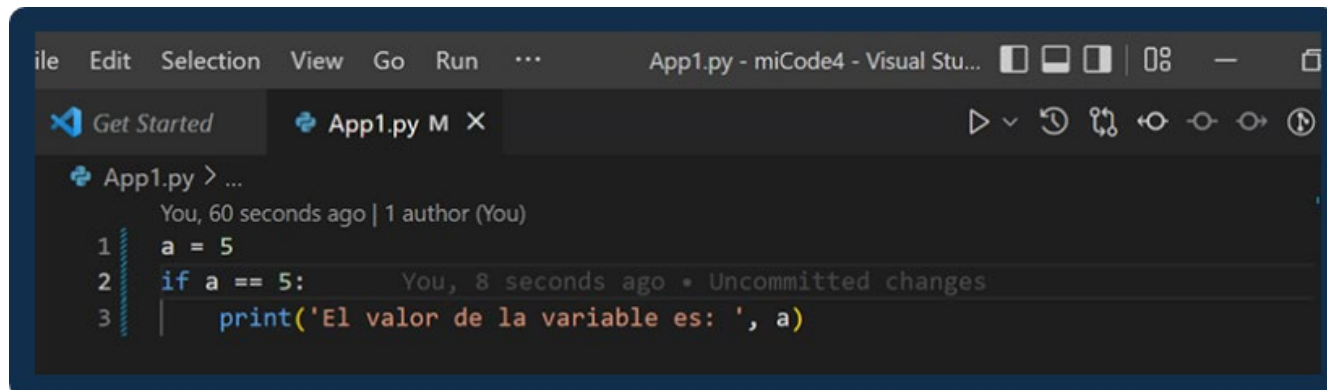
En la programación son muy importantes, así como las funciones, las clases y objetos. Los programas de computadora deben decidir de acuerdo con la información que se le suministra al sistema, realizar validaciones de acuerdo con las necesidades que este tenga que ejecutar y luego entregar al usuario los resultados esperados.

A continuación se pueden ver todos estos conceptos relacionados:

## **Estructuras de control simple**

Las estructuras de control simple permiten evaluar una condición dada y de acuerdo con la condición, y si esta se cumple, ejecutar las líneas de código que se encuentran dentro de la misma.

Estas se utilizan para tomar decisiones que solo dependen de que se cumpla una única condición.



```
App1.py - miCode4 - Visual Stu...
Get Started App1.py M X
App1.py > ...
You, 60 seconds ago | 1 author (You)
1 a = 5
2 if a == 5: You, 8 seconds ago • Uncommitted changes
3     print('El valor de la variable es: ', a)
```

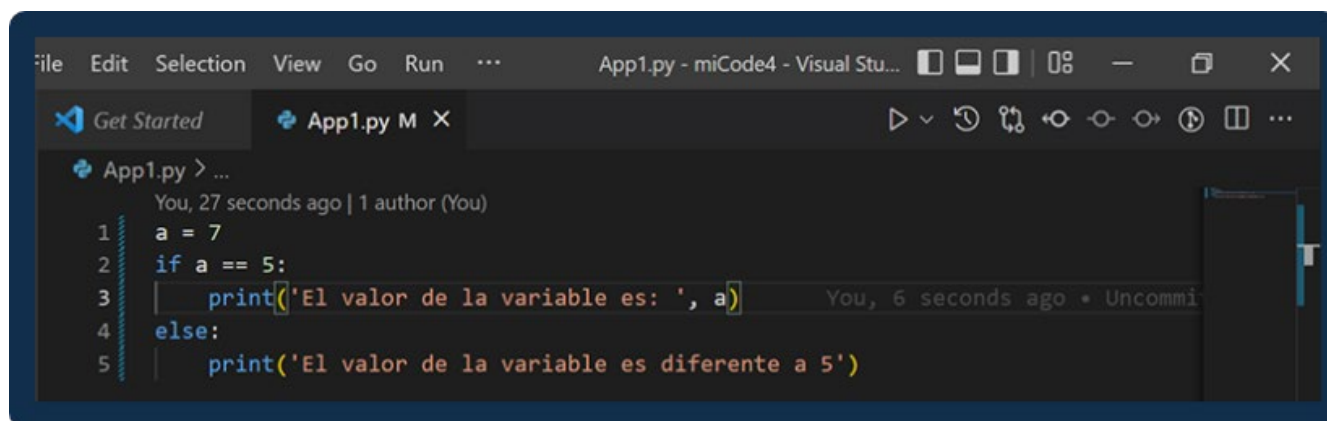
Como se puede observar, este es el resultado de la aplicación de una estructura de control simple, la cual permite validar si una variable establecida en una condición, se cumple para realizar las acciones que se le programan. Para este ejemplo, se envía el mensaje, pero se puede colocar cualquier tipo de instrucción o conjunto de instrucciones que, en caso de cumplirse la condición, realizarán las acciones que allí se establecen.

### Estructuras de control dobles

Las estructuras de control dobles o también llamadas compuestas permiten evaluar una condición inicial y si esta condición no se cumple permite negar el resultado de la primera condición y establecer nuevas acciones.

Acciones que se ejecutarán si ese proceso se llega a cumplir. Este tipo de estructuras se utilizan cuando se necesita validar más de una condición al tiempo.





```

App1.py > ...
You, 27 seconds ago | 1 author (You)
1 a = 7
2 if a == 5:
3     print('El valor de la variable es: ', a)
4 else:
5     print('El valor de la variable es diferente a 5')
  
```

Como se puede observar, se utiliza un comando de negación para la condición principal.

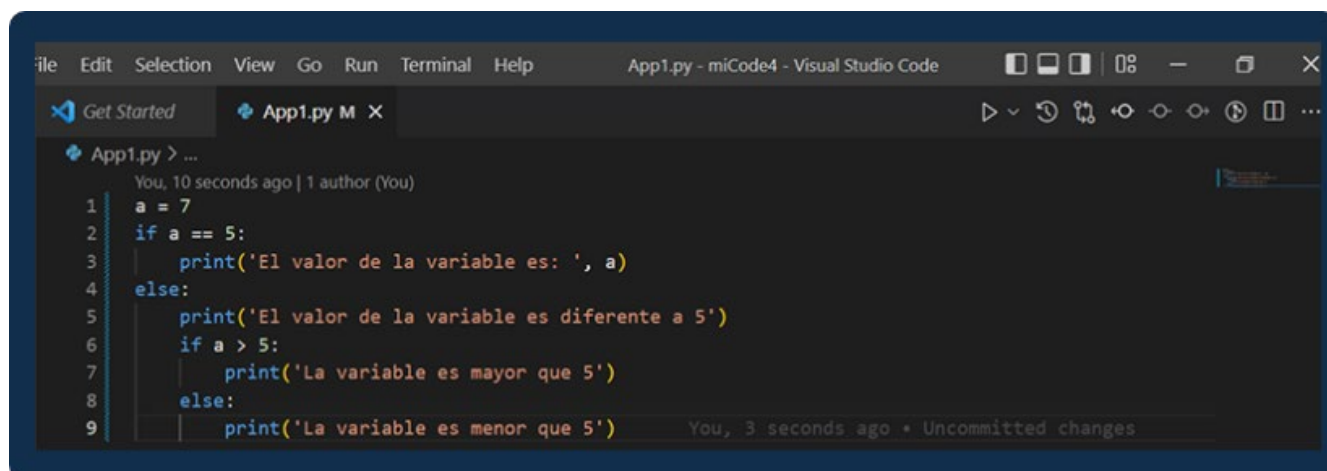
En este caso se ha cambiado el valor de la variable (a) de 5 a 7, por lo tanto la primera condición que verifica es si es el valor de la variable.

La instrucción else permite negar la condición principal y realizar otro grupo de acciones que pueden ejecutarse. Esto adicionalmente le da mayor versatilidad al programa ya que se pueden realizar muchas más acciones que en el condicional simple.

## Estructuras de control múltiples

Las estructuras de control múltiples permiten evaluar el valor que puede tener una variable teniendo en cuenta que muchas de esas condiciones, dependen de otras condiciones para poder ejecutar sus acciones.

De esta manera se ejecuta una serie de instrucciones por cada condición evaluada.



```

App1.py - miCode4 - Visual Studio Code
App1.py M X
App1.py > ...
You, 10 seconds ago | 1 author (You)
1 a = 7
2 if a == 5:
3     print('El valor de la variable es: ', a)
4 else:
5     print('El valor de la variable es diferente a 5')
6     if a > 5:
7         print('La variable es mayor que 5')
8     else:
9         print('La variable es menor que 5')
You, 3 seconds ago • Uncommitted changes

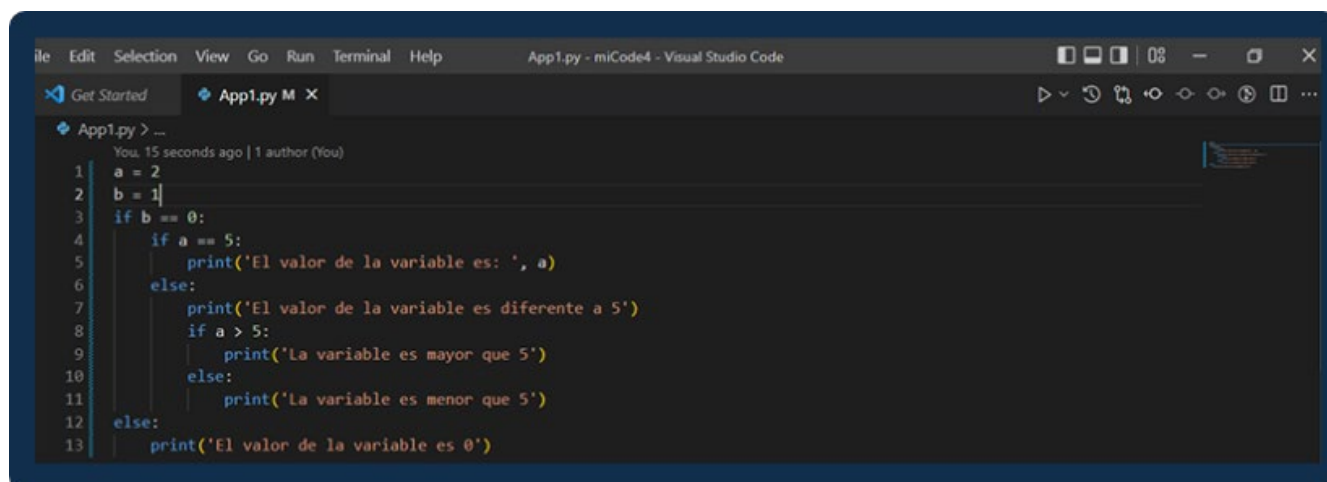
```

Como se puede observar, a diferencia de la estructura de control anterior, esta permite que se realicen evaluaciones de condiciones si la negación de la primera falla. Lo que hace tener un árbol de decisión que contempla la posibilidad de realizar la elección de una cantidad de instrucciones a ejecutar dependiendo de los valores posibles que se puedan evaluar para una variable o conjunto de ellas.

### Estructuras de control anidadas

Las estructuras condicionales anidadas permiten elegir entre varias opciones o alternativas posibles de forma encadenada, en función del cumplimiento o no de una determinada condición.

Este tipo de condicionales son muy útiles al momento de negar o aceptar el primer bloque condicional, es decir, si el bloque principal falla. Automáticamente no ingresa en ninguno de los condicionales, pero si ingresa permite evaluar diferentes opciones.



```

1  a = 2
2  b = 1
3  if b == 0:
4      if a == 5:
5          print('El valor de la variable es: ', a)
6      else:
7          print('El valor de la variable es diferente a 5')
8          if a > 5:
9              print('La variable es mayor que 5')
10             else:
11                 print('La variable es menor que 5')
12     else:
13         print('El valor de la variable es 0')

```

Claramente en la variable, se aprecia el uso de otra variable para realizar una evaluación inicial. Si esta se cumple, se validan las condiciones internas.

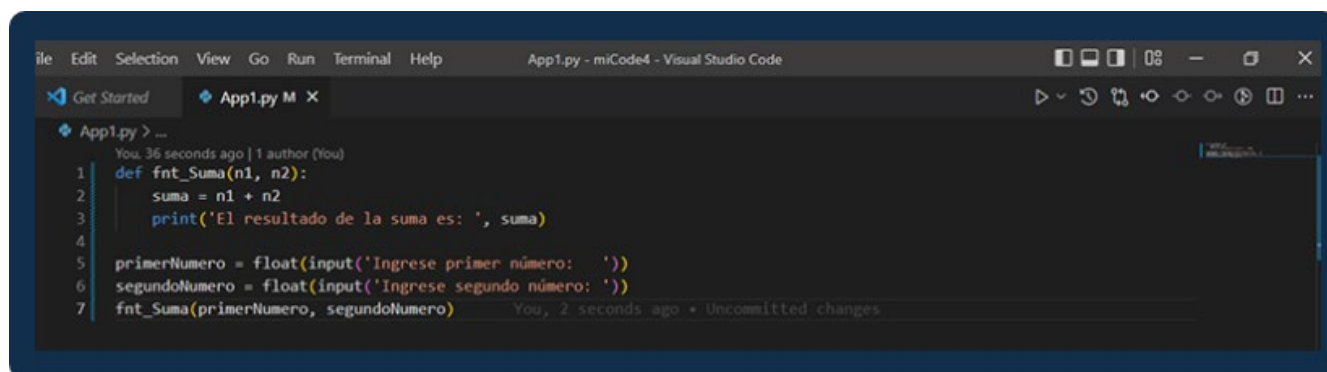
En caso de no cumplirse se niegan todas las condiciones e ingresa por el else para enviar el respectivo mensaje de rechazo.

## Funciones

El uso de funciones para cualquier programa de computadora es necesario.

Las funciones son como una especie de caja, las cuales, pueden contener una serie de códigos, que a su vez pueden ser invocados las veces que sean necesarios dentro del programa; entra dentro de los conceptos llamados recursividad. Es importante tener en cuenta que se debe contar con unas reglas para crear y llamar las funciones en el caso de Python.

Entra dentro de los conceptos llamados recursividad, es importante tener en cuenta que se debe contar con unas reglas para crear y llamar las funciones en el caso de Python.



```
1 def fnt_Suma(n1, n2):
2     suma = n1 + n2
3     print('El resultado de la suma es: ', suma)
4
5 primerNumero = float(input('Ingrese primer número: '))
6 segundoNumero = float(input('Ingrese segundo número: '))
7 fnt_Suma(primerNumero, segundoNumero)
```

Como se observa, para este caso, de aplicación de funciones se debe utilizar la palabra reservada `def` seguido del nombre de la función.

Es importante utilizar un estándar en el caso de las funciones, se utiliza la palabra `fnt` seguida del símbolo `_` para dar a entender que se trata de una función; y de esta manera se puede identificar de manera correcta dentro del código.

Otro aspecto es que las funciones reciben parámetros, en el caso de la función llamada `fnt_Suma`, recibe dos parámetros uno llamado `n1` y otro llamado `n2` los cuales se pueden evidenciar en la última línea de código.

Cuando se invoca la función, recibe los valores recolectados en las variables para ser enviados y realizar el proceso que se muestra en la imagen.

Esta es una manera de realizar toda la lógica del código dentro de un mismo espacio y ayuda también a entender mucho mejor el código fuente del programa.

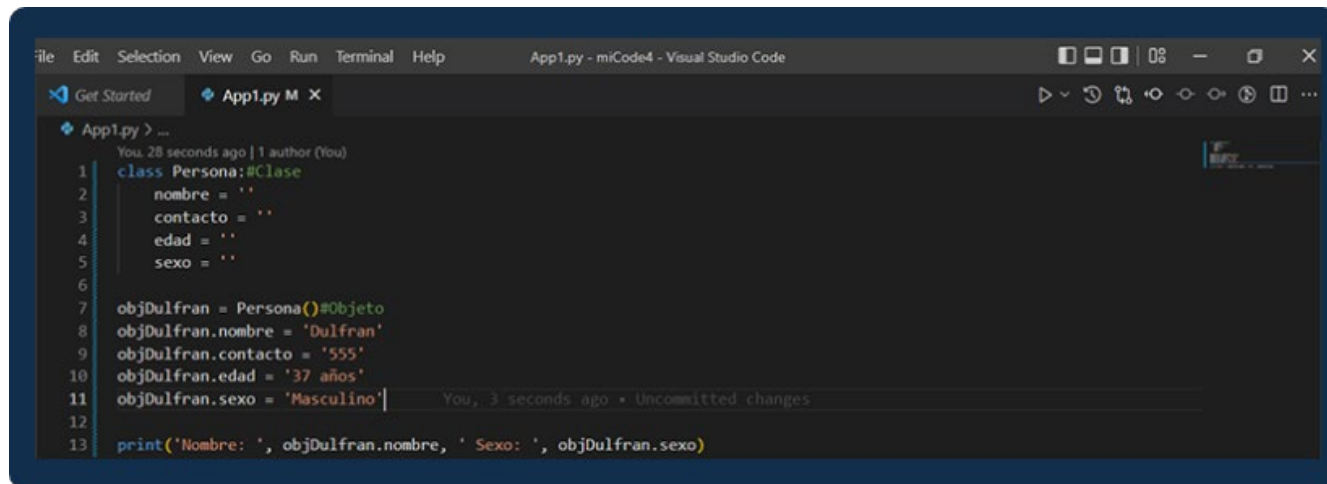
## Clases y objetos

Una clase es una plantilla para el objetivo de la creación de objetos de datos, según un modelo predefinido.

Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje. Ejemplo de una clase: Persona, Usuario, Cliente.

En Python las clases se crean de la siguiente manera: `class` y el nombre de la clase - `class Persona`:

Las clases están compuestas por atributos los cuales representan una característica de la misma; por ejemplo, si se sabe qué características tiene una persona, en este caso sería: Identificación, nombre, edad, contacto, sexo, entre otras. Estos atributos permiten almacenar la información dentro de la clase para luego poder ser utilizados cuando se requiera.



```
App1.py - miCode4 - Visual Studio Code
Get Started App1.py M X
App1.py > ...
You, 28 seconds ago | 1 author (You)
1 class Persona:#Clase
2     nombre = ''
3     contacto = ''
4     edad = ''
5     sexo = ''
6
7 objDulfran = Persona()#Objeto
8 objDulfran.nombre = 'Dulfran'
9 objDulfran.contacto = '555'
10 objDulfran.edad = '37 años'
11 objDulfran.sexo = 'Masculino'
12
13 print('Nombre: ', objDulfran.nombre, ' Sexo: ', objDulfran.sexo)
```

Como se puede observar, se ha creado una clase llamada Persona la cual tiene los atributos de nombre, contacto, edad y sexo. Luego se ha creado un objeto, ya que el mismo permite poder acceder a los atributos de la clase y modificar sus valores.

Haciendo por último, una impresión de la información contenida en los atributos nombre y sexo.

A continuación, se puede mostrar a través del siguiente video una ejemplificación de lo visto hasta este momento, en el que se irá también familiarizando con los conceptos y uso del aplicativo Visual Studio Code seleccionado:

**Video 3.** Iniciando e interactuando con nuestro entorno de desarrollo



[Enlace de reproducción del video](#)

**Síntesis del video: iniciando e interactuando con nuestro entorno de desarrollo**

Apreciado aprendiz, bienvenido a nuestro módulo de variables y comprensión de nuestro entorno Python. Como podemos observar, este es nuestro entorno de desarrollo que vamos a estar utilizando para nuestro ejercicio y para nuestra sesión.

Lo primero que vamos a aprender en esta sesión es cómo crear un nuevo archivo y cómo realizar la codificación en nuestro entorno de Visual Studio Code. Previamente, debemos realizar la instalación, la cual encontrarás en el componente número 1, donde explicamos paso a paso cómo realizar la instalación de esta aplicación.

Inicialmente, vamos a crear una nueva plataforma de control. En este caso, para crear nuestro archivo de desarrollo, nos vamos a la opción que dice "Archivo" y "Nuevo archivo". Una vez que ya tenemos desplegada esta opción, vamos a escoger la opción que dice "Python File" para crear nuestro archivo de codificación o nuestro archivo de código fuente. Presionamos clic y ya aquí tenemos nuestro lienzo para empezar a realizar la codificación de nuestro programa.

El siguiente paso que vamos a realizar es guardar nuestro archivo en una dirección física de nuestro disco duro. Por lo tanto, nos vamos a la opción que dice "Archivo", opción "Guardar", y luego nos va a desplegar esta ventana donde vamos a seleccionar la carpeta donde vamos a guardar nuestro documento. Para nuestro caso, vamos a utilizar la ventana del escritorio para guardar allí nuestro documento. Procedemos a crear el nombre de nuestro documento. Vamos a llamarlo "app\_mi\_primer\_programa". Debemos tener en cuenta que no creamos espacios en blanco, sino que en el caso particular podemos utilizar guión bajo para hacer la separación de cada uno de nuestros agregados en el nombre del programa.

Procedemos a dar clic en "Guardar", y como podemos observar en la parte superior, ya se encuentra desplegado nuestro programa dentro de la carpeta escritora.

Algo muy importante que debemos tener en cuenta en nuestra programación en Python es que, a diferencia de otros lenguajes de programación, no se requiere una declaración particular de las variables que vamos a utilizar. En nuestro caso, para crear una variable en Python, simplemente debemos colocar el nombre de la variable, por ejemplo, "numero", seguido del símbolo igual y luego el número o el dato que queremos asignar en ella. Por ejemplo, "numero = 5". Como podemos observar, hemos asignado el valor 5 a la variable "numero". Si podemos observar, el valor que nos ofrece en este caso esa variable. Debemos utilizar una instrucción llamada "print". Dentro de la instrucción "print", vamos a enviar la variable que nosotros deseamos que se muestre. En este caso, nuestra variable "numero". Vamos a colocar aquí, por ejemplo, "El resultado de la variable es:", dos puntos, terminamos la comilla, coma y el número de la variable, que en nuestro caso se llama "numero".

Para poder ejecutar una vez instalado nuestra plataforma, nuestro programa, procedemos a darle clic en el botón de ejecución. Una vez que le damos clic, empieza a desplegarse la plataforma de terminal. Como podemos observar, aquí nos está mostrando el resultado. En este caso, vamos a bajar un poco la terminal y aquí podemos observar el resultado de la variable 5, que es el valor que acabamos de asignar a nuestra variable.

Algo muy importante tener en cuenta al momento que estamos programando en nuestro lenguaje Python es eliminar la terminal una vez que hemos terminado de realizar la ejecución, para que no quede cargada en la memoria y no se nos



obstaculice en este caso ese proceso. Entonces, presionamos aquí donde dice "Kill terminal", le damos clic y automáticamente podemos continuar con nuestra codificación.

Ahora, ¿qué sucedería si en el caso deseamos que se capture una información o un dato capturado desde el teclado? En este caso, no es un dato que está por defecto asignado en una variable, sino que nosotros deseamos capturarlos del teclado. Para esto, debemos utilizar una instrucción que se llama "input". Pero debemos tener en cuenta que el dato que queremos capturar aquí es un dato numérico. Por lo tanto, colocamos "int", que significa entero, "input", que sería la instrucción para capturar el dato, y luego el mensaje. Entonces, "Ingresa un valor numérico". Y listo, aquí va a aparecer una opción en la cual le va a permitir al usuario poder ingresar el dato que nosotros queremos ingresar para nuestra variable. Aquí sería "valor\_numerico".

Y aquí tenemos en este caso ya todo capturado. Ahora, ¿qué vamos a hacer? Vamos a ejecutarlo a ver qué resultado nos ofrece. Ejecutamos...

Y observemos que aquí ya nos está pidiendo una información. A diferencia del caso de ejecución anterior, tenemos un mensaje que nos indica que debemos ingresar un dato. Vamos a colocar aquí el dato número 3. Le damos enter y nos muestra que el resultado de la variable es 3. Estamos mostrando aquí lo que el usuario ingresó desde el teclado.

Entonces, muy importante tener en cuenta cuando un dato se captura es del teclado. Luego, tener las opciones específicas de muestreo. Vamos a hacer un

ejercicio muy sencillo en el cual vamos a capturar dos números y luego vamos a mostrar el resultado de la suma de esos dos números.

Lo primero que requerimos es guardar esta información. "numero\_uno" será igual a "input" y acá le indicamos "Ingrese el primer número". Luego, vamos a indicarle al usuario que guarde o que nos ingrese el segundo número. "Ingrese el segundo número". Ya tenemos capturados los dos datos que el usuario nos va a ingresar desde el teclado.

Cuál es nuestro siguiente paso, realizar la operación matemática de la suma. Entonces, para ello, vamos a utilizar una nueva variable llamada "suma". Y a esta variable le vamos a asignar la operación matemática de suma del primer valor capturado y el segundo valor. En este caso, tendríamos "numero\_uno + numero\_dos". Y luego vamos a utilizar una instrucción que sería "print", la que estuvimos utilizando anteriormente para mostrar el resultado como tal de la aplicación de nuestra operación matemática. Le decimos "El resultado de la suma es:", dos puntos, y vamos a colocar aquí la variable que contiene ese valor como tal.

Entonces, aquí estaríamos capturando en nuestra primera línea el primer número. Capturamos el segundo número. Capturamos, en este caso, la operación matemática. Y luego vamos a mostrar el resultado al usuario en pantalla. Vamos a ejecutar nuestra aplicación...

Y aquí nos está solicitando que ingresemos el primer número. Vamos a colocar el número 5.

Ingresar el segundo número. Vamos a colocarle un 8.

Y aquí tenemos el resultado de la suma. En este caso, la operación matemática que se acaba de realizar es una suma, y nos está mostrando el resultado de esa operación matemática.

Bueno, mis estimados aprendices. Este es el proceso de codificación de una aplicación, uso de variables y operadores como tal. Espero que haya sido de mucha ayuda este vídeo y que podamos seguir trabajando en nuestro curso. Bienvenidos nuevamente y gracias por estar aquí.

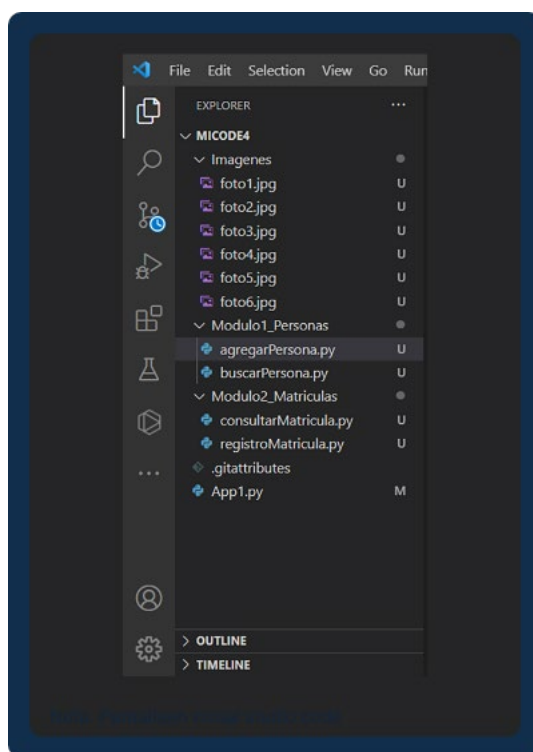
¡Felices códigos para todos!

## 8. Gestión de proyectos en Python

La gestión de proyectos en Python es la forma en cómo se pueden administrar diferentes espacios, carpetas y directorios que se puedan llegar a utilizar dentro del aplicativo a desarrollar.

Es importante tener en cuenta que esto va a permitir organizar de manera mucho más específica el programa, ya que de acuerdo con las necesidades que este requiera se crearán dichos espacios.

**Figura 4.** Espacios en un proyecto Python



Como se puede ver en la imagen del proyecto en Python, se requiere el uso de más de una carpeta para su organización.

En la imagen se visualiza la manera en cómo se puede gestionar y visualizar un proyecto en Python, a diferencia de los observados anteriormente este se encuentra distribuido en diferentes carpetas y directorios los cuales de acuerdo con la necesidad del programador puede agregar o quitar carpetas y directorios.

Es importante tener en cuenta que la gestión del proyecto no solamente se requiere para la realización y organización del proyecto en sí, sino que también permite realizar una interpretación de cómo está distribuido el programa en términos de lógica.

Un ejemplo de esto es que muchas veces en las empresas cuando los proyectos son demasiado extensos se procede a asignar responsabilidades funcionales a diferentes miembros del proyecto. Por ejemplo, a un programador se le puede asignar el módulo de factura y al otro el módulo de inventarios, ambos módulos pertenecen al mismo proyecto pero están en dos espacios distintos; esto ayuda adicionalmente a mejorar la seguridad y el orden dentro del proyecto.

Ahora bien, es importante comprender que el uso de las herramientas que se han estado explorando a través de este componente formativo juegan un papel importante, ya que permiten en este caso particular controlar ciertos aspectos del proyecto que requieren de un gran cuidado y verificación constante.

Como es el caso de los repositorios para el almacenamiento del código fuente del programa, se hace referencia a la herramienta GitHub, esta herramienta permite almacenar en un servidor de repositorio el código fuente del programa de una manera segura y sencilla. En muchas ocasiones uno de los problemas que se presentan en el trabajo de proyectos de software es precisamente dónde reposarán los archivos fuentes ya que estos son los que no solamente tienen las instrucciones y la lógica para

hacer funcionar el proyecto, sino que también van a permitir realizar la modificación o agregación de nuevas funcionalidades al programa.

En algunas ocasiones se almacenan estos archivos en discos duros y/o computadoras de respaldo las cuales pueden llegar a fallar en cualquier momento; por eso, la recomendación siempre está dirigida a la utilización de este tipo de servicios (como GitHub) para evitar pérdidas de archivos o en su defecto código fuente construido.

Se puede poner como ejemplo el siguiente escenario:

- Se está desarrollando un proyecto importante y se está construyendo una parte de la lógica del código la cual tiene un grado de complejidad que lleva varias horas de realización, pero se ha olvidado guardar los respectivos cambios.
- Se va el fluido eléctrico, esto significa que se ha perdido el trabajo realizado y en el peor de los casos, no se vuelve a recuperar.
- Se contempla la idea que en la computadora donde se está realizando la codificación del proyecto se puede averiguar. Pero si no se realizó una copia de seguridad puede afectar de gran manera el correcto desenlace del proyecto.
- Para evitar casos comunes como este ejemplo, los repositorios de código son utilizados por millones de programadores, ingenieros y entusiastas de la programación a nivel mundial, para mitigar el error de perder información que pueda comprometer el proyecto que estamos trabajando.

Todo lo desarrollado hasta este momento, lleva a la reflexión sobre la importancia de almacenar el código fuente de manera segura y confiable para evitar que este se pueda perder o pueda ser afectado por temas de virus o cualquier otro tipo de amenazas que se encuentran en el día a día de la industria del software.

También es importante la utilización de las buenas prácticas al momento de codificar, el uso de una aplicación para hacer que el programa mantenga un esquema de orden y nomenclaturas que permitan tanto su modificación como agregación de nuevas funciones de manera sencilla y eficiente.

Finalmente, se ha podido explorar las diferentes estructuras de control que se utilizan para validar las opciones y operaciones que se deben llevar a cabo dentro de un programa, el manejo de clases y objetos para organizar la información de una manera adecuada. Es importante tener en cuenta que el manejo de funciones permite separar las diferentes operaciones que se deben realizar en el programa y cómo este debe recibir la información que le entregará al usuario y la respuesta que se debe dar de acuerdo con los procesos necesarios por los requerimientos y necesidades del cliente para su desarrollo.

## Síntesis

Para garantizar la calidad del trabajo, la satisfacción de todos los públicos involucrados y cumplir con los requerimientos y necesidades del cliente, es necesario el cumplimiento de los pasos y puntos vistos a través del componente, para que, junto a una correcta Gestión del Proyecto, se logre un adecuado entorno de desarrollo y codificación; teniendo en cuenta lo siguiente:





## Material complementario

Tema	Referencia	Tipo de material	Enlace del Recurso
1. ¿Qué es un editor de código?	Visual Studio Code. (s.f.). Getting Started. Docs.	Manual	<a href="https://code.visualstudio.com/docs">https://code.visualstudio.com/docs</a>
5. Manejo y control de versiones	Dauzon, S. (2022). Git: Controle la gestión de sus versiones (conceptos, utilización y casos prácticos). Ediciones ENI.	Libro	<a href="https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_biblioteca_eniEPT3GIT">https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_biblioteca_eniEPT3GIT</a>
5. Manejo y control de versiones	Don Eber. (2021). Instalación de Git en Windows paso a paso   [2021 2022]. [Video] YouTube.	Video	<a href="https://www.youtube.com/watch?v=cYLapo1FFmA">https://www.youtube.com/watch?v=cYLapo1FFmA</a>
5. Manejo y control de versiones	Espitia, W. (2021). Cómo crear una cuenta de GitHub desde cero en 2021. [Video]. YouTube.	Video	<a href="https://www.youtube.com/watch?v=jwFSIEi_d7E">https://www.youtube.com/watch?v=jwFSIEi_d7E</a>
7. Creación y manejo de variables, estructuras de control, funciones, clases y objetos	Commit That Line! (2020). Las funciones en Python   ¿Para qué sirven y cómo se usan? [Video]. YouTube.	Video	<a href="https://www.youtube.com/watch?v=hLRoDs4wNCU">https://www.youtube.com/watch?v=hLRoDs4wNCU</a>
8. Gestión de proyectos en Python	Hinojosa Gutiérrez, Á. P. (2016). Python: Paso a paso. Ediciones de la U, Ra-Ma.	Libro	<a href="https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_ebooks0004998">https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_ebooks0004998</a>
8. Gestión de proyectos en Python	Python.org (s.f.). Python 3.10.5 documentation.	Manual	<a href="https://docs.python.org/3/">https://docs.python.org/3/</a>

## Glosario

**Clase:** es una plantilla para el objetivo de la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje.

**Editor de código:** es un programa que permite la creación y edición de archivos de código fuente.

**Entornos virtuales de desarrollo:** es la creación de directorios aislados los cuales permiten realizar la configuración y componentes que sólo requiere tu proyecto sin necesidad de afectar todo el sistema en general.

**Estructuras de control:** son utilizadas para validar y tomar decisiones dentro de un programa y que a su vez permiten la ejecución de bloques de código.

**Función:** las funciones son como una especie de caja las cuales pueden contener una serie de códigos los cuales pueden ser invocados las veces que sea necesaria dentro del programa.

**GitHub:** es un repositorio de código que se encuentra alojado en la nube y se utiliza para guardar archivos de código fuente y realizarle seguimiento a los mismos con respecto a los cambios o actualizaciones que estos puedan recibir.

## Referencias bibliográficas

Dauzon, S. (2022). Git: controle la gestión de sus versiones (conceptos, utilización y casos prácticos). 2ª Edición. ed. Ediciones ENI. [https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena\\_biblioteca\\_eniEPT3GIT](https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_biblioteca_eniEPT3GIT)

Hinojosa Gutiérrez, Á. P. (2016). Python: Paso a Paso. Primera Edición. ed. Ediciones De La U, Ra-Ma, [https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena\\_ebooks0004998](https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_ebooks0004998)

## Créditos

Nombre	Cargo	Regional y Centro de Formación
Claudia Patricia Aristizábal	Líder del Ecosistema	Dirección General
Rafael Neftalí Lizcano Reyes	Responsable de Línea de Producción	Centro Industrial del Diseño y la Manufactura - Regional Santander
Dulfran Antonio Montaña Montaña	Experto Temático	Centro De Diseño Y Metrología - Regional Distrito Capital
Zvi Daniel Grosman Landáez	Diseñadora instruccional	Centro de Gestión Industrial - Regional Distrito Capital
Andrés Felipe Velandia Espitia	Asesor Metodológico	Centro de Diseño y Metrología - Regional Distrito Capital
Jhon Jairo Rodríguez Pérez	Corrector de estilo	Centro de Diseño y Metrología - Regional Distrito Capital
Yerson Fabian Zarate Saavedra	Diseñador de Contenidos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Emilsen Alfonso Bautista	Desarrollador Full-Stack	Centro Industrial del Diseño y la Manufactura - Regional Santander
Zuleidy María Ruiz Torres	Validador de Recursos Educativos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Luis Gabriel Urueta Álvarez	Validador de Recursos Educativos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Daniel Ricardo Mutis Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro Industrial del Diseño y la Manufactura - Regional Santander