Actividad ED3.4: Personalizando Visual Studio Code

# Introducción

Esta práctica tiene como objetivo personalizar Visual Studio Code para adaptarlo a tus necesidades como entorno de desarrollo en **Python**. Configurarás el editor, crearás atajos de teclado, instalarás y configurarás extensiones, gestionarás perfiles y snippets, entre otras personalizaciones, para mejorar la experiencia de programación.

# Objetivos de la Práctica

1. Configurar y personalizar el entorno de trabajo en **VS Code** para el desarrollo en Python.
2. Crear y gestionar perfiles de configuración, incluyendo la exportación de un perfil.
3. Usar y personalizar atajos de teclado y snippets de código.
4. Configurar autosave y vistas de explorador de archivos.
5. Integrar IntelliSense y configurar un linter y reglas de estilo específicas para Python.

# Instrucciones

## Paso Inicial: Configuración del Idioma en Visual Studio Code

Si quieres tener Visual Studio Code en español, debes realizar las siguientes acciones:

#### Abrir el Administrador de Extensiones:

* + En Visual Studio Code, dirígete al panel izquierdo y selecciona el ícono de

**Extensiones** (o usa el atajo Ctrl+Shift+X).

#### Buscar el Paquete de Idioma Español:

* + En la barra de búsqueda de la sección de Extensiones, escribe "Spanish Language Pack for Visual Studio Code" o simplemente "Español".
  + Selecciona el paquete **"Spanish Language Pack for Visual Studio Code"**

(generalmente desarrollado por Microsoft) y haz clic en **Instalar**.

#### Cambiar el Idioma a Español:

* + Una vez instalado el paquete de idioma, aparecerá una notificación en la esquina inferior derecha con la opción **Cambiar el idioma de Visual Studio Code a Español**. Haz clic en esa opción.
  + VS Code se reiniciará automáticamente para aplicar el cambio de idioma.

#### Verificar que el Editor esté en Español:

* + Cuando Visual Studio Code se reinicie, verifica que todos los menús y opciones estén en español. Puedes comprobar esto observando el menú principal o las opciones de configuración.

**Nota**: Si en el futuro deseas cambiar el idioma nuevamente, puedes hacerlo desde Archivo > Preferencias > Configuración > Configuración de idioma.

## Configuración Inicial del Editor de Código

#### Tema y Personalización del Editor:

* + - Cambia el tema de color del editor en Archivo > Preferencias > Tema de Color.
    - Selecciona un tema oscuro o claro de acuerdo a tus preferencias y ajusta la configuración de fuente en Archivo > Preferencias > Configuración
      * Text Editor > Font.
        + Puedes intentar instalar la fuente JetBrains Mono en tu Sistema Operativo y, posteriormente, configurarla en VSCode. De esta manera, el editor lucirá muy similar a Intellij IDEA.

#### Configurar Autosave:

* + - En Archivo > Preferencias > Configuración, busca "files.autoSave" y selecciona la opción afterDelay para que el editor guarde automáticamente los archivos después de un breve periodo de inactividad.

## Configuración de Atajos de Teclado

#### Personalización de Atajos:

* + - Ve a Archivo > Preferencias > Métodos abreviados de Teclado.
    - Cambia los atajos de teclas para abrir el terminal (Ctrl + ñ), comentar líneas (Ctrl + /), y ejecutar un archivo Python (F5).
    - Documenta en una tabla los atajos personalizados que cambiaste, con la acción asociada a cada uno.

## Instalación y Gestión de Extensiones

#### Añadir Extensiones Recomendadas:

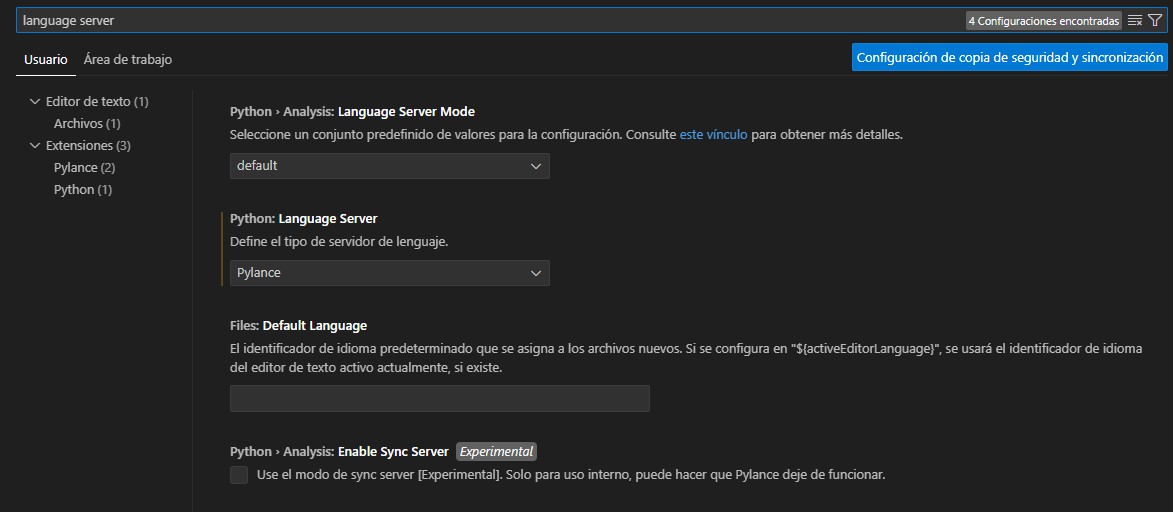
* + - Busca e instala las siguientes extensiones:
* **Python** (Microsoft)
* **Pylance** (para mejorar IntelliSense en Python)
* **Pylint** (para soportar el linter pylint)
* **autopep8** (para disponer de style guide pep8 en VSCode)
* **autoDocstring** (para crear docstrings automáticamente en funciones Python)

#### Eliminar Extensiones No Necesarias:

* Revisa las extensiones instaladas y elimina las que no necesites para este entorno de Python.

#### Configurar y Ajustar las Extensiones:

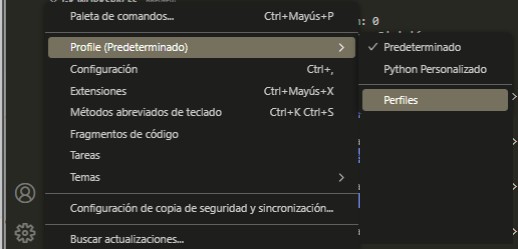
* + - Configura **Pylance** para que IntelliSense ofrezca autocompletado y sugerencias en Python. Puedes hacer esto desde Archivo > Preferencias > Configuración > Python > Language Server y seleccionando **Pylance**.



## Crear y Gestionar Perfiles de Configuración

#### Crear un Perfil de Configuración:

* + - Entra a Perfil > Crear Perfil de Configuración y guarda la configuración actual como "Python Personalizado".

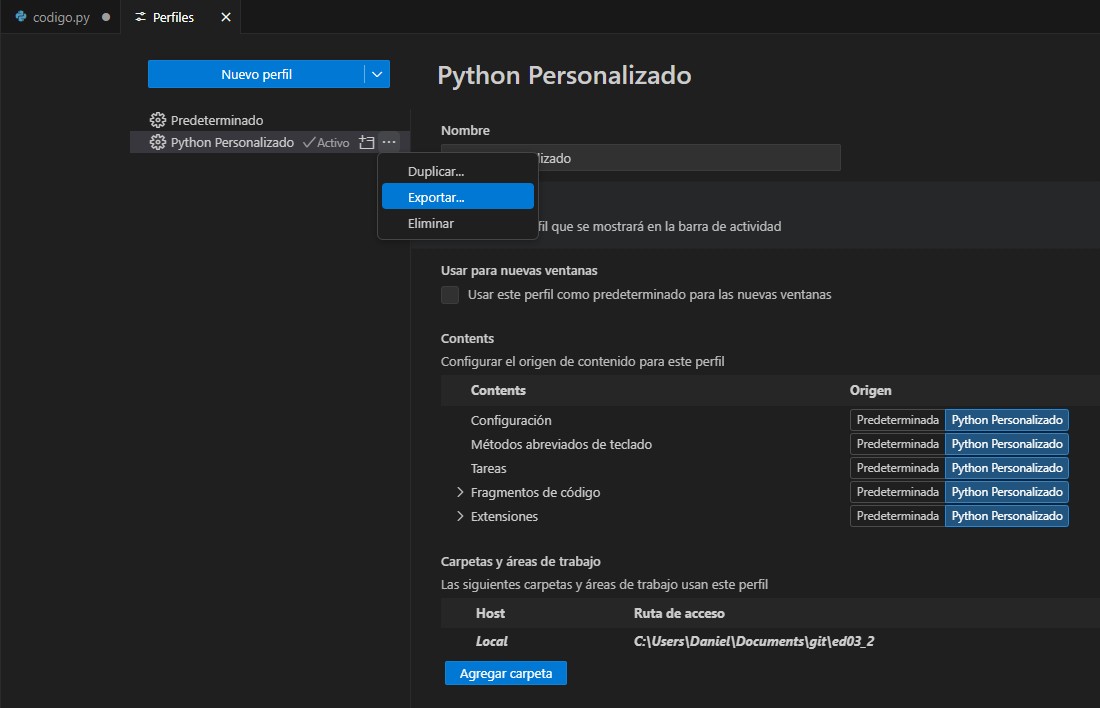


#### Modificar el Perfil:

* + - Realiza cambios en la configuración, como ajustar temas, extensiones y atajos, y guarda el perfil.

#### Exportar el Perfil:

* + - Exporta tu perfil desde Perfil > Exportar Perfil... y guarda el archivo en un lugar accesible. Esto generará un archivo .code-profile que puedes compartir con otros.



## Crear y Usar Snippets de Código

#### Crear un Snippet para una Función Python:

* + - Ve a Archivo > Preferencias > Configurar fragmentos de código
    - Selecciona Python y crea un snippet para una función que genere un docstring automáticamente, similar al siguiente:

{

"Python Function": { "prefix": "func", "body": [

"def ${1:func\_name}(${2:params}):", " \"\"\"",

"

"

"

],

${3:Description of the function}",

\"\"\"",

$0",

"description": "Snippet para una función Python con docstring"

}

}

Otro snippet, por ejemplo, permitiría crear un print con una variable por parámetro:

{

"Print with params": { "prefix": "pr2", "body": [

"print(\"${1:print\_msg}\", ${2:params})", " $0"

],

"description": "Snippet crear un print con parámetro"

}

}

#### Prueba el Snippet:

* + - En un archivo Python, escribe func o pr2 y observa cómo el snippet se expande automáticamente al tipo de código especificado.

*¿Qué otros snippet de código te parece interesante definir?*

## Configuración de la Vista del Explorador de Archivos

#### Personalizar la Vista del Explorador:

* + - En Archivo > Preferencias > Configuración, busca "explorer.compactFolders" y desactívalo si deseas ver todas las carpetas completamente expandidas.
    - Activa "explorer.sortOrder" para que los archivos Python se muestren primero.

## 7 Configuración de IntelliSense, Linter y Style Guide

### IntelliSense: Autocompletado y Sugerencias de Código

**IntelliSense** es una tecnología de autocompletado que ayuda a los desarrolladores mientras escriben código. En Visual Studio Code, **IntelliSense sugiere variables, funciones, módulos**

**y sintaxis** a medida que se escribe, ayudando a reducir errores de escritura y a mejorar la velocidad de programación.

Para Python, la extensión **Pylance** es el motor de IntelliSense recomendado en VS Code:

* Proporciona autocompletado específico de Python, basándose en los módulos, funciones y variables que hayas definido o importado en tu archivo.
* Ofrece información adicional sobre cada función o clase (por ejemplo, qué parámetros requiere y su orden).
* Mejora la navegación en el código, sugiriendo opciones relevantes en función del contexto.

**Extensión necesaria**: [Pylance](https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance).

### Linter: Análisis de Errores y Problemas en el Código

Un **linter** es una herramienta que analiza el código en busca de errores, advertencias y patrones que podrían causar problemas de ejecución o de estilo. En Python, los linters ayudan a detectar problemas comunes antes de que el código sea ejecutado, como:

* Variables no utilizadas.
* Referencias a funciones o variables inexistentes.
* Estilos de código que no cumplen con las normas establecidas (como PEP8, si está configurado).

El linter **Pylint** es comúnmente utilizado para Python y es compatible con VS Code:

* **Pylint** revisa cada línea de código en busca de errores sintácticos y recomendaciones de estilo.
* Puede configurarse para seguir las normas de **PEP8**, que es el estándar de estilo para Python, y marcar advertencias cuando algo no cumple con estas normas.

**Extensión necesaria**: [Python](https://marketplace.visualstudio.com/items?itemName=ms-python.python) (la extensión Python de Microsoft, que incluye la opción de configurar Pylint como linter).

### Reglas de Estilo (PEP8): Normas de Formato para Escribir Código

**PEP8** es una guía de estilo que especifica cómo debería estructurarse el código Python para que sea más fácil de leer y mantener. Establece reglas sobre:

* El uso de espacios y tabulaciones.
* Longitud de las líneas.
* Espacios en blanco entre funciones y clases.
* Nombres de variables, funciones y constantes.

Seguir PEP8 ayuda a mantener un estilo consistente en el proyecto, facilitando la colaboración y la lectura del código.

**Extensión necesaria**: [Autopep8](https://marketplace.visualstudio.com/items?itemName=ms-python.autopep8)

### Cómo Usar autopep8 en Visual Studio Code

#### Instala la Extensión de Python:

* + Asegúrate de tener instalada la extensión **Python** de Microsoft, ya que permite integrar y ejecutar autopep8.

#### Instala autopep8:

* + Para usar autopep8, primero debes instalarlo en tu entorno de Python.

En la terminal de VS Code, ejecuta:

pip install autopep8

#### Configura autopep8 en VS Code:

* + Ve a Archivo > Preferencias > Configuración.

Busca "Python Formatting Provider" y selecciona **autopep8** como el proveedor de formato:

"python.formatting.provider": "autopep8"

También puedes ajustar la configuración para que el código se formatee automáticamente al guardar:

"editor.formatOnSave": true

#### Formatear el Código Manualmente:

* + También puedes formatear el código en cualquier momento manualmente:
    - Haz clic derecho en el editor y selecciona **Format Document**.
    - O usa el atajo de teclado (generalmente Shift + Alt + F en Windows/Linux o Shift + Option + F en macOS).

### Cuándo Usar autopep8 en Lugar de un Linter

* **autopep8** es ideal para **corregir automáticamente** el estilo y aplicar las normas PEP8 sin intervención manual.
* **Pylint** o **Flake8**, en cambio, son útiles para **verificar** el cumplimiento de estilo y detectar errores y advertencias, pero **no ajustan el código automáticamente**.

## Consejos y Recursos

* **Atajos de teclado**: Puedes consultar la lista de atajos de teclado predeterminados en

Archivo > Ayuda > Ver todos los comandos.

* **Linter y PEP8**: La guía de estilo PEP8 se encuentra [aquí](https://www.python.org/dev/peps/pep-0008/) y es útil para ajustar las configuraciones de Pylint.
* **IntelliSense y Pylance**: Aprovecha la ayuda de autocompletado para ver sugerencias en tiempo real y comprobar los parámetros de las funciones Python en el editor.

# Entrega

Debes entregar:

#### Informe, en formato pdf, incluyendo

* + **Capturas de pantalla** de las personalizaciones realizadas:
    - Vista de la interfaz personalizada.
    - Atajos de teclado modificados.
    - Snippets creados y funcionando.
    - Configuración de autosave y explorador de archivos.
  + Pasos seguidos para configurar cada parte.
  + Principales problemas encontrados y cómo los solucionaron.
  + Ejemplo de cómo IntelliSense y el linter funcionan en un archivo de Python.
  + Cualquier otro aspecto a destacar.
* **Archivo de perfil exportado** (.code-profile).

Debes entregar el informe con el nombre

#### Apellidos\_nombre\_ed03\_4.pdf

La actividad será calificada como **apto** o **no apto**.