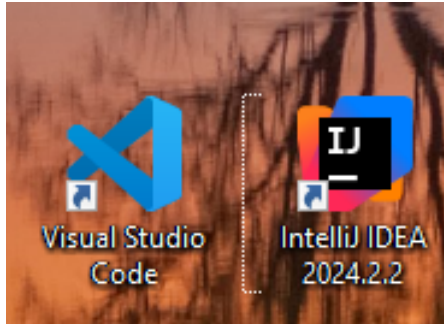


2.1. Instalación de entornos de desarrollo, propietarios y libres (CE 2.a)

Tarea individual:

Elige dos entornos de desarrollo: Instala ambos en tu equipo y captura pantallas del proceso de instalación.



Al tener estos dos entornos de desarrollo ya instalados, no he podido hacer captura paso a paso del proceso de instalación, pero aquí está la evidencia de que ambos se encuentran instalados perfectamente en mi equipo.

Preguntas evaluativas:

1.-¿Qué diferencias encontraste en el proceso de instalación entre el IDE propietario y el libre?

En el proceso de instalación del software propietario (IntelliJ), tuvimos que luchar toda la clase para conseguir una licencia aportando nuestros correos del instituto para decirles que somos un centro educativo y que necesitamos una licencia por un año, sin embargo, en el libre, tan solo tuvimos que dirigirnos a la página de descarga y lo descargamos sin necesitar ningún permiso de licencia ni pagar ni nada por el estilo.

2.-¿Qué ventajas identificaste en cada uno de los entornos durante la instalación?

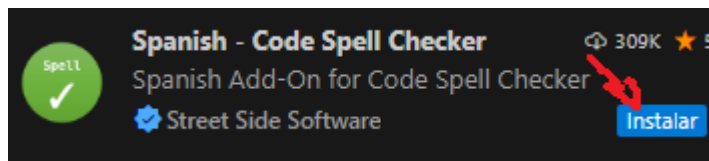
Que visual studio code es muchísimo más sencillo de instalar.

2.2. Gestión de módulos y extensiones en el entorno de desarrollo (CE 2.b)¶

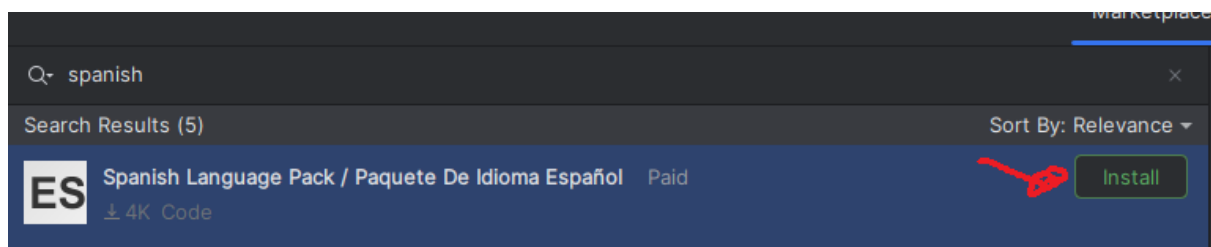
Tarea individual:

En cada IDE, agrega extensiones o módulos que amplíen su funcionalidad. Por ejemplo, suponiendo habiendo trabajado con VSC y IntelliJ IDEA, instala una extensión para trabajar con **Python** en **Visual Studio Code** o un plugin para **Kotlin** en **IntelliJ IDEA**.

En Visual Studio



En IntelliJ



Preguntas evaluativas:

1.-¿Cómo fue el proceso de instalación de extensiones o módulos en cada IDE?

En ambos siento que es bastante fácil e intuitivo encontrar los diferentes plugins ya que ambos entornos poseen de una pestaña llamada “plugins” que hace posible se puedan buscar las diferentes extensiones que suben los usuarios a la plataforma.

2.-¿Qué beneficios proporcionan las extensiones o plugins que instalaste para el desarrollo de tus proyectos?

El de visual studio code integra un autocorrector al español, ideal para poder documentar los códigos sin faltas de ortografía y el de IntelliJ, pone IntelliJ en español.

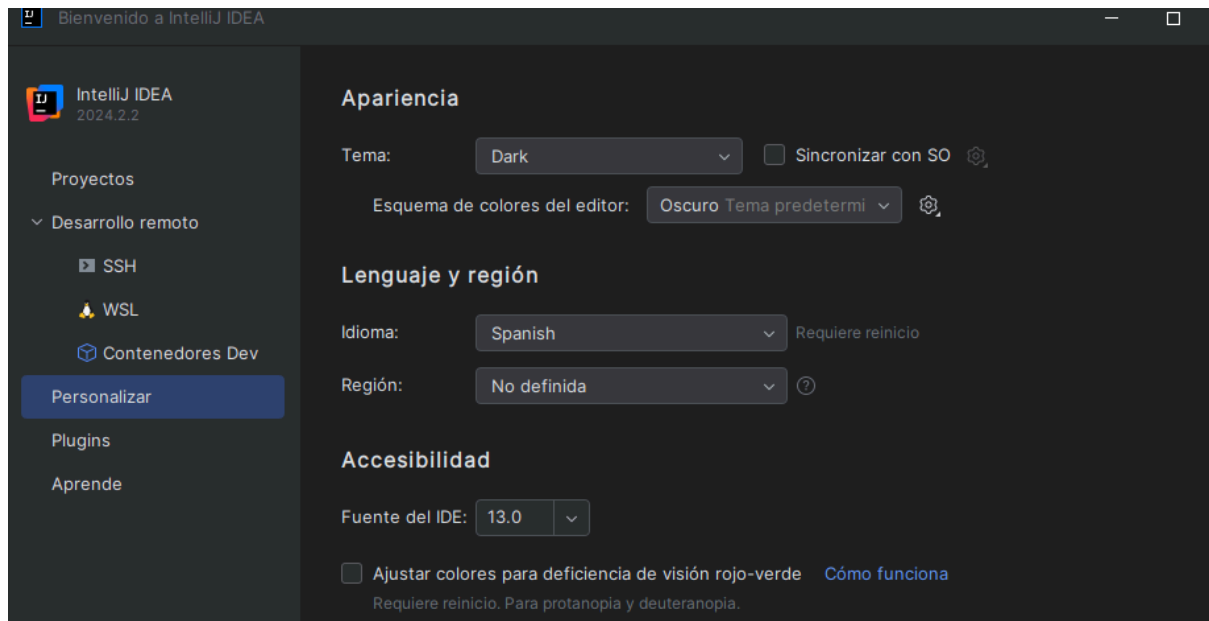
Evidencia: Captura de pantalla del panel de extensiones o plugins instalados en cada IDE.

2.3. Personalización y automatización del entorno (CE 2.c)

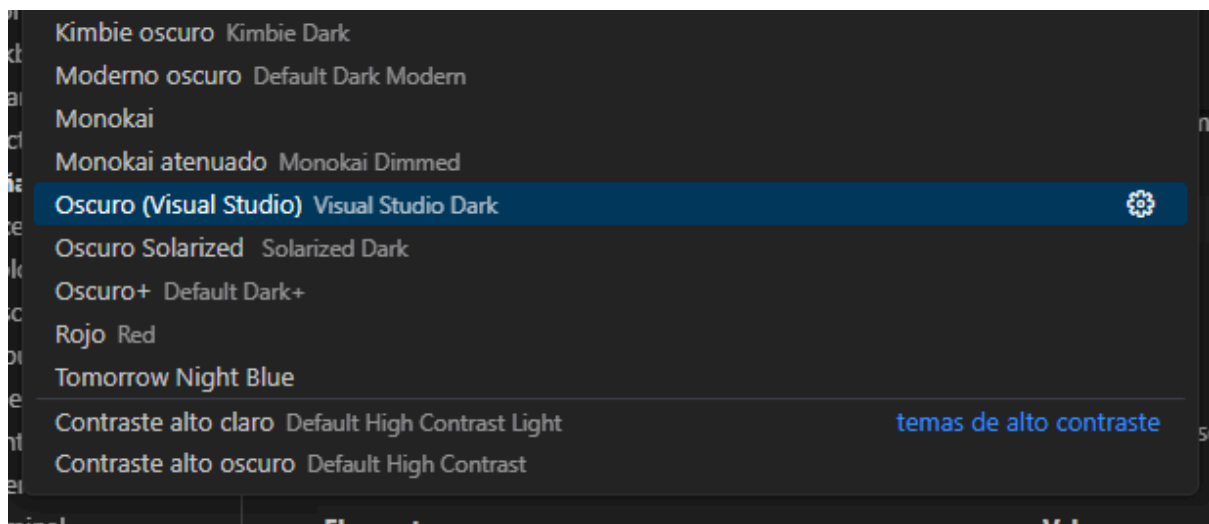
Tarea individual:

Personaliza el entorno de trabajo en cada IDE (tema, atajos de teclado) y automatiza una tarea, como la ejecución de pruebas o la compilación de código.

IntelliJ



Visual Studio Code



Preguntas evaluativas:

1.-¿Qué aspectos del entorno personalizaste y cómo mejoró tu experiencia de desarrollo?

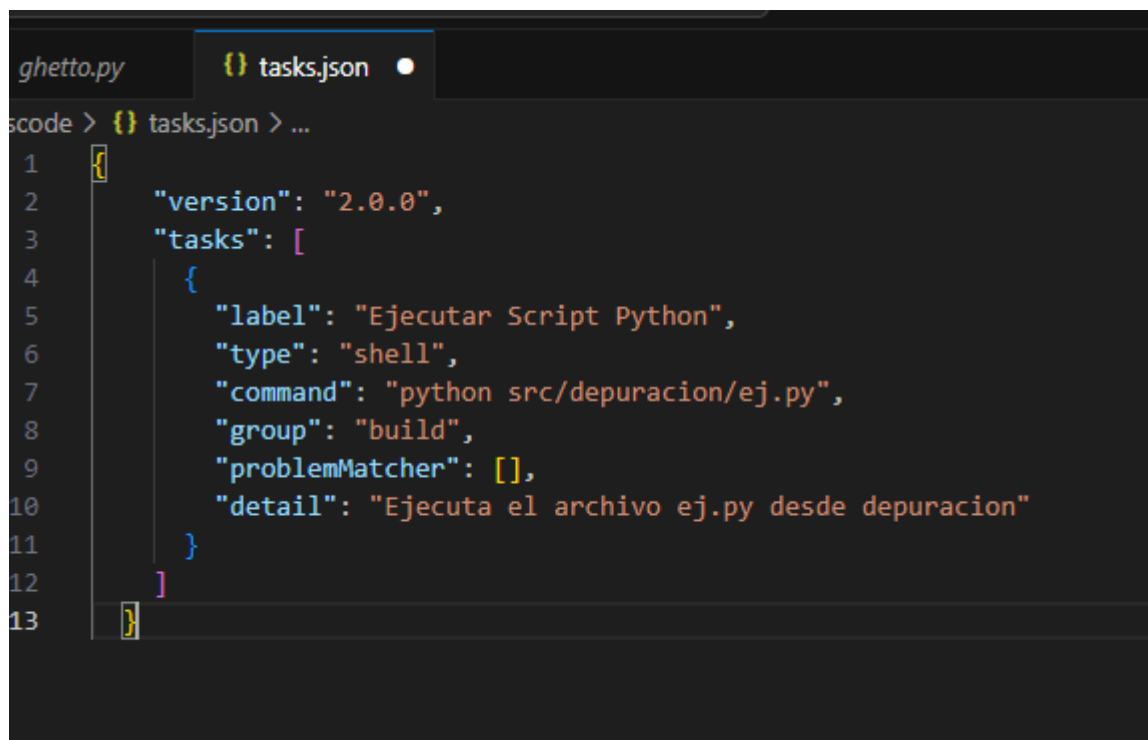
En IntelliJ, he cambiado el idioma y el color de fondo. En Visual Studio Code, cambié el color de fondo.

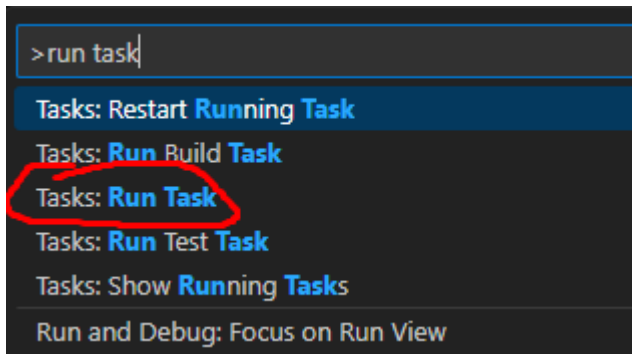
2.-¿Cómo configuraste la automatización de tareas y en qué te benefició durante el trabajo?

VISUAL STUDIO CODE



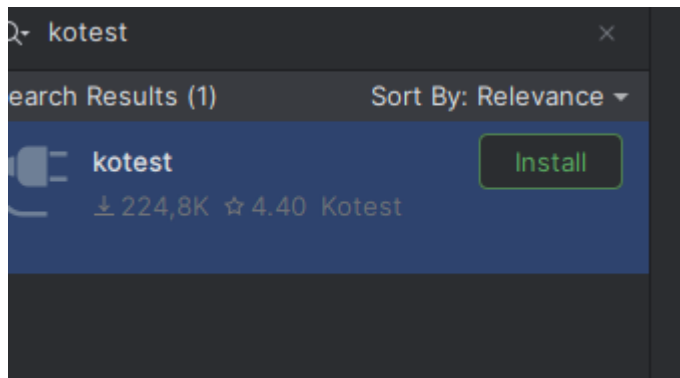
Creamos carpeta .vscode en nuestro proyecto y creamos un archivo llamado "tasks.json", en este archivo configuraremos con ciertos parametros que tarea queremos automatizar.





Pulsaremos control shift p y buscaremos “run task” para seleccionar la tarea creada.

EN INTELLIJ



Ahora activaremos la opción de que cada vez que guardemos o hagamos cambios se automaticen las pruebas.

Me parece que nos beneficia a la hora de repetir tareas pero también es verdad que en kotlin es muy costoso y rebuscado el como se hace en mi opinión.

Evidencia: Captura de pantalla mostrando la personalización del entorno y la automatización de tareas en cada IDE. El antes y el después.

2.4. Configuración del sistema de actualización del entorno de desarrollo (CE 2.d)

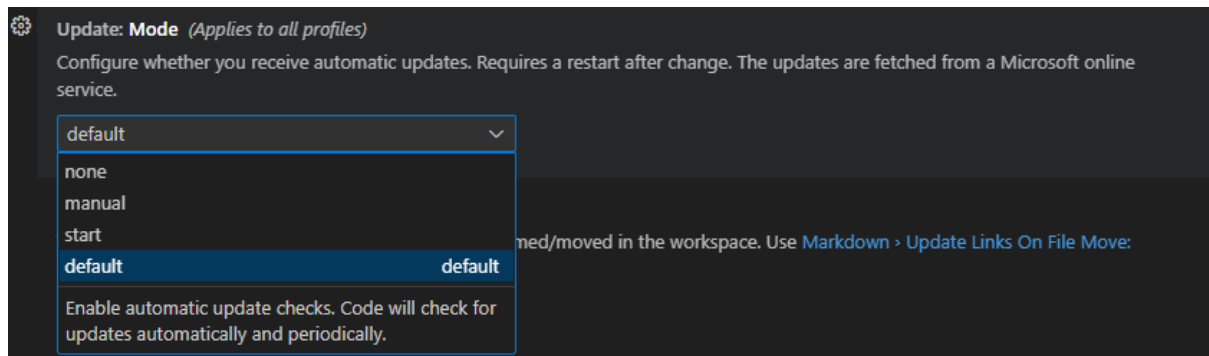
Tarea individual:

Configura el sistema de actualizaciones automáticas o manuales en ambos IDEs para asegurarte de que están al día con las últimas versiones y mejoras.

Preguntas evaluativas:

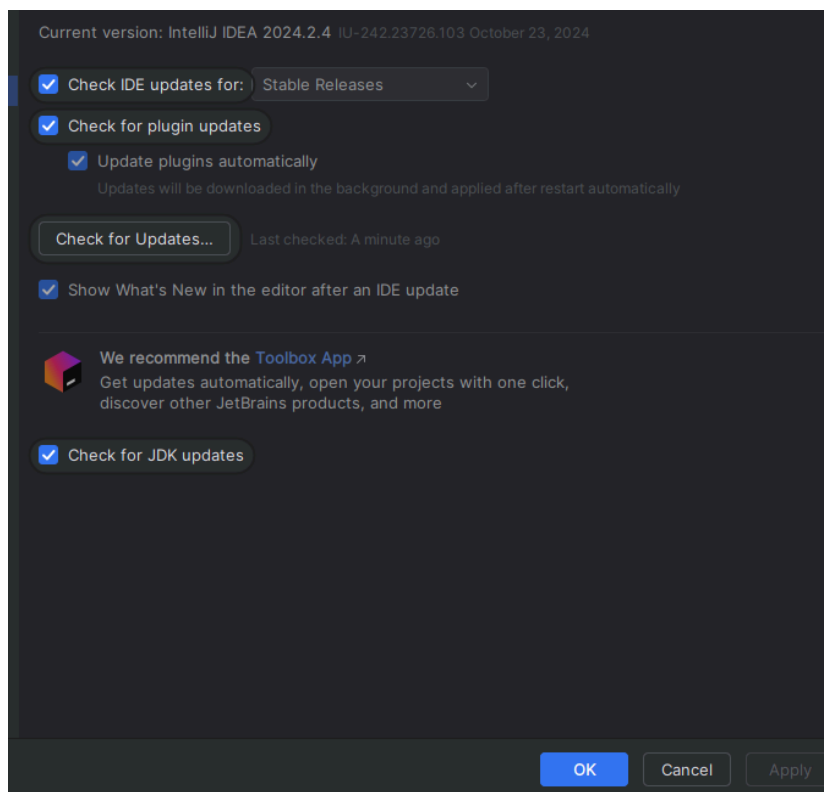
-¿Cómo configuraste las actualizaciones automáticas en cada IDE?

EN VISUAL



Vamos a las opciones y nos dirigimos a update mode, aquí ponemos default para que se actualice automáticamente.

EN INTELIJ



Vamos a settings>update y marcamos estas opciones y le damos a aplicar.

-¿Por qué es importante mantener el IDE actualizado en proyectos de desarrollo?

Porque si no puede dar problema en los plugins, las bibliotecas que hayamos cogido para nuestros proyectos, errores a la hora de corregir el código entre muchas cosas que pueden dar pie a generar problemas de incompatibilidad por no tener la última versión.

Evidencia: Captura de pantalla de la configuración de actualizaciones en cada IDE.

2.5. Generación de ejecutables a partir de código fuente en distintos lenguajes en un mismo IDE (CE 2.e)¶

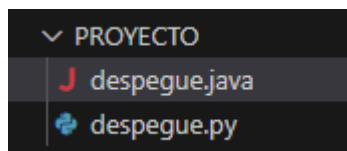
Tarea individual:

Escribe un programa que cuente de 10 a 0 y luego imprima "¡Despegue!". Usa un solo IDE para generar y ejecutar este programa en dos lenguajes diferentes (por ejemplo, Java y Kotlin en IntelliJ IDEA).

Preguntas evaluativas:

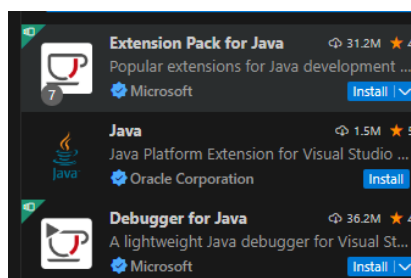
-¿Cuál fue el proceso para ejecutar el mismo programa en diferentes lenguajes dentro del mismo IDE?

Creemos una carpeta que contenga los dos archivos (podríamos incluso organizar más nuestro proyecto creando una carpeta para los archivos java y otros para los de python)



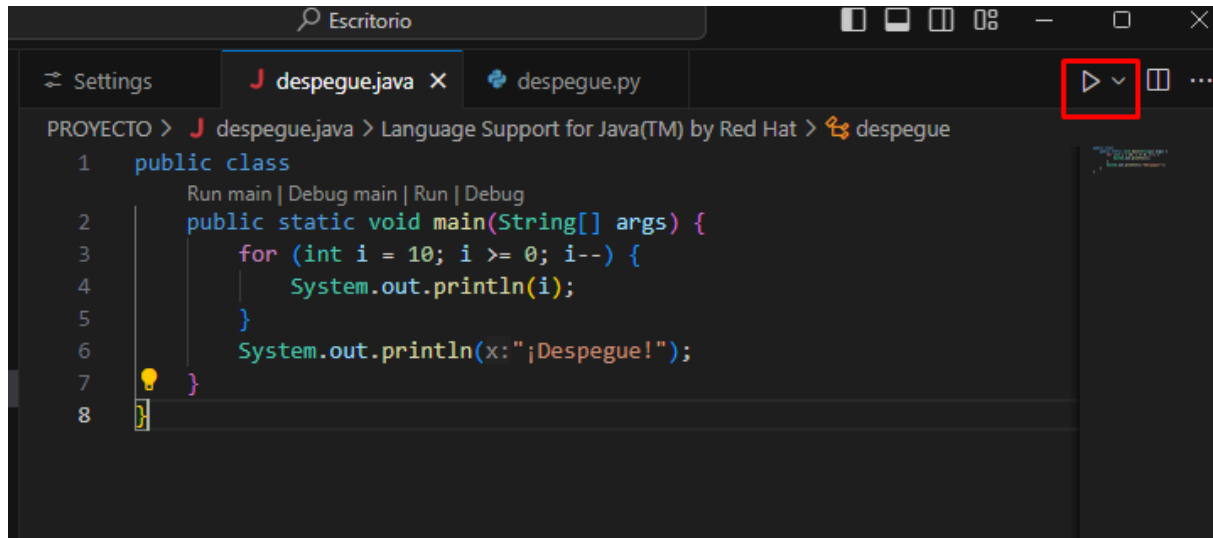
INSTALAMOS EXTENSIONES

Como yo ya tenía las extensiones correspondientes de python descargaré solo las de java.



Después de instalar el compilador y los plugins correspondientes, ya nos dejará ejecutar nuestros códigos:

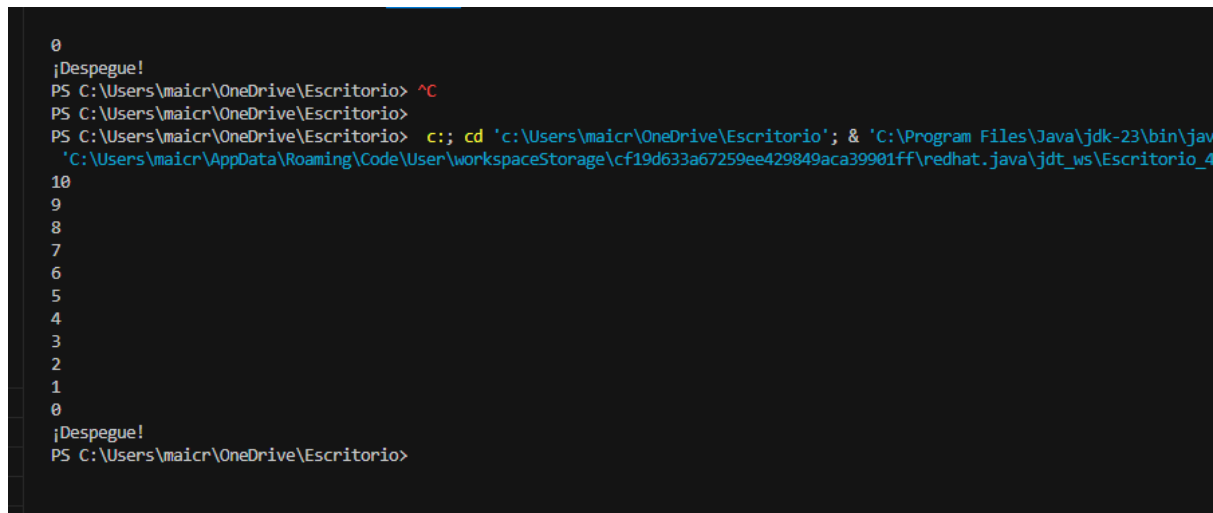
JAVA



The screenshot shows an IDE window titled 'Escritorio'. The tab bar shows 'despegue.java' and 'despegue.py'. The 'despegue.java' tab is active, displaying the following Java code:

```
1 public class
2     Run main | Debug main | Run | Debug
3     public static void main(String[] args) {
4         for (int i = 10; i >= 0; i--) {
5             System.out.println(i);
6         }
7         System.out.println(x: "¡Despegue!");
8     }
```

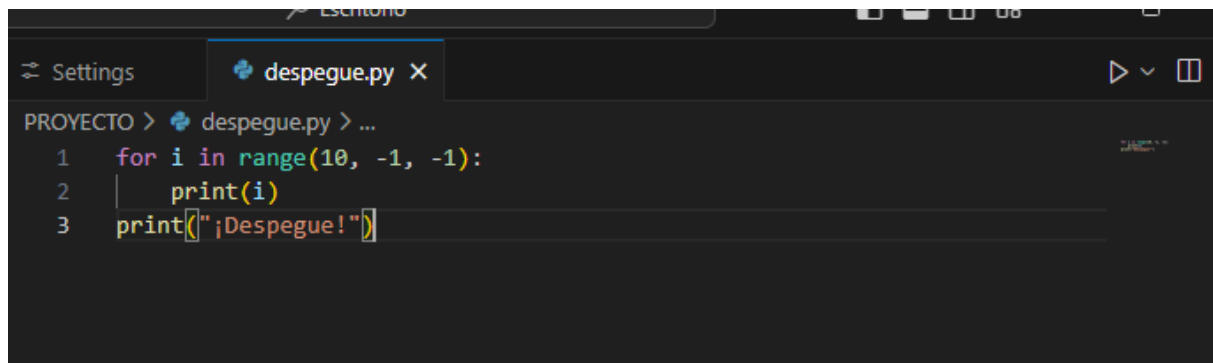
A red box highlights the Run button (a play icon) in the top right corner of the IDE window.



The screenshot shows a terminal window with the following output:

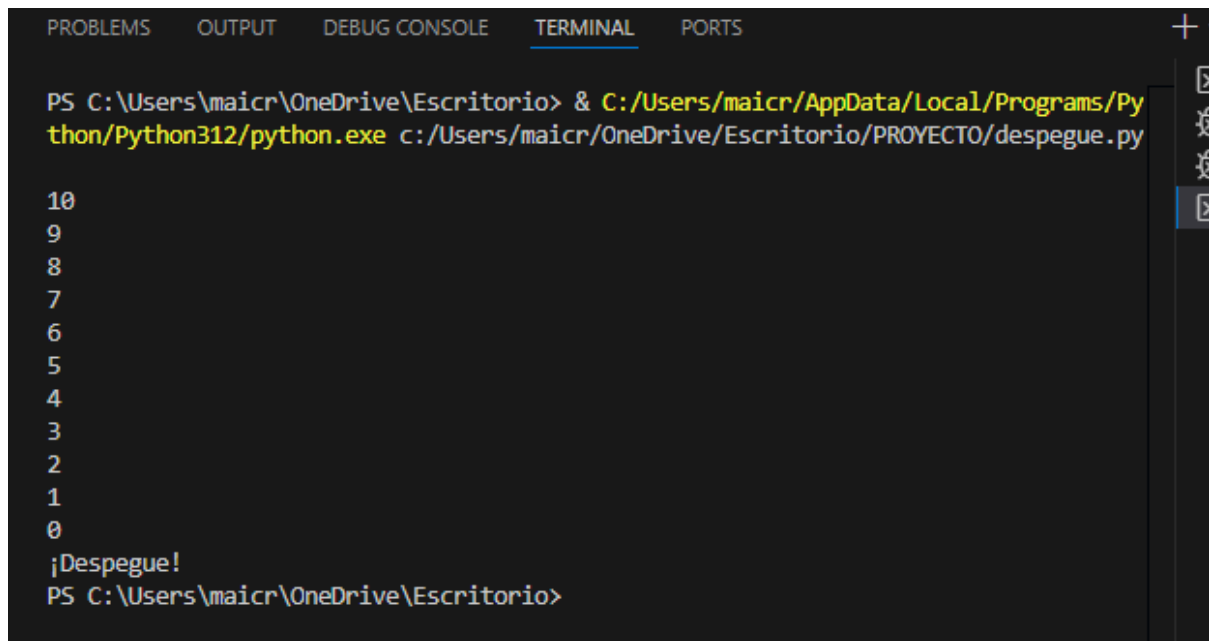
```
0
¡Despegue!
PS C:\Users\maicr\OneDrive\Escritorio> ^C
PS C:\Users\maicr\OneDrive\Escritorio>
PS C:\Users\maicr\OneDrive\Escritorio> c:; cd 'c:\Users\maicr\OneDrive\Escritorio'; & 'C:\Program Files\Java\jdk-23\bin\jav
'C:\Users\maicr\AppData\Roaming\Code\User\workspaceStorage\cf19d633a67259ee429849aca39901ff\redhat.java\jdt_ws\Escritorio_4
10
9
8
7
6
5
4
3
2
1
0
¡Despegue!
PS C:\Users\maicr\OneDrive\Escritorio>
```

PYTHON



The screenshot shows an IDE window titled 'Escritorio'. The tab bar shows 'despegue.py'. The 'despegue.py' tab is active, displaying the following Python code:

```
1 for i in range(10, -1, -1):
2     print(i)
3 print("¡Despegue!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\maicr\OneDrive\Escritorio> & C:/Users/maicr/AppData/Local/Programs/Python/Python312/python.exe c:/Users/maicr/OneDrive/Escritorio/PROYECTO/despegue.py

10
9
8
7
6
5
4
3
2
1
0
¡Despegue!
PS C:\Users\maicr\OneDrive\Escritorio>
```

-¿Qué diferencias encontraste en la generación del ejecutable entre los dos lenguajes?

En java hay muchas mas líneas de código para hacer las cosas y tienes que especificarle muchísimas cosas que en python ya te lo dan todo casi hecho.

Evidencia: Captura de pantalla mostrando la ejecución del programa en ambos lenguajes dentro del mismo IDE.

2.6. Generación de ejecutables con diferentes IDEs a partir del mismo código fuente (CE 2.f)¶

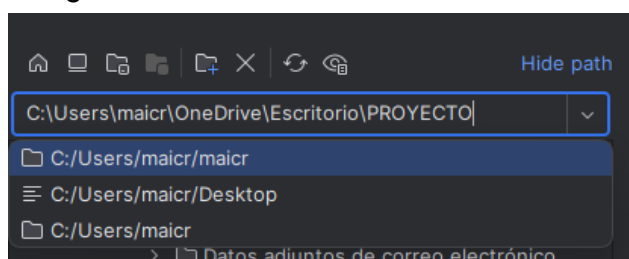
Tarea individual:

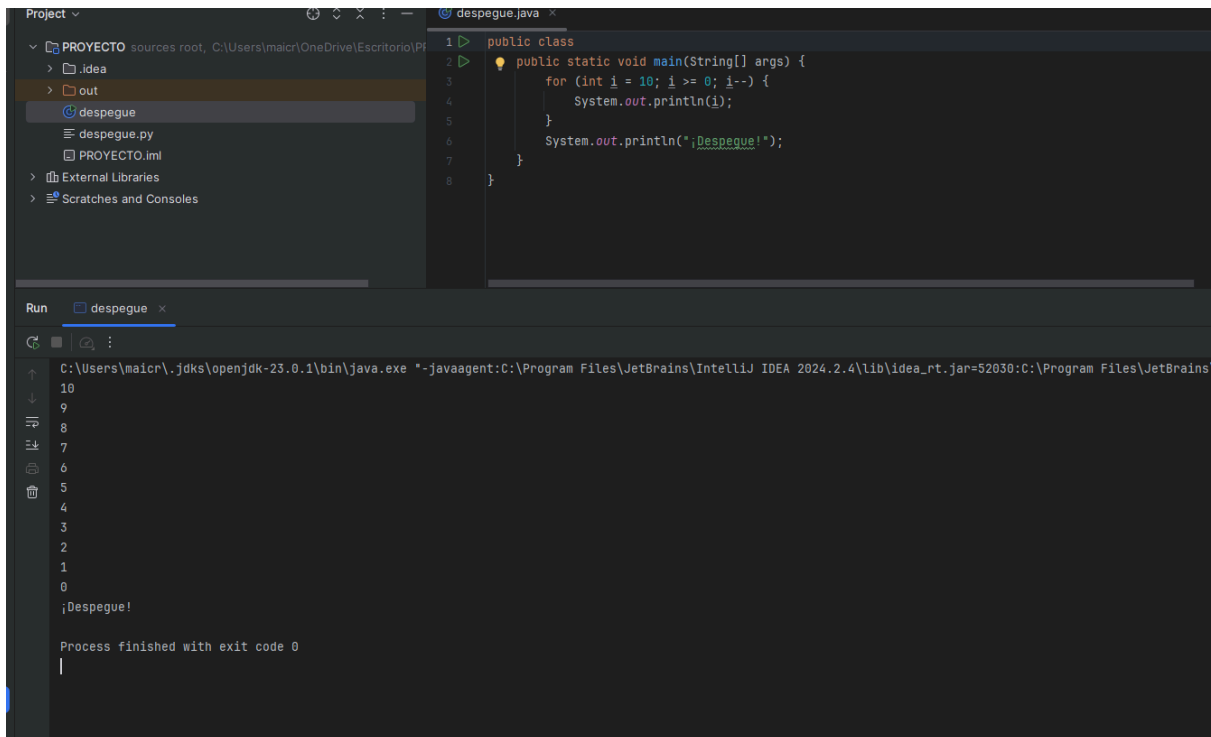
Escribe un programa en Python que cuente de 10 a 0 y luego imprima "¡Despegue!". Ejecuta el programa en los IDEs seleccionados.

Preguntas evaluativas:

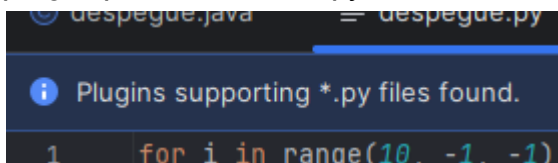
-¿Qué diferencias encontraste al ejecutar el mismo código fuente en diferentes IDEs?

Con la misma carpeta usada en visual studio code, vamos a probar estos dos códigos de nuevo en IntelliJ

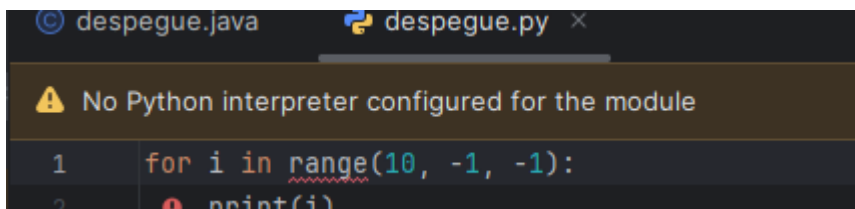




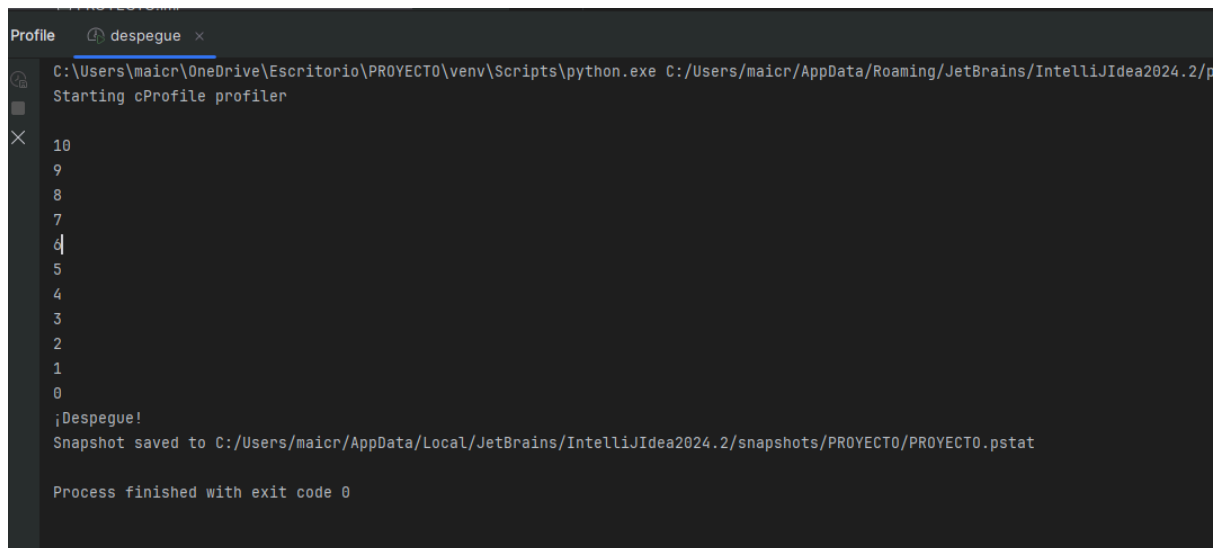
Parece ser que el código de Java, lo he logrado ejecutar sin necesidad de ningún plugin, probemos el de python.



Vemos que para python, nos piden un plugin. Descargamos el plugin. Una vez lo instalemos, reiniciamos y ya veremos que podemos ejecutar el código. O eso creía...



Ahora IntelliJ nos dirá que no se encuentra un interprete de python configurado. Vamos a tener que tocar cosas en las opciones.



```
Profile  despegue x
C:\Users\maicr\OneDrive\Escritorio\PROYECTO\venv\Scripts\python.exe C:/Users/maicr/AppData/Roaming/JetBrains/IntelliJIdea2024.2/p
Starting cProfile profiler

10
9
8
7
6
5
4
3
2
1
0

¡Despegue!
Snapshot saved to C:/Users/maicr/AppData/Local/JetBrains/IntelliJIdea2024.2/snapshots/PROYECTO/PROYECTO.pstat

Process finished with exit code 0
```

Una vez lo solucionemos, ya nos dejará ejecutar.

-¿Cuál de los IDEs te pareció más cómodo o eficiente para ejecutar el código Python o el lenguajes que hayas elegido? ¿Por qué?

Visual Studio Code sin lugar a dudas. Porque python en IntelliJ necesitas configurar muchas cosas, sin embargo java me parece mejor en intellij porque como hemos visto me lo ha detectado automáticamente, al igual que pasa con kotlin, porque al final imagino que estarán preparados para trabajar con estos dos lenguajes, sin embargo python, necesitas configurar más cosas que en visual studio code no son necesarias.

Evidencia: Captura de pantalla mostrando la ejecución del programa en ambos IDES.