GRUPO BLR

PYTHON Y PENSAMIENTO COMPUTACIONAL PARA DOCENTES: GUÍA DIDÁCTICA CON ENFOQUE STEAM

ISBN: 978-9907-802-89-4

Julio Antonio Encalada Cuenca
Walter Andrés Benítez Martínez
Britney Dayana Mallaguari Bolívar

2026

PYTHON Y PENSAMIENTO COMPUTACIONAL PARA DOCENTES: GUÍA DIDÁCTICA CON ENFOQUE STEAM

Julio Antonio Encalada Cuenca

Walter Andrés Benítez Martínez

Britney Dayana Mallaguari Bolívar

AUTORES



UTMACH

Este libro ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad científica.

©Grupo Editorial BLR
Universidad Técnica de Machala
Riobamba – Ecuador
Correo: publicaciones@grupoblr.com
https://grupoblr.com/libros-investigo
REPOSITORIO



Encalada, J., Benítez., W., Mallaguari, B. (2026) Python y pensamiento computacional para docentes: Guía didáctica con enfoque STEAM. Grupo Editorial BLR.

© Julio Antonio Encalada Cuenca
Walter Andrés Benítez Martínez
Britney Dayana Mallaguari Bolívar

Python y pensamiento computacional para docentes: Guía didáctica con enfoque STEAM

ISBN: 978-9907-802-89-4

El copyright promueve la libertad de expresión, protege la diversidad de ideas y conocimiento, además apoya la libre expresión. Se prohíbe de manera rigurosa la producción o el almacenamiento de esta publicación, ya sea en su totalidad o en parte, está estrictamente prohibido por ley, incluyendo el diseño de la portada, así como su difusión a través de cualquiera de sus medios, ya sean electrónicos, mecánicos, ópticos, de grabación o incluso de fotocopia, sin permiso de los propietarios de los derechos de autor.

Filiación de autores

Julio Antonio Encalada Cuenca

Universidad Técnica de Machala

Correo Electrónico: jencalada@utmachala.edu.ec

ORCID: https://orcid.org/0000-0002-8120-2047

Walter Andrés Benítez Martínez

Universidad Técnica de Machala

Correo Electrónico: wbenitez1@utmachala.edu.ec

ORCID: https://orcid.org/0009-0004-0010-8932

Britney Dayana Mallaguari Bolívar

Universidad Técnica de Machala

Correo Electrónico: bmallagua1@utmachala.edu.ec

ORCID: https://orcid.org/0009-0006-7558-978X

PREFACIO

Introducción

En el siglo XXI, la educación enfrenta un reto fundamental: preparar a estudiantes para un futuro donde programación y razonamiento lógico serán esenciales, por tal motivo, Python no es solo una herramienta de codificación, sino un medio por el cuál potencia el pensamiento computacional. Según Orjuela Díaz y Rodríguez Vanegas (2022), el uso de Python en educación "fomenta significativamente la capacidad de análisis y la resolución de problemas en estudiantes secundarios" (p. 27).

Experiencias de aprendizaje gamificado con Python también evidencian que la motivación aumenta cuando los estudiantes participan en actividades interactivas y desafiantes, lo cual resulta clave para futuros docentes que buscan dinamizar sus clases (García & Naranjo, 2025). Asimismo, programas educativos enfocados en pensamiento computacional han demostrado ser efectivos cuando los docentes comprenden y aplican estrategias pedagógicas alineadas con dicho enfoque (Camacho & Grané, 2023).

Finalmente, este libro está dirigido a futuros docentes que buscan integrar la programación con Python en su práctica pedagógica, su propósito es mostrar cómo este lenguaje puede fortalecer el pensamiento computacional y enriquecer el aprendizaje de los estudiantes mediante experiencias claras, dinámicas y significativas.

En esta obra, pensamiento computacional asume como eje formativo transversal, más allá del aprendizaje técnico de Python, se busca que futuros docentes desarrollen habilidades cognitivas como descomposición de problemas, reconocimiento de patrones, abstracción y diseño de algoritmos, estas competencias permiten estructurar el razonamiento, organizar información y resolver situaciones de manera lógica y sistemática, Python no se presenta como un fin en sí mismo, sino como un medio pedagógico que facilita el desarrollo del pensamiento computacional dentro de un enfoque interdisciplinario STEAM, donde la programación se integra con ciencia, tecnología, ingeniería, arte y matemáticas generando experiencias de aprendizaje significativas.

Objetivo general

El objetivo de este libro es proporcionar a los futuros docentes una guía formativa que inicia en los aspectos más elementales de la programación con Python y avanza hacia aplicaciones prácticas en el aula; se busca que los educadores adquieran estrategias claras para desarrollar en sus estudiantes el pensamiento computacional, abstracción, diseño de algoritmos y resolución lógica de problemas, asimismo, la obra pretende constituirse en un recurso accesible y estructurado que apoye la innovación pedagógica y contribuya a un aprendizaje activo y significativo en el ámbito escolar.

Estructura

- **Capítulo 1:** presenta instalación y configuración del entorno de trabajo, explicando cómo preparar Visual Studio Code (VSC) y extensión de Python para iniciar el proceso de programación.

- **Capítulo 2:** aborda los fundamentos de la programación con Python, destacando los conceptos básicos del pensamiento computacional y la importancia de iniciar desde los elementos esenciales del lenguaje.

- **Capítulo 3:** introduce las variables, los operadores y las estructuras de control condicionales, mostrando cómo estos recursos permiten construir programas simples aplicables a situaciones educativas.

- **Capítulo 4:** desarrolla el uso de ciclos, funciones y procedimientos, resaltando su papel en la creación de programas estructurados y reutilizables que fortalecen el razonamiento lógico.

- **Capítulo 5:** analiza las listas, los diccionarios y el manejo de archivos, explicando cómo estas herramientas amplían las posibilidades de almacenamiento y procesamiento de datos en proyectos didácticos.

Características pedagógicas del libro

- **Enfoque Constructivista:** El aprendizaje de la programación en Python se concibe como un proceso activo en el que los estudiantes construyen su conocimiento a partir de la práctica y la experimentación. Se fomenta el ensayo, el error y la reflexión

sobre los resultados, lo que permite que los futuros docentes desarrollen autonomía y confianza al enseñar programación. De esta manera, se prioriza el razonamiento lógico sobre la memorización de sintaxis (Liao, 2024).

- **Aprendizaje Basado en Proyectos (ABP):** Cada capítulo incluye proyectos prácticos diseñados para aplicar los contenidos de Python en contextos educativos reales. El ABP no solo consolida destrezas técnicas como el uso de variables, funciones o ciclos, sino que también potencia competencias transversales como el trabajo en equipo, la creatividad y la resolución de problemas, convirtiendo la programación en un medio para "aprender haciendo" (Colussi & Monjelat, 2023).

- **Progresión gradual de contenidos:** El libro está diseñado de forma progresiva, iniciando con la instalación del entorno y los primeros programas, para luego avanzar a temas más complejos como listas, funciones, archivos y aplicaciones interdisciplinarias. Esta secuenciación permite que incluso quienes carecen de experiencia previa puedan desarrollar competencias en programación de manera escalonada, sin saltos conceptuales (Sánchez-García et al., 2015).

- **Enfoque interdisciplinario:** Se presentan capítulos específicos en los que Python se aplica a matemáticas, ciencias, tecnología/ingeniería y arte. Este enfoque muestra la versatilidad del lenguaje como recurso pedagógico, al conectar el pensamiento computacional con diferentes áreas del currículo escolar y

favorecer aprendizajes significativos y contextualizados (Villa & Buenaño, 2025).

- **Accesibilidad e inclusión:** El libro está concebido para que cualquier docente, incluso sin formación previa en programación, pueda comprender y enseñar Python. Se utilizan ejemplos claros, lenguaje sencillo y actividades guiadas, con el fin de democratizar el acceso al pensamiento computacional y garantizar que la programación sea un recurso pedagógico inclusivo y accesible (Meseguer & Serrano, 2024).

El presente libro ofrece a los futuros docentes una guía integral para la enseñanza de la programación con Python, partiendo de los fundamentos esenciales del lenguaje y avanzando hacia aplicaciones prácticas en distintas áreas del conocimiento. Se presenta la información clave, proponiendo actividades didácticas y ofreciendo recursos pedagógicos que permiten al docente adaptar y enriquecer la experiencia según su contexto y las características de sus estudiantes.

La programación con Python favorece el desarrollo del pensamiento crítico, la creatividad y las habilidades para la resolución de problemas. Al integrarse con áreas como Ciencia, Tecnología, Ingeniería, Arte y Matemáticas, se fomenta un aprendizaje activo, motivador y conectado con la realidad escolar, donde los futuros docentes pueden guiar a sus estudiantes en experiencias significativas y colaborativas (Monsálvez, 2017).

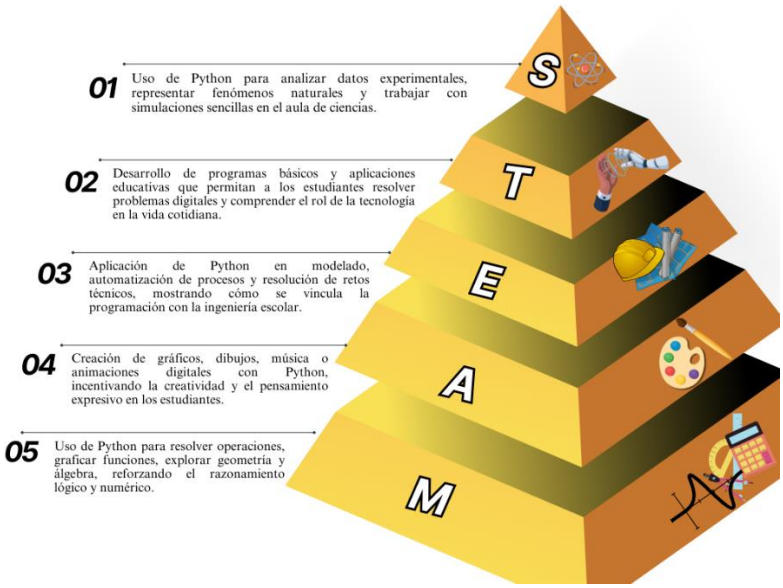


Figura 1. Pirámide de aplicación interdisciplinaria de Python en la educación.

Nota. En esta figura se representan las áreas de aplicación de la programación con Python dentro de STEAM.

Desde esta perspectiva, la programación con Python se concibe como una oportunidad para fortalecer el pensamiento crítico, la creatividad y la resolución de problemas en contextos escolares. Al integrarse con un enfoque interdisciplinario STEAM, se promueve un aprendizaje activo y significativo, conectado con la realidad educativa y orientado a la formación de estudiantes capaces de analizar, crear y colaborar. En este proceso, el rol del docente resulta clave para guiar experiencias de aprendizaje que trasciendan la enseñanza técnica y favorezcan el desarrollo de competencias cognitivas y pedagógicas fundamentales. La organización de los capítulos del libro permite avanzar de manera gradual y aplicar la programación en contextos de matemáticas, ciencias, tecnología, ingeniería y arte, siempre bajo un enfoque

constructivista que prioriza la práctica, la exploración y la creatividad en el aula (Perea, 2023a).

ÍNDICE

PREFACIO i

ÍNDICE viii

ÍNDICE DE TABLAS xiv

ÍNDICE DE FIGURAS xv

CAPÍTULO I 20

1 INSTALACIÓN Y PREPARACIÓN DEL ENTORNO DE PROGRAMACIÓN CON PYTHON 20

1.1 Introducción: La importancia de preparar el entorno de programación 21

1.2 Definiciones básicas 22

1.2.1 Definición de programación 22

1.2.2 Definición de pensamiento computacional 24

1.2.3 Definición de programación Python 26

1.2.4 Definición de editor de programación 26

1.3 Instalación y configuración del entorno 29

1.3.1 Instalación de Python 29

1.3.2 Configuración de la extensión de Python en Visual Studio Code 30

1.3.3 Primer programa en Python 30

1.3.4 Dificultades iniciales comunes con Python y soluciones..... 32

1.3.5 Analogías pedagógicas..... 33

1.4 Actividades didácticas con Python..... 33

1.5 Mini-proyecto integrador del capítulo 42

1.6 Integración STEAM del capítulo 44

1.7 Recursos digitales complementarios del capítulo 46

1.7.1 Recurso Audiovisual 46

1.7.2 Recurso de programación..... 46

1.8 Resumen del capítulo 47

1.9 Autoevaluación del capítulo..... 49

CAPÍTULO II 52

2 FUNDAMENTOS DE LA PROGRAMACIÓN CON PYTHON 52

2.1 Introducción: La programación como estrategia educativa 53

2.2 Pensamiento computacional y su relación con Python 53

2.3 La programación en el aula como herramienta pedagógica..... 55

2.4 Actividades didácticas: Explorando el pensamiento computacional 56

2.5 Mini-proyecto: Construcción de instrucciones paso a paso 59

2.6 Integración STEAM del capítulo 60

2.7 Recursos digitales complementarios del capítulo 62

2.7.1 Fichas didácticas 62

2.8 Resumen del capítulo 62

2.9 Autoevaluación del capítulo II 63

CAPÍTULO III 67

3 VARIABLES, OPERADORES Y ESTRUCTURAS CONDICIONALES EN PYTHON 67

3.2 Introducción: Primeros pasos prácticos en Python 68

3.3 Elementos básicos de Python 68

3.3.1 Variables 68

3.3.2 Tipos de datos 70

3.3.3 Operadores 72

3.3.4 Comentarios en el código 74

3.3.5 Estructuras condicionales 75

3.4 Actividades didácticas con Python 78

3.5 Mini-proyecto: Clasificador de notas 86

3.6 Integración STEAM del mini-proyecto 88

3.7 Recursos digitales complementarios del capítulo 89

3.8 Resumen del capítulo 90

3.9 Autoevaluación del capítulo 91

CAPÍTULO IV 94

4 CICLOS, FUNCIONES Y PROCEDIMIENTOS EN PYTHON 94

4.1 Introducción: La lógica detrás de la repetición 95

4.2 Ciclos en Python 96

4.2.1 Ciclo for 96

4.2.2 Ciclo while 97

4.2.3 Dificultades comunes y soluciones 98

4.3 Funciones y procedimientos 99

4.3.1 Definición de función 99

4.3.2 Funciones con valor de retorno 101

4.3.3 Procedimientos 101

4.3.4 Dificultades comunes y soluciones en procedimientos y funciones……...103

4.4 Actividades didácticas con Python..... 103

4.5 Mini-proyecto: Contador de asistencia 121

4.6 Integración STEAM del mini-proyecto 123

4.7 Recursos digitales complementarios del capítulo 124

4.8 Resumen del capítulo 125

4.9 Autoevaluación del capítulo..... 126

CAPÍTULO V 129

5 LISTAS, DICCIONARIOS Y MANEJO DE ARCHIVOS EN PYTHON..... 129

5.1 Introducción: Cómo guardar información en nuestros programas…….130

5.2 Listas en Python 130

5.2.1 Dificultades comunes y soluciones en el uso de listas..... 134

5.3 Diccionarios en Python 134

5.3.1 Dificultades comunes y soluciones en el uso de diccionarios.. 136

5.4 Manejo de archivos en Python 137

5.4.1 Dificultades comunes y soluciones en el uso de archivos..... 139

5.5 Guardar información en formato CSV 139

5.6 Mini-proyecto: Registro de calificaciones 159

5.7 Integración STEAM del mini-proyecto 161

5.8 Recursos digitales complementarios del capítulo 162

5.9 Resumen del capítulo 163

5.10 Autoevaluación del capítulo 164

REFERENCIAS BIBLIOGRÁFICAS 167

ÍNDICE DE TABLAS

Tabla 1. Problemas frecuentes en la instalación inicial de Python. 32

Tabla 2. Tipos de datos básicos en Python. 71

Tabla 3. Operadores básicos en Python. 73

Tabla 4. Dificultades comunes y soluciones en ciclos 98

Tabla 5. Dificultades comunes y soluciones en procedimientos y funciones. 103

Tabla 6. Dificultades comunes y soluciones en el uso de listas. ... 134

Tabla 7. Dificultades comunes y soluciones en el uso de diccionarios 136

Tabla 8. Dificultades comunes y soluciones en el uso de archivos 139

ÍNDICE DE FIGURAS

Figura 1. Pirámide de aplicación interdisciplinaria de Python en la educación. vi
Figura 2. Pantalla del instalador de Python. 29
Figura 3. Extensión de Python en Visual Studio Code. 30
Figura 4. Ejecución del primer programa en Python mediante consola. 31
Figura 5. Ejercicio de presentación con ingresos por teclado y ejecución mediante consola. 34
Figura 6. Ejecución de ejercicio / transformación de información predefinida. 35
Figura 7. Desarrollo y ejecución de ejercicio de confirmación de Información mediante consola. 36
Figura 8. Desarrollo y ejecución de suma de dos números mediante ingreso por teclado. 37
Figura 9. Resta de dos números. 38
Figura 10. Multiplicación de dos números mediante ejecución por consola e ingreso de valores por teclado. 39
Figura 11. División de dos números con resultado limitado a 2 decimales. 40
Figura 12. Organización de información mediante captura de datos en Python. 41
Figura 13. Código completo en Python del mini proyecto. 43
Figura 14. Código QR - Videotutorial de instalación y primer programa en Python. 46
Figura 15. Código QR - Descarga de archivos del capítulo. 47

Figura 16 . Componentes del pensamiento computacional aplicados a Python y la educación. 54
Figura 17. Código QR – Fichas didácticas 62
Figura 18. Representación visual de variables en Python. 69
Figura 19. Ejemplo de variables en Python. 70
Figura 20. Ejemplo práctico de tipos de datos en Python. 72
Figura 21. Ejemplo práctico de operadores en Python. 74
Figura 22. Ejemplo de comentarios en Python. 75
Figura 23. Ejemplo básico de condicional if-else en Python. 77
Figura 24. Cálculo del área de un rectángulo. 78
Figura 25. Actividad didáctica con condicionales en Python. 79
Figura 26. Transformación de tiempo (hora(s) y minuto(s). 80
Figura 27. Cálculo del nivel de energía (escala de 0 a 5). 81
Figura 28. Reparto equitativo de dulces. 82
Figura 29. Cálculo de descuento en una compra. 83
Figura 30. Simulador de batería del celular mediante Python. 84
Figura 31. Alerta de clima escolar. 85
Figura 32. Desarrollo del Mini-proyecto. 87
Figura 33. Código QR para descarga de archivos complementarios del Capítulo 3. 90
Figura 34. Estructura Repetitiva for en Python. 96
Figura 35. Estructura Repetitiva while en Python. 97
Figura 36. Definición y ejecución de funciones en Python. 100
Figura 37. Funciones con valores de retorno en Python. 101
Figura 38. Procedimientos en Python. 102
Figura 39. Playlist automática. 104
Figura 40. Empaquetado de productos. 105

Figura 41. Rutina de ejercicios. 106
Figura 42. Semáforo en conteo. 106
Figura 43. Simulación de aplausos. 107
Figura 44. Cuenta de lanzamiento. 108
Figura 45. While - Acceso con intento limitado. 109
Figura 46. Adivina el número. 110
Figura 47. Juego hasta rendirse. 111
Figura 48. Costo total. 111
Figura 49. Inflar globos. 112
Figura 50. Llenar tanque. 113
Figura 51. Intentos de tiro. 114
Figura 52. Consumo de agua. 115
Figura 53. Anuncio en el aula. 116
Figura 54. Mensaje de celebración. 117
Figura 55. Encender luz. 118
Figura 56. Recordatorio de tarea. 118
Figura 57. Preparar comida. 119
Figura 58. Rutina antes de dormir. 120
Figura 59. Preparar audio. 121
Figura 60. Sistema de registro de asistencia en el aula. 122
Figura 61. Código QR para descarga de archivos complementarios del Capítulo 4. 124
Figura 62. Creación y visualización de una lista en Python. 131
Figura 63. Modificación de una lista con los métodos append() y remove(). 132
Figura 64. Ejecución del programa de listas en Python. 132
Figura 65. Recorrido de una lista con ciclo for en Python. 133

Figura 66. Creación y visualización de un diccionario en Python. 135
Figura 67. Acceso a un valor específico de un diccionario en Python. 135
Figura 68. Agregar un nuevo dato al diccionario en Python..... 136
Figura 69. Creación y escritura de un archivo de texto en Python. 137
Figura 70. Lectura del contenido de un archivo de texto en Python. 138
Figura 71. Creación de un archivo CSV con datos en Python. ... 140
Figura 72. Playlist personalizada dentro de Python..... 141
Figura 73. Lista de compras inteligente dentro de Python, ejemplificación. 142
Figura 74. Menú del día depende de la disponibilidad del ingreso. 143
Figura 75. Palabras motivadoras en cada línea con Python..... 144
Figura 76. Paquetes por entregar con Python. 145
Figura 77. Mochila organizada por prioridad en Python con ejecución del ejercicio por consola. 146
Figura 78. Marca de celular. 147
Figura 79. Estados del semáforo. 148
Figura 80. Animal y sonido dentro de Python con ejecución por consola. 149
Figura 81. Color y emoción..... 150
Figura 82. Deporte y lugar..... 151
Figura 83. Vehículo y uso, ejemplificaciones en Python. 151

Figura 84. Metas y resultados mediante Python presentado por consola. 152
Figura 85. Crear un archivo simple. 153
Figura 86. Leer archivo simple. 154
Figura 87. Agregar un texto sencillo en un archivo.txt dentro de Python. 155
Figura 88. Mostrar contenido línea por línea en la ejecución por consola 156
Figura 89. Guardar un nombre en un archivo de texto. 157
Figura 90. Lectura de nombre almacenado en archivo. 158
Figura 91. Código QR para descarga de archivos complementarios del Capítulo 5. 163

CAPÍTULO I

1 INSTALACIÓN Y PREPARACIÓN DEL ENTORNO DE PROGRAMACIÓN CON PYTHON

Objetivo del capítulo

Al finalizar este capítulo, los lectores podrán desarrollar competencias básicas para instalar y configurar el entorno de programación con Python, comprendiendo su función como herramienta accesible y flexible para la enseñanza. Además, podrán reconocer la importancia de contar con un editor especializado como Visual Studio Code y con la extensión de Python, entendiendo su papel como puente entre las instrucciones del usuario y la ejecución en la computadora, este capítulo también busca que los futuros docentes valoren la simplicidad de Python como recurso pedagógico inclusivo, capaz de motivar a estudiantes de diversos contextos y necesidades, brindándoles confianza para dar sus primeros pasos en la programación. Asimismo, se busca que comprendan la instalación del entorno como un primer ejercicio de pensamiento computacional, basado en la secuencia ordenada de pasos, la verificación de resultados y la resolución sistemática de posibles errores.

Aprendizajes Esperados

- Definición de programación.
- Definición de programación Python.
- Definición de editor de programación.

- Diferenciar un editor de texto como Word o bloc de notas, de un editor de programación como Visual Studio Code.
- Realizar correctamente la instalación de Python y verificar su funcionamiento.
- Instalar la extensión de Python en Visual Studio Code.
- Crear, guardar y ejecutar un archivo básico con extensión .py.

1.1 Introducción: La importancia de preparar el entorno de programación

Antes de aprender a programar, es fundamental disponer de un entorno adecuado que facilite el aprendizaje. La instalación de Python y la preparación de herramientas como Visual Studio Code permiten a los estudiantes y docentes contar con un espacio amigable, accesible y eficaz para escribir código. De acuerdo con Prokopyev et al. (2020), el uso de Python en la formación inicial docente fomenta la adquisición de competencias tecnológicas básicas y facilita la enseñanza de conceptos computacionales complejos mediante ejemplos simples. Asimismo, la instalación y configuración inicial constituye un primer paso hacia la inclusión digital, garantizando que personas con distintos niveles de experiencia puedan participar en actividades de programación (Sbampato et al., 2025).

Python se distingue de otros lenguajes porque combina claridad en la sintaxis, disponibilidad gratuita y aplicabilidad en múltiples campos como educación, ciencia, ingeniería y análisis de datos. Visual Studio Code, por su parte, es un editor ampliamente utilizado que se adapta

tanto a principiantes como a profesionales, y que permite integrar extensiones como la de Python para una experiencia completa (Vera et al., 2025).

1.2 Definiciones básicas

1.2.1 Definición de programación

La programación puede definirse como el proceso sistemático mediante el cual se diseñan, escriben y organizan instrucciones que una computadora es capaz de interpretar y ejecutar para resolver problemas o realizar tareas específicas. Estas instrucciones se estructuran a partir de algoritmos, entendidos como secuencias finitas y ordenadas de pasos lógicos orientados a la obtención de un resultado. En este sentido, programar no implica únicamente escribir código, sino también comprender un problema, analizar sus posibles soluciones y traducirlas a un lenguaje formal que pueda ser procesado por una máquina (Morales et al., 2009).

Desde una perspectiva computacional, la programación constituye un puente entre el pensamiento humano y ejecución automatizada de proceso, a través de lenguajes de programación, el ser humano logra comunicarse con la computadora utilizando estructuras sintácticas y semánticas definidas, este proceso requiere precisión, claridad y coherencia, ya que incluso pequeños errores pueden impedir la correcta ejecución de un programa, por ello, la programación se asocia estrechamente con el desarrollo de habilidades como el razonamiento lógico, la atención al detalle y la capacidad de estructurar ideas de manera ordenada, programación ha sido reconocida como una

herramienta clave para el desarrollo del pensamiento computacional, más allá de su dimensión técnica, permite a estudiantes aprender a descomponer problemas complejos en partes manejables y cotidianas, identificar patrones, abstraer información relevante y diseñar soluciones eficientes, de esta manera señala Perea (2023), programación favorece la construcción de un pensamiento estructurado que puede transferirse a diversas áreas del conocimiento, fortaleciendo así la formación integral del estudiante.

Asimismo, diversos enfoques pedagógicos destacan que la programación debe ser entendida como un proceso creativo y no únicamente técnico, programar implica diseñar soluciones originales, experimentar con diferentes alternativas y construir productos digitales que respondan a necesidades específico, aquel error adquiere valor formativo, ya que permite al estudiante reflexionar sobre su proceso, identificar fallas y mejorar progresivamente su solución, la visión contribuye a desarrollar una actitud crítica y resiliente frente a desafíos, aspectos fundamentales en la educación contemporánea.

En el contexto actual caracterizado por la transformación digital, programación se posiciona como una competencia esencial para la ciudadanía, no solo se trata de formar programadores profesionales, sino de desarrollar en los estudiantes capacidad de comprender cómo funcionan las tecnologías que utilizan diariamente y cómo pueden crear sus propias soluciones digitales, este modo, programación se convierte en un medio para fomentar la autonomía, innovación y participación activa en una sociedad cada vez más mediada por la tecnología.

Desde una perspectiva más formal, la programación también puede definirse como la disciplina que estudia el diseño de algoritmos y su implementación en lenguajes de programación para la construcción de software, desde este enfoque se resalta carácter científico de la programación, ya que implica el uso de métodos estructurados, validación de resultados y optimización de soluciones, la programación se vincula con áreas como la ingeniería de software, lógica matemática y teoría de computación, consolidándose como un campo de conocimiento con fundamentos propios (Osorio et al., 2023).

Por otro lado, desde una visión aplicada, programación puede entenderse como una herramienta para resolución de problemas del mundo real mediante uso de tecnología, como de programas informáticos, es posible automatizar procesos, analizar datos, crear aplicaciones interactivas y desarrollar soluciones innovadoras en distintos contextos, como la educación, la salud o la industria, perspectiva enfatiza el valor práctico de la programación, destacando su utilidad para transformar ideas en productos funcionales que impactan directamente en la sociedad.

1.2.2 Definición de pensamiento computacional

El pensamiento computacional se define como un proceso cognitivo orientado a la resolución de problemas mediante la formulación de soluciones que pueden ser ejecutadas por una computadora o por un ser humano de manera sistemática (Pérez Angulo, 2019). Implica descomponer problemas complejos en partes más manejables, identificar patrones, abstraer información relevante y diseñar algoritmos

claros, esta manera de pensar no se limita al ámbito de la informática, sino que se aplica en diversas disciplinas para estructurar soluciones eficientes y replicables.

Desde una perspectiva educativa, pensamiento computacional es una habilidad fundamental que permite a los estudiantes desarrollar capacidades analíticas y de razonamiento lógico, consiste en aprender a plantear problemas de manera estructurada, utilizando estrategias como descomposición, abstracción y secuenciación de pasos incorporación en el currículo fomenta no solo competencias digitales, sino también habilidades transferibles a la vida cotidiana y al ámbito profesional (Cervera-Manjarrez et al., 2023a).

Otra definición concibe el pensamiento computacional como una metodología para abordar problemas basada en principios de la ciencia de la computación, incluye automatización de procesos, creación de modelos y simulaciones, y evaluación de soluciones en términos de eficiencia y viabilidad, consecuente a esto, el pensamiento computacional permite optimizar recursos y tomar decisiones fundamentadas en procesos lógicos y estructurados.

Finalmente, el pensamiento computacional puede entenderse como una forma de pensamiento que integra creatividad, lógica y pensamiento crítico para diseñar soluciones innovadoras, no solo se trata de programar, sino desarrollar una mentalidad orientada a la resolución de problemas complejos, considerando múltiples variables y posibles escenarios reales, esta competencia es clave en nuestro contexto, donde

la tecnología la información demandan habilidades para analizar, interpretar y generar soluciones efectivas.

1.2.3 Definición de programación Python

Python es un lenguaje de programación interpretado, de alto nivel y multiplataforma. Se caracteriza por su sintaxis sencilla y clara, lo que lo convierte en una excelente opción educativa. Según Soto Gómez (2021), Python es el lenguaje textual más adecuado en la enseñanza secundaria debido a su legibilidad y simplicidad. Además, es utilizado en áreas como ciencia de datos, inteligencia artificial, desarrollo web y educación, lo que amplía las oportunidades de aprendizaje práctico.

1.2.4 Definición de editor de programación

Un editor de programación es un software diseñado para escribir y ejecutar código, ofreciendo funciones que un editor de texto común como Word o Bloc de notas no tiene, resaltado de sintaxis, depuración, integración con compiladores y ejecución de programas (Navas-Bonilla et al., 2025a).

Visual Studio Code, en particular, ha sido adoptado en entornos educativos por su accesibilidad y versatilidad. Castañón et al. (2022) señalan que un editor moderno como Visual Studio Code ayuda a los estudiantes a concentrarse en la lógica del programa y no en aspectos técnicos innecesarios.

1.2.4.1 Pensamiento computacional en la preparación del entorno

La instalación y configuración del entorno de programación no debe entenderse únicamente como un procedimiento técnico previo al aprendizaje de la programación, sino como una experiencia formativa inicial que introduce al estudiante en los fundamentos del pensamiento computacional. Este proceso implica la ejecución de una serie de acciones organizadas que requieren atención, comprensión de instrucciones y verificación de resultados, desde una perspectiva educativa, cada paso realizado durante la instalación de Python y configuración del editor de código representa una oportunidad para que el estudiante desarrolle habilidades cognitivas relacionadas con la organización, secuenciación y toma de decisiones, elementos esenciales en la resolución de problemas.

Pensamiento computacional, definido como capacidad de formular problemas y diseñar soluciones de manera estructurada, se manifiesta desde estas primeras actividades durante la preparación del entorno, el estudiante aplica procesos de descomposición al dividir la tarea general en subtareas más pequeñas, como descargar el instalador, ejecutar el programa y seleccionar opciones específicas, así mismo, reconocimiento de patrones se evidencia cuando identifica errores comunes y aprende a resolverlos mediante procedimientos similares, acciones, aunque simples en apariencia, constituyen la base de una forma de pensamiento lógico que posteriormente se trasladará al desarrollo de algoritmos y programas más complejos.

Otro aspecto relevante es la verificación constante del funcionamiento del sistema, lo cual introduce al estudiante en una dinámica de evaluación continua, al comprobar si Python se ejecuta correctamente o si Visual Studio Code (VSC) reconoce el lenguaje, el estudiante desarrolla habilidades de validación y control, propias de los procesos computacionales, enfoque fomenta una actitud crítica frente al resultado de sus acciones, promoviendo la revisión sistemática y la corrección de errores, error deja de ser percibido como una falla y pasa a ser interpretado como una fuente de aprendizaje que impulsa la mejora continua.

Además, la configuración del entorno permite al estudiante comprender la relación entre diferentes componentes tecnológicos, como el S.O, el lenguaje de programación y el editor de código, comprensión favorece la abstracción, ya que el estudiante comienza a identificar cuáles son los elementos esenciales para que un programa funcione correctamente, diferenciándolos de aquellos que no son relevantes capacidad de abstraer información es fundamental en el pensamiento computacional, puesto que permite simplificar problemas complejos y enfocarse en los aspectos clave para su resolución, concebir la instalación del entorno como una actividad de aprendizaje implica reconocer su valor pedagógico dentro del proceso formativo. En lugar de ser una fase mecánica, se transforma en espacio donde el estudiante experimenta, toma decisiones, enfrenta dificultades y construye conocimiento de manera activa, enfoque contribuye al desarrollo de la autonomía, confianza y resiliencia digital, competencias esenciales para futuros docentes que deberán guiar a sus estudiantes en el uso crítico y creativo de la tecnología, preparación del entorno se consolida como el primer

paso en la formación de un pensamiento estructurado, lógico y orientado a la resolución de problemas.

1.3 Instalación y configuración del entorno

1.3.1 Instalación de Python

Acceder a www.python.org

Descargar la versión más reciente (Python 3.13.7).

Ejecutar el instalador descargado.

Al iniciar la instalación no olvides seleccionar la casilla Add Python to PATH.



Figura 2. Pantalla del instalador de Python.

Nota. Activar esta opción garantiza que Python pueda ejecutarse desde cualquier parte del sistema.

1.3.2 Configuración de la extensión de Python en Visual Studio Code

Abrir Visual Studio Code.

Seleccionar el ícono de Extensiones en la barra lateral izquierda.

Buscar “Python” y seleccionar la oficial de Microsoft.

Instalarla.



Figura 3. Extensión de Python en Visual Studio Code.

Nota. La extensión oficial de Python facilita la ejecución de programas en Visual Studio Code y proporciona funciones adicionales como la selección del intérprete y la depuración del código.

1.3.3 Primer programa en Python

Crear una carpeta (Opcional).

Habilita las extensiones de nombre de archivos (ver > mostrar > extensiones de nombre de archivos).

Crea un nuevo documento de texto (bloc de notas), llamado python.py.

Arrastra el archivo (Python.py) al entorno de Visual Studio Code.

Código Python:

print ("Hola, mundo")

Ejecutar en Visual Studio Code con Run Python File.

Verificar la consola que haya devuelto el “Hola Mundo”.

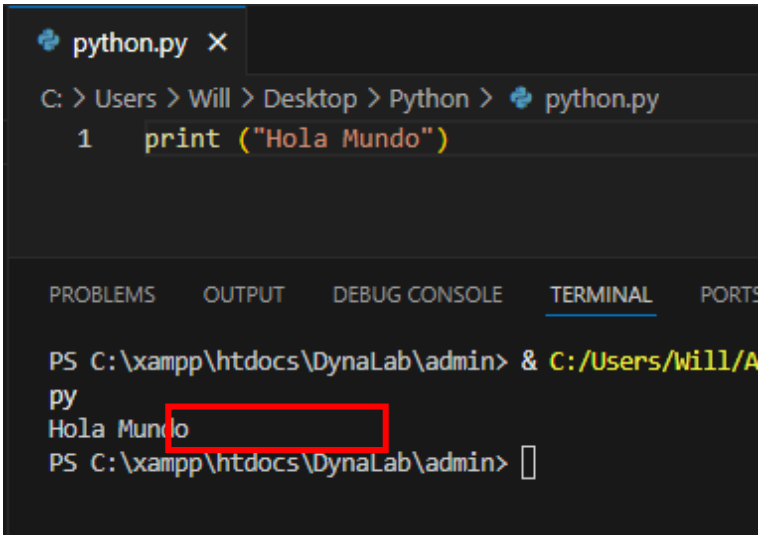


Figura 4. Ejecución del primer programa en Python mediante consola.

Nota. Este programa confirma que el entorno está listo para usarse, ejecutando un ejemplo simple y dejando preparado todo para el desarrollo de futuros ejercicios.

Desde la perspectiva del pensamiento computacional, primer programa permite comprender la relación entre instrucción y resultado, evidenciando cómo una secuencia lógica produce un efecto concreto, experiencia inicial que fortalece comprensión del algoritmo como conjunto ordenado de pasos que generan una salida específica.

1.3.4 Dificultades iniciales comunes con Python y soluciones

Tabla 1. Problemas frecuentes en la instalación inicial de Python.

Problema	**Posible causa**	**Solución**
El comando Python no funciona.	No se activó Add to PATH.	Reinstalar Python activando la casilla.
Visual Studio Code no detecta Python.	No se instaló la extensión.	Instalar la extensión oficial de Microsoft.
Archivo guardado como .txt	Error al elegir formato	Guardar con extensión .py
No aparece salida.	Ejecución incorrecta.	Usar la opción Run Python File.

Nota. Esta tabla resume los errores más comunes de principiantes.

1.3.5 Analogías pedagógicas

- **Python = lápiz digital**: se utiliza para escribir instrucciones.
- **Visual Studio Code = cuaderno**: espacio donde se registran y ejecutan los programas.
- **Extensión de Python en Visual Studio Code = traductor**: interpreta las instrucciones y las convierte en acciones.

Estas comparaciones permiten a estudiantes con poca experiencia comprender más fácilmente los conceptos técnicos (Navas-Bonilla et al., 2025).

1.4 Actividades didácticas con Python

Como ejercicio de aplicación práctica, se propone crear un programa sencillo que solicite al usuario su nombre y muestre un saludo personalizado en pantalla.

Esta actividad busca reforzar la comprensión de la interacción básica entre el usuario y la computadora, además de motivar a los futuros docentes a visualizar cómo un recurso tan simple puede convertirse en una herramienta introductoria para sus estudiantes.

Código Python:

```
nombre = input ("¿Cómo te llamas? :")

print ("Bienvenido/a,", nombre)
```

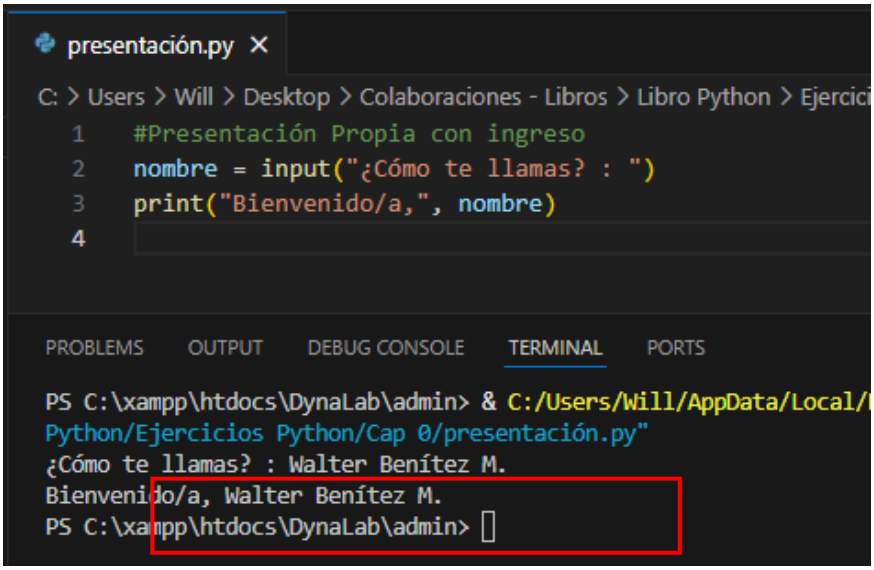


Figura 5. Ejercicio de presentación con ingresos por teclado y ejecución mediante consola.

Nota. Ejercicio que refuerza la interacción básica entre usuario y computadora, y puede replicarse como dinámica en entornos educativos y con datos propios.

Orientación docente: Este mismo ejemplo puede adaptarse para que los estudiantes comprendan el concepto de entrada y salida en programación, vinculando la teoría con la práctica de manera sencilla, motivadora y cotidiana.

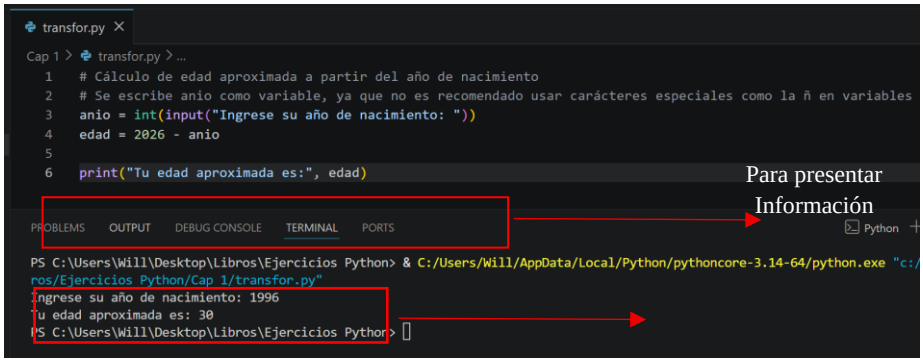


Figura 6. Ejecución de ejercicio / transformación de información predefinida.

Nota. Este ejercicio permite al estudiante transformar un dato de entrada en un resultado mediante una operación simple, introduciendo el uso de conversión de datos con int( ).

Orientación docente: Este ejercicio puede utilizarse para explicar cómo la programación no solo recibe datos, sino también los procesa, se relaciona con cálculos cotidianos, fortaleciendo el razonamiento lógico del alumnado.

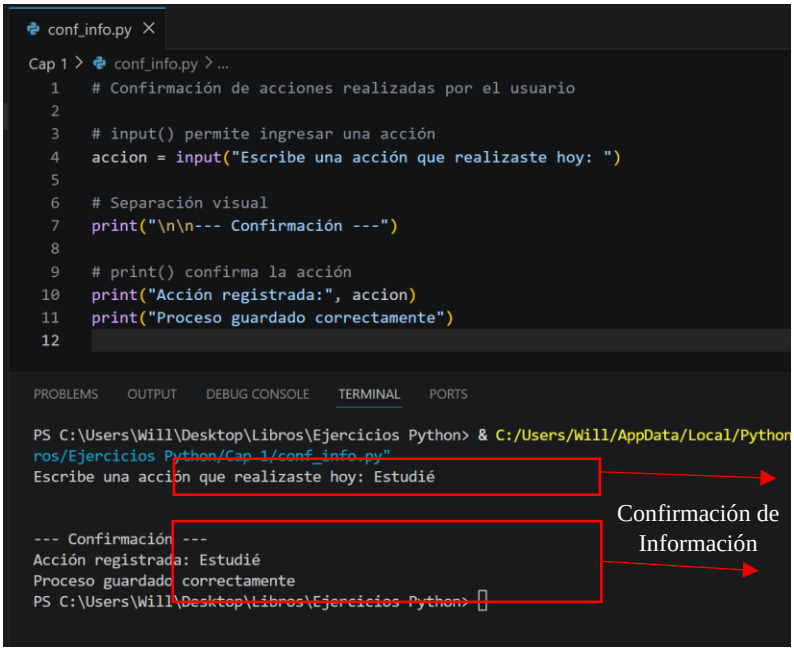


Figura 7. Desarrollo y ejecución de ejercicio de confirmación de Información mediante consola.

Nota. Ejercicio que permite al estudiante registrar y confirmar acciones mediante un programa sencillo y entendible.

Orientación docente: Actividad que refuerza la idea de que programas pueden recibir información y devolver respuestas, favoreciendo la interacción básica.

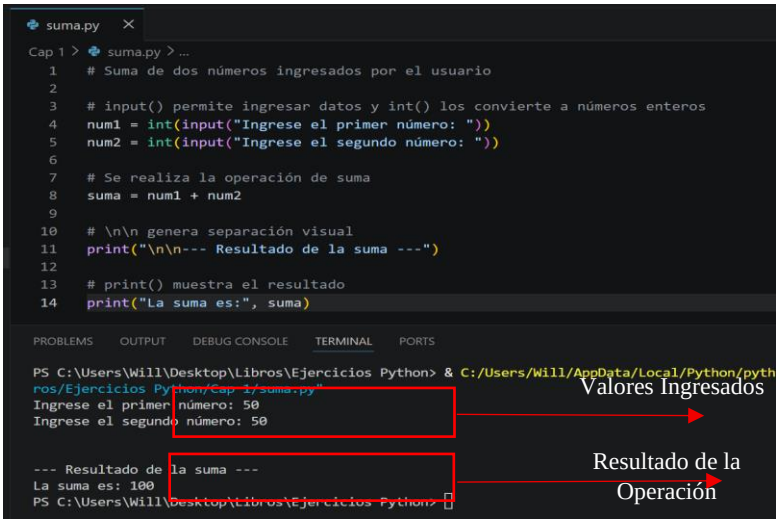


Figura 8. Desarrollo y ejecución de suma de dos números mediante ingreso por teclado.

Nota. Este ejercicio permite al estudiante realizar una operación básica de suma a partir de datos ingresados por el mismo y entendiendo cómo funciona una de las operaciones matemáticas.

Orientación docente: Esta actividad permite evidenciar cómo un programa puede recibir datos, procesarlos mediante una operación y devolver un resultado. Favorece la comprensión de la lógica básica de entrada – proceso – salida, fundamental en el pensamiento computacional.

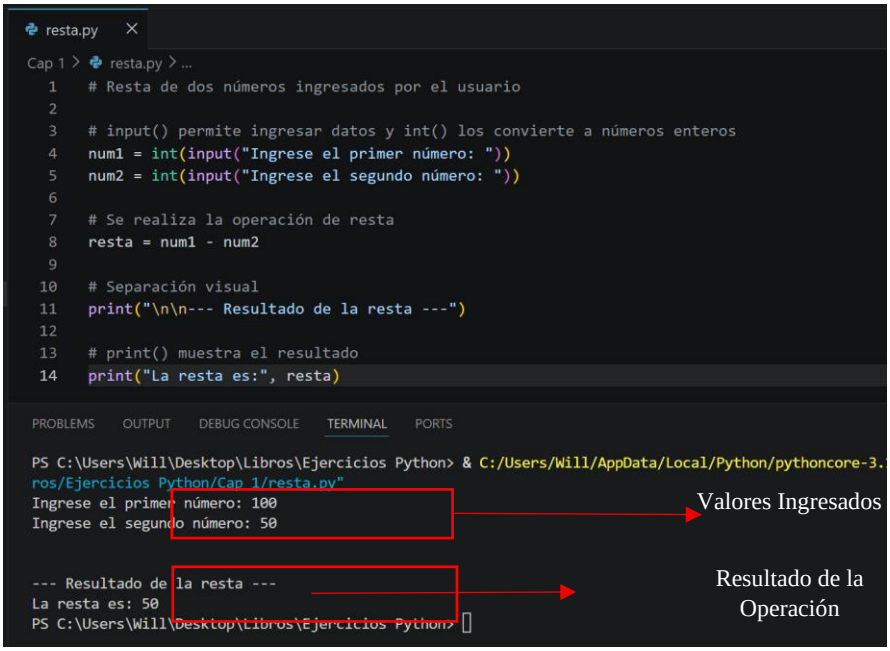


Figura 9. Resta de dos números.

Nota. Este ejercicio permite al estudiante calcular diferencia entre dos valores a partir de datos ingresados por teclado, evidenciando cómo un programa recibe información, realiza un proceso y genera un resultado, asimismo, refuerza el uso de operaciones aritméticas básicas y la manipulación de datos en Python.

Orientación docente: Esta actividad puede utilizarse para relacionar programación con operaciones matemáticas sencillas, facilitando que el estudiante comprenda cómo una secuencia de instrucciones permite obtener un resultado concreto y fácil, asimismo, contribuye al desarrollo de pensamiento lógico al identificar el proceso de entrada, operación y salida dentro de un programa cotidiano.

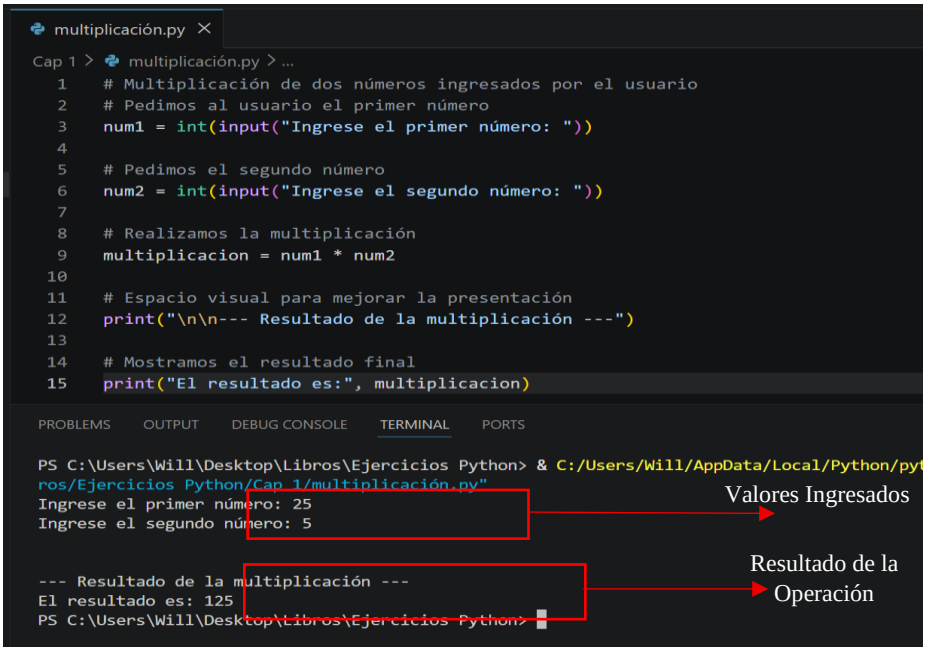


Figura 10. Multiplicación de dos números mediante ejecución por consola e ingreso de valores por teclado.

Nota. Este ejercicio introduce al estudiante en la aplicación de operaciones aritméticas mediante programación, utilizando datos proporcionados por el usuario, a través de esta actividad, se evidencia el flujo básico de un programa: entrada de datos, procesamiento y generación de resultados.

Orientación docente: El docente puede emplear esta actividad para integrar la programación con el área de Matemáticas, facilitando la comprensión de operaciones básicas desde un enfoque práctico, permite reforzar el pensamiento computacional al evidenciar cómo una secuencia de instrucciones organizada produce un resultado específico.

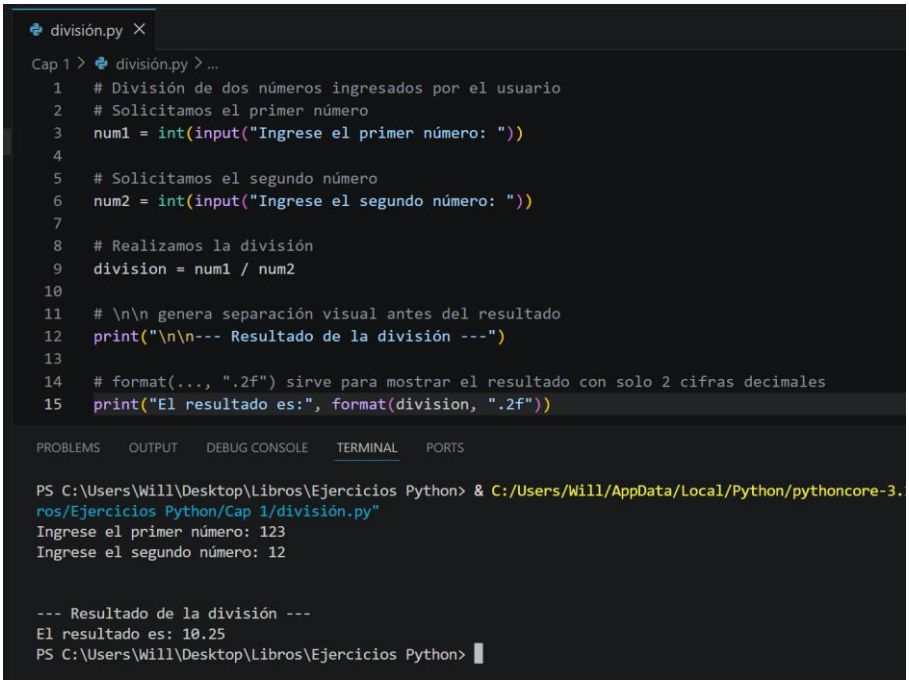


Figura 11. División de dos números con resultado limitado a 2 decimales.

Nota. Este ejercicio permite al estudiante realizar una operación de división y visualizar el resultado de forma organizada, incorporando una presentación más clara de los valores decimales.

Orientación docente: Esta actividad permite comprender cómo un programa no solo realiza operaciones, sino también puede controlar la forma en que se muestran los resultados, favorece la interpretación de datos numéricos y el desarrollo del pensamiento lógico aplicado a la presentación de información.

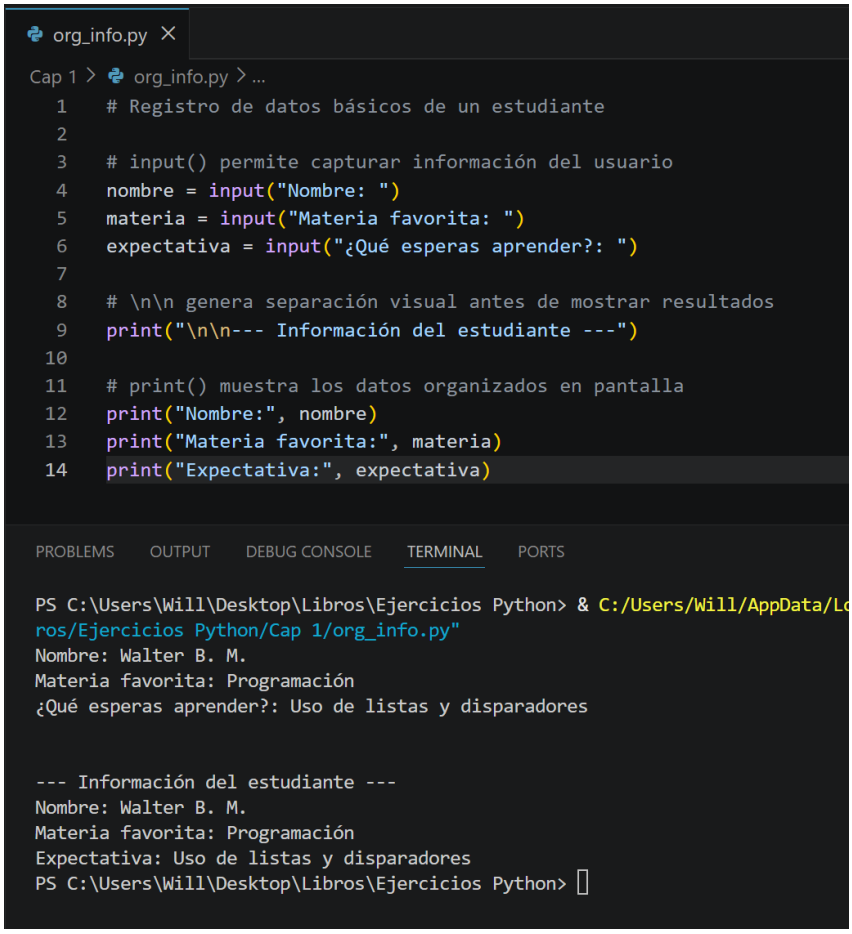


Figura 12. Organización de información mediante captura de datos en Python.

Nota. Este ejercicio permite al estudiante organizar múltiples datos de entrada y mostrarlos de forma estructurada en pantalla.

Orientación docente: Esta actividad favorece la comprensión de cómo se pueden recopilar y presentar datos de manera ordenada mediante la programación, asimismo, puede relacionarse con situaciones del entorno escolar, promoviendo la organización de información y el desarrollo del pensamiento lógico.

1.5 Mini-proyecto integrador del capítulo

Con el propósito de fortalecer los conocimientos adquiridos en este capítulo, se propone un mini-proyecto que amplía la interacción básica trabajada previamente, incorporando múltiples entradas de datos y una pequeña transformación de la información; actividad que permite al estudiante avanzar desde un ejercicio guiado hacia una aplicación más autónoma, consolidando la lógica secuencial y la construcción de programas simples.

Enunciado del ejercicio:

Diseña un programa en Python que solicite al usuario su nombre, edad y ciudad, y luego genere un mensaje personalizado, programa deberá realizar un cálculo sencillo para mostrar cuántos años tendrá el usuario dentro de cinco años.

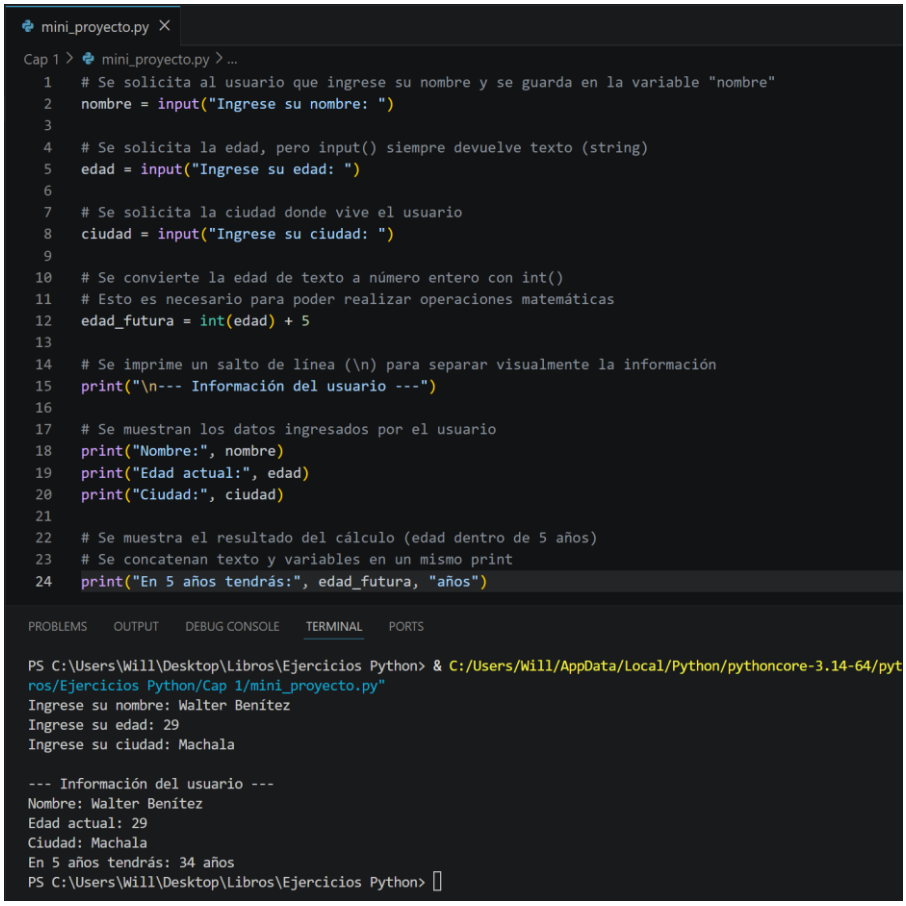


Figura 13. Código completo en Python del mini proyecto.

Nota. La imagen muestra ejecución del mini proyecto desarrollado en Python, donde se evidencia cómo el programa solicita datos al usuario, los procesa y presenta resultados de forma organizada, observa el uso de entrada de datos (input), conversión de tipos (int) y salida en pantalla (print), permitiendo visualizar la edad actual y la edad futura del usuario de manera clara.

Explicación:

A diferencia de la actividad educativa del apartado anterior, este mini-proyecto integra múltiples datos y realiza una pequeña operación con la información ingresada, introduce el uso de conversión de tipos mediante int(), lo que permite realizar cálculos básicos; representa un avance en la complejidad del programa, manteniéndose dentro de un nivel accesible para principiantes, el pensamiento computacional, este ejercicio fortalece la organización de datos, la secuenciación de instrucciones y la transformación de información para generar nuevas salidas.

Orientación docente:

El docente puede ampliar esta actividad solicitando al estudiante que agregue nuevos datos o modifique el formato de salida, promoviendo la creatividad y personalización del programa.

1.6 Integración STEAM del capítulo

Science (Ciencia): La preparación del entorno de programación introduce a los estudiantes en procesos científicos básicos como la observación, la verificación y la experimentación. Al instalar Python y ejecutar el primer programa, el estudiante comprueba hipótesis simples (por ejemplo, si el entorno funciona correctamente), analiza resultados en la consola y reflexiona sobre posibles errores, desarrollando habilidades propias del método científico aplicadas al ámbito tecnológico.

Technology (Tecnología): Capítulo que se centra directamente en el uso de herramientas tecnológicas fundamentales para la programación

instalación de Python, uso de Visual Studio Code y la configuración de extensiones permiten al estudiante comprender cómo interactúan el software, sistema operativo y lenguaje de programación, fortaleciendo competencias digitales esenciales para el ejercicio docente en contextos educativos actuales.

Engineering (Ingeniería): Configuración del entorno de programación implica seguir una secuencia ordenada de pasos, identificar errores comunes y aplicar soluciones, lo cual refleja un proceso propio del pensamiento ingenieril, estudiante planifica, ejecuta, prueba y ajusta el sistema hasta lograr que el primer programa funcione correctamente, desarrollando habilidades de resolución de problemas y pensamiento sistemático.

Arts (Arte): Aunque el enfoque inicial es técnico, el primer programa en Python permite introducir elementos creativos mediante mensajes personalizados y actividades de interacción básica con el usuario, escritura de saludos y personalización del texto fomentan la expresión digital y comunicación creativa, sentando las bases para futuras aplicaciones visuales, narrativas o expresivas en proyectos educativos.

Mathematics (Matemáticas): Preparación del entorno de programación se apoya en conceptos matemáticos implícitos como la secuencia de pasos, orden lógico y precisión en la ejecución de instrucciones, uso correcto de comandos, rutas y versiones refuerza el razonamiento lógico-matemático.

1.7 Recursos digitales complementarios del capítulo

1.7.1 Recurso Audiovisual

Como apoyo audiovisual, puede escanear el código QR de la Figura 14 para acceder al videotutorial donde se explican todos los pasos desde la descarga e instalación hasta la creación y ejecución del primer programa en Python.



Figura 14. Código QR - Videotutorial de instalación y primer programa en Python.

Nota. Al escanear el código, se abrirá el videotutorial en YouTube con los pasos de instalación de Python, Visual Studio Code, configuración de la extensión y ejecución del primer programa.

1.7.2 Recurso de programación

Con el fin de apoyar el proceso de enseñanza y aprendizaje, se pone a disposición un conjunto de recursos digitales que incluyen los archivos

utilizados en el videotutorial y los códigos de las actividades desarrolladas en este capítulo.

Estos materiales pueden descargarse mediante el código QR correspondiente, lo que permite a los docentes acceder fácilmente a los ejemplos, modificarlos y utilizarlos como base para nuevas experiencias en el aula.



Figura 15. Código QR - Descarga de archivos del capítulo.

Nota. Los recursos incluyen el archivo python.py (primer programa con "Hola, mundo") y presentación.py (programa que solicita el nombre del usuario y devuelve un saludo personalizado).

1.8 Resumen del capítulo

En este capítulo se presentó la base para iniciar el camino en la programación, entendida como el proceso de dar instrucciones a una computadora para que realice tareas específicas, introdujo a Python

como un lenguaje de programación ideal para principiantes gracias a su sintaxis clara, sencilla y accesible, lo que lo convierte en una herramienta inclusiva en entornos educativos.

Asimismo, se explicó la importancia de contar con un editor especializado como Visual Studio Code, el cual permite escribir, organizar y ejecutar programas de manera eficiente del proceso práctico, se abordaron pasos esenciales para instalar Python, configurar la extensión correspondiente en Visual Studio Code y verificar el correcto funcionamiento del entorno.

También se presentaron las primeras nociones de interacción con el usuario mediante el uso de funciones básicas como input( ) y print( ), permitiendo entrada y salida de datos, asimismo, herramientas constituyen el punto de partida para la construcción de programas sencillos y comprensibles, favoreciendo un aprendizaje progresivo, se ejecutó el primer programa básico con el comando print("Hola, mundo"), marcando el inicio del aprendizaje activo y práctico en el uso de Python.

Por último, a través de actividades prácticas y un mini-proyecto, se consolidaron los conocimientos adquiridos, permitiendo a los estudiantes crear programas básicos interactivos, también el enfoque pedagógico, este capítulo establece las bases necesarias para avanzar hacia contenidos más complejos, promoviendo la confianza y motivación en el aprendizaje de la programación.

1.9 Autoevaluación del capítulo

Pregunta 1: ¿Qué diferencia a un editor de texto común de un editor de programación?

a) Permite agregar colores en el texto.
b) Puede ejecutar código y trabajar con extensiones.
c) Solo sirve para escribir documentos largos.
d) No abre archivos .py

Pregunta 2: ¿Qué función cumple la casilla Add to PATH en la instalación de Python?

a) Aumenta la velocidad del procesador.
b) Permite ejecutar Python desde cualquier carpeta del sistema.
c) Instala automáticamente Visual Studio Code.
d) Evita virus durante la instalación.

Pregunta 3: ¿Cuál fue el primer programa escrito en Python en este capítulo?

a) say ("Hola, mundo")
b) echo ("Hola, mundo")
c) print ("Hola, mundo")
d) output ("Hola, mundo")

Pregunta 4: ¿Qué se necesita para que Visual Studio Code ejecute programas en Python?

a) Conexión a internet permanente.
b) La extensión de Python instalada.

c) Instalar Microsoft Word.

d) Guardar el archivo en .docx.

Pregunta 5: ¿Qué comando permite verificar que Python está instalado?

a) python --install

b) python --run

c) python --version

d) python –path

Pregunta 6: ¿Qué es Python?

a) Un sistema operativo

b) Un lenguaje de programación

c) Un editor de texto

d) Un programa de diseño

Pregunta 7: ¿Qué función permite mostrar información en pantalla?

a) input()

b) print()

c) open()

d) range()

Pregunta 8: ¿Qué función permite ingresar datos desde el usuario?

a) print()

b) input()

c) write()

Pregunta 9: ¿Qué es el pensamiento computacional?

a) Usar computadoras diariamente
b) Resolver problemas de forma lógica y estructurada
c) Escribir código avanzado

Pregunta 10: Desde el enfoque educativo, la programación permite:

a) Memorizar código
b) Desarrollar pensamiento lógico y resolución de problemas
c) Evitar el uso de computadoras
d) Solo imprimir texto

CAPÍTULO II

2 FUNDAMENTOS DE LA PROGRAMACIÓN CON PYTHON

Objetivo del capítulo

Al finalizar este capítulo, los lectores podrán comprender los fundamentos de la programación como estrategia educativa, vinculándolos con el desarrollo del pensamiento computacional y la resolución sistemática de problemas en contextos escolares. Asimismo, se busca que los lectores identifiquen los principios básicos que hacen de Python una herramienta accesible y adecuada para la enseñanza y el aprendizaje de la programación, reconociendo su potencial pedagógico para favorecer el razonamiento lógico, la creatividad y la aplicación progresiva de soluciones en el aula.

Aprendizajes esperados

- Explicar la relación entre programación y pensamiento computacional.
- Identificar los elementos esenciales que hacen de Python un lenguaje accesible para principiantes.
- Reconocer cómo el uso de la programación fomenta la creatividad y el razonamiento lógico.
- Relacionar las etapas del pensamiento computacional con la práctica educativa.
- Valorar la programación como recurso pedagógico innovador.

2.1 Introducción: La programación como estrategia educativa

La programación no solo consiste en escribir líneas de código; representa además una estrategia educativa que promueve el pensamiento lógico, la resolución de problemas y la creatividad en los estudiantes. Python, por su sintaxis clara y facilidad de comprensión, es una excelente opción para docentes y alumnos que inician en la programación.

El pensamiento computacional se posiciona como fundamento de esta perspectiva. García-Peñalvo et al. (2019) sostienen que "el pensamiento computacional (PC) es una destreza cognitiva que debe adquirirse desde etapas tempranas para adaptarse a una sociedad informatizada" (p. 19). Asimismo, PC se relaciona con habilidades como la abstracción, la descomposición, la formulación de algoritmos y la depuración del código.

2.2 Pensamiento computacional y su relación con Python

El pensamiento computacional es una capacidad que permite enfrentar problemas con métodos estructurados, utilizando herramientas propias de la informática. Diversos autores han señalado que el pensamiento computacional ha alcanzado una madurez conceptual que permite su integración al currículo escolar (Cervera-Manjarrez et al., 2023b).

Cada línea de código en Python se convierte en una oportunidad para que los estudiantes ejerzan descomposición (dividir un problema en partes), identificación de patrones, abstracción de información irrelevante y creación de algoritmos coherentes.



Figura 16 . Componentes del pensamiento computacional aplicados a Python y la educación.

Nota. La figura muestra cuatro componentes del pensamiento computacional integrados con Python y práctica educativa, destacando cómo cada fase combina lógica de la programación con estrategias de enseñanza para favorecer el aprendizaje.

En el marco de esta obra, pensamiento computacional no se aborda como un contenido aislado, sino como competencia transversal que orienta el aprendizaje de la programación, Python actúa como herramienta pedagógica que permite poner en práctica estas habilidades cognitivas de manera progresiva, facilitando su transferencia a contextos educativos reales.

2.3 La programación en el aula como herramienta pedagógica

En el contexto educativo, la programación no solo cumple un fin tecnológico, sino también pedagógico. Al implementarse en el aula, se convierte en una herramienta transversal que favorece el trabajo colaborativo, la creatividad y la resolución de problemas. Desde esta mirada, el docente no solo enseña a programar, sino que guía procesos de pensamiento estructurado, reflexión y resolución de problemas en el aula.

Python, por su sencillez, permite a los docentes diseñar actividades que simulan situaciones reales: desde juegos matemáticos hasta proyectos de ciencias. Esto no solo motiva al estudiante, sino que también lo convierte en protagonista activo de su propio aprendizaje (Ascencio Jordán et al., 2023).

Además, la incorporación de la programación en el aula permite fortalecer el desarrollo del pensamiento computacional en los estudiantes, promoviendo habilidades como la descomposición de problemas, el reconocimiento de patrones y la construcción de soluciones paso a paso, competencias resultan esenciales en un entorno educativo que busca formar individuos capaces de adaptarse a contextos cambiantes y resolver situaciones complejas de manera lógica y organizada, programación trasciende el aprendizaje técnico y se convierte en un medio para potenciar el razonamiento y la autonomía intelectual.

Por otra parte, uso de lenguajes como Python facilita la integración interdisciplinaria, ya que puede ser aplicado en diversas áreas del

conocimiento como matemáticas, ciencias naturales e incluso en proyectos sociales, versatilidad permite que los estudiantes comprendan la utilidad práctica de lo aprendido, estableciendo conexiones entre la teoría y la realidad, asimismo, fomenta el aprendizaje basado en proyectos, donde los estudiantes trabajan en equipo para diseñar soluciones, fortaleciendo habilidades sociales y comunicativas.

Finalmente, programación en el aula contribuye a la formación de ciudadanos digitales críticos y responsables, interactuar con herramientas tecnológicas, los estudiantes no solo desarrollan competencias técnicas, sino también una comprensión más profunda sobre el funcionamiento de la tecnología en la sociedad, permitiendo asumir un rol activo en la construcción de soluciones innovadoras, preparándolos para enfrentar los desafíos del mundo actual y futuro desde una perspectiva ética y creativa.

2.4 Actividades didácticas: Explorando el pensamiento computacional

Actividad 1: Descomposición (problema cotidiano)

Enunciado:

Un estudiante quiere preparar un jugo natural. Divide el proceso en pasos.

Ejemplo esperado:

- Lavar frutas
- Pelar

- Licuar
- Servir

Aplicación en Python (representación de pasos):

- print("Paso 1: Lavar las frutas")
- print("Paso 2: Pelar las frutas")
- print("Paso 3: Licuar")
- print("Paso 4: Servir")

Propósito:

Aquí Python solo representa el algoritmo (no interacción)

Actividad 2: Reconocimiento de patrones

Enunciado:

Observa los siguientes mensajes y detecta qué se repite:

- print("Hola Ana")
- print("Hola Luis")
- print("Hola Carlos")

Pregunta:

¿Qué parte se repite?

¿Qué cambia?

Propósito:

Identificar estructuras repetitivas (sin usar ciclos aún)

Actividad 3: Abstracción

Enunciado:

De la siguiente información:

Nombre

Edad

Color favorito

Número de cédula

¿Cuál usarías para un saludo?

Aplicación en Python:

- nombre = "Ana"
- print("Hola,", nombre)

Propósito:

Se ignoran datos innecesarios

Actividad 4: Algoritmo

Enunciado:

Organiza los pasos para encender una computadora:

Aplicación en Python:

- print("Paso 1: Encender la computadora")
- print("Paso 2: Esperar que cargue el sistema")
- print("Paso 3: Iniciar sesión")
- print("Paso 4: Abrir un programa")

Propósito:

Refuerza secuencia lógica

2.5 Mini-proyecto: Construcción de instrucciones paso a paso

Enunciado:

Diseñar un programa en Python que describa paso a paso cómo realizar una actividad cotidiana (por ejemplo: preparar un café, estudiar para un examen o enviar un correo).

Código en Python:

- print("=== Cómo preparar un café ===")
- print("Paso 1: Hervir agua")
- print("Paso 2: Colocar café en la taza")

- print("Paso 3: Agregar agua caliente")
- print("Paso 4: Mezclar")
- print("Paso 5: Servir")

Justificación pedagógica:

Descomposición: dividir la actividad en pasos

Patrones: estructura repetitiva de “Paso + acción”

Abstracción: centrarse solo en pasos relevantes

Algoritmo: secuencia lógica ordenada

2.6 Integración STEAM del capítulo

Science (Ciencia): Pensamiento computacional promueve habilidades cognitivas propias del enfoque científico, como la observación, el análisis, la formulación de hipótesis y la resolución sistemática de problemas, comprender fundamentos de programación, estudiantes desarrollan una forma de razonamiento estructurado que puede aplicarse al análisis de fenómenos naturales, experimentos escolares y procesos de investigación básica.

Technology (Tecnología): Este capítulo introduce la programación como una competencia digital esencial en la educación actual; Python se presenta como una herramienta tecnológica accesible que permite a los futuros docentes comprender el funcionamiento de los sistemas

digitales y preparar a sus estudiantes para interactuar críticamente con tecnología, no solo como usuarios, sino como creadores de soluciones.

Engineering (Ingeniería): La programación vincula el pensamiento ingenieril al requerir descomposición de problemas, planificación de soluciones y creación de algoritmos claros y eficientes, a través del pensamiento computacional, estudiantes aprenden a diseñar procesos paso a paso, evaluar alternativas y mejorar soluciones, habilidades fundamentales en la ingeniería y en la resolución de problemas complejos.

Arts (Arte): El aprendizaje de la programación fomenta la creatividad al permitir múltiples formas de abordar un mismo problema; Python ofrece un entorno flexible donde estudiantes pueden explorar ideas, proponer soluciones originales y expresar su pensamiento de manera personal, el enfoque fortalece la creatividad, comunicación y expresión de ideas en contextos educativos.

Mathematics (Matemáticas): El pensamiento computacional se apoya en principios matemáticos como lógica, secuencia, abstracción y análisis de patrones, trabajar con programación, estudiantes refuerzan el razonamiento lógico-matemático, comprendiendo cómo las estructuras formales y los algoritmos permiten representar y resolver problemas de manera precisa y ordenada.

2.7 Recursos digitales complementarios del capítulo

2.7.1 Fichas didácticas

Como apoyo audiovisual, puede escanear el código QR de la Figura 17 para acceder a las fichas didácticas donde se ejemplifica los componentes que incluye el pensamiento computacional.



Figura 17. Código QR – Fichas didácticas

Nota. Al escanear el código, se visualizarán fichas didácticas relacionadas con los componentes del pensamiento computacional.

2.8 Resumen del capítulo

En este capítulo se abordaron los fundamentos de la programación como una estrategia educativa orientada al desarrollo del pensamiento computacional, destacó que programar no consiste únicamente en escribir código, sino estructurar soluciones a problemas mediante

procesos lógicos y organizados; Python se presentó como un lenguaje accesible, claro y adecuado para docentes y estudiantes que inician en la programación.

Se profundizó en el concepto de pensamiento computacional como una competencia clave en la educación actual, destacando sus cuatro componentes principales: descomposición, reconocimiento de patrones, abstracción y algoritmos. Estos elementos permiten analizar problemas complejos, identificar regularidades, centrarse en lo esencial y construir soluciones paso a paso, favoreciendo un aprendizaje estructurado y significativo, asimismo, se analizó el papel de la programación en el aula como una herramienta pedagógica que promueve la creatividad, el trabajo colaborativo y la resolución de problemas, evidenció que Python no solo permite desarrollar habilidades técnicas, sino también competencias cognitivas aplicables a distintas áreas del conocimiento dentro del enfoque STEAM.

Finalmente, mediante actividades prácticas y un mini-proyecto, se consolidaron los conceptos abordados, permitiendo a los estudiantes aplicar el pensamiento computacional en situaciones cotidianas, experiencias refuerzan la idea de que la programación puede ser integrada de manera progresiva en el aula, facilitando un aprendizaje activo, contextualizado y orientado a la construcción del conocimiento.

2.9 Autoevaluación del capítulo II

Pregunta 1. ¿Qué es el pensamiento computacional?

a) Escribir código en cualquier lenguaje

b) Resolver problemas mediante procesos lógicos y estructurados
c) Usar computadoras diariamente
d) Aprender solo programación

Pregunta 2. ¿Cuál de los siguientes NO es un componente del pensamiento computacional?

a) Descomposición
b) Abstracción
c) Programación avanzada
d) Algoritmos

Pregunta 3. La descomposición consiste en:

a) Memorizar código
b) Dividir un problema en partes más pequeñas
c) Crear gráficos
d) Ejecutar programas

Pregunta 4. El reconocimiento de patrones permite:

a) Identificar similitudes en problemas
b) Escribir código más rápido
c) Usar internet
d) Instalar programas

Pregunta 5. La abstracción implica:

a) Considerar toda la información
b) Ignorar lo irrelevante y centrarse en lo importante
c) Usar funciones

d) Crear archivos

Pregunta 6. Un algoritmo es:

a) Un programa complejo
b) Una secuencia ordenada de pasos para resolver un problema
c) Un lenguaje de programación
d) Un editor de texto

Pregunta 7. ¿Por qué Python es adecuado para principiantes?

a) Porque es obligatorio
b) Porque tiene sintaxis clara y sencilla
c) Porque es difícil
d) Porque solo sirve para expertos

Pregunta 8. En el aula, la programación permite:

a) Solo usar computadoras
b) Desarrollar pensamiento lógico y creatividad
c) Evitar el trabajo en grupo
d) Memorizar comandos

Pregunta 9. En la actividad de patrones, ¿qué cambia en los mensajes?

a) El saludo
b) El nombre
c) El código
d) El programa

Pregunta 10. El mini-proyecto del capítulo permite:

a. Memorizar código
b. Aplicar pensamiento computacional en situaciones reales
c. Instalar Python
d. Crear videojuegos

CAPÍTULO III

3 VARIABLES, OPERADORES Y ESTRUCTURAS CONDICIONALES EN PYTHON

Objetivo del capítulo

Al finalizar este capítulo, los lectores serán capaces de aplicar los conceptos de variables, tipos de datos, operadores y comentarios en Python, así como utilizar estructuras condicionales para la toma de decisiones en programas educativos. Además, se busca que los lectores desarrollen la capacidad de descomponer problemas, establecer criterios lógicos y diseñar soluciones algorítmicas simples, reconociendo errores comunes en principiantes y estrategias didácticas para su abordaje en el aula.

Aprendizajes esperados

- Definir el concepto de variable en el contexto de Python y su aplicación educativa.
- Identificar y clasificar los principales tipos de datos.
- Usar operadores aritméticos, relacionales y lógicos en programas básicos.
- Documentar el código mediante comentarios claros y útiles.
- Aplicar estructuras condicionales if/else en programas educativos.
- Elaborar un mini-proyecto que combine los recursos estudiados.

3.2 Introducción: Primeros pasos prácticos en Python

Este capítulo se centra en los elementos básicos del lenguaje Python que todo principiante necesita conocer. A partir de aquí, aprenderemos cómo declarar variables, identificar los principales tipos de datos, utilizar operadores, escribir comentarios y aplicar estructuras condicionales. Estos conceptos son la base para construir programas simples que servirán como apoyo en la enseñanza y el aprendizaje de la programación. Desde esta perspectiva, el aprendizaje de estos elementos básicos permite a los estudiantes comenzar a estructurar problemas, analizar condiciones y tomar decisiones lógicas mediante algoritmos sencillos.

3.3 Elementos básicos de Python

3.3.1 Variables

Las variables son espacios de memoria que permiten almacenar datos para su uso posterior en el programa, pueden contener datos de distintos tipos, como números, palabras o valores lógicos, y reciben un nombre que las identifica dentro del programa.

Una manera sencilla de comprenderlo es pensar en las variables como si simularan cajas en la vida real, que normalmente se utilizan para guardar objetos. Cada caja tiene una etiqueta (el nombre de la variable) y dentro de ella se coloca un contenido específico (el dato).

Así, cuando el programa necesita acceder a esa información, basta con llamar al nombre de la variable para “abrir la caja” y utilizar lo que hay dentro.



Figura 18. Representación visual de variables en Python.

Nota. La figura muestra las variables como cajas con nombre y contenido, representando cómo almacenan datos en Python.

Ejemplo en Python:

```
nombre = "Walter Benítez Martínez"

edad = 28

print ("Me llamo:", nombre, "y tengo", edad, "años")
```

En este caso, la variable nombre almacena las palabras “Walter Benítez Martínez” y la variable edad guarda el número “28”, al ejecutar el programa, Python muestra el contenido de ambas variables en la pantalla.

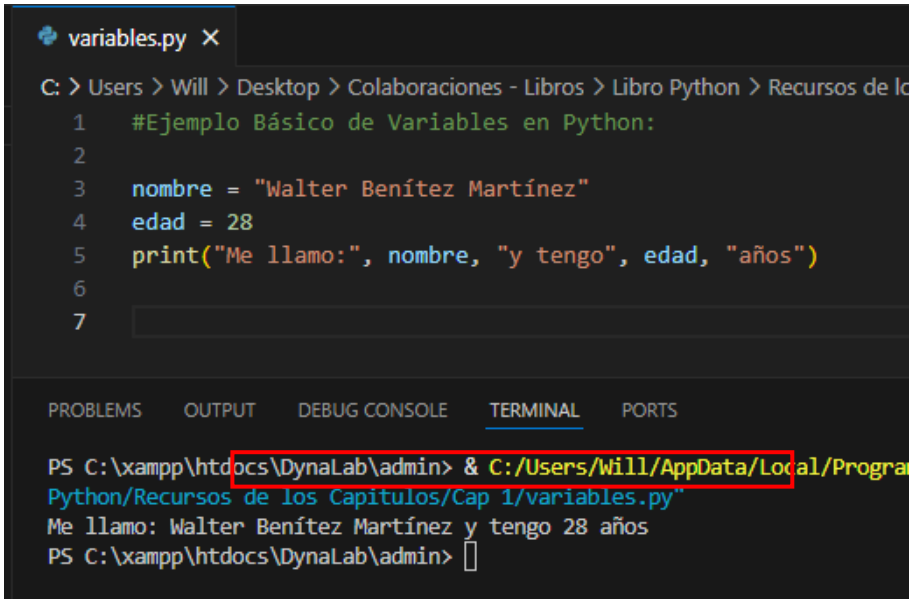


Figura 19. Ejemplo de variables en Python.

Nota. Las variables facilitan la personalización de los programas, ya que permiten trabajar con diferentes datos sin necesidad de cambiar la estructura del código.

Desde el pensamiento computacional, el uso de variables favorece la abstracción, ya que permite representar información relevante sin necesidad de trabajar directamente con valores concretos, facilitando la comprensión y reutilización de soluciones.

3.3.2 Tipos de datos

En la vida cotidiana usamos distintos tipos de información: números, palabras, medidas o respuestas de “sí” y “no”. En Python, estos se llaman tipos de datos.

Tabla 2. Tipos de datos básicos en Python.

Tipo de dato	Descripción	Ejemplo en Python
Enteros (int)	Números completos, positivos o negativos.	5, 100, -23
Decimales (float)	Números con parte decimal o fraccionaria.	3.14, 1.60
Texto (str)	Palabras o frases escritas entre comillas.	"Hola"
Lógicos (bool)	Valores de verdadero o falso.	True, False

Nota. La tabla resume tipos de datos más usados en Python, mostrando su nombre, una breve explicación y ejemplos prácticos para principiantes en programación.

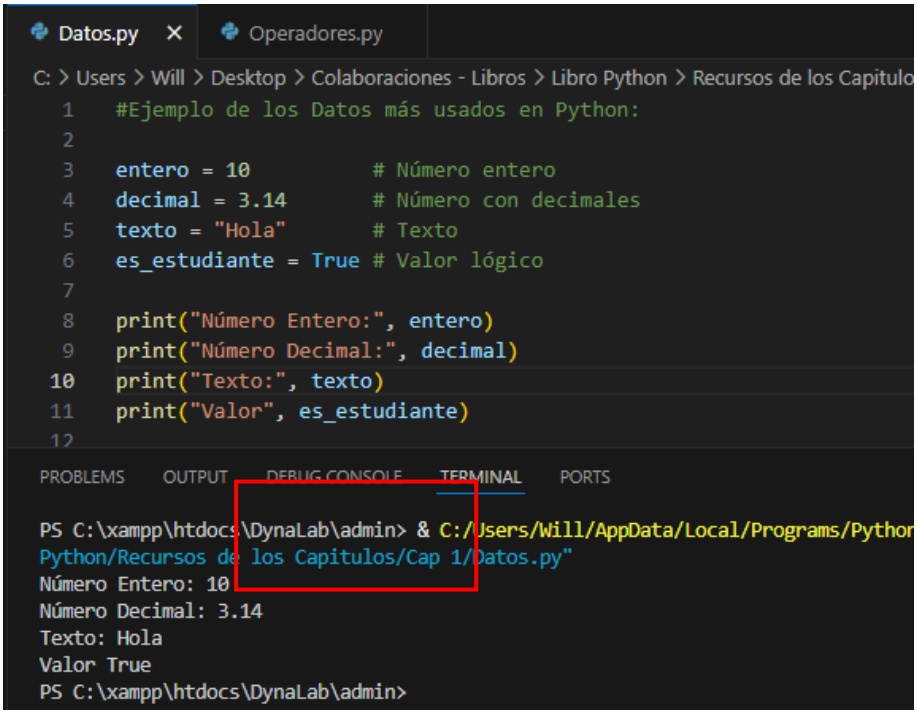


Figura 20. Ejemplo práctico de tipos de datos en Python.

Nota. Figura que evidencia cómo los diferentes tipos de datos pueden declararse en Python y visualizarse con la instrucción print(), lo que facilita su comprensión en contextos educativos.

Identificar y utilizar correctamente los tipos de datos contribuye al pensamiento computacional al permitir clasificar información y seleccionar las operaciones adecuadas para su procesamiento.

3.3.3 Operadores

Los operadores son los “signos” o “palabras” que permiten hacer cálculos o comparaciones.

Tabla 3. Operadores básicos en Python.

Tipo de operador	**Descripción**	**Ejemplos en Python**
Aritméticos	Permiten realizar operaciones matemáticas básicas.	+ (suma), - (resta), * (multiplicación), / (división)
Relacionales	Sirven para comparar valores y devolver un resultado verdadero o falso.	> (mayor que), < (menor que), == (igual), != (diferente)
Lógicos	Combinan condiciones lógicas para evaluar situaciones complejas.	and (y), or (o), not (no)

Nota. Los operadores permiten que un programa realice cálculos, compare valores y tome decisiones, convirtiéndose en herramientas esenciales para la programación en Python.

Para complementar la explicación de los operadores, a continuación, se muestra un ejemplo sencillo en Python que utiliza operadores aritméticos y relacionales, evidenciando cómo la computadora interpreta estas instrucciones.

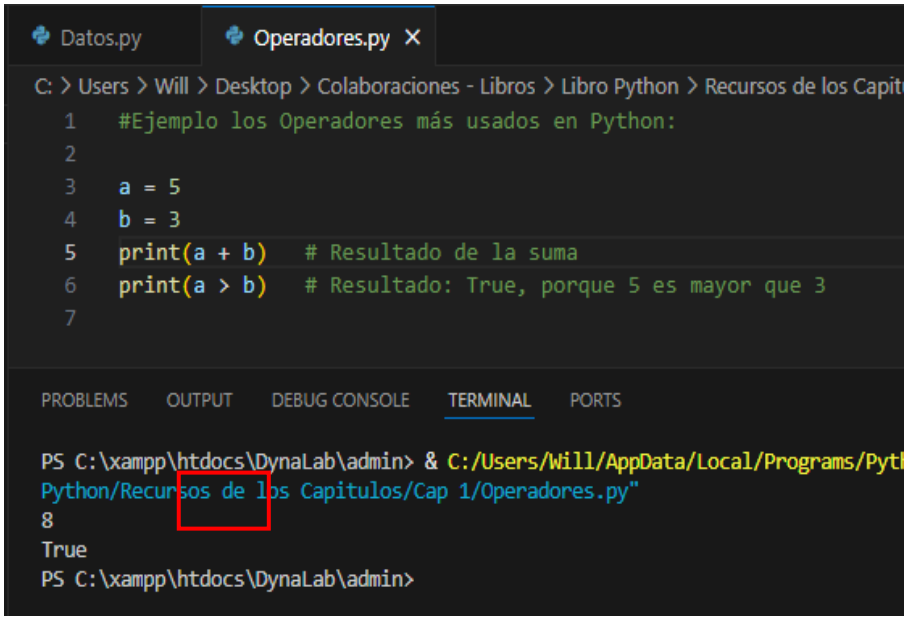


Figura 21. Ejemplo práctico de operadores en Python.

Nota. La figura muestra cómo los operadores permiten realizar cálculos matemáticos y comparaciones lógicas, siendo herramientas fundamentales para la resolución de problemas en programación.

El uso de operadores aritméticos, relacionales y lógicos permite establecer reglas y condiciones, fortaleciendo la capacidad de analizar situaciones y tomar decisiones basadas en criterios definidos.

3.3.4 Comentarios en el código

Un comentario es una explicación escrita dentro del programa, pero que la computadora no ejecuta. En Python se escriben con el símbolo #.

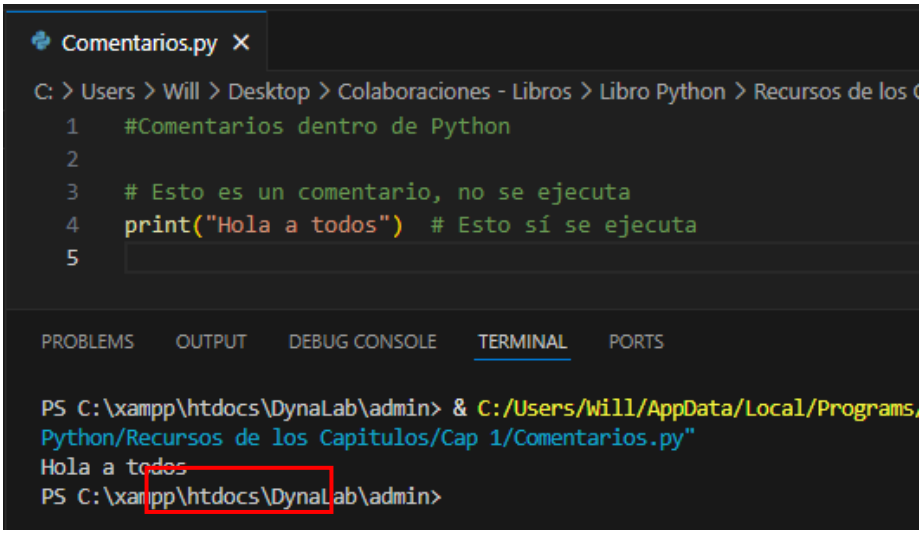


Figura 22. Ejemplo de comentarios en Python.

Nota. La figura muestra cómo los comentarios sirven para explicar el código sin que la computadora los ejecute, mientras que las instrucciones sí son interpretadas y ejecutadas por Python.

3.3.5 Estructuras condicionales

Las condicionales permiten que el programa tome decisiones. Se escriben con if (si), elif (si no, entonces) y else (en caso contrario).

if simple: Evalúa una condición y, si es verdadera, ejecuta un bloque de código.

```
edad = 20

if edad >= 18:

  print("Eres mayor de edad")
```

if – else: Permite ejecutar un bloque si la condición se cumple y otro bloque si no se cumple.

```
edad = 15

if edad >= 18:

    print("Eres mayor de edad")

else:

    print("Eres menor de edad")
```

if - elif – else: Se usa cuando hay múltiples condiciones posibles.

```
nota = 8

if nota >= 9:

    print("Excelente")

elif nota >= 7:

    print("Aprobado")

else:

    print("Reprobado")
```

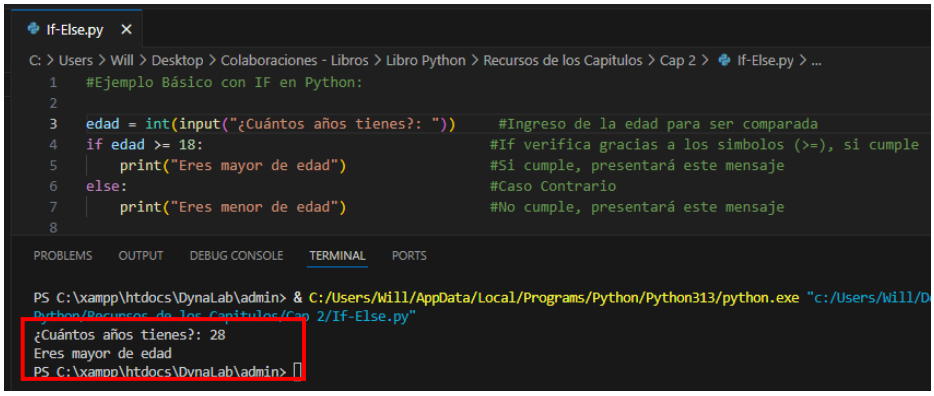


Figura 23. Ejemplo básico de condicional if-else en Python.

Nota. La figura muestra cómo el programa solicita la edad del usuario y, mediante una condición, decide si es mayor o menor de edad, ejemplificando el uso de estructuras condicionales.

Las estructuras condicionales constituyen un elemento central del pensamiento computacional, ya que permiten descomponer un problema en posibles escenarios, evaluar condiciones y ejecutar acciones específicas según los resultados.

Este tipo de razonamiento resulta fundamental para automatizar decisiones y modelar situaciones reales del entorno educativo.

3.4 Actividades didácticas con Python

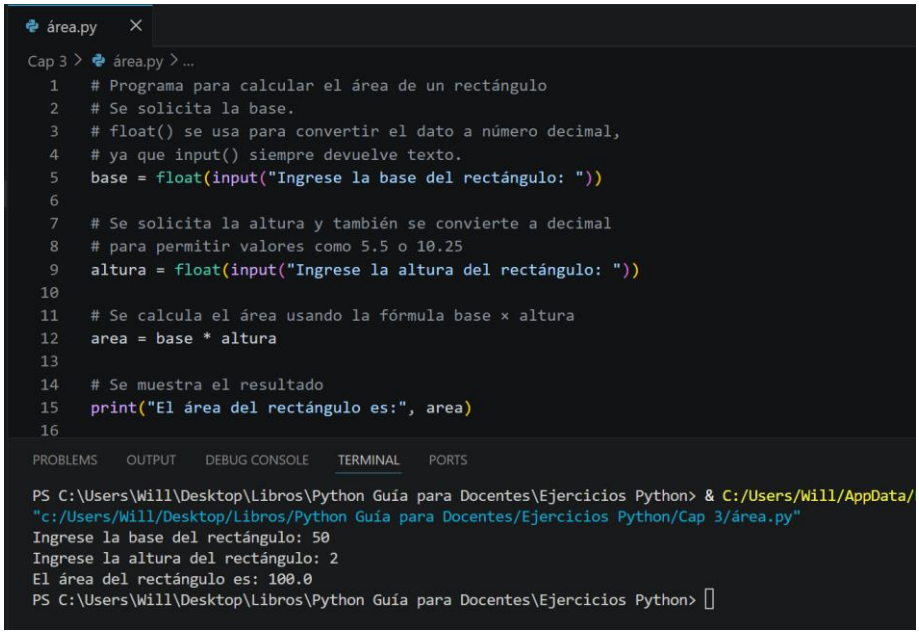


Figura 24. Cálculo del área de un rectángulo.

Nota. Este ejercicio permite al estudiante comprender cómo los datos ingresados por el usuario pueden convertirse a valores numéricos mediante float(), facilitando la realización de operaciones matemáticas, refuerza el uso de variables para almacenar información y procesarla mediante una fórmula sencilla.

Orientación docente: Esta actividad puede integrarse con contenidos de Matemáticas, específicamente con el cálculo de áreas, permite evidenciar cómo una fórmula conocida se traduce en instrucciones dentro de un programa. Asimismo, fortalece la comprensión del proceso entrada – operación – salida, favoreciendo el desarrollo del pensamiento lógico y la aplicación práctica del conocimiento.

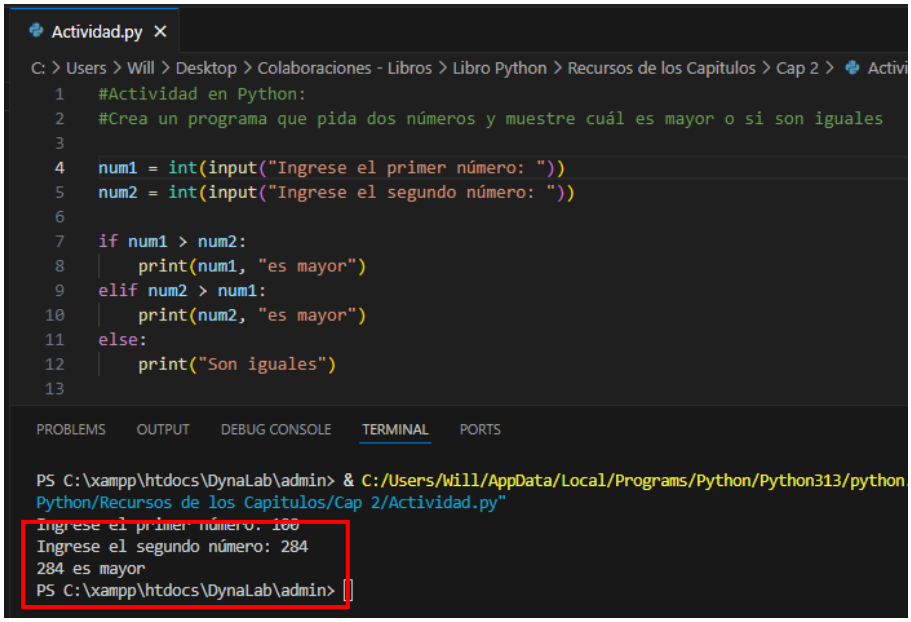


Figura 25. Actividad didáctica con condicionales en Python.

Nota. La figura muestra un programa que solicita dos números al usuario y determina cuál es mayor o si son iguales, aplicando condicionales múltiples con if, elif y else.

Esta actividad promueve el pensamiento computacional al exigir que el estudiante compare valores, establezca condiciones y seleccione una respuesta lógica a partir de un algoritmo simple.

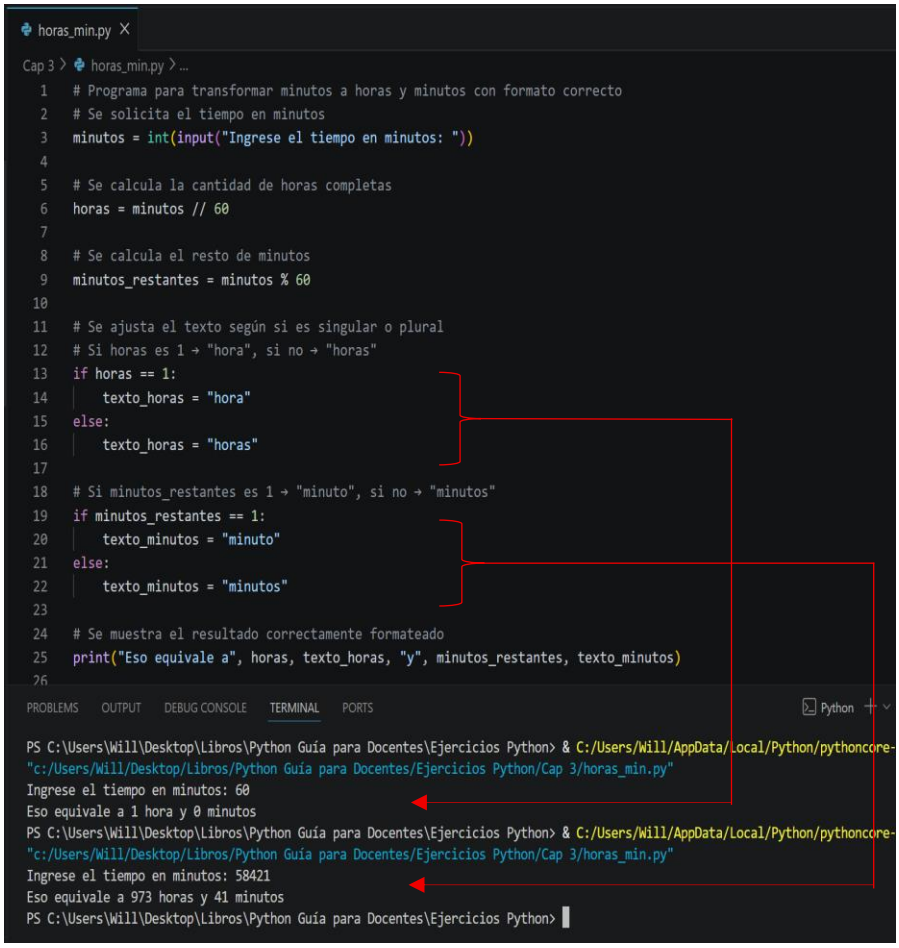


Figura 26. Transformación de tiempo (hora(s) y minuto(s).

Nota. Este ejercicio mejora la presentación del resultado al adaptar el lenguaje según la cantidad, diferenciando entre singular y plural. Además, refuerza el uso de variables y operaciones básicas para transformar datos en información más clara y precisa.

Orientación docente: Esta actividad permite evidenciar cómo la programación no solo realiza cálculos, sino que también puede mejorar la forma en que se comunican los resultados. Introduce el uso de

condicionales simples para adaptar el mensaje, fortaleciendo el pensamiento lógico y la atención al detalle en la construcción de programas.

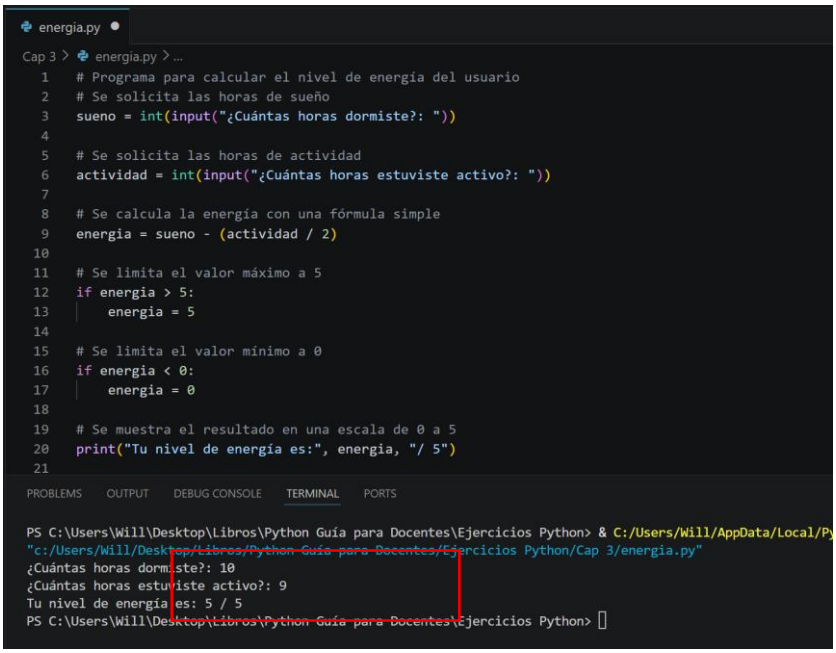


Figura 27. Cálculo del nivel de energía (escala de 0 a 5).

Nota. Este ejercicio permite al estudiante trabajar con variables, operaciones aritméticas y una escala de valores para representar un resultado, a través del uso de límites, se controla que el resultado se mantenga dentro de un rango específico (de 0 a 5), facilitando su interpretación.

Orientación docente: Esta actividad puede emplearse para mostrar cómo la programación no solo realiza cálculos, sino que también organiza los resultados de forma comprensible mediante escalas, introduce el uso de condicionales para controlar valores, fortaleciendo

el pensamiento lógico y la capacidad de estructurar soluciones más realistas.

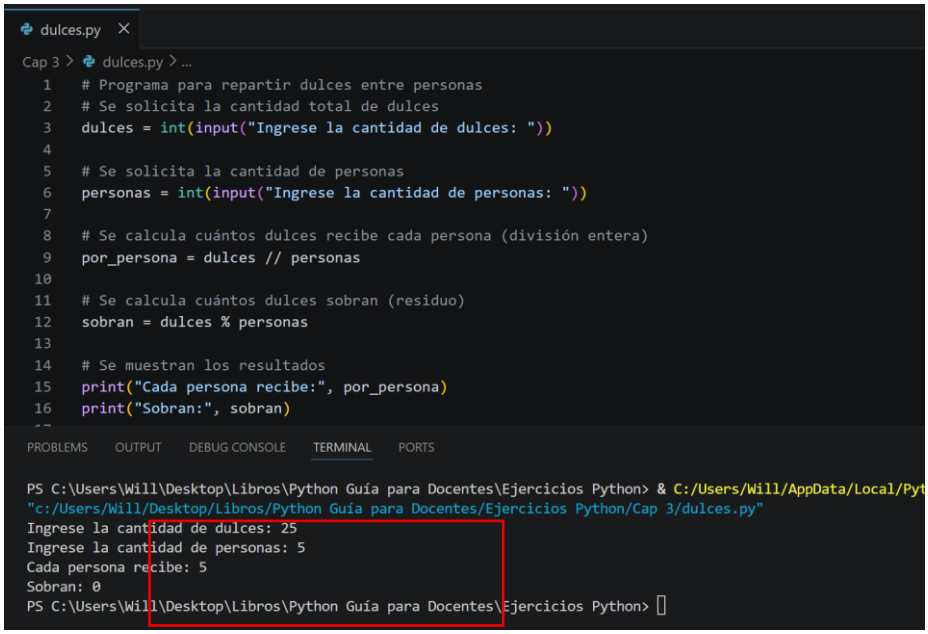


Figura 28. Reparto equitativo de dulces.

Nota. Este ejercicio utiliza la división entera y el residuo para repartir una cantidad de forma equitativa, mostrando cuánto corresponde a cada persona y qué cantidad sobra. Así se comprende mejor el uso de estos operadores.

Orientación docente: Sirve para explicar cómo dividir recursos de manera justa y cómo interpretar los resultados de una división, además, permite introducir el concepto de residuo en situaciones reales.

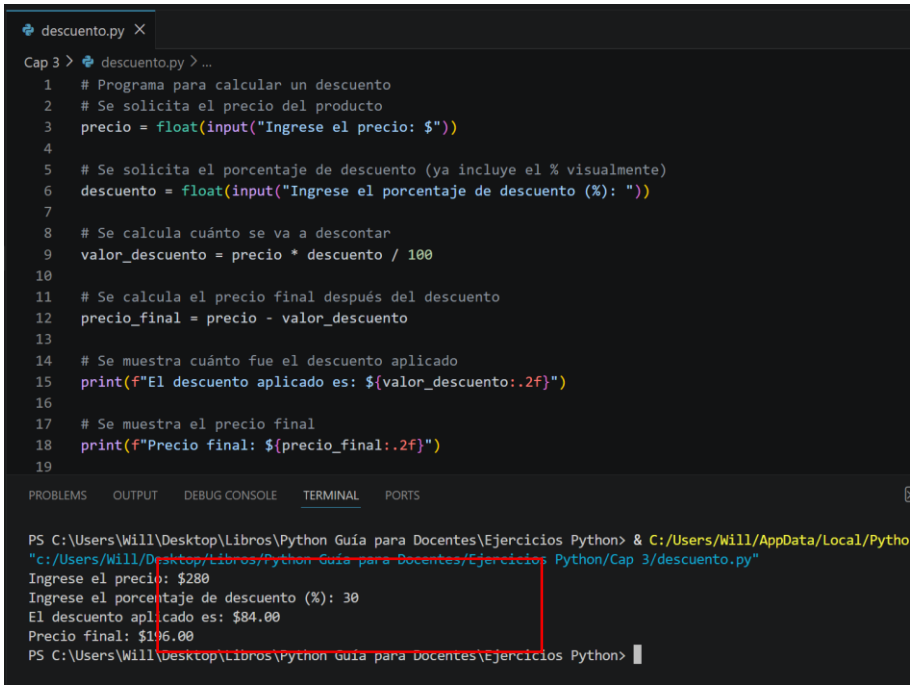


Figura 29. Cálculo de descuento en una compra.

Nota. Este ejercicio permite calcular tanto el valor del descuento como el precio final de un producto, utilizando operadores aritméticos, a través de este proceso, el estudiante comprende cómo un porcentaje se transforma en un valor real y cómo este afecta al precio original, refuerza el uso de variables para almacenar datos intermedios y finales dentro de un mismo programa.

Orientación docente: Esta actividad puede emplearse para explicar el cálculo de porcentajes en situaciones reales como compras o promociones, docente puede guiar al estudiante para que identifique cada paso del proceso: cálculo del descuento y obtención del precio final, también permite fomentar la interpretación de resultados, ayudando a comprender cuánto se ahorra y cómo cambia el valor inicial.

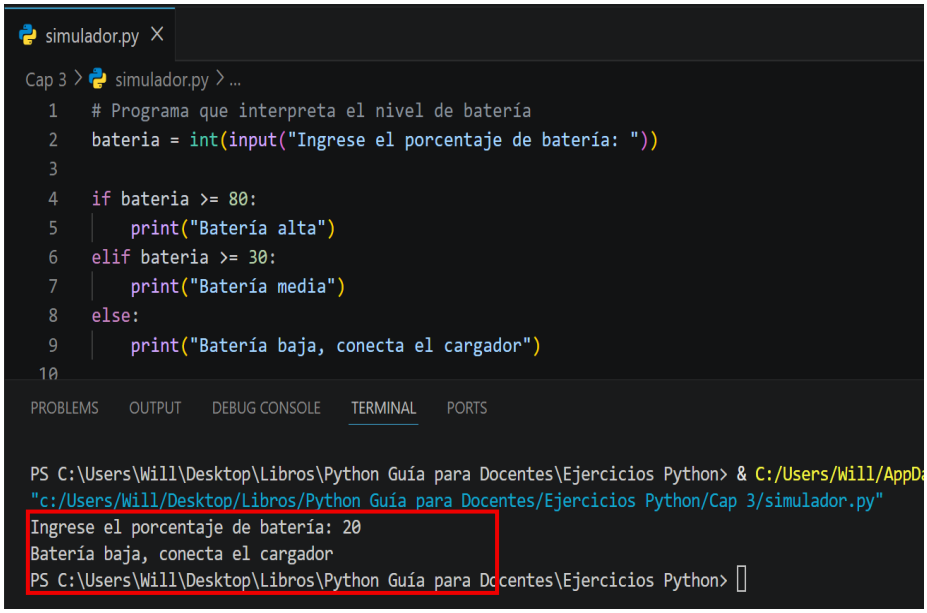


Figura 30. Simulador de batería del celular mediante Python.

Nota. Refuerza operadores relacionales y toma de decisiones con valores numéricos.

Orientación docente: Permite conectar la programación con la vida diaria, haciendo que el estudiante comprenda decisiones automáticas como en dispositivos reales.

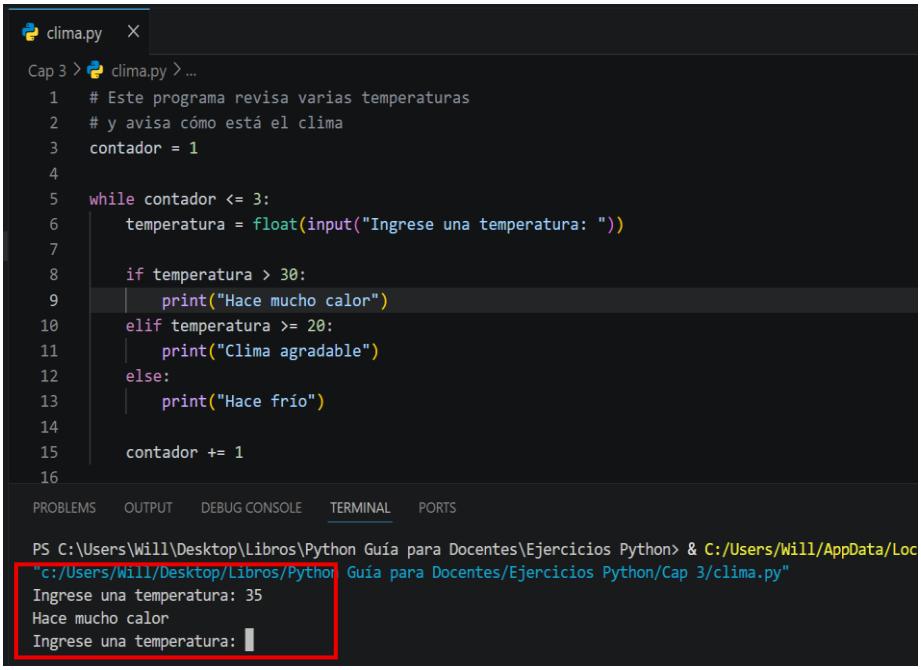


Figura 31. Alerta de clima escolar.

Nota. Este ejercicio combina una estructura repetitiva con condicionales para analizar varias temperaturas seguidas, así, el estudiante practica cómo un programa puede repetir una acción y tomar decisiones en cada intento.

Orientación docente: Es una actividad distinta a las comunes de notas o descuentos, pero sigue siendo sencilla de comprender, además, permite relacionar la programación con situaciones cotidianas y reforzar la lógica paso a paso.

3.5 Mini-proyecto: Clasificador de notas

Este mini-proyecto ayuda a los docentes a relacionar la programación con el sistema educativo, mostrando cómo Python puede simular procesos de evaluación.

Enunciado del ejercicio:

Diseña un programa en Python que solicite al usuario ingresar la calificación de un estudiante y muestre en pantalla un mensaje con su desempeño (por ejemplo: Excelente = 10 / 9, Muy bien = 8, Aprobado 7 o Reprobado 0 - 6).

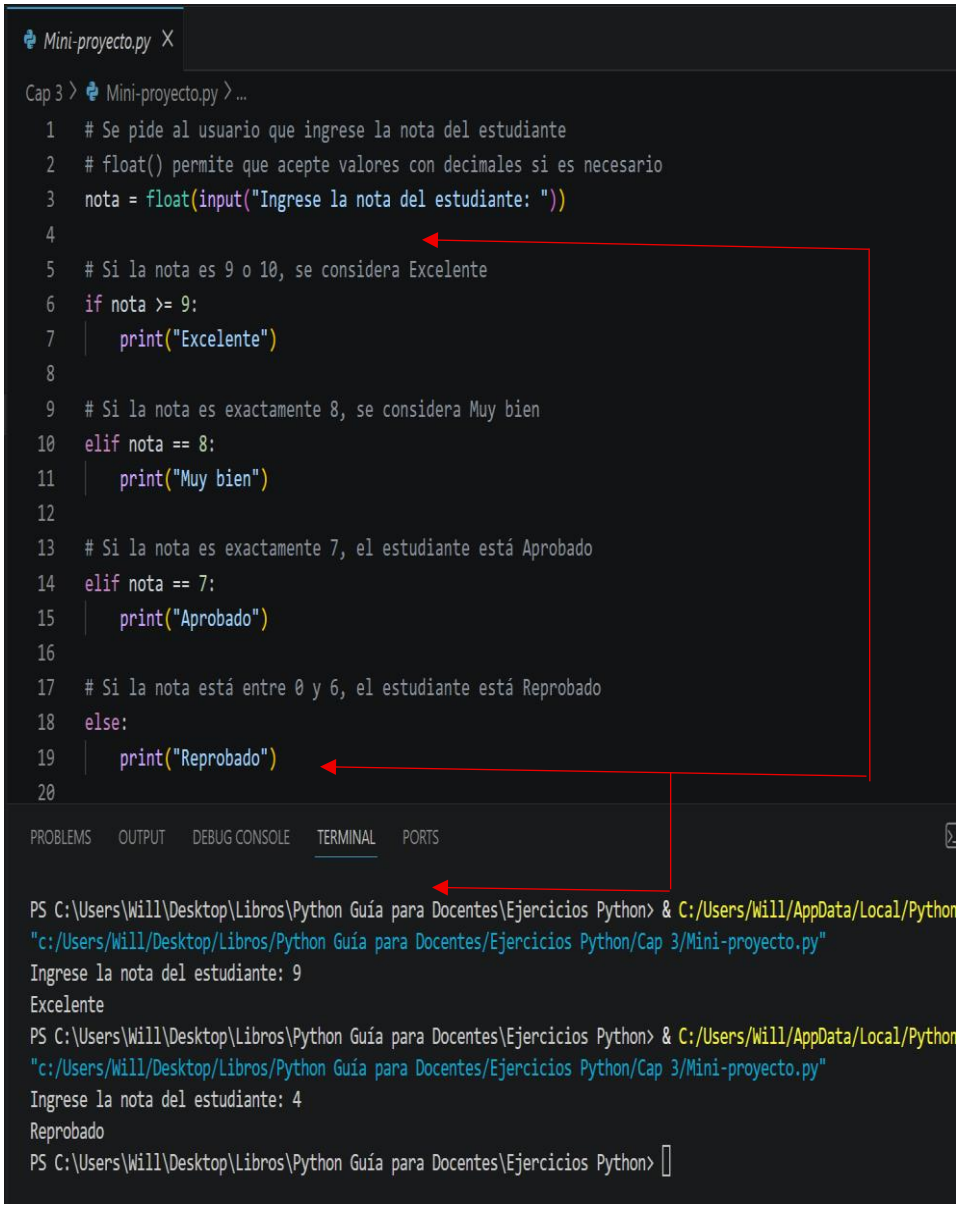


Figura 32. Desarrollo del Mini-proyecto.

Nota. Este mini-proyecto permite al estudiante clasificar el rendimiento académico a partir de una calificación ingresada, utilizando estructuras condicionales en Python, a través de esta actividad, se evidencia cómo

un programa puede evaluar diferentes rangos de valores y generar respuestas específicas según criterios establecidos, refuerza el uso de variables, operadores relacionales y la toma de decisiones dentro de un algoritmo.

Esta actividad puede emplearse para vincular la programación con procesos reales del contexto educativo, como la evaluación del desempeño estudiantil, permite que el estudiante comprenda cómo un sistema automatiza la toma de decisiones a partir de condiciones previamente definidas, asimismo, favorece el desarrollo del pensamiento computacional al requerir la interpretación de datos, la definición de criterios y la construcción de soluciones lógicas mediante estructuras condicionales.

Desde el pensamiento computacional, este mini-proyecto integra descomposición del problema, definición de criterios de evaluación y automatización de decisiones mediante estructuras condicionales, simulando procesos reales del contexto educativo.

3.6 Integración STEAM del mini-proyecto

Science (Ciencia): El proyecto se relaciona con procesos de observación, análisis e interpretación de datos educativos, asimismo, permite comprender cómo las calificaciones representan información cuantitativa que puede clasificarse y transformarse siguiendo criterios objetivos.

Technology (Tecnología): El uso de Python para automatizar la clasificación de notas evidencia cómo la tecnología puede optimizar

tareas administrativas escolares, mejorar la eficiencia del cálculo manual y garantizar precisión en los resultados.

Engineering (Ingeniería): El diseño del algoritmo requiere planificar, estructurar, probar y depurar el código siguiendo un proceso iterativo propio de la ingeniería, estudiante aplica habilidades de resolución de problemas al diseñar condiciones lógicas que clasifican los datos correctamente.

Arts (Arte): Estudiante puede explorar formas creativas de presentar los resultados, como mensajes personalizados, colores en la interfaz o representaciones visuales de los niveles de desempeño, fomentando el diseño expresivo y comunicación visual en proyectos digitales.

Mathematics (Matemáticas): La clasificación del rendimiento académico utiliza comparaciones numéricas, intervalos, operadores relacionales y lógica matemática, estudiante fortalece su razonamiento al analizar rangos, límites y valores dentro de un sistema de evaluación.

3.7 Recursos digitales complementarios del capítulo

Para reforzar los contenidos de este capítulo, se han preparado archivos digitales que contienen los ejemplos prácticos trabajados en clase, incluyendo programas con variables, tipos de datos, operadores y estructuras condicionales, mismo recursos que permiten a los docentes y estudiantes experimentar directamente con los códigos, adaptarlos según sus necesidades y reutilizarlos como base para nuevas actividades educativas.

El acceso a los materiales se facilita mediante un código QR, lo que garantiza una descarga rápida y sencilla de los archivos complementarios.



Figura 33. Código QR para descarga de archivos complementarios del Capítulo 3.

Nota. El código QR proporciona acceso directo a los programas y ejemplos trabajados en este capítulo, permitiendo a docentes y estudiantes descargar, modificar y reutilizar los recursos en el aula.

3.8 Resumen del capítulo

En este capítulo se introdujeron los elementos fundamentales del lenguaje Python que permiten comenzar a construir programas con mayor nivel de complejidad, abordó el concepto de variables como espacios de almacenamiento de información, permitiendo representar datos dentro de un programa, asimismo, se explicó importancia de los tipos de datos, los cuales determinan la naturaleza de la información que se manipula, como números, texto o valores lógicos.

También se estudiaron los operadores, entendidos como herramientas que permiten realizar cálculos, comparaciones y evaluaciones lógicas, operadores aritméticos, relacionales y lógicos constituyen la base para procesar información y establecer condiciones dentro de un programa, facilitando la toma de decisiones en función de ciertos criterios, aspecto clave del capítulo fue el uso de estructuras condicionales, como if, elif y else, las cuales permiten que un programa ejecute diferentes acciones según se cumplan determinadas condiciones, tipo de estructuras representa un avance significativo en la programación, ya que introduce la capacidad de tomar decisiones, simulando situaciones reales donde existen múltiples posibilidades.

Actividades prácticas y un mini-proyecto, se integraron conceptos desarrollados, permitiendo a los estudiantes aplicar variables, operadores y condicionales en la resolución de problemas sencillos, enfoque del pensamiento computacional, estos elementos fortalecen la capacidad de abstracción, análisis lógico y toma de decisiones, contribuyendo a la construcción progresiva de programas más complejos.

3.9 Autoevaluación del capítulo

Pregunta 1: ¿Qué es una variable en Python?

a) Un programa completo.
b) Un espacio de memoria para almacenar datos.
c) Una instrucción condicional.
d) Un operador aritmético.

Pregunta 2: ¿Cuál de los siguientes NO es un tipo de dato en Python?

a) int
b) float
c) str
d) liston

Pregunta 3: ¿Qué operador se utiliza para comparar igualdad?

a) =
b) ==
c) !=
d) and

Pregunta 4: ¿Cuál de los siguientes es un tipo de dato numérico decimal?

a) int
b) str
c) float
d) bool

Pregunta 5: ¿Qué tipo de dato representa valores de verdadero o falso?

a) int
b) float
c) str
d) bool

Pregunta 6: ¿Cuál es el resultado de la expresión 5 > 3?

a) 5
b) 3
c) True
d) False

Pregunta 7: Escribe un ejemplo de comentario en Python.

Pregunta 8: Programa un condicional que muestre “Aprobado” si la nota es mayor o igual a 7 y “Reprobado” en caso contrario.

Pregunta 9: ¿Para qué sirven los operadores lógicos?

a) Para imprimir datos
b) Para combinar condiciones
c) Para guardar información
d) Para crear variables

Pregunta 10: En la estructura if, ¿qué ocurre si la condición es verdadera?

a) Se ignora el código
b) Se ejecuta el bloque de instrucciones
c) Se detiene el programa
d) Se genera un error

CAPÍTULO IV

4 CICLOS, FUNCIONES Y PROCEDIMIENTOS EN PYTHON

Objetivo del capítulo

Al finalizar este capítulo, los lectores serán capaces de aplicar los ciclos, las funciones y los procedimientos en Python como recursos para estructurar programas más eficientes y reutilizables en contextos educativos. Además, se busca que los lectores desarrollen el pensamiento algorítmico y la capacidad de abstracción, automatizando tareas repetitivas y diseñando soluciones organizadas que favorezcan la resolución lógica de problemas en el aula.

Aprendizajes esperados

- Comprender la estructura y funcionamiento de los ciclos for y while en Python.
- Reconocer la importancia de las funciones y procedimientos como recursos de organización del código.
- Diseñar y ejecutar funciones simples que incluyan parámetros y valores de retorno.
- Aplicar los ciclos y funciones en mini-proyectos educativos contextualizados.
- Promover el pensamiento lógico mediante la construcción de programas estructurados y reutilizables.

4.1 Introducción: La lógica detrás de la repetición

En la vida cotidiana repetimos acciones constantemente: saludamos a varias personas, realizamos ejercicios o revisamos listas de tareas. En programación ocurre algo similar; cuando una instrucción debe repetirse varias veces, se utilizan **ciclos**.
Python permite automatizar estas repeticiones con estructuras simples como for y while, evitando escribir el mismo código múltiples veces, las funciones permiten agrupar bloques de código que se pueden reutilizar en diferentes partes de un programa. Esto no solo mejora la eficiencia, sino que también favorece el desarrollo del pensamiento lógico y la planificación secuencial**,** competencias esenciales para los futuros docentes.

Según Tinoco et al. (2023), enseñar programación con Python en contextos educativos "contribuye al desarrollo de competencias cognitivas y a la comprensión de la lógica computacional mediante la práctica sistemática y la reutilización de código" (p. 30).

Por su parte, Raschka et al. (2020) destacan que Python, al ser un lenguaje de propósito general, permite "automatizar tareas repetitivas y aplicar estructuras modulares que simplifican la enseñanza de algoritmos y procesos educativos" (p. 8).

Las estructuras repetitivas permiten organizar el pensamiento en secuencias lógicas, favoreciendo la automatización de procesos y el desarrollo del pensamiento computacional mediante la programación.

4.2 Ciclos en Python

4.2.1 Ciclo for

El ciclo for se utiliza cuando se conoce cuántas veces se desea repetir una acción. Por ejemplo, si queremos imprimir los números del 1 al 5:



Figura 34. Estructura Repetitiva for en Python.

Nota. El ciclo for repite una instrucción un número fijo de veces, facilitando la automatización de tareas repetitivas en Python.

En este caso:

- range(1,6) genera los números del 1 al 5 (el 6 no se incluye).
- i actúa como un contador que cambia su valor en cada repetición.
- El bloque dentro del for se repite hasta que el rango termina.

Analogía pedagógica:

El ciclo for es como pasar lista en clase: sabes exactamente cuántos estudiantes hay y repites el saludo hasta terminar la lista. Desde el pensamiento computacional, el ciclo for permite identificar patrones de repetición y controlar secuencias de acciones de manera ordenada.

De acuerdo con Gómez, (2021), “los ciclos constituyen uno de los pilares del razonamiento computacional, al permitir que los estudiantes comprendan la noción de secuencia y repetición dentro del algoritmo” (p. 12).

4.2.2 Ciclo while

El ciclo while se usa cuando no se sabe con certeza cuántas veces se repetirá una acción, pero existe una condición lógica que debe cumplirse.

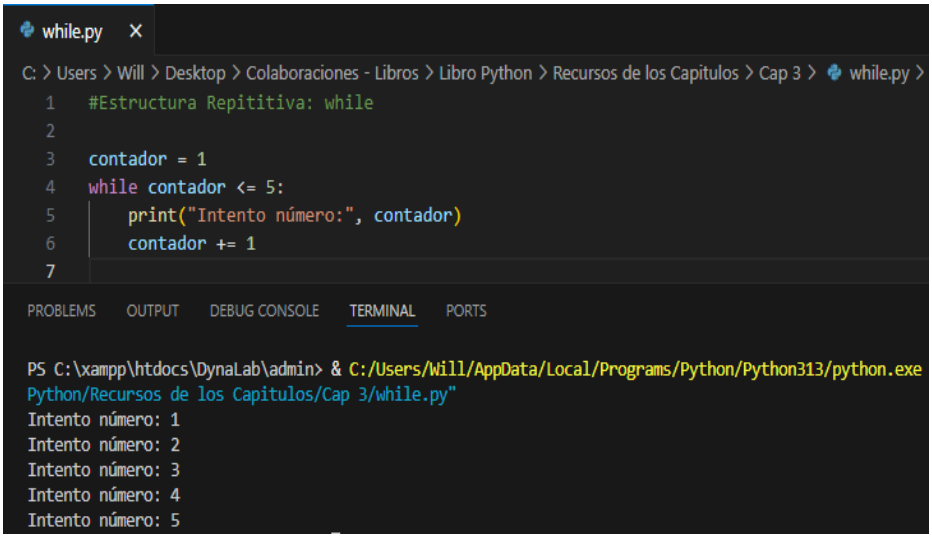


Figura 35. Estructura Repetitiva while en Python.

Nota. El ciclo while ejecuta instrucciones mientras una condición sea verdadera, ideal para repeticiones con fin indeterminado o controladas por una variable.

Este ciclo se ejecuta mientras la condición contador <= 5 sea verdadera. Cuando la condición deja de cumplirse, el ciclo se detiene automáticamente.

Analogía pedagógica:

El ciclo while es como practicar una tabla de multiplicar hasta que el estudiante la diga correctamente; no sabes cuántas veces lo intentará, pero el ciclo continúa hasta lograrlo. El ciclo while favorece el razonamiento lógico al exigir el control de condiciones y la verificación constante de un estado para detener la repetición.

La utilización de ciclos también promueve la autonomía y la experimentación, pilares del aprendizaje activo, como señalan Jaimes et al. (2018), en su análisis sobre pensamiento computacional en educación media.

4.2.3 Dificultades comunes y soluciones

Tabla 4. Dificultades comunes y soluciones en ciclos

Problema	**Posible causa**	**Solución**
El ciclo no se detiene	No se actualiza la variable de control	Asegúrate de usar contador += 1 o una condición que cambie

El ciclo se ejecuta infinitamente	Condición siempre verdadera	Revisa la condición del while o el rango del for
No se ejecuta el ciclo	Condición inicial no se cumple	Verifica los valores iniciales de la variable
Se obtienen resultados incorrectos	Error en la lógica del ciclo	Revisa la secuencia de instrucciones dentro del ciclo

Nota. La tabla resume errores frecuentes al programar en Python y sus soluciones, ayudando a identificar y corregir fallos en ciclos y funciones durante el aprendizaje.

4.3 Funciones y procedimientos

Las funciones permiten abstraer procesos complejos en bloques simples y reutilizables, lo que facilita la comprensión modular de los problemas. Esta capacidad es central en el pensamiento computacional, ya que permite organizar soluciones, reducir la complejidad y reutilizar algoritmos en distintos contextos educativos.

Diferenciar entre funciones y procedimientos fortalece la organización lógica del código y la planificación de soluciones, competencias propias del pensamiento computacional.

4.3.1 Definición de función

Una función es un conjunto de instrucciones que realiza una tarea específica.
Se define con la palabra clave def y puede recibir parámetros (datos que necesita para funcionar) y devolver un resultado.

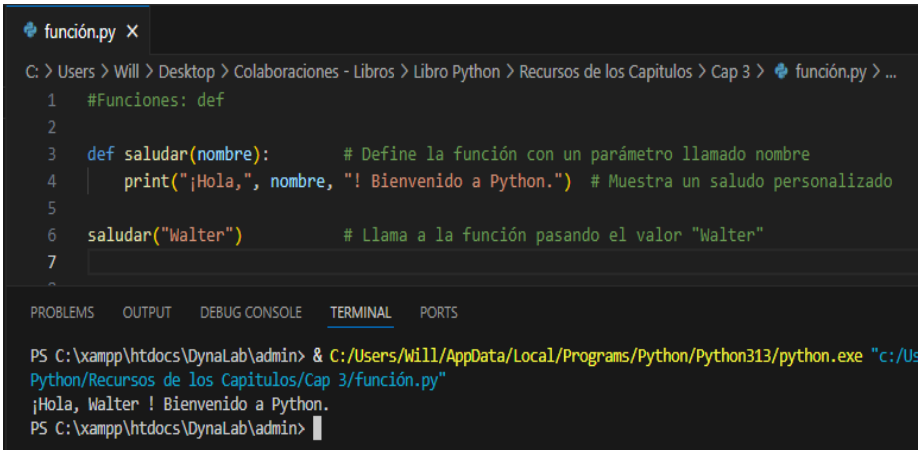


Figura 36. Definición y ejecución de funciones en Python.

Nota. Definir una función no la ejecuta automáticamente; es necesario llamarla por su nombre para que realice la acción programada.

Para usar la función:

saludar("Walter")

Aquí, la función recibe un parámetro (nombre) y muestra un saludo personalizado.
Las funciones hacen que el código sea más ordenado, legible y reutilizable.

Como expone Chimarro-Amaguaña et al. (2023), “la enseñanza basada en funciones promueve la comprensión modular del código y favorece la autonomía de los estudiantes al resolver problemas complejos mediante fragmentos más simples” (p. 15).

4.3.2 Funciones con valor de retorno

Una función puede devolver un resultado al programa principal mediante la instrucción return.



Figura 37. Funciones con valores de retorno en Python.

Nota. Las funciones con return devuelven un resultado al programa principal, permitiendo reutilizar cálculos o datos en otras partes del código.

return envía el valor calculado de vuelta al programa principal, lo que permite reutilizarlo en otras operaciones.

4.3.3 Procedimientos

En este libro, se utiliza el término procedimiento como recurso didáctico para referirse a una función que realiza una acción sin especificar un valor de retorno. En Python, estas funciones se definen mediante def y, cuando no incluyen una sentencia return explícita, devuelven implícitamente el valor None.

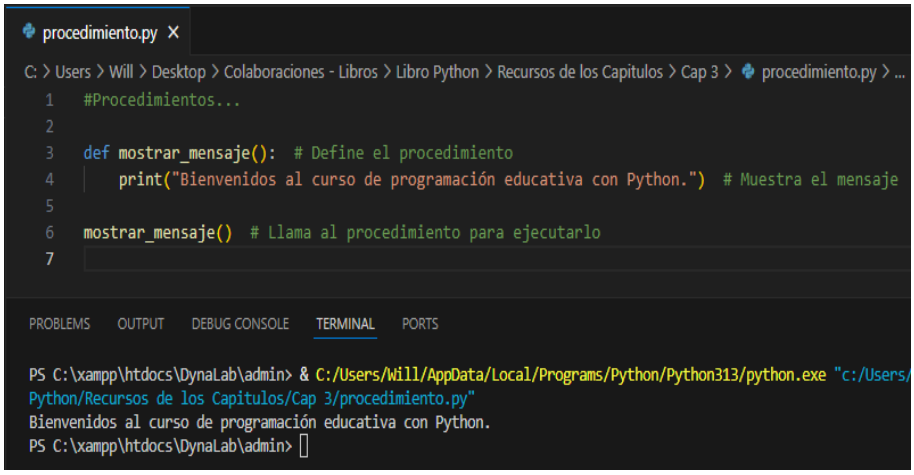


Figura 38. Procedimientos en Python.

Nota. Un procedimiento ejecuta una acción sin devolver valores; debe llamarse explícitamente para mostrarse en pantalla.

Estas estructuras son ideales en educación para mostrar mensajes, procesar datos o realizar tareas repetitivas dentro de los proyectos escolares.

Según Tinoco et al. (2023), enseñar a diferenciar funciones y procedimientos "permite comprender la estructura jerárquica de los programas, fortaleciendo la claridad y legibilidad del código en el aula" (p. 31).

4.3.4 Dificultades comunes y soluciones en procedimientos y funciones

Tabla 5. Dificultades comunes y soluciones en procedimientos y funciones.

Problema	Posible causa	Solución
La función no muestra nada	Falta la instrucción print() o return	Verifica que la función devuelva o muestre resultados
Error de ejecución en la función	Mala indentación	Revisa la correcta indentación del código
La función no recibe datos correctamente	Parámetros mal definidos	Verifica los argumentos al llamar la función
Se obtiene un resultado incorrecto	Error en la lógica interna	Revisa las operaciones dentro de la función

Nota. La tabla resume errores frecuentes al programar en Python y sus soluciones, ayudando a identificar y corregir fallos en funciones y procedimientos durante el aprendizaje.

4.4 Actividades didácticas con Python

Crea un programa que pida un número y muestre su tabla de multiplicar del 1 al 10 utilizando un ciclo for.

Ejemplo:

Pide al usuario un número y lo convierte a entero

```
numero = int(input("Ingresa un número: "))

# Repite el ciclo desde 1 hasta 10

for i in range(1, 11):

# Muestra la tabla de multiplicar del número ingresado

    print(numero, "x", i, "=", numero * i)
```

Orientación docente: Esta actividad puede emplearse en clases de Matemáticas o Computación para reforzar el aprendizaje de las tablas y el razonamiento lógico mediante la programación.

Además, esta actividad promueve el pensamiento computacional al requerir el control de secuencias, la automatización de cálculos y la aplicación de un algoritmo repetitivo con un propósito claro.

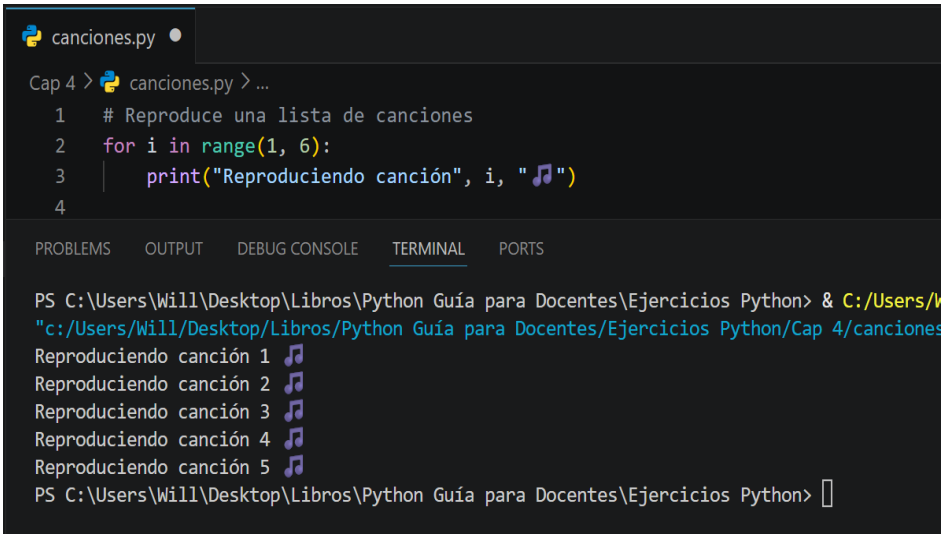


Figura 39. Playlist automática.

Nota. Este ejercicio permite comprender cómo el ciclo for repite una acción un número definido de veces, simulando la reproducción automática de canciones.

Orientación docente: Puede relacionarse con aplicaciones reales como reproductores musicales, ayudando a entender la automatización de procesos repetitivos.



Figura 40. Empaquetado de productos.

Nota. Refuerza el uso de ciclos para representar procesos repetitivos en contextos reales. Permite comprender cómo una acción se repite automáticamente.

Orientación docente: Puede vincularse con procesos logísticos o industriales. Se recomienda variar las repeticiones para observar cambios en el resultado.



Figura 41. Rutina de ejercicios.

Nota. Permite entender cómo un ciclo representa repeticiones controladas dentro de una actividad. Se evidencia el uso de contadores para organizar acciones.

Orientación docente: Puede relacionarse con actividades físicas para hacer el aprendizaje más dinámico. Además, fortalece la idea de repetición estructurada en programación.

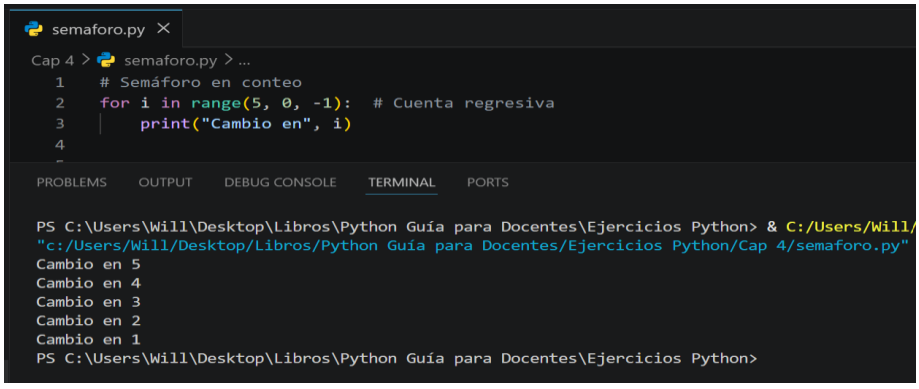


Figura 42. Semáforo en conteo.

Nota. Introduce el uso de ciclos con decremento, permitiendo trabajar secuencias inversas. Refuerza el control del flujo en programas.

Orientación docente: Puede relacionarse con sistemas reales como semáforos o temporizadores. Ayuda a comprender cómo controlar el tiempo y las secuencias en programación.

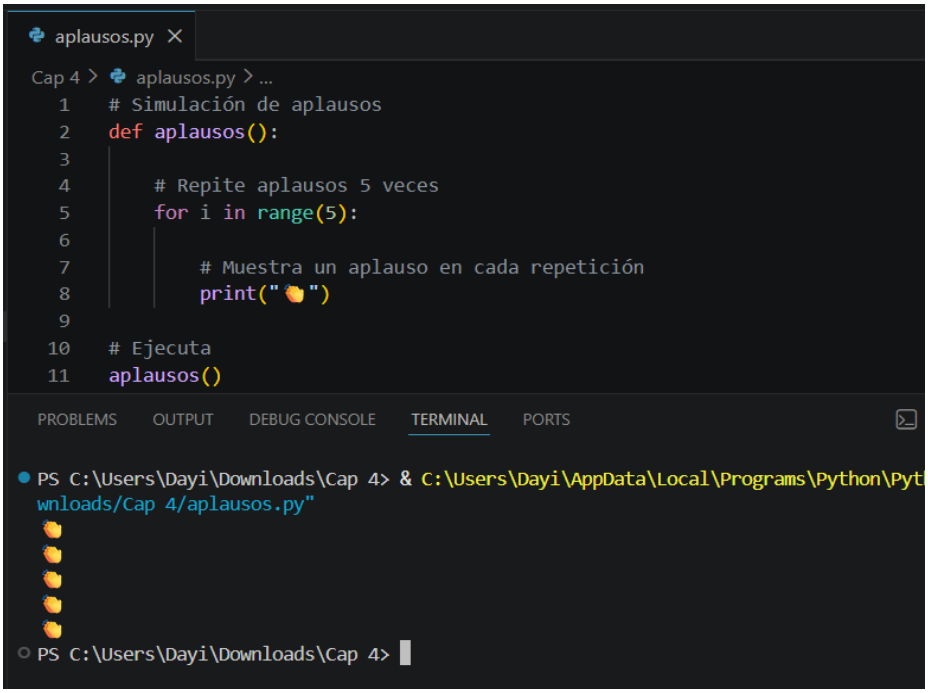


Figura 43. Simulación de aplausos.

Nota. Este ejercicio permite trabajar ciclos mediante la repetición de una acción simple, reforzando la automatización de tareas.

Orientación docente: Puede usarse para representar sonidos o eventos. Se recomienda variar las repeticiones para observar los cambios y hacer el aprendizaje más dinámico.

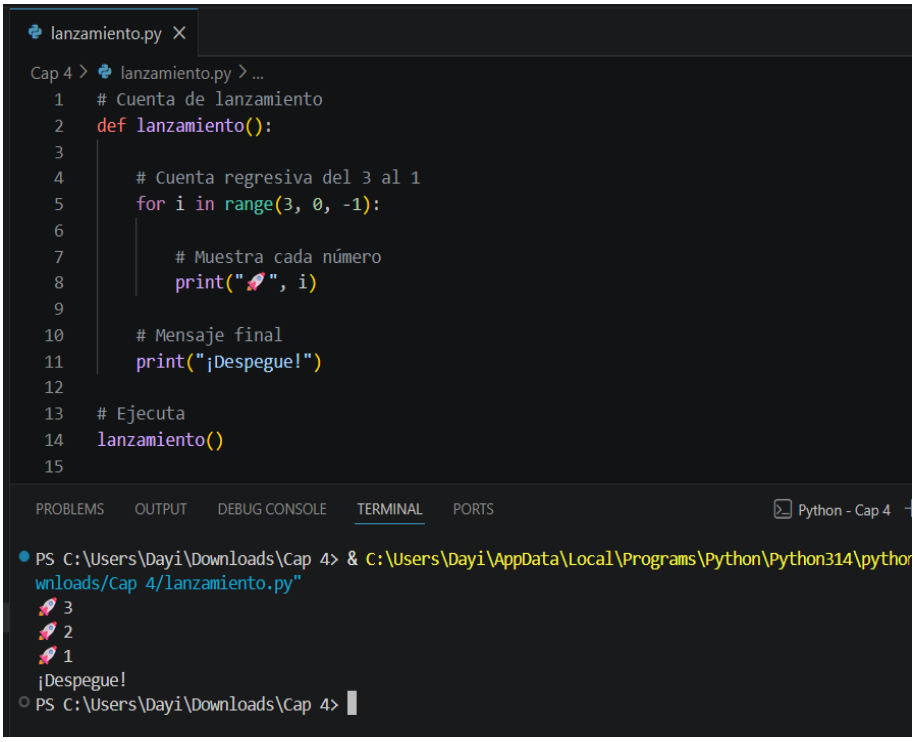


Figura 44. Cuenta de lanzamiento.

Nota. Permite trabajar ciclos en orden descendente, mostrando cómo un programa puede contar hacia atrás paso a paso. Refuerza el control del flujo del programa y ayuda a comprender cómo se modifica el valor en cada repetición.

Orientación docente: Ideal para explicar ciclos inversos de forma sencilla. Hace el aprendizaje más dinámico y visual, y se puede pedir al estudiante cambiar el número inicial para observar cómo varía la cuenta regresiva.

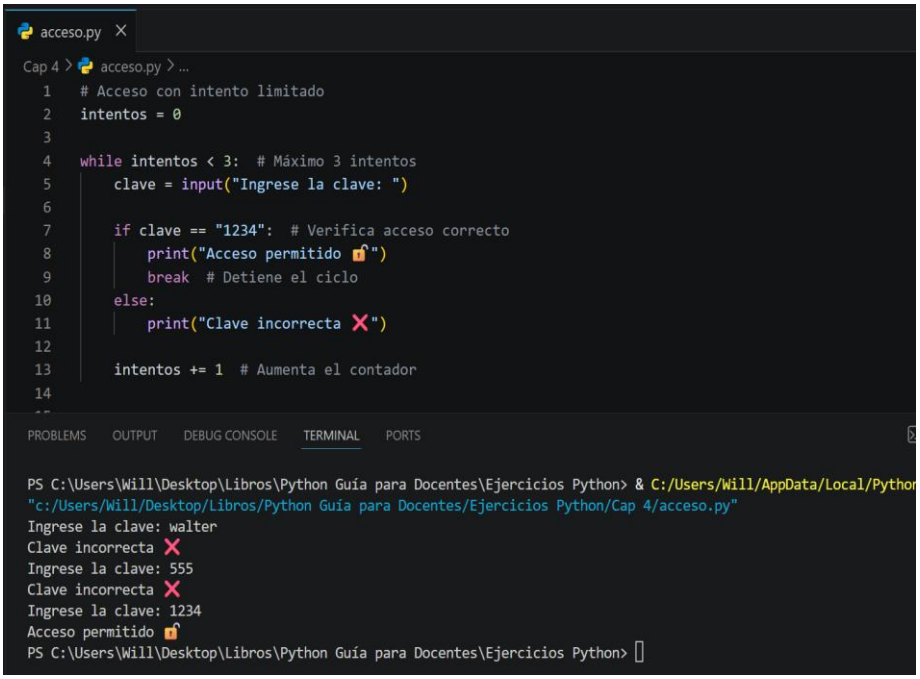


Figura 45. While - Acceso con intento limitado.

Nota. Combina repetición y condicionales para controlar intentos de acceso, permitiendo limitar acciones y validar datos.

Orientación docente: Se relaciona con sistemas de seguridad como contraseñas. Refuerza el uso de contadores y se puede modificar intentos o clave para practicar.

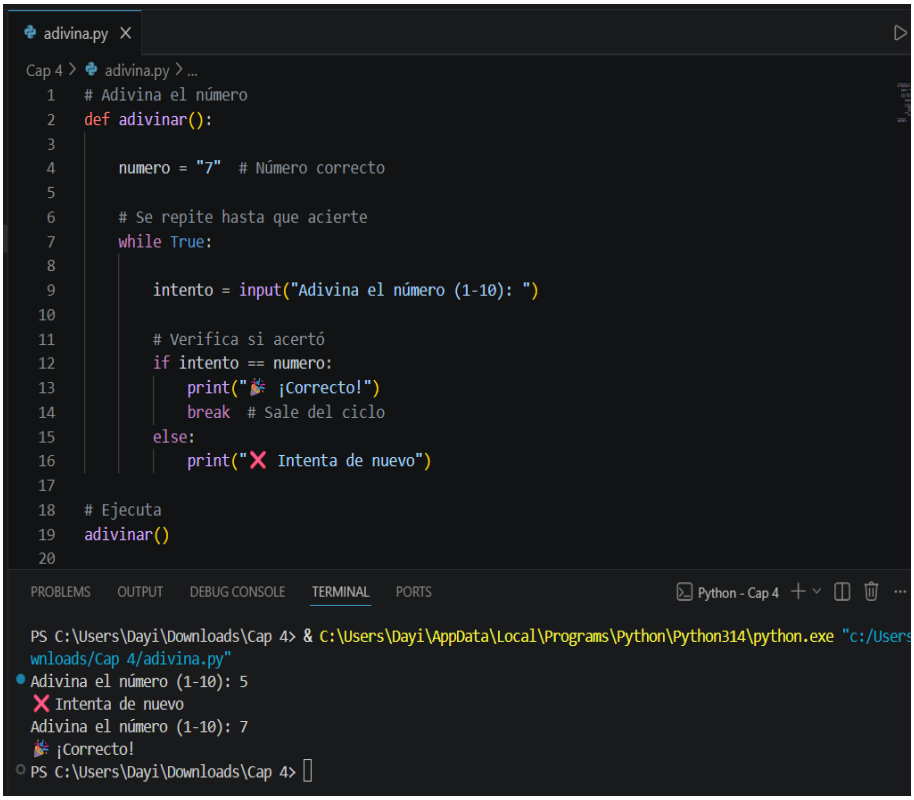


Figura 46. Adivina el número.

Nota. Este ejercicio permite trabajar ciclos infinitos controlados con break. Refuerza la repetición hasta cumplir una condición.

Orientación docente: Se puede cambiar el número. Ayuda a comprender cómo detener un ciclo.

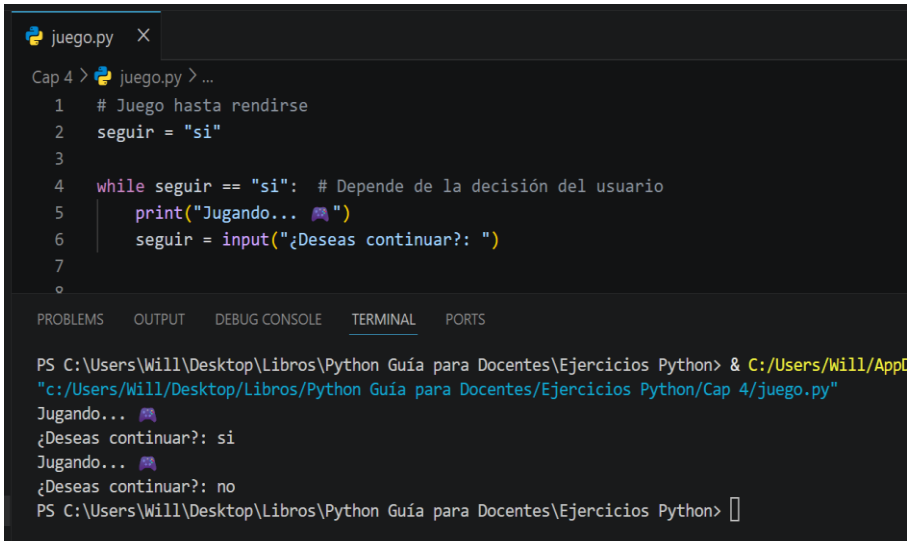


Figura 47. Juego hasta rendirse.

Nota. Permite entender ciclos que dependen de decisiones del usuario. Se evidencia el control dinámico del flujo del programa.

Orientación docente: Favorece la interacción usuario-programa. Ayuda a comprender ciclos abiertos con final condicionado.



Figura 48. Costo total.

Nota. Permite aplicar funciones en situaciones reales de cálculo, reforzando el uso de variables y operaciones. Además, ayuda a organizar el código y reutilizar procesos de forma más clara.

Orientación docente: Se puede vincular con ejemplos como compras o pequeños negocios. Facilita la comprensión del uso práctico de funciones y se puede pedir al estudiante modificar los datos para observar cambios en los resultados.

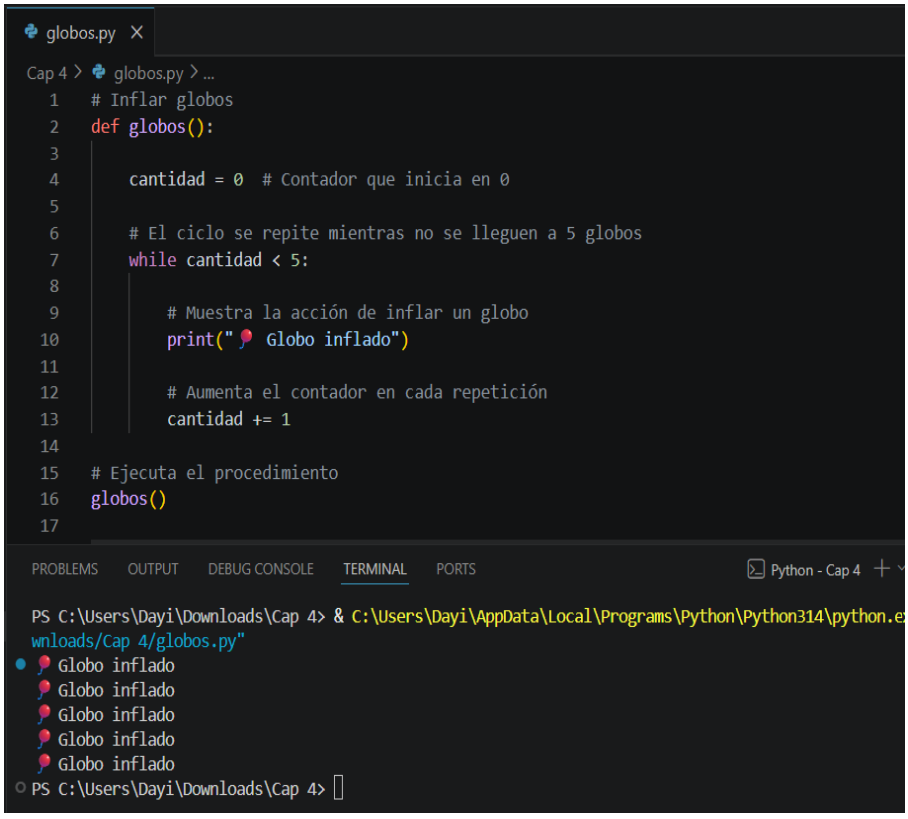


Figura 49. Inflar globos.

Nota. Este ejercicio permite comprender cómo funciona un ciclo while controlado por un contador. El programa repite una acción hasta que se

cumple una condición específica. Es una forma sencilla de visualizar la repetición automática en programación.

Orientación docente: Se recomienda explicar que el ciclo depende de una condición (cantidad < 5) y que el contador es clave para evitar que el ciclo sea infinito. También se puede pedir al estudiante cambiar el número de globos para observar cómo cambia el comportamiento del programa.

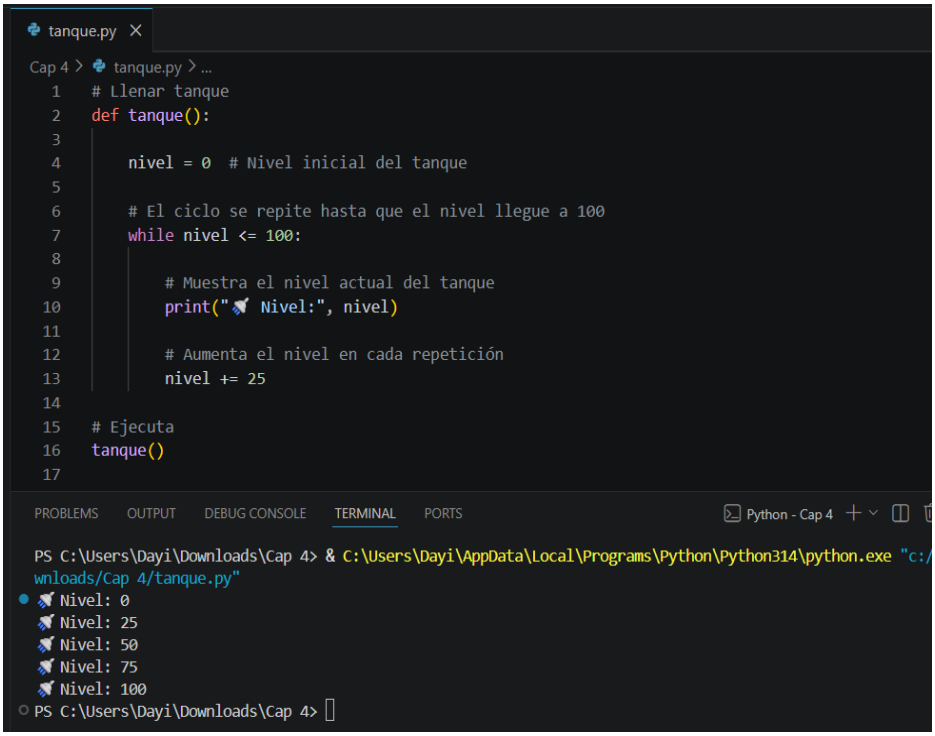


Figura 50. Llenar tanque.

Nota. Permite trabajar ciclos con incrementos definidos. El estudiante comprende cómo una variable puede cambiar progresivamente dentro de un ciclo.

Orientación docente: Se recomienda explicar que el incremento controla cuántas veces se repite el ciclo. También se puede pedir cambiar el valor de aumento para observar diferencias.

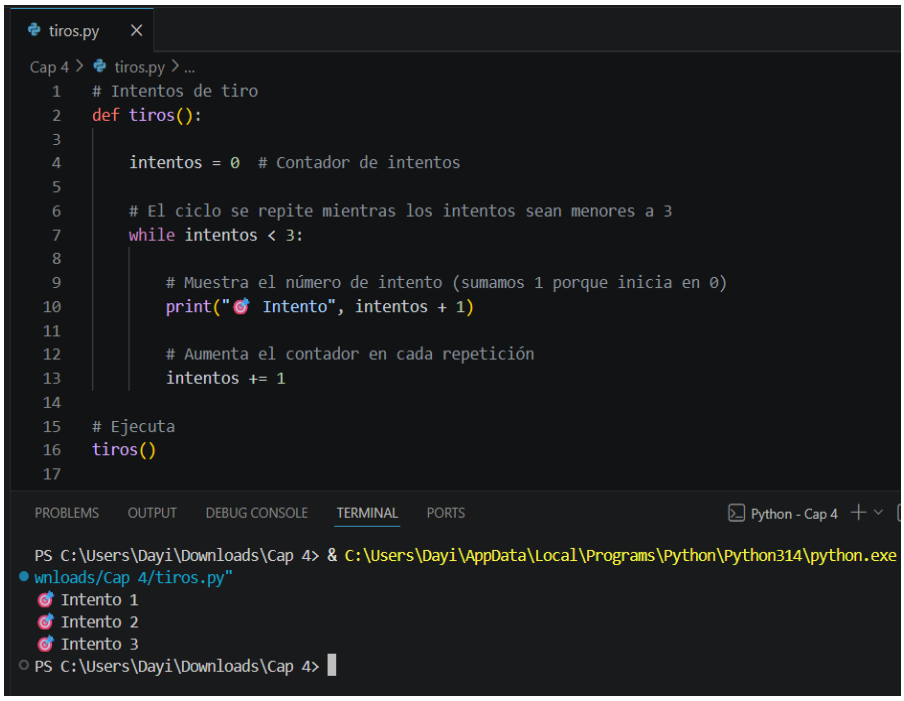


Figura 51. Intentos de tiro.

Nota. Este ejercicio permite comprender cómo se controla el número de repeticiones mediante un contador. Refuerza la idea de ciclos con límite definido.

Orientación docente: Se puede explicar la diferencia entre el valor real del contador y lo que se muestra al usuario (i + 1). También es útil pedir a los estudiantes modificar el número de intentos.



Figura 52. Consumo de agua.

Nota. Este ejercicio permite trabajar funciones y estructuras condicionales para evaluar el consumo de agua en litros, incorporando

validación de datos para evitar valores irreales. Además, muestra cómo un programa puede clasificar información por rangos y generar respuestas adecuadas según cada caso, fortaleciendo la lógica de decisión.

Orientación docente: Puede vincularse con hábitos saludables, permitiendo analizar datos reales y reflexionar sobre el bienestar. También se puede pedir al estudiante probar distintos valores y ajustar los rangos, para observar cambios y reforzar la toma de decisiones y el uso de funciones.

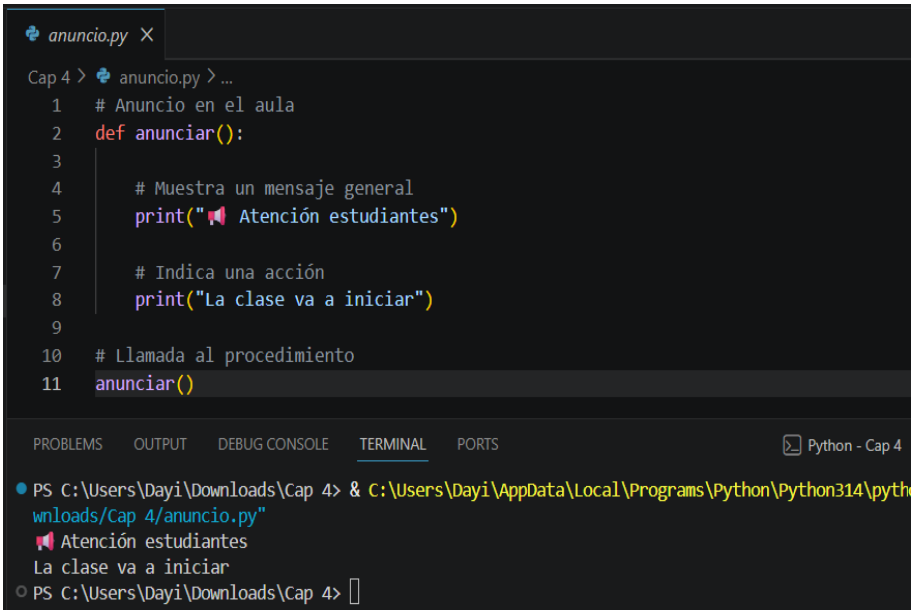


Figura 53. Anuncio en el aula.

Nota. Este ejercicio introduce el uso de procedimientos como bloques que ejecutan acciones sin devolver valores. Permite organizar instrucciones simples dentro de una función.

Orientación docente: Ideal para iniciar el tema de procedimientos. Ayuda a que el estudiante entienda la estructura básica sin complejidad.

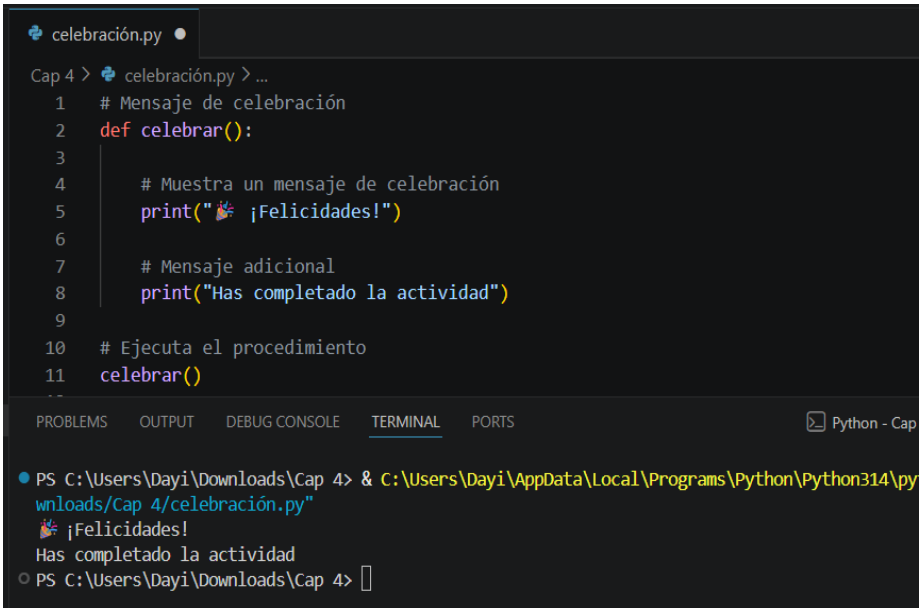


Figura 54. Mensaje de celebración.

Nota. Este ejercicio permite trabajar procedimientos como bloques de código que ejecutan acciones sin devolver valores. Refuerza la idea de agrupar instrucciones relacionadas en una sola estructura.

Orientación docente: Se recomienda explicar que los procedimientos ayudan a organizar el código y evitar repeticiones. El estudiante puede modificar los mensajes para personalizar el ejercicio.

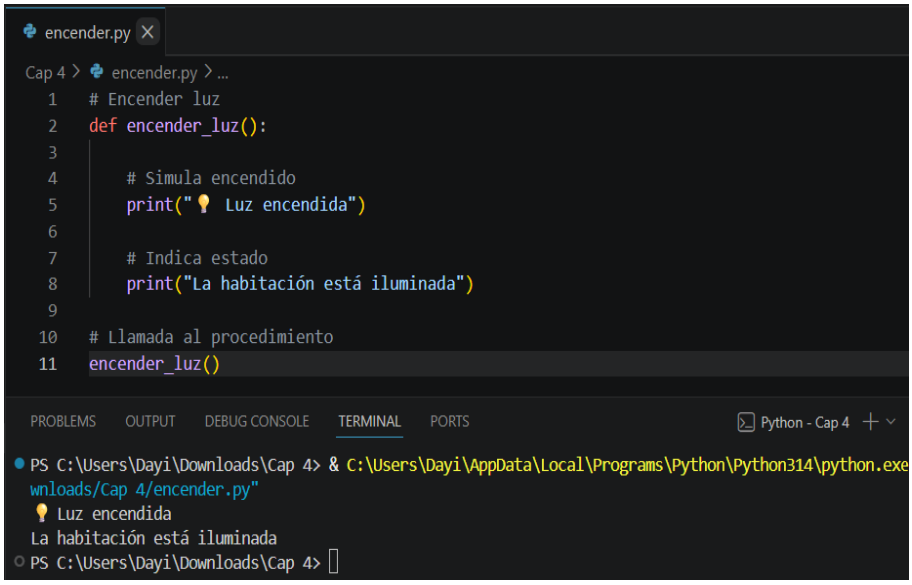


Figura 55. Encender luz.

Nota. Permite representar acciones cotidianas mediante procedimientos, mostrando cómo un programa organiza y reutiliza instrucciones.

Orientación docente: Relaciona la programación con la vida diaria. Se puede pedir crear ejemplos propios para reforzar el aprendizaje.



Figura 56. Recordatorio de tarea.

Nota. Permite generar mensajes automáticos con procedimientos, reforzando la organización del código y su función como recordatorio.

Orientación docente: Puede emplearse para integrar programación con hábitos de estudio. Favorece la comprensión de cómo la tecnología puede apoyar la organización personal.

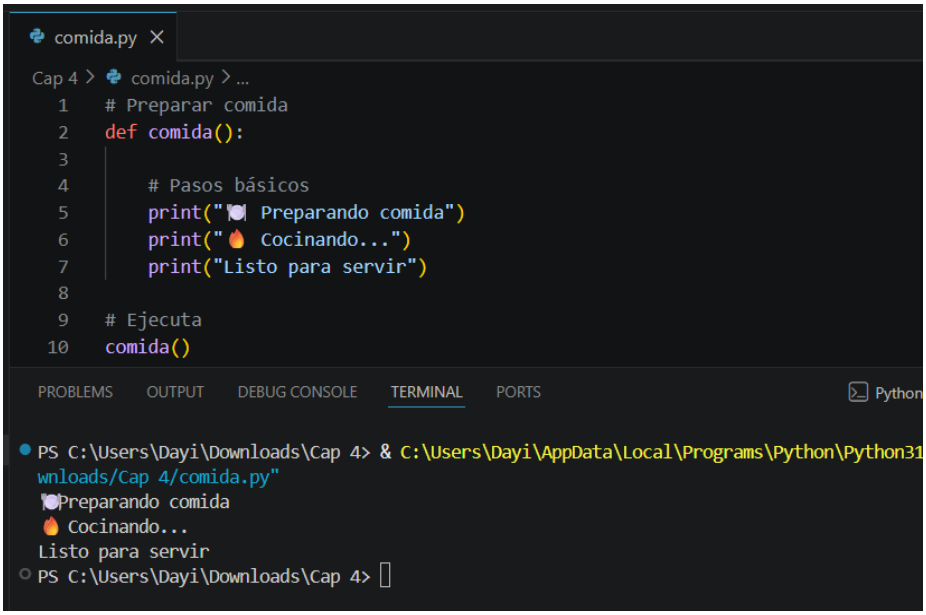


Figura 57. Preparar comida.

Nota. Permite representar procesos cotidianos mediante procedimientos, reforzando la secuencia lógica y la organización de tareas en pasos. Además, ayuda a comprender cómo estructurar acciones de forma clara, ordenada y reutilizable dentro de un programa.

Orientación docente: Se relaciona con rutinas diarias, como actividades personales o escolares. Se puede pedir crear ejemplos propios y

modificar los pasos para observar cambios, reforzando la comprensión y haciendo el aprendizaje más significativo.

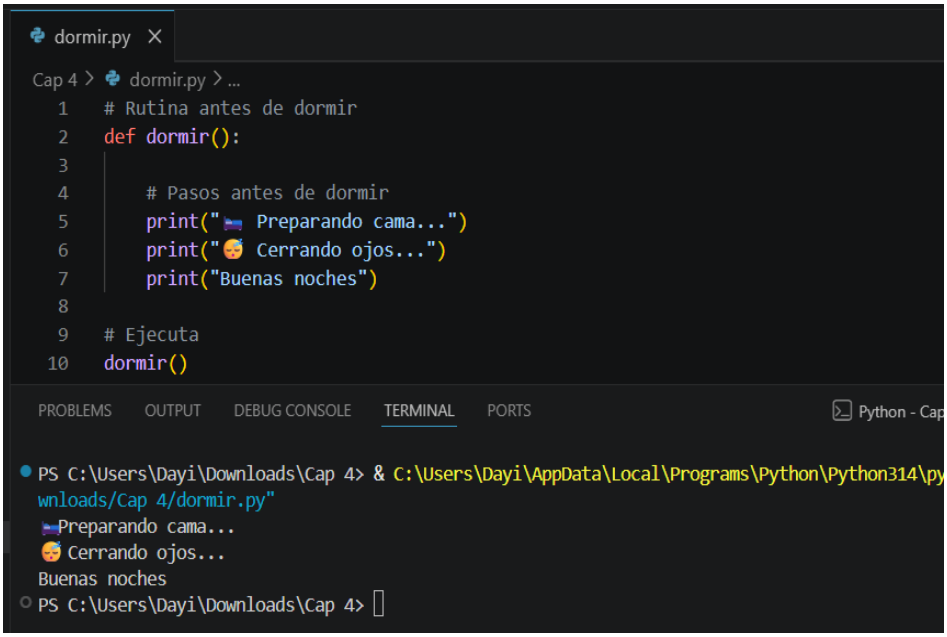


Figura 58. Rutina antes de dormir.

Nota. Permite representar rutinas con procedimientos, reforzando la secuencia lógica y la organización de pasos. Además, ayuda a comprender cómo dividir una tarea en partes más simples y reutilizar instrucciones dentro del programa.

Orientación docente: Se relaciona con hábitos diarios, como actividades personales o escolares. Se puede pedir crear ejemplos propios y modificar las rutinas para observar cambios, fortaleciendo la comprensión y el aprendizaje práctico.

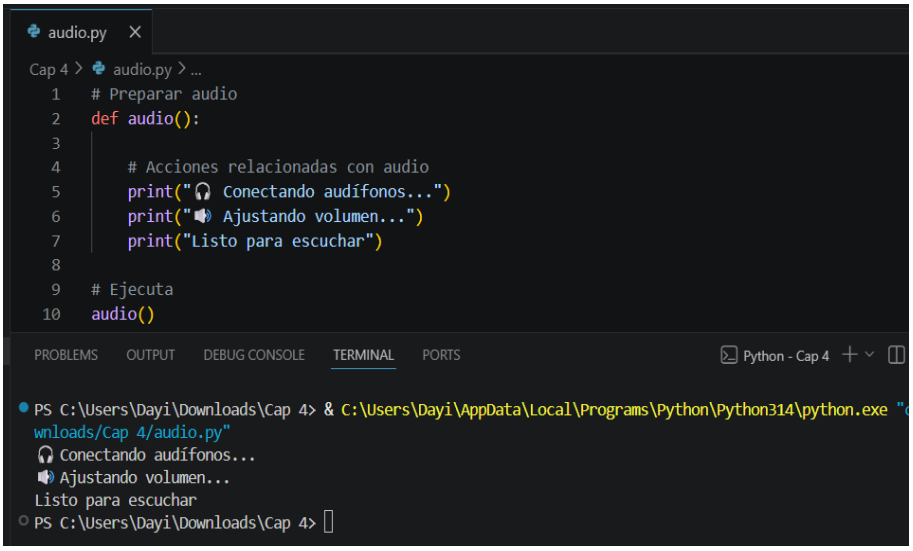


Figura 59. Preparar audio.

Nota. Permite trabajar procedimientos con acciones tecnológicas, reforzando la organización del código. Además, ayuda a comprender cómo agrupar instrucciones para simular el funcionamiento de dispositivos o procesos digitales.

Orientación docente: Puede vincularse con dispositivos reales, como celulares o computadoras. Favorece el aprendizaje contextual y se puede pedir al estudiante proponer ejemplos propios relacionados con la tecnología que usa a diario.

4.5 Mini-proyecto: Contador de asistencia

Este mini-proyecto permite aplicar los ciclos y funciones en una situación educativa.
El objetivo es simular un sistema de asistencia en el aula.

Enunciado del ejercicio:

Diseña un programa en Python que solicite el número de estudiantes presentes y luego muestre un saludo para cada uno.

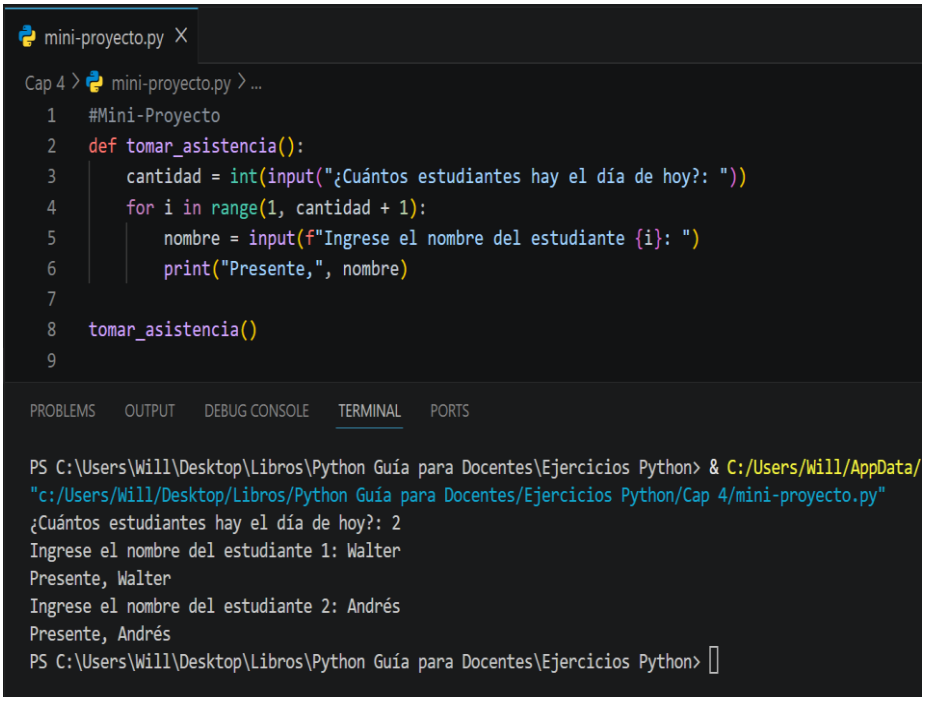


Figura 60. Sistema de registro de asistencia en el aula.

Nota. Este mini-proyecto permite registrar la asistencia de estudiantes utilizando un ciclo y una función. El programa solicita datos, los procesa y muestra resultados, evidenciando cómo se automatizan tareas repetitivas en programación. Este programa aplica un ciclo for y una función personalizada. El docente puede adaptarlo para registrar notas, calificaciones o tareas entregadas.

Desde el pensamiento computacional, este mini-proyecto integra descomposición del problema, uso de secuencias repetitivas y

abstracción mediante funciones, automatizando un proceso habitual del entorno educativo.

4.6 Integración STEAM del mini-proyecto

Science (Ciencia): El programa permite observar y analizar datos reales del aula, como la cantidad de estudiantes presentes. Esto introduce a los alumnos en prácticas científicas básicas de recolección y organización de información, útiles para estudios de comportamiento, asistencia o patrones de participación.

Technology (Tecnología): Python automatiza una tarea cotidiana del entorno escolar: registrar asistencia. El uso de funciones y ciclos muestra cómo la tecnología puede reemplazar procesos manuales, reducir errores y mejorar la eficiencia en actividades administrativas y académicas.

Engineering (Ingeniería): El estudiante aplica pensamiento ingenieril al diseñar un algoritmo que sigue pasos claros: solicitar datos, iterar sobre ellos y generar una salida organizada. La función tomar_asistencia() ejemplifica el ciclo de ingeniería: planificar → construir → probar → corregir.

Arts (Arte): Los mensajes personalizados (“Presente, nombre”) pueden adaptarse visualmente, permitiendo a los estudiantes experimentar con estilos de texto, interacciones creativas o interfaces más expresivas. Se fomenta la comunicación digital estética y clara.

Mathematics (Matemáticas): El uso del ciclo for involucra conteo, secuencias numéricas y control de rangos (1 hasta cantidad + 1). El

estudiante practica razonamiento matemático al comprender cómo la iteración recorre valores consecutivos y cómo se relacionan número de estudiantes y número de repeticiones.

4.7 Recursos digitales complementarios del capítulo

Para reforzar los contenidos de este capítulo, se han elaborado archivos digitales que contienen los ejemplos prácticos desarrollados en clase, incluyendo programas con ciclos, funciones, procedimientos y mini-proyectos educativos. Estos materiales permiten a los docentes y estudiantes experimentar directamente con el código, comprender su funcionamiento y adaptar los ejemplos a sus propios contextos de aprendizaje.

Los archivos se encuentran disponibles en una carpeta de Google Drive, donde pueden descargarse fácilmente para su uso, modificación y reutilización en futuras actividades, el acceso a estos materiales se facilita mediante un código QR, garantizando una forma rápida, práctica y segura de obtener los recursos complementarios del capítulo.



Figura 61. Código QR para descarga de archivos complementarios del Capítulo 4.

Nota. El código QR permite acceder directamente a los archivos complementarios del capítulo, que incluyen los ejemplos de ciclos, funciones, procedimientos y el mini-proyecto de asistencia en Python.

4.8 Resumen del capítulo

En este capítulo se abordaron las estructuras de control que permiten automatizar procesos y organizar de mejor manera los programas en Python. Se estudiaron los ciclos, específicamente for y while, los cuales permiten repetir instrucciones de forma controlada. Estos elementos son fundamentales para evitar la repetición manual de código y mejorar la eficiencia en la programación.

Asimismo, se analizó el funcionamiento de los ciclos en distintos contextos, destacando la importancia de las condiciones de control y la correcta actualización de variables para evitar errores como ciclos infinitos. Se enfatizó que el uso adecuado de estas estructuras contribuye al desarrollo del pensamiento lógico y a la resolución de problemas de forma sistemática.

Otro aspecto importante del capítulo fue la introducción de funciones y procedimientos, los cuales permiten organizar el código en bloques reutilizables. Las funciones facilitan la modularidad del programa, mejoran su legibilidad y permiten reutilizar soluciones sin necesidad de reescribir código, lo cual es una práctica fundamental en el desarrollo de software.

Finalmente, mediante actividades prácticas y ejercicios aplicados, se integraron los conceptos de ciclos y funciones en la resolución de

problemas. Desde el enfoque del pensamiento computacional, estas herramientas fortalecen la descomposición de problemas, la automatización de procesos y la construcción de soluciones más eficientes y organizadas.

4.9 Autoevaluación del capítulo

Pregunta 1: ¿Qué diferencia existe entre un ciclo for y un ciclo while?

Pregunta 2: ¿Qué palabra se utiliza para definir una función en Python?

Pregunta 3: ¿Cuál es la finalidad de la instrucción return?

Pregunta 4: ¿Qué sucede si en un ciclo while no se actualiza la variable de control?

Pregunta 5: Diseña una función que calcule el área de un rectángulo a partir de su base y altura.

Pregunta 6: ¿Qué permite un ciclo en programación?

a) Guardar datos

b) Repetir instrucciones

c) Crear variables

d) Eliminar errores

Pregunta 7: ¿Cuál es la principal diferencia entre for y while?

a) No hay diferencia

b) for repite un número definido de veces, while depende de una condición

c) while es más rápido

d) for no usa variables

Pregunta 8: ¿Qué ocurre si no se actualiza la variable en un ciclo while?

a) El programa termina

b) Se genera un error inmediato

c) Puede generarse un ciclo infinito

d) No ocurre nada

Pregunta 9: ¿Cuál es el resultado de este código?

```
for i in range(3):
    print(i)
```

a) 1 2 3

b) 0 1 2

c) 0 1 2 3

d) 3 2 1

Pregunta 10: ¿Qué hace la función range(5)?

a) Imprime números

b) Genera una secuencia de 0 a 4

c) Genera números del 1 al 5

d) Crea una lista vacía

CAPÍTULO V

5 LISTAS, DICCIONARIOS Y MANEJO DE ARCHIVOS EN PYTHON

Objetivo del capítulo

Al finalizar este capítulo, los lectores podrán aplicar las listas, los diccionarios y el manejo de archivos en Python para guardar, organizar y recuperar información en contextos educativos. Además, se busca que los lectores desarrollen la capacidad de estructurar datos, identificar patrones y procesar información de manera sistemática, fortaleciendo el pensamiento computacional aplicado al análisis y gestión de datos.

Aprendizajes esperados

- Comprender el concepto y la utilidad de las listas en Python.
- Aprender a crear y modificar diccionarios para guardar información.
- Identificar cómo leer y escribir datos en archivos de texto o CSV.
- Desarrollar proyectos simples que almacenen y procesen información educativa.
- Reconocer la importancia del almacenamiento de datos para la programación práctica.

5.1 Introducción: Cómo guardar información en nuestros programas

Hasta ahora hemos aprendido a repetir acciones y a crear funciones que realizan tareas específicas. Sin embargo, ¿qué pasa si queremos guardar varios datos al mismo tiempo? Python nos permite hacerlo fácilmente gracias a las listas**,** los diccionarios y los archivos.

Con ellas, podemos guardar y organizar información de una forma parecida a cómo lo hacemos en la vida real, una lista del supermercado, una libreta con datos o una hoja donde registramos nombres y calificaciones.

De acuerdo con Schnieder y Williams (2023), “la automatización de tareas computacionales y el procesamiento de datos con Python favorecen la creación de recursos educativos interactivos para docentes e investigadores” (p. 4).

Por su parte, Rivera et al. (2023) afirman que “las herramientas digitales basadas en Python facilitan el aprendizaje práctico y promueven la organización de la información en entornos educativos” (p. 66).

Estas herramientas no solo permiten almacenar información, sino que también desarrollan la capacidad de organizar datos, reconocer patrones y estructurar soluciones, habilidades fundamentales del pensamiento computacional.

5.2 Listas en Python

Una lista en Python es como una lista normal, puedes escribir varios elementos dentro de ella, mantenerlos en orden, y agregarlos o quitarlos cuando quieras.

Se crean con corchetes [ ] y los elementos van separados por comas.

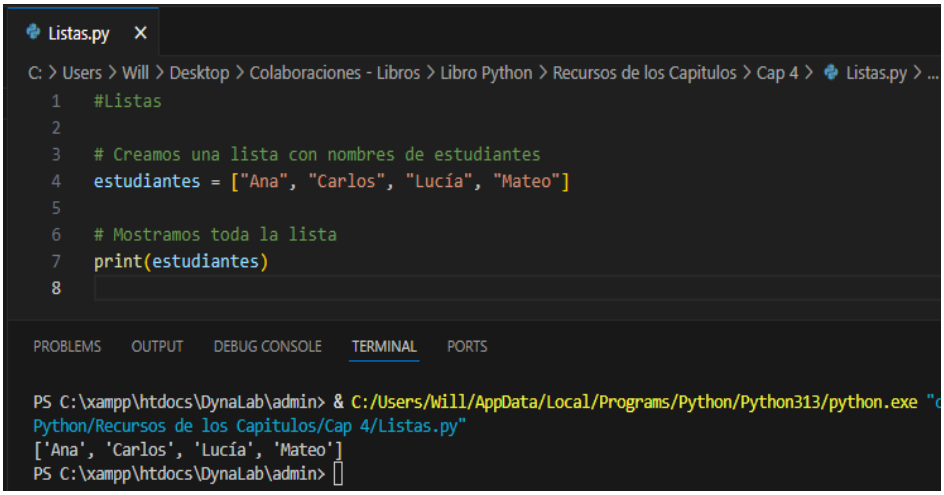


Figura 62. Creación y visualización de una lista en Python.

Nota. El programa define una lista llamada estudiantes con varios nombres y muestra su contenido completo en pantalla, demostrando cómo almacenar y visualizar conjuntos de datos en una sola estructura.

Desde el pensamiento computacional, las listas permiten agrupar información relacionada y recorrerla de manera sistemática, favoreciendo la identificación de patrones y la automatización del procesamiento de datos.

Agregar o eliminar elementos

```
 9   # Agregar un nuevo estudiante
10   #.append = Agregar
11   estudiantes.append("Valeria")
12
13   # Eliminar un estudiante de la lista
14   #.remove = Eliminar
15   estudiantes.remove("Carlos")
```

Figura 63. Modificación de una lista con los métodos append() y remove().

Nota. El método append() permite añadir nuevos elementos al final de la lista, mientras que remove() elimina un elemento específico. Estas acciones facilitan la gestión dinámica de datos dentro de un programa educativo.

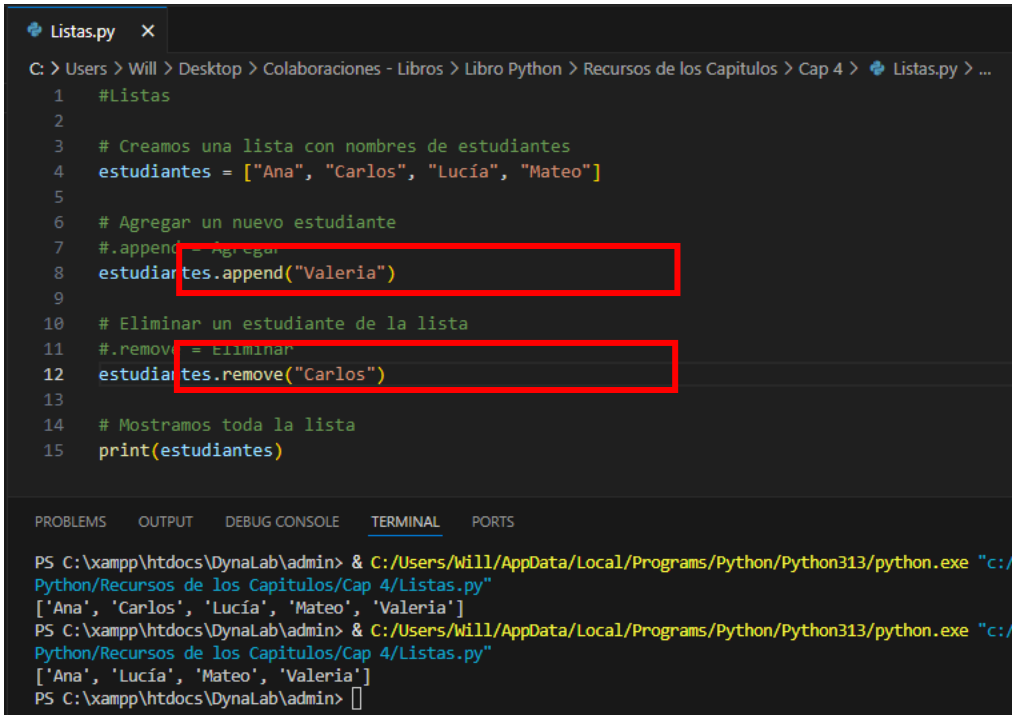


Figura 64. Ejecución del programa de listas en Python.

Nota. El código crea una lista de estudiantes, agrega el nombre Valeria y elimina Carlos, mostrando finalmente la lista actualizada en pantalla.

Recorrer una lista



Figura 65. Recorrido de una lista con ciclo for en Python.

Nota. El ejemplo muestra cómo recorrer una lista de estudiantes mediante un ciclo for y visualizar un mensaje personalizado para cada elemento.

Las listas son muy útiles para guardar nombres, materiales, notas o actividades.
Cada elemento ocupa un lugar (posición) y puedes acceder a él fácilmente.

5.2.1 Dificultades comunes y soluciones en el uso de listas

Tabla 6. Dificultades comunes y soluciones en el uso de listas.

Problema	**Posible causa**	**Solución**
Error de índice	Se intenta acceder a una posición inexistente	Verifica con len(lista) el tamaño antes de usar índices
Se obtiene un dato incorrecto	Índice mal utilizado	Revisa la posición correcta dentro de la lista
La lista no cambia	No se modifica correctamente	Usa métodos como append() o remove() según el caso

Nota. Las listas requieren un manejo adecuado de índices y métodos para evitar errores comunes durante su uso.

5.3 Diccionarios en Python

Un diccionario es como una libreta donde cada palabra (clave) tiene un significado (valor).
Por ejemplo, puedes tener una palabra “nombre” que guarda “Ana” o una palabra “nota” que guarda 9.5.

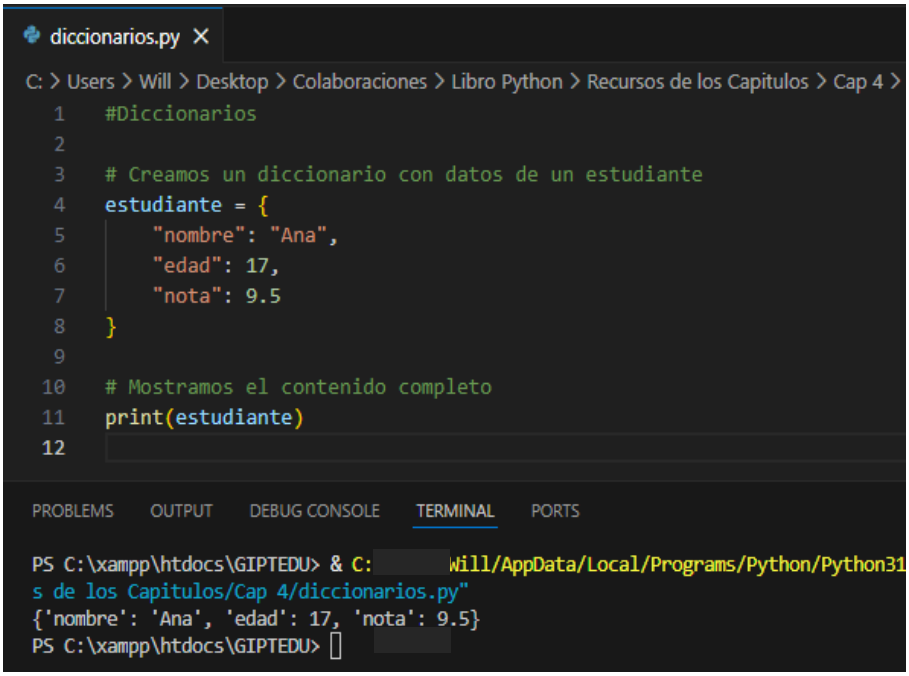


Figura 66. Creación y visualización de un diccionario en Python.

Nota. El ejemplo presenta un diccionario en Python que almacena datos básicos de un estudiante y muestra su contenido mediante la instrucción print().

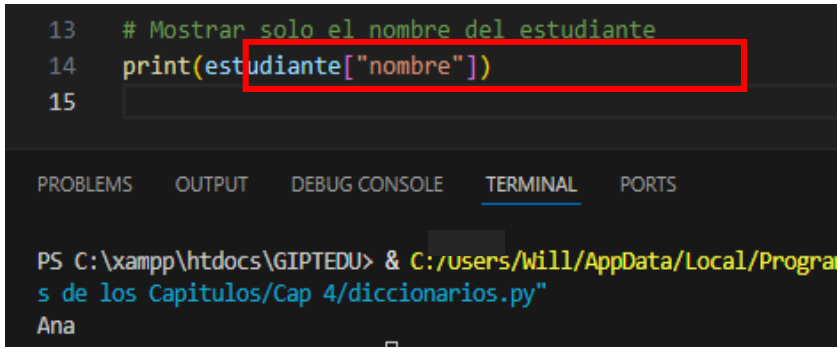


Figura 67. Acceso a un valor específico de un diccionario en Python.

Nota. El ejemplo muestra cómo acceder y visualizar un valor específico de un diccionario utilizando su clave correspondiente.

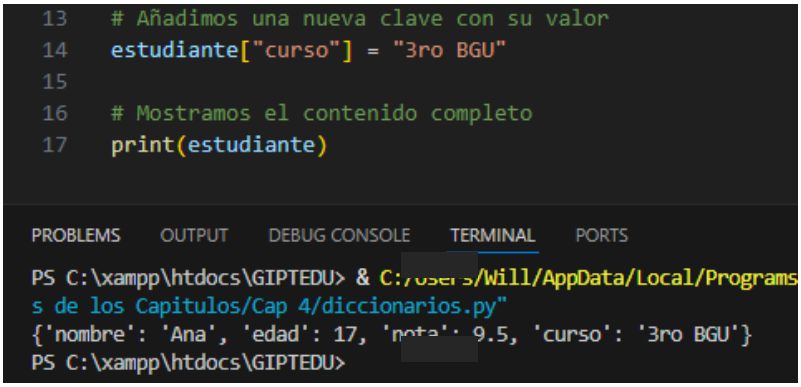


Figura 68. Agregar un nuevo dato al diccionario en Python.

Nota. El ejemplo muestra cómo agregar una nueva clave con su valor a un diccionario y visualizar el contenido actualizado.

El uso de diccionarios fortalece la abstracción al permitir asociar conceptos con información específica mediante claves, facilitando la organización estructurada de datos en contextos educativos.

5.3.1 Dificultades comunes y soluciones en el uso de diccionarios

Tabla 7. Dificultades comunes y soluciones en el uso de diccionarios

Problema	**Posible causa**	**Solución**
Clave no encontrada	La clave no existe en el diccionario	Usa if "clave" in diccionario: para validar antes de acceder
Valor incorrecto	Error al asignar datos	Verifica las claves y valores al momento de definir el diccionario

No se actualiza el valor	Se sobrescribe incorrectamente	Revisa la sintaxis al modificar el diccionario

Nota. Los diccionarios permiten organizar datos mediante claves, por lo que es importante validar su existencia antes de acceder a ellas.

5.4 Manejo de archivos en Python

Hasta ahora los datos desaparecen cuando el programa termina. Pero si queremos guardar esa información para usarla más adelante, necesitamos un archivo.
Un archivo es como una hoja donde escribimos los resultados para poder leerlos otro día.

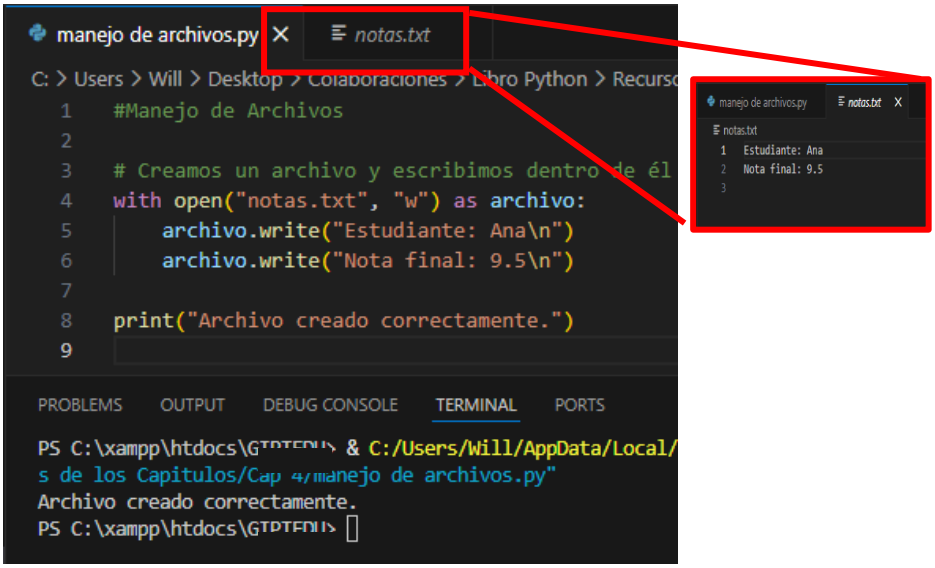


Figura 69. Creación y escritura de un archivo de texto en Python.

Nota. El ejemplo muestra cómo crear un archivo de texto y escribir información en él utilizando la función open().

Explicación: "w" significa modo escritura, crea o reemplaza un archivo, cada línea se escribe con archivo.write() y \n sirve para saltar de línea.

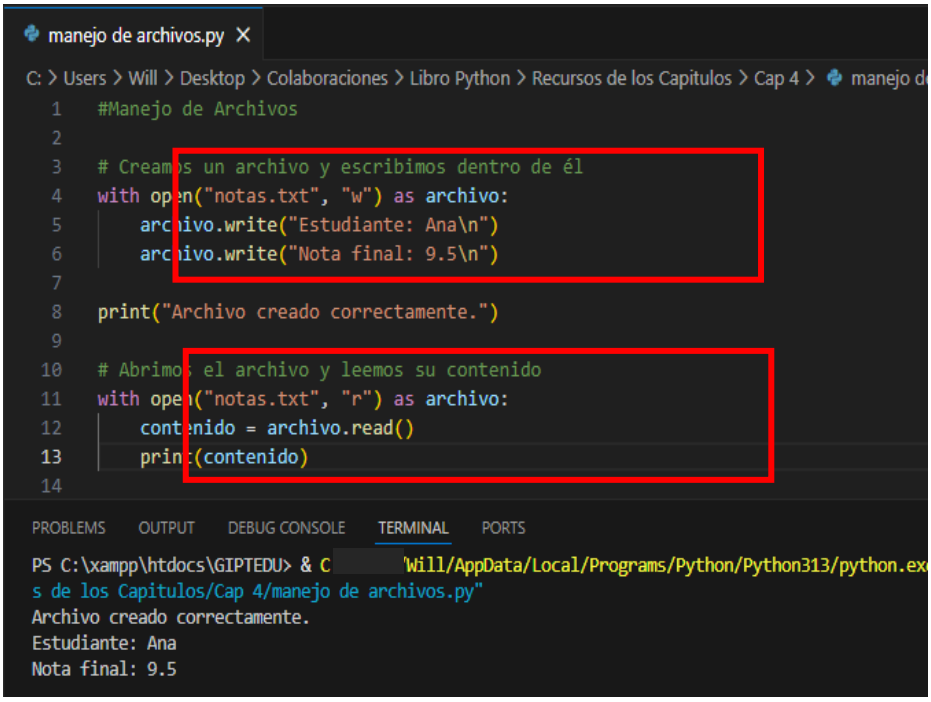


Figura 70. Lectura del contenido de un archivo de texto en Python.

Nota. El ejemplo muestra cómo abrir un archivo de texto y leer su contenido mediante el modo de lectura.

El manejo de archivos amplía el pensamiento computacional al permitir conservar información, procesarla posteriormente y analizar resultados, integrando secuencia lógica y organización de datos en un mismo proceso.

5.4.1 Dificultades comunes y soluciones en el uso de archivos

Tabla 8. Dificultades comunes y soluciones en el uso de archivos

Problema	Posible causa	Solución
El archivo no se crea	Error en la ruta o permisos	Revisa el nombre del archivo y la ubicación del programa
No se guarda la información	No se cierra correctamente el archivo	Usa with open() para asegurar el cierre automático
Error al leer el archivo	Formato o ruta incorrecta	Verifica la ruta y el modo de apertura (r, w, a)

Nota. El manejo de archivos requiere especial atención en rutas, modos de apertura y cierre adecuado para evitar pérdida de información.

5.5 Guardar información en formato CSV

Los archivos CSV son archivos de texto que organizan los datos mediante valores separados por delimitadores. La coma es el separador más utilizado, aunque también pueden emplearse otros delimitadores según la configuración o el entorno. Son útiles para registrar calificaciones, listas o inventarios.

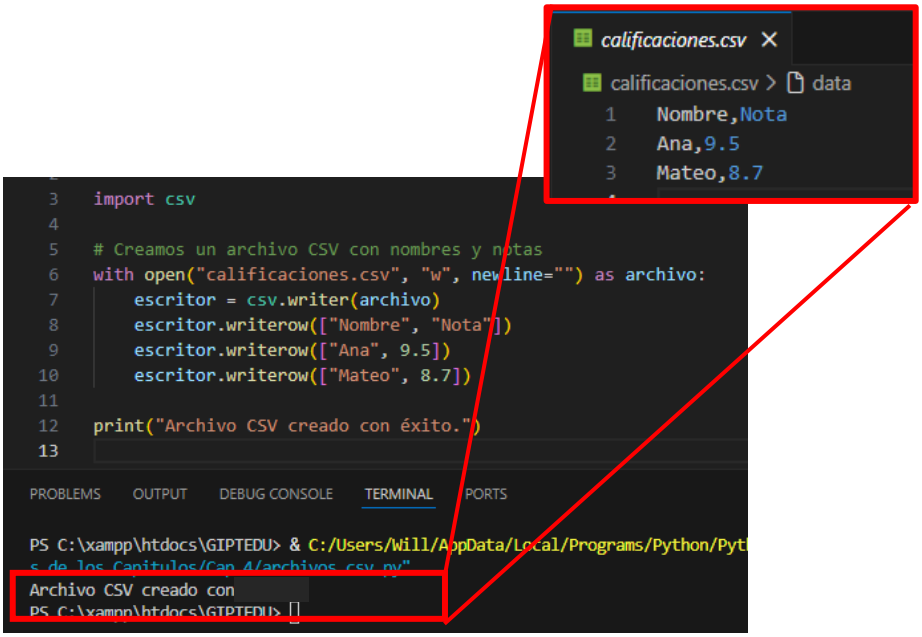


Figura 71. Creación de un archivo CSV con datos en Python.

Nota. El ejemplo muestra cómo crear un archivo CSV y almacenar información organizada en filas y columnas.

Los archivos CSV permiten estructurar información en filas y columnas, facilitando el análisis sistemático de datos y promoviendo habilidades básicas de procesamiento y organización propias del pensamiento computacional.

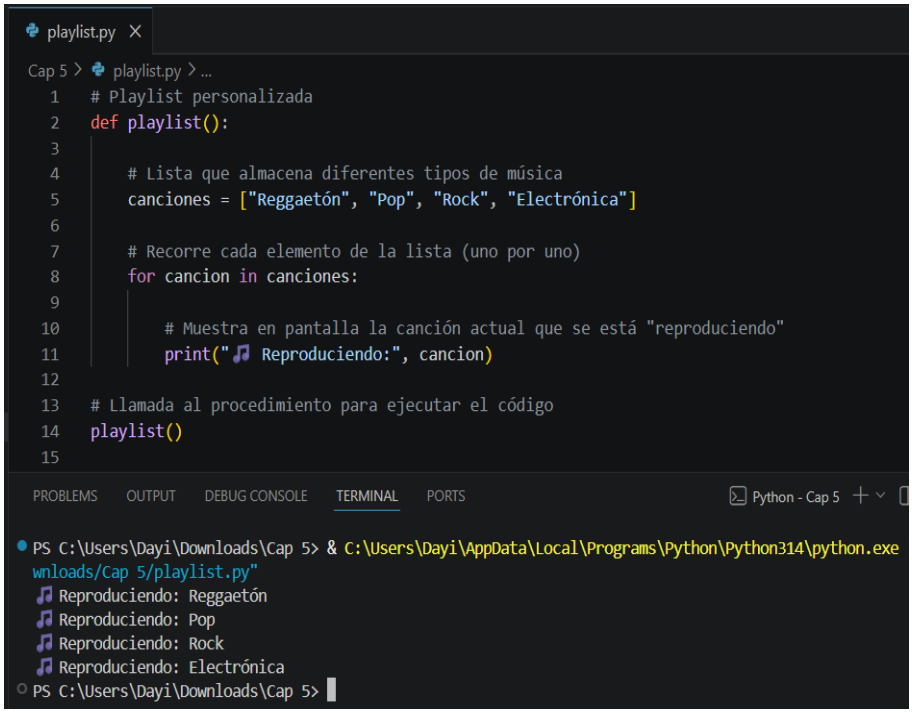


Figura 72. Playlist personalizada dentro de Python.

Nota. Este ejercicio permite usar listas para almacenar varios elementos en una sola variable y recorrerlos de forma ordenada, facilitando organización de datos.

Orientación docente: Puede relacionarse con ejemplos como reproductores de música, estudiantes modifiquen la lista con sus propias canciones para reforzar el aprendizaje.

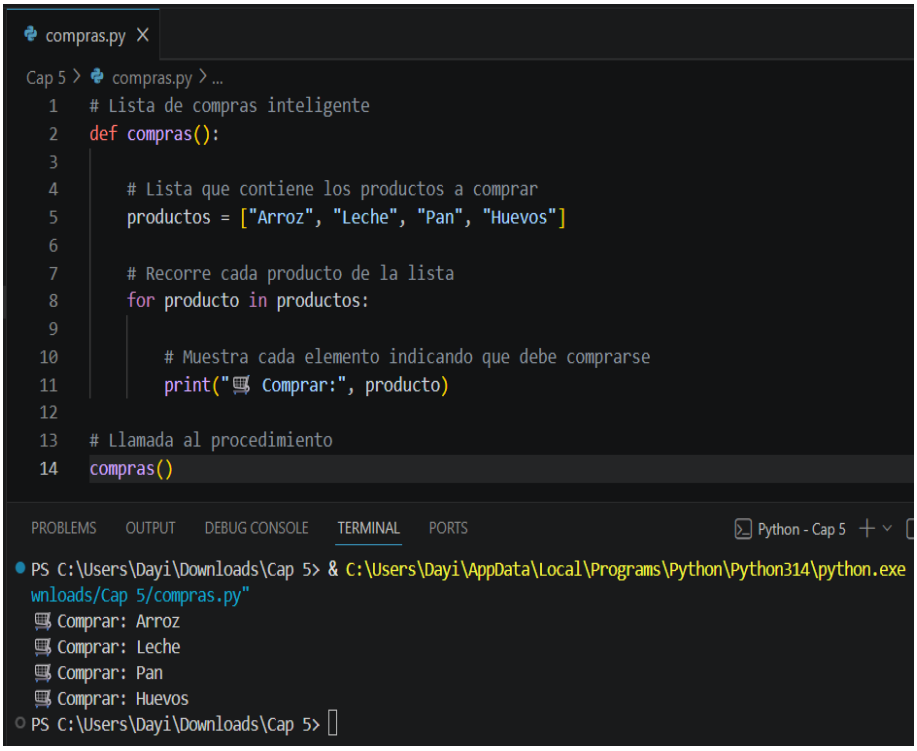


Figura 73. Lista de compras inteligente dentro de Python, ejemplificación.

Nota. Permite usar listas organizando información en un solo conjunto y recorrer cada elemento, reforzando su uso en situaciones cotidianas.

Orientación docente: Puede vincularse con actividades como hacer compras, por tal motivo se sugiere que el estudiante cree su propia lista para fortalecer el aprendizaje.

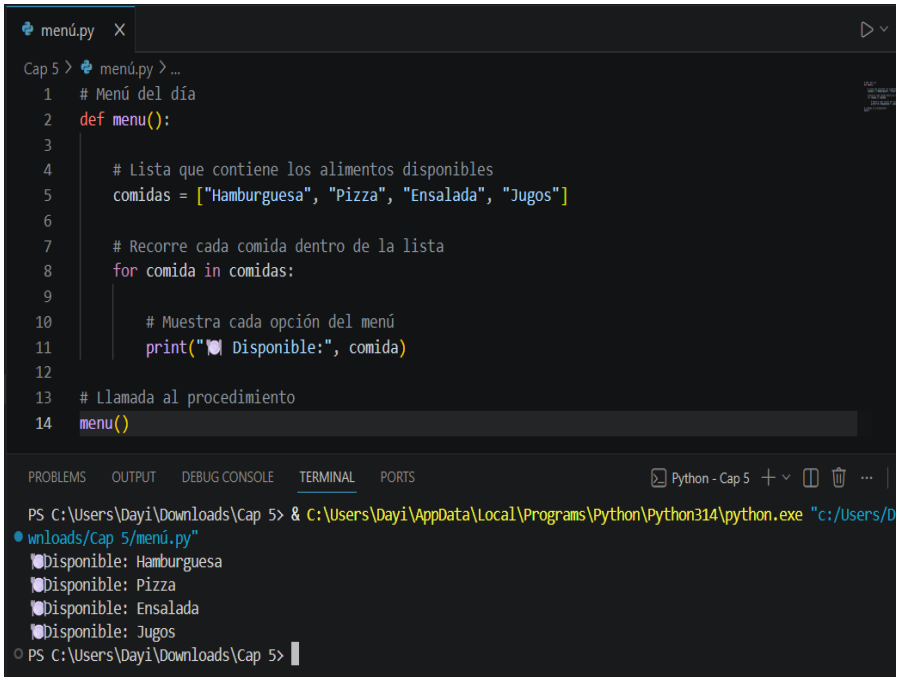


Figura 74. Menú del día depende de la disponibilidad del ingreso.

Nota. Permite trabajar listas como estructuras que almacenan múltiples opciones, estudiante aprende a recorrer cada elemento y a mostrar información de manera ordenada, fortaleciendo el manejo de datos agrupados.

Orientación docente: Puede vincularse con emprendimientos o negocios como restaurantes, asimismo, pedir al estudiante que cree su propio menú, favoreciendo la creatividad y el aprendizaje aplicado.

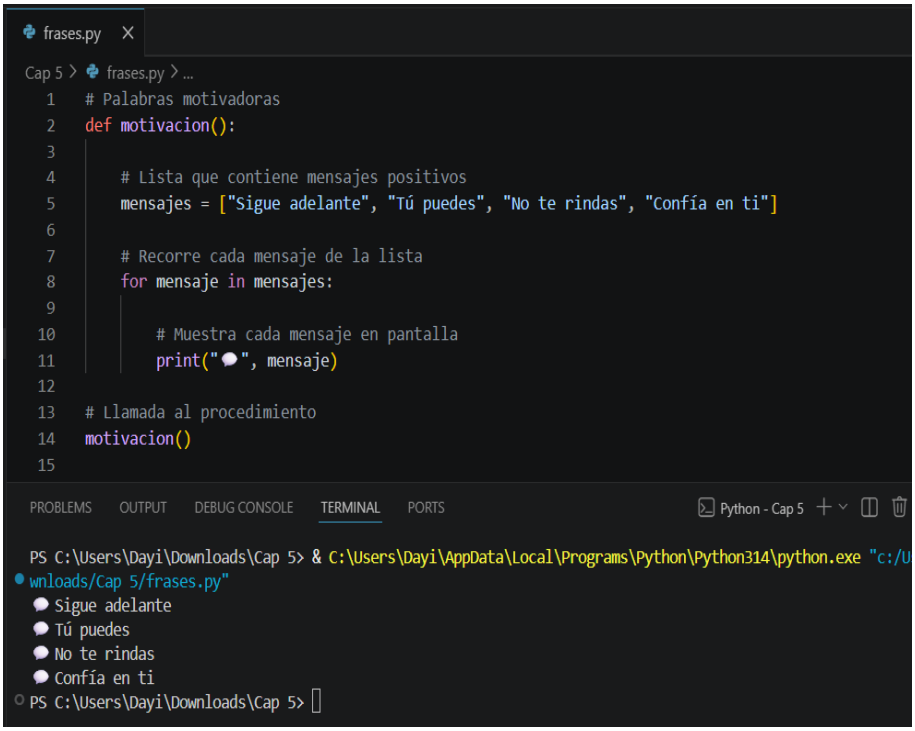


Figura 75. Palabras motivadoras en cada línea con Python.

Nota. Permite trabajar listas de texto como elementos informativos, estudiante comprende cómo almacenar y mostrar mensajes, reforzando el manejo de cadenas dentro de listas.

Orientación docente: Puede integrarse con educación emocional, fortaleciendo el aprendizaje significativo, recomendamos invitar a estudiantes a crear sus propios mensajes, promoviendo la participación y creatividad de los mismos.

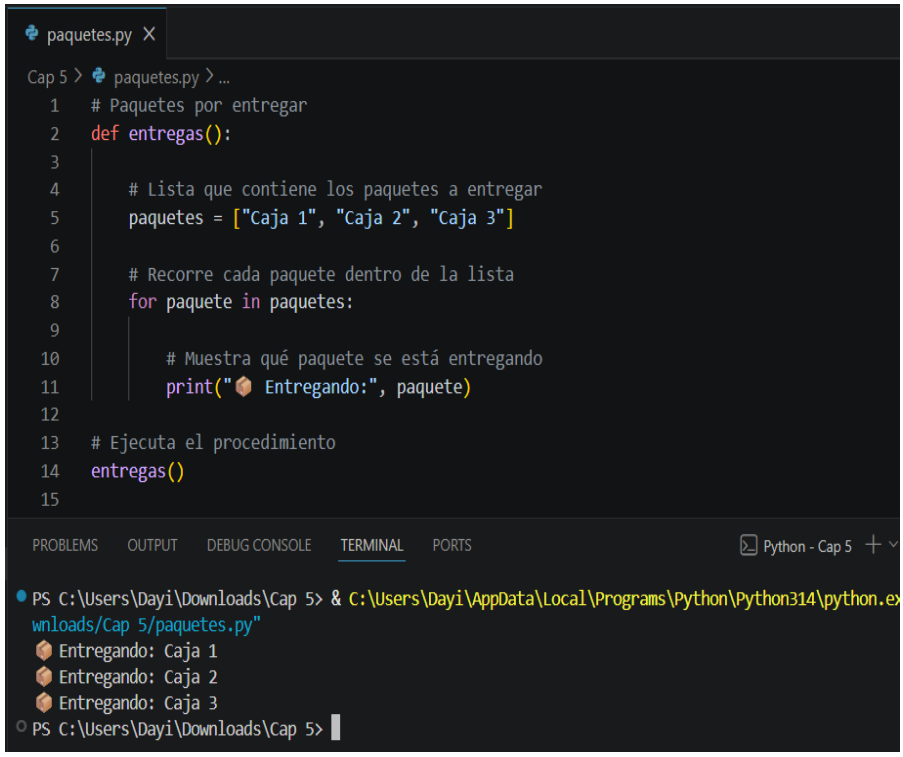


Figura 76. Paquetes por entregar con Python.

Nota. Este ejercicio permite usar listas, ejecutar acciones repetitivas sobre varios elementos, reforzando lógica de recorrido y orden en un programa.

Orientación docente: Puede vincularse con ejemplos de logística o envíos, asimismo, puede pedir añadir más elementos a la lista para comprender cómo crecen los datos de manera clara.

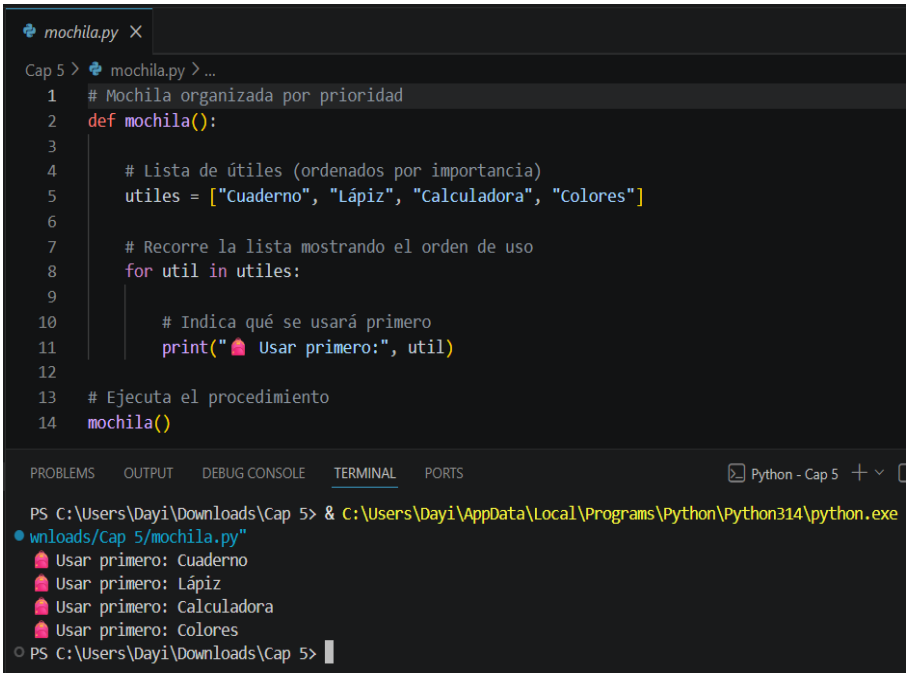


Figura 77. Mochila organizada por prioridad en Python con ejecución del ejercicio por consola.

Nota. Este ejercicio permite trabajar listas como estructuras ordenadas, donde cada elemento tiene una posición específica, evidenciando cómo el orden de datos influye en la forma en que se presentan o utilizan.

Orientación docente: Se puede explicar la importancia del orden en las listas, además, se puede pedir al estudiante reorganizar los elementos para observar cambios en la salida del ejercicio.



Figura 78. Marca de celular.

Nota. Permite trabajar diccionarios de forma simple, utilizando una sola clave y valor. Refuerza la comprensión inicial del concepto.

Orientación docente: Se puede usar como primer ejemplo antes de agregar más datos. Favorece el aprendizaje progresivo.

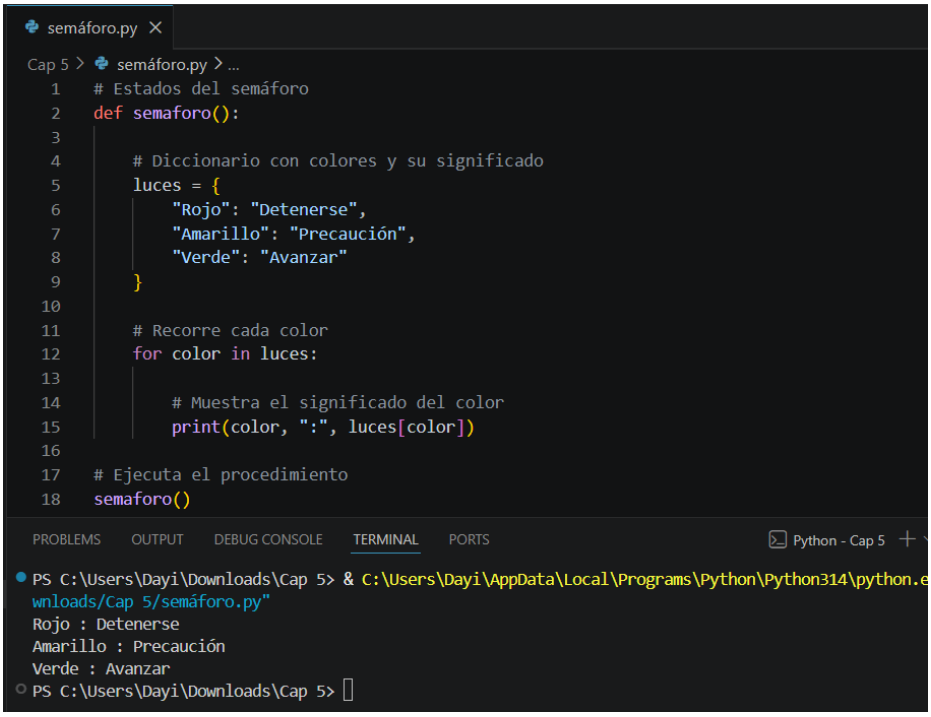


Figura 79. Estados del semáforo.

Nota. Este ejercicio permite trabajar diccionarios como asociaciones entre conceptos, lo que refuerza la interpretación de datos organizados.

Orientación docente: Puede integrarse con educación vial para favorecer el aprendizaje significativo mediante ejemplos reales.

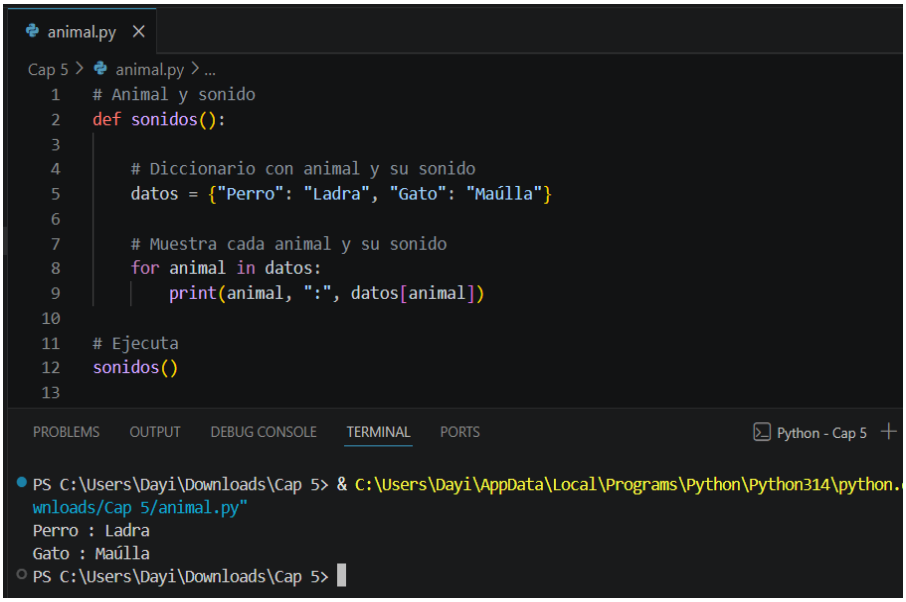


Figura 80. Animal y sonido dentro de Python con ejecución por consola.

Nota. Este ejercicio permite representar relaciones simples entre elementos usando diccionarios, ayudando a organizar información de forma clara y práctica.

Orientación docente: Ideal para iniciar con ejemplos sencillos, se puede pedir al estudiante cambiar los animales o agregar nuevos para reforzar la comprensión.

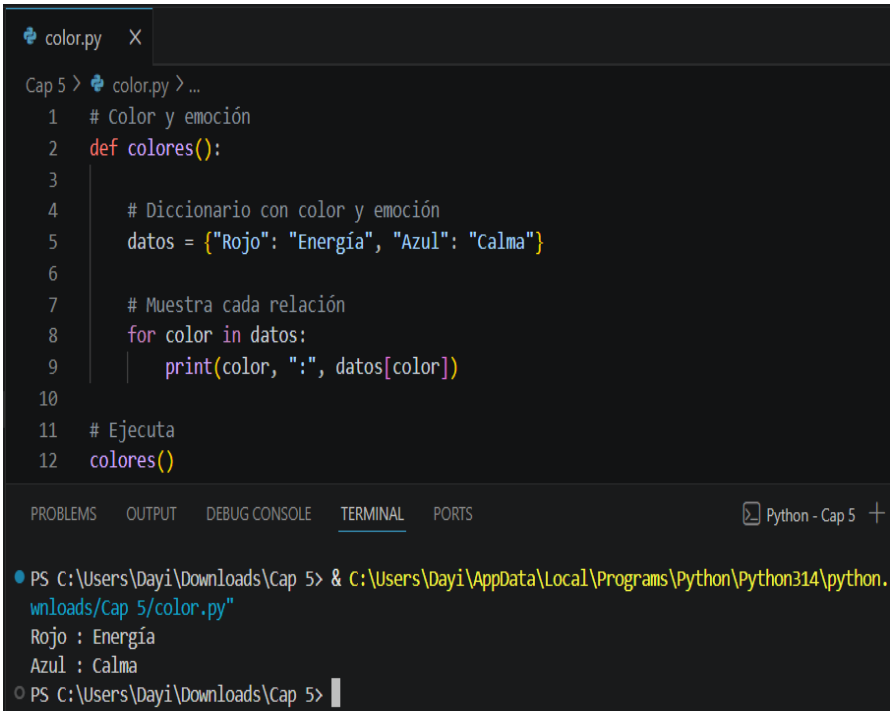


Figura 81. Color y emoción.

Nota. Permite trabajar asociaciones abstractas mediante diccionarios, usando texto para relacionar ideas y reforzar la comprensión de este tipo de estructuras.

Orientación docente: Puede integrarse con educación emocional, por ejemplo asociando emociones con situaciones. Esto favorece un aprendizaje más significativo y cercano al estudiante.

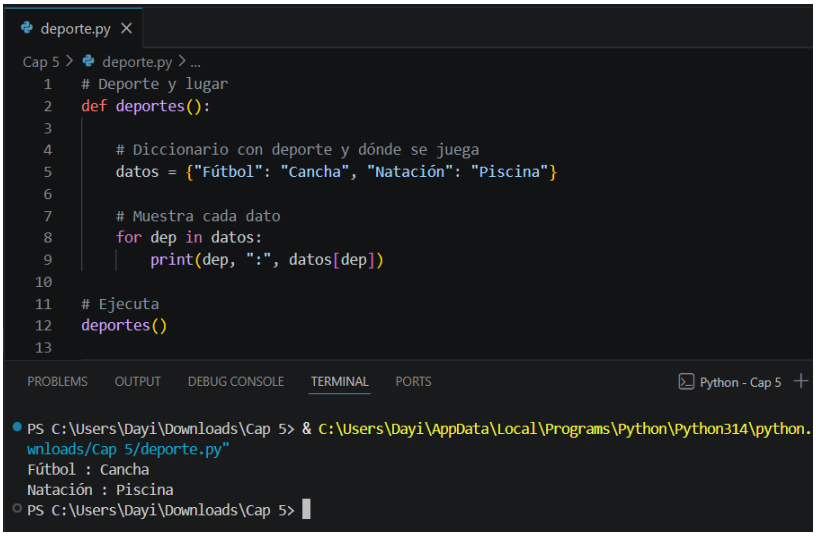


Figura 82. Deporte y lugar.

Nota. Este ejercicio permite relacionar actividades con su contexto, refuerza el uso práctico de diccionarios.

Orientación docente: Se puede pedir agregar más deportes ayudando a reforzar la comprensión.



Figura 83. Vehículo y uso, ejemplificaciones en Python.

Nota. Este ejercicio permite relacionar objetos con su utilidad mediante diccionarios, facilitando la comprensión de cómo organizar información en contextos reales y cotidianos, ayuda a identificar la relación clave-valor de forma clara y con ejemplos cotidianos.

Orientación docente: Puede vincularse con la vida diaria, como objetos del hogar o del aula e incluso se puede pedir al estudiante agregar más ejemplos propios, que refuerza el concepto de manera sencilla y sin generar dificultad.

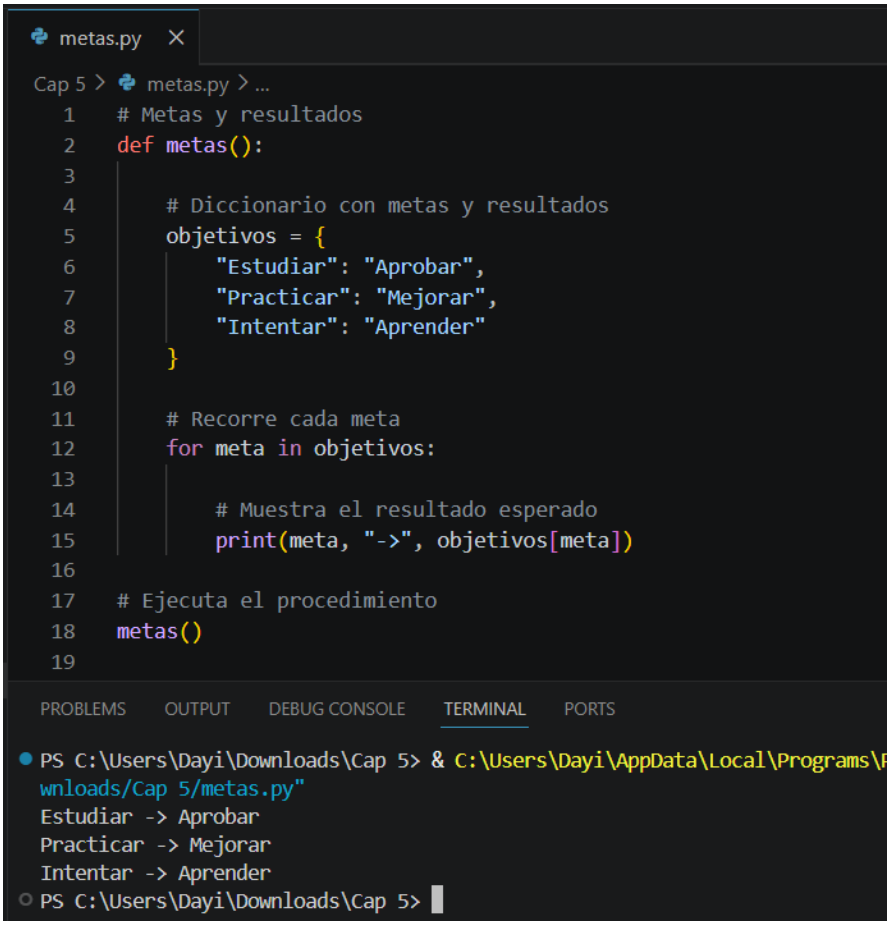


Figura 84. Metas y resultados mediante Python presentado por consola.

Nota. Este ejercicio permite representar relaciones entre acciones y resultados mediante diccionarios, fortaleciendo comprensión de cómo estructuran pares clave-valor, ayuda a visualizar cómo una acción puede generar un resultado específico.

Orientación docente: Puede utilizarse para motivar al estudiante, relacionando acciones con metas personales o logros; se recomienda pedir ejemplos propios, favoreciendo la conexión entre la programación y la vida cotidiana para conllevar una integración natural.

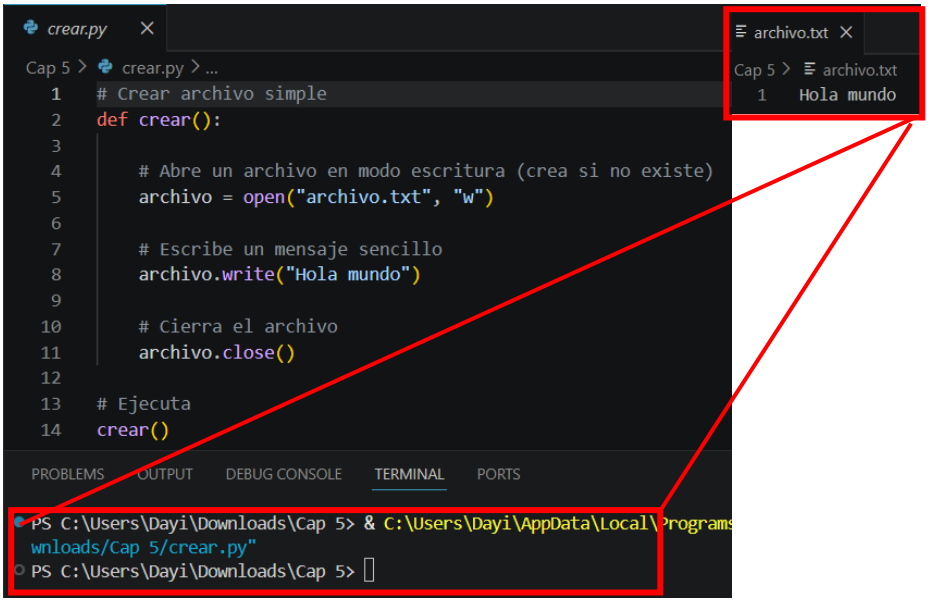


Figura 85. Crear un archivo simple.

Nota. Este ejercicio permite crear un archivo de tipo texto (.txt) llamado archivo.txt en la misma carpeta del programa. Si el archivo no existe, Python lo crea automáticamente; si ya existe, lo sobrescribe. Se utiliza para introducir el manejo básico de archivos y el guardado de información.

Orientación docente: Se recomienda explicar que el modo "w" crea archivos de texto (.txt) de forma automática. El docente puede pedir a los estudiantes revisar la carpeta del proyecto para observar el archivo generado, reforzando así la relación entre el código y el resultado real.

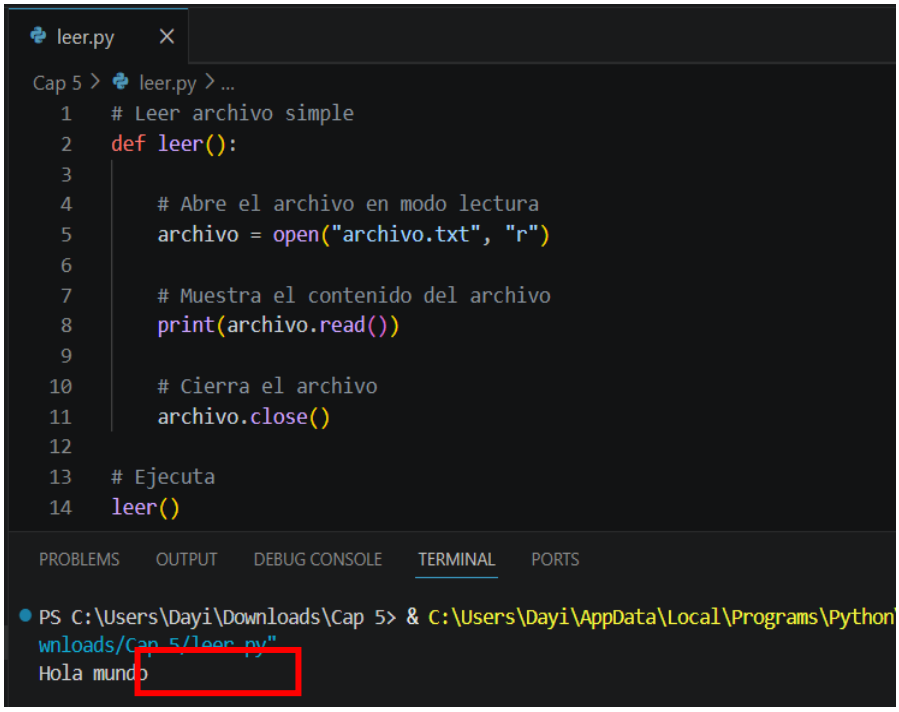


Figura 86. Leer archivo simple.

Nota. Permite leer el contenido de un archivo previamente creado, reforzando el uso del modo lectura y la comprensión de cómo acceder a información almacenada.

Orientación docente: Se recomienda ejecutarlo después del ejercicio anterior, ya que, esto ayuda a comprender cómo recuperar y visualizar la información guardada de forma práctica.

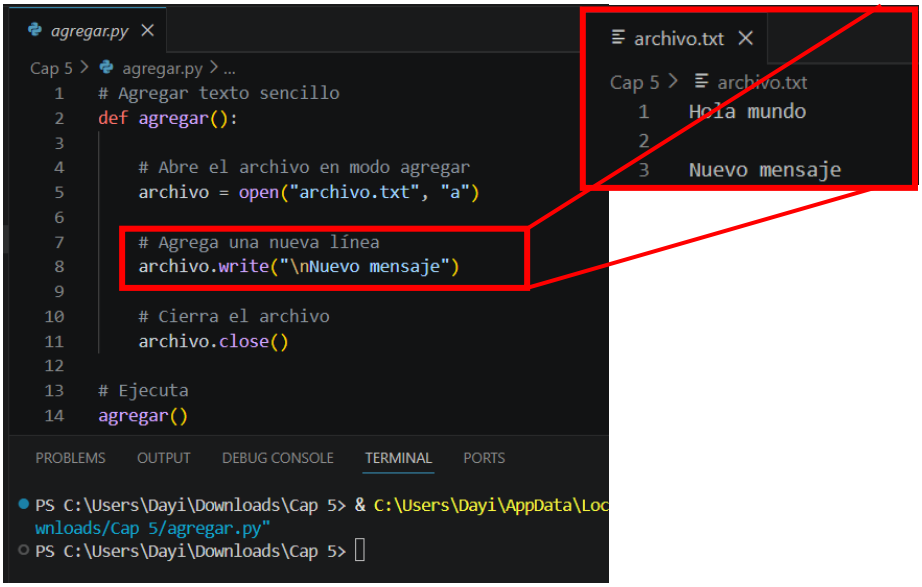


Figura 87. Agregar un texto sencillo en un archivo.txt dentro de Python.

Nota. Permite añadir información a un archivo sin borrar lo anterior, reforzando el uso del modo append (“a”) y consolidando lo trabajado en ejercicios previos.

Orientación docente: Explicar la diferencia entre “w” (sobrescribe) y “a” (agrega), por tal motivo, se recomienda ejecutarlo varias veces para observar cómo se acumula la información.

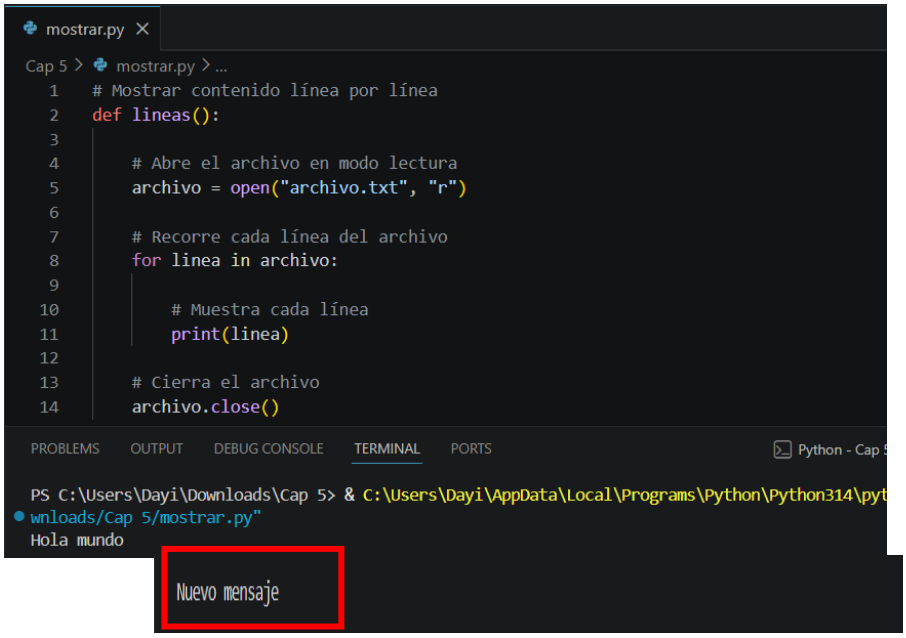


Figura 88. Mostrar contenido línea por línea en la ejecución por consola

Nota. Este ejercicio permite leer un archivo de forma ordenada línea por línea, utilizando ciclos para recorrer su contenido, por eso, refuerza la comprensión del manejo de archivos y cómo procesar información de manera secuencial en un programa, ayuda a identificar cómo cada línea puede ser tratada de forma independiente, lo que resulta útil en múltiples aplicaciones reales.

Orientación docente: Puede trabajarse con textos simples como listas de nombres, tareas o mensajes, normalmente se recomienda que el/los estudiantes modifique el contenido del archivo y vuelva a ejecutarlo para observar los cambios. Esto favorece la comprensión del recorrido de datos, el uso de ciclos y la relación entre almacenamiento y lectura de información.

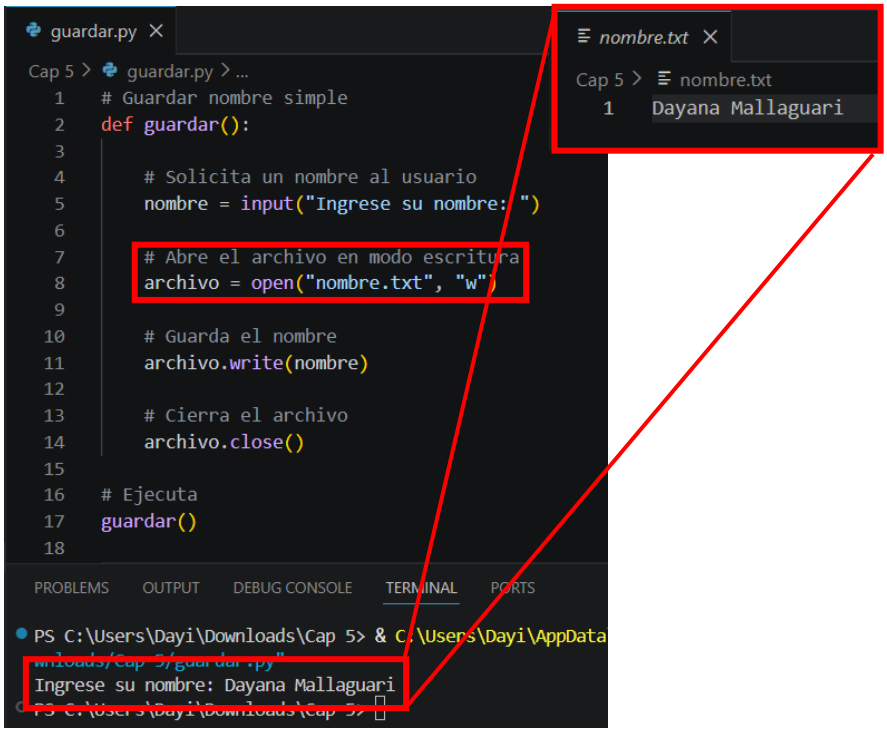


Figura 89. Guardar un nombre en un archivo de texto.

Nota. Este ejercicio permite crear un archivo de texto (nombre.txt) y guardar dentro de él un dato ingresado por el usuario, en este caso su nombre, refuerza la comprensión de cómo un programa puede generar archivos y almacenar información de forma permanente, la cual puede ser consultada o utilizada posteriormente.

Orientación docente: Es importante explicar que el programa no solo guarda un dato, sino que crea un archivo ".txt" en el sistema y escribe dentro de él el nombre ingresado, asimismo, recomendamos que el estudiante abra el archivo después de ejecutar el programa para verificar el contenido, puede repetir el ejercicio con diferentes nombres para

observar cómo se sobrescribe la información y así comprender mejor el uso del modo escritura ("w").

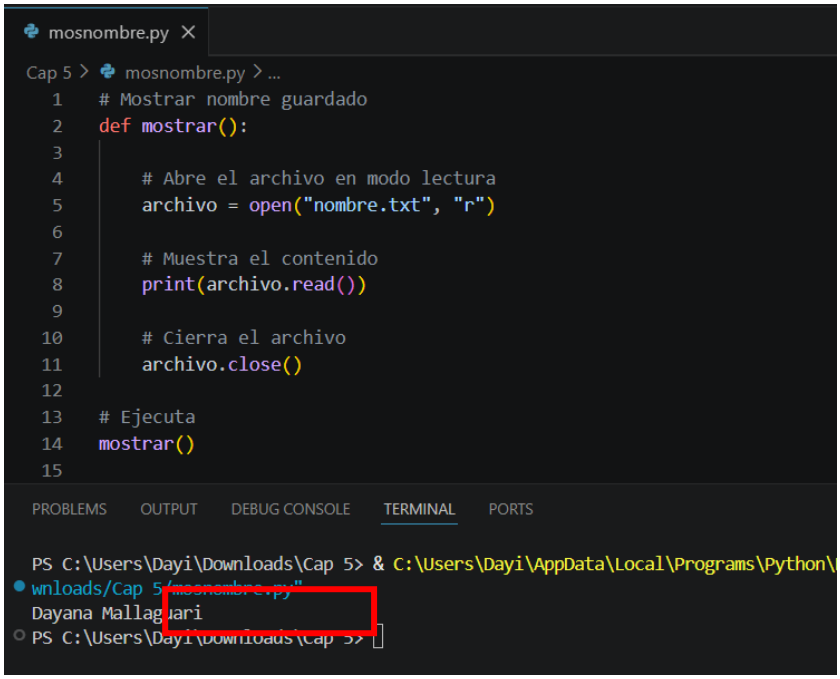


Figura 90. Lectura de nombre almacenado en archivo.

Nota. Este ejercicio permite leer y mostrar en pantalla el nombre que fue previamente guardado en el archivo “nombre.txt”, refuerza la relación entre escritura y lectura de archivos, ya que utiliza la información generada en el ejercicio anterior, ayuda a comprender cómo los datos almacenados pueden recuperarse y utilizarse dentro de un programa en otro momento.

Orientación docente: Ideal para cerrar el ciclo de aprendizaje junto con el ejercicio de guardado, recomendamos que el estudiante ejecute primero el programa que crea el archivo y guarda el nombre, y luego este para leerlo, evidenciando la conexión entre ambos procesos,

también se puede pedir que modifique el nombre varias veces y observe cómo cambia el resultado al leer el archivo, fortaleciendo así la comprensión práctica del manejo de archivos.

5.6 Mini-proyecto: Registro de calificaciones

Enunciado:

Diseña un programa que permita registrar el nombre y la nota de varios estudiantes, guardarlos en un archivo y mostrar el promedio general.

Solución:

mini-proyecto.py

C: > Users > Will > Desktop > Colaboraciones > Libro Python > Recursos de los Capitulos > Cap 4 > mini-proyecto.

```python
#Mini-Proyecto

import csv
# Creamos una lista vacía para guardar los datos
calificaciones = []
# Pedimos la cantidad de estudiantes
cantidad = int(input("¿Cuántos estudiantes desea registrar?: "))
# Repetimos el proceso para cada estudiante
for i in range(cantidad):
    nombre = input(f"Ingrese el nombre del estudiante {i+1}: ")
    nota = float(input("Ingrese la nota final: "))
    calificaciones.append([nombre, nota])
# Guardamos los datos en un archivo CSV
with open("calificaciones.csv", "w", newline="") as archivo:
    escritor = csv.writer(archivo)
    escritor.writerow(["Nombre", "Nota"])
    escritor.writerows(calificaciones)
# Calculamos el promedio
suma = 0
for estudiante in calificaciones:
    suma += estudiante[1]

promedio = suma / len(calificaciones)
print("El promedio general es: ", promedio)
print("\n Calificaciones:", calificaciones)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\xampp\htdocs\GIPTEDU> & C:/Users/Will/AppData/Local/Programs/Python/Python
s de los Capitulos/Cap 4/mini-proyecto.py"
¿Cuántos estudiantes desea registrar?: 2
Ingrese el nombre del estudiante 1: Walter
Ingrese la nota final: 10.00
Ingrese el nombre del estudiante 2: Will
Ingrese la nota final: 9.85
El promedio general es: 9.925

 Calificaciones: [['Walter', 10.0], ['Will', 9.85]]
PS C:\xampp\htdocs\GIPTEDU>
```

Figura 35. Solución del mini-proyecto: Registro y promedio de calificaciones en Python.

Nota. El programa registra calificaciones de estudiantes, las guarda en un archivo CSV y calcula el promedio general.

Desde el pensamiento computacional, este mini-proyecto integra organización de datos, procesamiento matemático y análisis estructurado de información, simulando situaciones reales del ámbito escolar y fortaleciendo la toma de decisiones basada en datos.

5.7 Integración STEAM del mini-proyecto

Science (Ciencia): El uso de listas, diccionarios y archivos permite a los estudiantes recopilar, organizar y analizar datos, habilidades fundamentales en procesos científicos; registrar información como calificaciones, nombres o resultados, se simulan prácticas propias de la ciencia, como la observación sistemática, el registro de datos y la interpretación de resultados, fortaleciendo el pensamiento analítico y la toma de decisiones basadas en evidencias.

Technology (Tecnología): Este capítulo desarrolla competencias digitales clave al introducir el almacenamiento y gestión de información mediante estructuras de datos y archivos digitales, manejo de archivos de texto y CSV muestra cómo la tecnología facilita la organización, conservación y reutilización de datos, preparando a los futuros docentes para integrar herramientas digitales en contextos educativos reales.

Engineering (Ingeniería): El diseño de programas que utilizan listas, diccionarios y archivos implica planificar estructuras de datos, definir procesos de almacenamiento y validar resultados, reflejando el pensamiento ingenieril, estudiantes siguen un proceso lógico de diseño,

prueba y corrección de soluciones, aplicando principios de eficiencia, organización y resolución de problemas.

Arts (Arte): La organización de la información y la presentación de resultados fomentan la creatividad digital, todo esto a través de mensajes personalizados, formatos de salida claros y estructuras bien organizadas, los estudiantes desarrollan habilidades de comunicación y expresión, comprendiendo que la programación también puede ser una herramienta creativa para presentar información de manera estética y comprensible.

Mathematics (Matemáticas): El trabajo con listas y archivos involucra operaciones matemáticas como conteo, sumas, promedios y conversiones numéricas; cálculo del promedio de calificaciones refuerza el razonamiento lógico-matemático, el manejo de datos numéricos y la precisión en los cálculos, evidenciando la relación directa entre programación y matemáticas.

5.8 Recursos digitales complementarios del capítulo

Para reforzar los contenidos de este capítulo, se han elaborado archivos digitales que contienen los ejemplos prácticos desarrollados a lo largo del mismo, incluyendo programas con listas, diccionarios, manejo de archivos de texto y archivos CSV, así como el mini-proyecto de registro de calificaciones; estos recursos permiten a los docentes y estudiantes experimentar directamente con el código, comprender su funcionamiento y adaptar los ejemplos a distintos contextos educativos.

Los archivos se encuentran disponibles en una carpeta digital para su descarga, uso y reutilización en actividades futuras, asimismo, el acceso

a estos materiales se facilita mediante un código QR, lo que garantiza una forma rápida, práctica y segura de obtener los recursos complementarios del capítulo.



Figura 91. Código QR para descarga de archivos complementarios del Capítulo 5.

Nota. El código QR permite acceder a los programas de listas, diccionarios, manejo de archivos, archivos CSV y al mini-proyecto de registro de calificaciones desarrollados en el mismo.

5.9 Resumen del capítulo

En este capítulo se abordó estructuras de datos fundamentales en Python, específicamente listas y diccionarios, así como manejo básico de archivos, estas herramientas permiten almacenar, organizar y manipular información de manera eficiente, lo cual nos facilita el desarrollo de programas que trabajan con conjuntos de datos, asimismo, se analizó el uso de listas como colecciones ordenadas de elementos, permitiendo acceder, modificar y recorrer datos mediante índices, se

estudiaron diccionarios como estructuras que almacenan información mediante pares clave-valor, facilitando una organización más flexible y significativa de datos en comparación con las listas.

Otro aspecto relevante es el manejo de archivos, lo cual permite guardar y recuperar información de forma persistente; se explicó cómo crear, escribir y leer archivos utilizando Python, destacando la importancia de modos de apertura y uso de estructuras seguras como with open ( ) para evitar errores comunes en la manipulación de archivos.

Por último, mediante actividades prácticas y ejercicios aplicados, se integraron listas, diccionarios y archivos en resolución de problemas sencillos, desde el enfoque del pensamiento computacional, estos contenidos fortalecen la organización de la información, estructuración de datos y capacidad de desarrollar soluciones más completas y aplicadas a contextos reales actuales.

5.10 Autoevaluación del capítulo

Pregunta 1: ¿Para qué se utiliza una lista en Python?

a) Para almacenar un solo dato.
b) Para guardar datos ordenados y recorrerlos.
c) Para ejecutar archivos externos.
d) Para crear interfaces gráficas.

Pregunta 2: ¿Cuál es la principal ventaja de usar un diccionario en Python?

a) Permite almacenar datos sin orden.

b) Facilita el acceso a valores mediante claves.

c) Solo puede contener números.

d) Reemplaza a las listas.

Pregunta 3: ¿Qué función cumple el modo "w" al abrir un archivo en Python?

a) Leer el contenido del archivo.

b) Agregar información sin borrar datos.

c) Crear o sobrescribir un archivo.

d) Ejecutar el archivo.

Pregunta 4: ¿Para qué se utiliza un archivo CSV en Python?

a) Para crear gráficos automáticos.

b) Para almacenar datos en forma de filas y columnas.

c) Para ejecutar programas más rápido.

d) Para guardar imágenes.

Pregunta 5: ¿Qué operación se realiza en el mini-proyecto del capítulo?

a) Conversión de texto a voz.

b) Registro de calificaciones y cálculo del promedio.

c) Creación de juegos interactivos.

d) Desarrollo de una página web.

Pregunta 6: ¿Cómo se define un diccionario?

a) []

b) ()

c) {}

d) <>

Pregunta 7: ¿Qué ocurre si se intenta acceder a una clave que no existe?

a) Se devuelve 0

b) Se genera un error

c) Se crea automáticamente

d) Se ignora

Pregunta 8: ¿Qué permite el uso de archivos en Python?

a) Mostrar datos en pantalla

b) Guardar información de forma permanente

c) Crear variables

d) Ejecutar ciclos

Pregunta 9: ¿Qué hace with open("archivo.txt", "w")?

a) Lee un archivo

b) Cierra el programa

c) Abre un archivo para escribir

d) Elimina el archivo

Pregunta 10: En un contexto educativo, el uso de listas y diccionarios permite:

a) Memorizar código

b) Organizar y gestionar información

c) Evitar el uso de datos

d) Solo imprimir resultados

REFERENCIAS BIBLIOGRÁFICAS

Ascencio Jordán, E., Gil Mateos, J. E., Ascencio Jordán, E., & Gil Mateos, J. E. (2023). Enseñanza de la programación: ¿Nueva didáctica o una didáctica diferente? *Revista Cubana de Educación Superior*, *42*(3). http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S0257-43142023000300018&lng=es&nrm=iso&tlng=es

Camacho, R. S., & Grané, M. (2023). Programas de pensamiento computacional en educación primaria: Una revisión sistemática. *Digital Education Review*, (44), 133-145. https://doi.org/10.1344/der.2023.44.133-145

Castañón, T. I. P., Gómez, J. A. R., & López, Y. C. (2022). Procedimiento metodológico para el uso de editores de texto desde el entorno web como herramienta educativa: Editores de texto desde el entorno web. *Didáctica y Educación ISSN 2224-2643*, *13*(5), 1-18. http://revistas.ult.edu.cu/index.php/didascalia/gateway/plugin/pubIdResolver/ark:/54724/DDE.v13i5.1292

Cervera-Manjarrez, N., Oquendo-González, E. J., Velásquez-Pérez, Y., Rose-Parra, C., Cervera-Manjarrez, N., Oquendo-González, E. J., Velásquez-Pérez, Y., & Rose-Parra, C. (2023a). Principios y estrategias para el desarrollo del pensamiento computacional. *Cienciamatria. Revista Interdisciplinaria de Humanidades, Educación, Ciencia y Tecnología*, *9*(17), 120-132. https://doi.org/10.35381/cm.v9i17.1128

Cervera-Manjarrez, N., Oquendo-González, E. J., Velásquez-Pérez, Y., Rose-Parra, C., Cervera-Manjarrez, N., Oquendo-González, E. J., Velásquez-Pérez, Y., & Rose-Parra, C. (2023b). Principios y estrategias para el desarrollo del pensamiento computacional. *Cienciamatria. Revista Interdisciplinaria de Humanidades, Educación, Ciencia y Tecnología*, *9*(17), 120-132. https://doi.org/10.35381/cm.v9i17.1128

Chimarro-Amaguaña, J. D., Chuqui-Barriga, F. A., Guamán-Cullispuma, D. P., & Quishpe-Farinango, C. I. (2023). El auge exponencial del lenguaje Python en el desarrollo tecnológico. *Revista Científica INGENIAR: Ingeniería, Tecnología e Investigación. ISSN: 2737-6249.*, *6*(12), 240-256.

Colussi, N., & Monjelat, N. (2023). Proyectos de programación grupal en Python: ABP en el redictado del ciclo inicial universitario. *JAIIO, Jornadas Argentinas de Informática*, *9*(9), 61-64.

García, V. H. M., & Naranjo, S. G. C. (2025). Aprendizaje Gamificado de Programación en Python: Diseño e Implementación en Segundo de Bachillerato Técnico. *Ciencia Latina Revista Científica Multidisciplinar*, *9*(2), 3125-3148. https://doi.org/10.37811/cl_rcm.v9i2.17130

García-Peñalvo, F. J., Conde, M. Á., Gonçalves, J., & Lima, J. (2019). Computational thinking and robotics in education. *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality, TEEM'19*, 2-5. https://doi.org/10.1145/3362789.3362957

Gómez, E. S. (2021). Python en Ingeniería en Ciencias Informáticas. *Serie Científica de la Universidad de las Ciencias Informáticas*, *14*(12), 1-15.

Jaimes, E. I. G., Chau, A. L., Mora, V. T., & Hernández, R. R. (2018). Estrategia didáctica de enseñanza y aprendizaje para programadores de software. *RIDE Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, *9*(17), 688-712. https://doi.org/10.23913/ride.v9i17.402

Liao, E. (2024). Research on Teaching Reform for Python Programming Curriculum Based on AIGC-CDIO-OBE Model. *2024 6th International Conference on Computer Science and Technologies in Education (CSTE)*, 106-109. https://doi.org/10.1109/CSTE62025.2024.00026

Meseguer, G. O., & Serrano, J. L. (2024). Implementación y formación del profesorado de educación primaria en pensamiento computacional: Una revisión sistemática. *RIED-Revista Iberoamericana de Educación a Distancia*, *27*(1), 255-287. https://doi.org/10.5944/ried.27.1.37572

Monsálvez, J. C. G. (2017). Python como primer lenguaje de programación textual en la Enseñanza Secundaria. *Education in the Knowledge Society (EKS)*, *18*(2), 147-162. https://doi.org/10.14201/eks2017182147162

Morales, M., Antonio, A., Mora, R. D., & Humberto, D. (2009). *NASPI: UNA NOTACIÓN ALGORÍTMICA ESTÁNDAR PARA PROGRAMACIÓN IMPERATIVA*. (3).

Navas-Bonilla, C. del R., Guerra-Arango, J. A., Oviedo-Guado, D. A., & Murillo-Noriega, D. E. (2025a). Inclusive education through technology: A systematic review of types, tools and characteristics. *Frontiers in Education*, *10*. https://doi.org/10.3389/feduc.2025.1527851

Navas-Bonilla, C. del R., Guerra-Arango, J. A., Oviedo-Guado, D. A., & Murillo-Noriega, D. E. (2025b). Inclusive education through technology: A systematic review of types, tools and characteristics. *Frontiers in Education*, *10*. https://doi.org/10.3389/feduc.2025.1527851

Orjuela Díaz, M., & Rodríguez Vanegas, E. A. (2022). *Desarrollo del pensamiento variacional y computacional a través del lenguaje Python en Educación Básica Secundaria en Colombia*. http://repository.unad.edu.co/handle/10596/47745

Osorio, Y. E. C., Pardo, I. D. T., & Gil, M. G. (2023). Modelos mentales y algoritmos de programación en estudiantes de media técnica en informática. *Revista Virtual Universidad Católica del Norte*, (69), 98-134.

Perea, E. A. B. (2023a). Pensamiento computacional y programación en la formación de estudiantes desde edades tempranas. *Revista Educación*, *47*(2).

https://www.redalyc.org/journal/440/44074795006/html/?utm_source=chatgpt.com

Perea, E. A. B. (2023b). Pensamiento computacional y programación en la formación de estudiantes desde edades tempranas. *Revista Educación*, 1-18. https://doi.org/10.15517/revedu.v47i2.53645

Pérez Angulo, J. A. (2019). El pensamiento computacional en la vida cotidiana. *Revista Scientific*, *4*(13), 293-306. https://doi.org/10.29394/Scientific.issn.2542-2987.2019.4.13.15.293-306

Prokopyev, M. S., Vlasova, E. Z., Tretyakova, T. V., Sorochinsky, M. A., Solovyeva, R. A., Prokopyev, M. S., Vlasova, E. Z., Tretyakova, T. V., Sorochinsky, M. A., & Solovyeva, R. A. (2020). Desarrollo de un curso de programación para estudiantes de una institución de educación superior de formación docente utilizando el lenguaje de programación Python. *Propósitos y Representaciones*, *8*(3). https://doi.org/10.20511/pyr2020.v8n3.484

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, *11*(4), 193. https://doi.org/10.3390/info11040193

Rivera, J., Ávila, G., & Gómez, N. M. (2023). INTEGRACIÓN EL LENGUAJE PYTHON Y RASPBERRY PI COMO ESTRATEGIA PEDAGÓGICA PARA EL LOGRO DE

COMPETENCIAS DIGITALES EN ESTUDIANTES UNIVERSITARIOS. *CIE Academic Journal*, *2*(2), 44-57.

Sánchez-García, J. E., Urías-Ruiz, M., & Gutiérrez-Herrera, B. E. (2015). Análisis De Los Problemas De Aprendizaje De La Programación Orientada a Objetos. *Ra Ximhai*, *11*(4), 289-304.

Sbampato, K., Arruda, H., & Silva, É. R. (2025). ChatGPT, Python, and Microsoft Excel. *Journal of the Medical Library Association*, *113*(1), 110-112. https://doi.org/10.5195/jmla.2025.2065

Schnieder, M., & Williams, S. (2023). Educational Mobile Apps for Programming in Python: Review and Analysis. *Education Sciences*, *13*(1), 66. https://doi.org/10.3390/educsci13010066

Soto Gómez, E. (2021). Python en Ingeniería en Ciencias Informáticas. *Serie Científica de la Universidad de las Ciencias Informáticas*, *14*(12), 1-15.

Tinoco, L. M. E., Aushay, A. E. C., & Ordóñez, J. C. D. (2023). Calidad de datos con Python: Un enfoque práctico. *Esprint Investigación*, *2*(2), 26-34. https://doi.org/10.61347/ei.v2i2.55

Vera, G. Y. B., Bonilla, E. J. G., Arteaga, W. A. Y., & Jaramillo, K. J. Y. (2025). VISUAL STUDIO CODE COMO ESTRATEGIA INNOVADORA EN EL PROCESO DE APRENDIZAJE EN LA ASIGNATURA DE DISEÑO WEB. *Journal of Science and Research*, *10*(INNOVA-2024). https://revistas.utb.edu.ec/index.php/sr/article/view/3626

Villa, G. G. C., & Buenaño, D. E. A. (2025). Integración del Pensamiento Computacional en Contextos Educativos: Revisión Sistemática de Literatura en artículos de acceso abierto de los últimos 10 años en SCOPUS. *Reincisol.*, *4*(8), 339-369. https://doi.org/10.59282/reincisol.V4(8)339-369

Python y pensamiento computacional para docentes: Guía didáctica con enfoque STEAM, se publicó en el mes de noviembre de 2026.

ISBN: 978-9907-802-89-4

Grupo Editorial BLR
Ecuador
Cel: +593 98 320 4362
https://grupoblr.com/
publicaciones@grupoblr.com

BIOGRAFÍA DE AUTORES

Julio Antonio Encalada Cuenca:
Es docente e investigador en la Universidad Técnica de Machala. Magíster en Educación Informática, cuenta con amplia experiencia en docencia e investigación en el campo de la robótica educativa, el pensamiento computacional y la metodología STEAM. Su trabajo promueve la integración de
tecnologías innovadoras en el aula.

Walter Andrés Benítez Martínez:
Es Tecnólogo en Análisis de Sistemas y estudiante de Pedagogía en Informática en la Universidad Técnica de Machala, cuenta con experiencia docente y en desarrollo de sistemas web, además posee formación en inteligencia artificial, programación, bases de datos, ciberseguridad, competencias digitales y metodologías educativas innovadoras.

Britney Dayana Mallaguari Bolívar:
Es estudiante de Séptimo Semestre de la Carrera de Pedagogía de las Ciencias Experimentales de Universidad Técnica de Machala, Ecuador. Le apasiona la educación, la innovación y el aprendizaje significativo con tecnologías emergentes. Ha fortalecido su formación mediante cursos y proyectos de tecnología educativa.

PYTHON Y PENSAMIENTO COMPUTACIONAL PARA DOCENTES: GUÍA DIDÁCTICA CON ENFOQUE STEAM

Estimado lector, este libro sitúa a la programación en Python como un catalizador pedagógico fundamental para el desarrollo del pensamiento computacional en la educación del siglo XXI, superando la visión instrumental de la codificación técnica para convertirla en un medio de estructuración cognitiva. Concebida específicamente para la formación de futuros docentes, la obra proporciona estrategias metodológicas que articulan la resolución lógica de problemas con la dinamización del aula mediante experiencias interactivas y entornos de aprendizaje gamificados.

El texto asume el pensamiento computacional como un eje formativo transversal, capacitando al futuro educador en competencias esenciales como la descomposición de problemas complejos, el reconocimiento sistemático de patrones, la abstracción y el diseño riguroso de algoritmos. Lejos de concebir el lenguaje Python como un fin aislado, la propuesta lo integra dentro del enfoque interdisciplinario STEAM (Ciencia, Tecnología, Ingeniería, Arte y Matemáticas), demostrando cómo el código enriquece la enseñanza transversal y potencia la creatividad pedagógica.

En su conjunto, la obra se consolida como una guía teórico-práctica indispensable para transformar la didáctica contemporánea, dotando a los educadores de herramientas aplicadas para diseñar situaciones de aprendizaje significativas, fortalecer el razonamiento analítico en los estudiantes y responder con solvencia a las demandas de la sociedad digital.

GRUPO BLR

Grupo Editorial BLR
Ecuador
Cel: +593 98 320 4362
https://grupoblr.com/
publicaciones@grupoblr.com

ISBN: 978-9907-802-89-4