|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Tecnólogo en implementación y gestión de bases de datos |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501096 desarrollar la solución de *software* de acuerdo con el diseño y metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-02  Codificar el algoritmo de la solución del problema de acuerdo con el lenguaje de programación seleccionado. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | CF16 |
| NOMBRE DEL COMPONENTE FORMATIVO | Construcción de algoritmos en un entorno de desarrollo (IDE) |
| BREVE DESCRIPCIÓN | Es a través de los lenguajes de programación se pueden crear diferentes soluciones y responder a millones de necesidades de la vida cotidiana por medio del desarrollo de programas o aplicaciones. Actualmente, están presentes en los diferentes sectores o industrias, pues contribuyen a desarrollar recursos capaces de responder a las deficiencias o vacíos del día a día. |
| PALABRAS CLAVE | Lenguaje de programación, programa, *python*, sintaxis, pseudocódigo. |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | Ciencias naturales, aplicadas y relacionadas |
| IDIOMA | Español |

**Tabla de contenidos**

**Introducción**

1. **Lenguajes de programación**

1.1. Definición y características

1.2. Tipos de lenguajes de programación

1.3. Sintaxis de los lenguajes de programación

**2. Entornos de desarrollo**

2.1. Estructuras de control

2.2. Ciclos

2.3. Estructura de datos

**3. Métodos y funciones**

**Introducción**

|  |
| --- |
| Cuadro de texto |
| Apreciado aprendiz, sea bienvenido a este momento de formación orientado a los procesos de implementación y gestión de soluciones de *software* de acuerdo con el diseño y metodologías de desarrollo, aplicando todo el conocimiento con criterios propios de desempeño en diferentes campos como el del sector académico, productivo e investigativo.  En el siguiente video conocerá, de forma general, la temática que se explorará a lo largo de este componente formativo.  ¡Muchos éxitos en este proceso de aprendizaje! |

**Guion de video introductorio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video motion | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | Construcción de algoritmos en un entorno de desarrollo (IDE) | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** |  | Música de fondo | Desde que aparecieron los primeros ordenadores hasta el día de hoy, no ha cesado la evolución y los avances en los diferentes sectores, especialmente en el sector tecnológico, donde las empresas más grandes a nivel mundial la utilizan cada vez más a través de una secuencia de acciones. Así mismo, ocurre en las actividades diarias de una persona, donde la mayor parte del tiempo termina realizando una secuencia de acciones con una relación de sucesión lógica que muchas veces, aunque no lo parezca, no ocurren en el mismo orden. | Desde que aparecieron los primeros ordenadores hasta el día de hoy, no ha parado la evolución y los avances de estos. |
| **2** |  | Música de fondo | Estas actividades que se hacen diariamente se terminan realizando de forma natural, aplicadas a cualquier tarea o acción que se hace en el día y que pasan desapercibidas haciendo que la serie de sucesos solo permita ver el resultado final. Cuando se habla de programar o definir el paso a paso de este, es de suma importancia la secuencia que lleva a la creación de los lenguajes de programación que son de distintos tipos de expresiones y reglas con una estructura lógica que sirven para generar tareas que satisfacen necesidades. | Cuando se habla de programar o definir el paso a paso de este, es de suma importancia la secuencia que lleva a la creación de los lenguajes de programación. |
| **3** |  | Música de fondo | Durante los primeros años, los procesos eran de forma lenta y hacían mucho más difícil su desarrollo hasta que ingresó la programación y con ella, los lenguajes en programación que poco a poco, con varias mejoras, lograron transformarse de lenguajes complicados de máquina a sofisticados lenguajes legibles, que a su vez, impulsaron nuevas tecnologías hasta el día de hoy; los lenguajes en programación son los responsables de que la tecnología haya podido avanzar a grandes pasos y se pueda disfrutar de una manera muchaomás rápida y práctica como se aprecia actualmente, con el objetivo de que el hombre pueda realizar los trabajos con la menor dificultad posible, ya que un ordenador permite realizar diferentes operaciones a una gran velocidad; pero es totalmente inservible si no fuera por los mismos programas que se ejecutan, la maquina no piensa de la misma forma que el ser humano, y no es inteligente para saber qué es lo que tiene que hacer, ni comprender mensajes subjetivos. | Poco a poco con varias mejoras lograron transformarse de lenguajes complicados de máquina a sofisticados lenguajes legibles que a su vez impulsaron nuevas tecnologías hasta el día de hoy |
| **4** |  | Música de fondo | Es aquí, donde la lógica humana tiene un papel muy importante en todo el conocimiento de la programación, porque debe aprender a comunicarse con la computadora, a partir de un lenguaje, creando una estructura de forma clara y en el orden adecuado para que la acción sea ejecutada correctamente, presentar de manera clara la secuencia que debe ser seguida para alcanzar los resultados esperados. A ese encadenamiento lógico en la programación, se le llama lógica de programación, y la descripción de cómo hacerlo se define como algoritmos. | A ese encadenamiento lógico en la programación, se le llama lógica de programación, y la descripción de cómo hacerlo la definimos como algoritmos. |
| **5** |  | Música de fondo | Por tal motivo, como programador, es importante conocer los conceptos básicos de programación, los tipos de lenguajes que se utilizan para el desarrollo y su funcionamiento en la interpretación de algoritmos, así como para dar solución a los problemas que puedan presentarse y facilitar las actividades que se realizan en el día a día, ser capaz de crear aplicaciones web y móviles, páginas web, *software* de programación entre otros. | Ser capaz de crear aplicaciones web y móviles, páginas web, *software* de programación entre otros. |
| **Nombre del archivo** | 228130\_v1 | | |  |

1. **Lenguajes de programación**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| **¿Qué es un programa?**  Según, (Joyanes 2020): “un programa es sólo una secuencia de instrucciones que indican a una computadora las tareas a realizar. Evidentemente se necesita proporcionar estas instrucciones en un lenguaje que la computadora pueda comprender”. En otras palabras, un programa de ordenadorr es un conjunto de instrucciones o sentencias propias de un determinado lenguaje de programación, ordenadas lógicamente con el fin de solucionar un problema específico. | |

|  |
| --- |
| Cuadro de texto |
| A lo largo de este componente formativo, se estudia la importancia de saber programar y la lógica que permite comprender cómo se construyen los programas en las computadoras tomando como base el lenguaje Python que es uno de los más usados actualmente y más fáciles de aprender. |

**1.1. Definición y características**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider Presentación | |
| **Introducción** | A continuación, se presenta información relevante sobre la importancia de saber programar y la eficacia de la lógica. Se sugiere leer con atención y tomar nota de los aspectos más relevantes para el aprendizaje. | |
| **¿Qué es un lenguaje?**  Conjunto de símbolos y palabras (vocabulario o léxico) y conjunto de reglas (sintaxis y semántica) que permiten agrupar los símbolos para formar las frases del lenguaje. | | Fractal Realms series. Design composed of fractal elements, grids  and symbols as a metaphor on the subject of education, science and technology  **Imagen:** 228130\_i1 |
| **¿Qué es programación?**  Es un programa que escribe una secuencia de frases del lenguaje, es decir, la programación es el conjunto de acciones que incluye el análisis, el diseño, la implementación, la prueba y la depuración de un código en un determinado lenguaje de programación para poder solucionar un problema con la computadora. | | Lenguaje de programación PHP. Man escribe código PHP. Preprocesador de hipertexto en el teclado. Coder se dedica a la programación en PHP. Lenguaje de programación moderno. Desarrollador de software de fondo azul.  **Imagen:** 228130\_i2 |
| **¿Qué es un lenguaje de programación?**  El lenguaje de programación es una herramienta que permite crear programas, expresar algoritmos con precisión y, mediante las instrucciones necesarias, crear comunicación persona-equipo con el fin de realizar determinadas tareas en el dispositivo. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. | | Imagen abstracta de Hacker Activity en forma de cielo estrellado o espacio, que consiste en puntos, líneas y formas en forma de planetas, estrellas y el universo. Ciberataque vector. Modo de color RGB  **Imagen:** 228130\_i3 |
| Un lenguaje de programación está constituido por un léxico, una sintaxis y una semántica que definen su estructura principal y le dan un significado a sus elementos y expresiones para resolver problemas que se pueden ver interpretados en apps, páginas webs, programas de *software, scripts* y otros elementos que hoy facilitan el día a día a los diferentes tipos de usuarios. | | **Imagen:** 228130\_i4 |
| De acuerdo con lo mencionado anteriormente, los lenguajes de programación vienen definidos con tres características principales que los identifican:  Tienen un ***léxico,*** que son las palabras propias o palabras claves utilizadas en un programa que funcionan como un conjunto de símbolos o elementos básicos dentro de un lenguaje. | | **Imagen:** 228130\_i5 |
| Estos símbolos o elementos básicos son los siguientes:  a. **Identificadores:** nombres que se dan a algunos elementos del lenguaje, por ejemplo, los nombres de las variables, los nombres de procedimientos, los nombres de las funciones, los nombres de los módulos, etc.  b. **Instrucciones:** palabras claves propias del lenguaje que representan sentencias de procesamiento y definición de otros elementos del lenguaje.  c. **Constantes:** valores que no cambian a lo largo del programa.  d. **Operadores:** símbolos que se usan para realizar operaciones entre variables y constantes.  e. **Comentarios:** son los caracteres que se utilizan para documentar los procesos de los programas. | | **Imagen:** 228130\_i6 |
| Tienen una **sintaxis*,*** que es un conjunto de reglas que definen las expresiones de texto que son correctas, así como la secuencia de símbolos que forman las diferentes sentencias del lenguaje. Estas reglas indican si una sentencia está bien escrita o no. | | **Imagen:** 228130\_i7 |
| Tienen una determinada **semántica** que define cuál será el resultado de la ejecución de un programa. Indica el significado de las expresiones y los tipos de datos del lenguaje. | | Pictogram code or html brackets icon..  **Imagen:** 228130\_i8 |

* 1. **Tipos de lenguajes de programación**

|  |
| --- |
| Cuadro de texto |
| Existen diversos tipos de lenguajes de programación, ya que cada uno es creado para resolver diferentes tipos de problemas con características similares; esto ha hecho que cada día aparezcan distintos tipos de lenguajes que ofrecen un estilo de programación fácil, pero con una funcionalidad mayor y con más potencia. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 2 |
| **Introducción** | A continuación, se presenta los distintos lenguajes de programación: |
| Futurista robot inteligencia artificial revolucionaria desarrollo de tecnología IA y concepto de aprendizaje de máquinas. Investigación mundial de la ciencia biónica robótica para el futuro de la vida humana. 3 D. Ilustración 3D  **Imagen:** 228130\_i9 | |
| Las diferencias que hay entre los tipos de lenguaje son:   1. Su propósito (general o específico para cierto tipo de tareas). 2. La sintaxis. 3. Los paradigmas de programación (imperativo, declarativo, orientado a objetos, funcional). 4. Las librerías disponibles, tales como f*rameworks* o comunidad que lo soporta. | |
| Los tipos de lenguajes de programación se clasifican:  De acuerdo con el nivel de programación, se encuentran de tres tipos:  a. **Lenguaje de máquina:** es el más primitivo y se basa en la numeración binaria, 0 y 1. Este lenguaje es utilizado directamente por máquinas o computadoras.  b. **Lenguajes de programación de bajo nivel:** es un poco más fácil de interpretar, pero puede variar de acuerdo con la máquina o computadora que se esté programando.  c. **Lenguajes de programación de alto nivel:** son los más utilizados;se usan palabras en inglés, lo cual facilita que una persona pueda intervenir más fácil que en los dos anteriores. | |
| De acuerdo con su orden cronológico de creación, se clasifican de la siguiente manera:   1. **Primera generación:** el lenguaje máquina y el ensamblador. 2. **Segunda generación:**primeros lenguajes de programación de alto nivel, los más reconocidos son: *Fortran y Cobol*. 3. **Tercera generación*:*** alto nivel imperativo, pero mucho más utilizados y vigentes en la actualidad como son: *Visual Basic*, C#, C++, *Java y Javascript.* 4. **Cuarta generación:** más cercanos a la época actual, es común encontrarlos en aplicaciones de gestión y manejo de bases de datos tales como: SQL, *Perl,* PHP, *Python* y Ruby. 5. **Quinta generación:** los más avanzados, creados para la inteligencia artificial y para el procesamiento de lenguajes naturales como son: *Mercury, Lisp y Prolog.* | |
| De acuerdo con la finalidad y herramientas en las que se use, se clasifican de la siguiente manera:   1. **De propósito general:** C, C++, C#, *Java, Go* y PHP 2. **Aplicaciones web:** *Ruby on Rails*, PHP y *Python* 3. **Multiuso:** *Java,* PHP, *Javascript y Python* | |
| De acuerdo con su forma de ejecución, se clasifican en:  a. **Lenguaje interpretado o de script:** se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja es que su lenguaje interpretado es más flexible y portable.  b. **Lenguaje compilado:** convierte un lenguaje “cualquiera” en código máquina”. Se pueden ver como traductores, permitiendo a la máquina interpretar lo que se quiere hacer. La ventaja de este lenguaje es que su ejecución es más rápida. | |

**1.3 Sintaxis de los lenguajes de programación**

|  |
| --- |
| Cuadro de texto |
| Se conoce como sintaxis a la parte visible de un lenguaje de programación. Es decir, que es el conjunto de reglas que debe seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.  En esta parte de la programación es muy importante el desarrollo de la lógica, ya que es la base de todo el conocimiento en programación; es decir; se debe aprender primero la lógica de programación, antes que la sintaxis de un lenguaje específico o determinado. Así se aprende a resolver problemas independientes de un lenguaje de programación para después tener la experticia de escribir un código que la computadora interpreta correctamente. |

**2. Entornos de desarrollo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cuadro de texto | | | | |
| En el mundo del *software* y la tecnología, un entorno de desarrollo integrado (IDE) por sus siglas en inglés *Integrated Development Environment*, es una herramienta informática que proporciona servicios integrales para facilitar al programador el desarrollo de *software* a través de un conjunto de procedimientos y aplicaciones que sirven para codificar, integrar, testear, validar, depurar y ejecutar programas.  Es decir, un entorno de desarrollo integrado es un espacio de trabajo en el que se puede llevar a cabo la gestión del desarrollo de *software* en todas sus etapas hasta que se implemente una versión real. | | | | |
| **Tipo de recurso** | Video motion | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | Entornos de desarrollo | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** |  | Música de fondo | Existen muchos tipos de entorno de desarrollos, este curso se centra en el uso y estudio de *Visual Studio Code* (VSC), utilizando en su mayoría, conocimientos y aplicaciones del lenguaje de programación *Python,* con el fin de aprender el uso dentro los entornos de desarrollo y así, facilitar el estudio de *software*, aprovechando las extensiones y funcionalidades de este a la hora de codificar o completar una aplicación mientras se desarrolla.  ¿Qué es *Visual Studio Code*?  Es un editor de código fuente ligero pero potente, creado por Microsoft, que se ejecuta desde el escritorio tanto en Windows como en macOs y Linux, es gratuito, de código abierto, está disponible para todos los sistemas operativos y permite trabajar con varios lenguajes de programación. Es una herramienta centrada en el código, lo que facilita la edición de diferentes tipos de archivos de código fuente y de proyectos basados en carpetas, así como codificar aplicaciones web y móviles sobre las plataformas más reconocidas con soporte integrado para una gran cantidad de lenguajes de programación y características de edición como *IntelliSense*, además de encontrar referencias de símbolos, llegar rápidamente a una definición de tipo, y mucho más y de fácil adaptación.    Muchas de las características de *Visual Studio Code* no se visualizan directamente, a través de los menús o la interfaz de usuario. Estas, se acceden mediante la paleta de comandos o a través de  archivos .json. | En este componente de construcción de algoritmos en un entorno de desarrollo se tomará como base el lenguaje de programación *Python*. |
| **2** |  | Música de fondo | Es un lenguaje de programación de alto nivel creado a finales de los 90, por el holandés Guido Van Rossum quien era aficionado a un grupo de humoristas ingleses de los años 60, “*Monty Python*”, y quienes sirvieron como inspiración para el nombre. La implementación de *Python* surge cuando su creador trabajaba en el Centro para las Matemáticas y la Informática de los Países Bajos en el proyecto de lenguaje de programación ABC, es decir, *Python* fue creado como sucesor del lenguaje ABC como un lenguaje de programación de propósito general o “lenguaje de *scripting”* porque fue pensado para ser usado en proyectos sencillos convirtiéndose de esta forma en un lenguaje de programación eficaz y fácil de aprender. | ¿Qué es *Python*? |
| **3** |  | Música de fondo | Para instalar *Visual Studio Code* hay que entrar a su página oficial y descargar la versión más adecuada para el sistema operativo que tenga nuestra computadora.  Para *trabajar Python en el VS Code*, es recomendable instalar las siguientes extensiones: | Configuración *Visual Studio Code* para trabajar con *Python* |
| **4** |  | Música de fondo | *Pytho*n: esta extensión, creada por Microsoft, da a VS *Code* muchas funcionalidades para programar en *Python,* como son el autocompletado, el formateo del código fuente, ayuda sobre los diferentes parámetros que aceptan las funciones, herramientas para hacer *debugging* y testear nuestro código *Python*, entre otras. | Esta extensión, creada por Microsoft, da a VS Code muchas funcionalidades para programar en Python como son: |
| **5** |  | Música de fondo | *AREPL for Python*: es muy útil, ya que evalúa nuestro código mientras lo escribimos y los resultados del mismo se muestran en una ventana dentro del editor. | Esta extensión es muy útil ya que: |
| **6** |  | Música de fondo | *Python Docstring Generator*: Este ayuda a generar, de manera eficiente, *docstrings* de los métodos y funciones que se creen. Para usarla sólo hay que abrir triples comillas, dobles en la primera línea del método o función y pulsar la tecla intro. Esto muestra una plantilla con los diferentes parámetros a llenar, y que se pueden recorrer fácilmente con la ayuda del tabulador. | Esta extensión ayuda a generar de manera eficiente d*ocstring*s de los métodos y funciones que se creen. |
| **Nombre del archivo** | 228130\_v2 | | |  |

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| ¡Importante!  El concepto de «lenguaje de *scripting*» se ha transformado ampliamente desde su creación, porque hoy en día *Python* es usado para desarrollar, de manera rápida, aplicaciones en diversos sectores, especialmente en las plataformas y apps.  Es un interpretador de instrucciones que permite usar un lenguaje en forma recíproca, es decir, permite trabajar de manera interactiva en una ventana, y, además, a través de programas que pueden desarrollarse y ensayar a medida que son construidos. Cuenta con implícitas estructuras de datos eficientes como conjuntos, diccionarios, listas y tuplas, con un enfoque simple, pero muy eficiente que permiten llevar a cabo múltiples tareas en pocas líneas de código y de manera legible. | |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | A continuación, se presentan las funciones y usos de Python | |
| **Figura 1**  *Funciones y usos de Python*    Se debe elaborar una infografía similar o igual a la que se presenta en la imagen, las referencias de las imágenes empleadas se encuentran en el anexo 1, el cual encontrará en la carpeta con la siguiente denominación CF16\_228130\_Anexo1. | | |
| **Código de la imagen** | 228130\_i10 | |
| **Punto caliente 1** | **¿Para qué sirve?**  Su objetivo principal es la automatización de procesos, permitir que las tareas sean más simples a través de la creación de códigos legibles que ahorran recursos y en especial tiempo. Uno de sus puntos más fuertes es que logra “comprobar los errores sobre la marcha” para ser solucionados cuando se afecta a la memoria, lo que mantiene la integridad de la matriz y evita dificultades a la hora de escribir el código. | punto caliente en este icono |
| **Punto caliente 2** | **¿Dónde se utiliza?**  Al ser un lenguaje de propósito general, se puede utilizar en diferentes campos para análisis de datos, crear aplicaciones de escritorios, bases de datos relacionales, NoSQL, criptomonedas, desarrollo de aplicaciones multimedia, desarrollo de videojuegos, desarrollo móvil y desarrollo web, prototipado de *software*, etc. | punto caliente en este icono |
| **Punto caliente 3** | **¿Cómo funciona?**  El intérprete de *Python* maneja módulos de código que son intercambiables en lugar de una extensa lista de sentencias de códigos estándar para los lenguajes de programación funcional.  Realizando los siguientes pasos:  ● **Paso 1:** el intérprete lee un código o instrucción, verifica y comprueba la sintaxis de cada línea; si encuentra un error, detiene la traducción y muestra un mensaje de error.  ● **Paso 2:** si no presenta errores, el intérprete lo traduce a un lenguaje intermedio llamado código *byte*.  ● **Paso 3:** el código del *byte* se envía a la máquina virtual *Python*, donde de nuevo se ejecuta el código del *byte* en PVM. Si se produce un error durante esta ejecución, ésta se detiene con un mensaje de error. | punto caliente en este icono |
| **Punto caliente 4** | **Características del lenguaje.**  *Python* se caracteriza especialmente en:  ✔ Es un lenguaje interpretado  ✔ Lenguaje sencillo, legible y elegante  ✔ Sintaxis simple, clara y sencilla  ✔ Tiene tipado dinámico  ✔ Es fuertemente tipado  ✔ Es multiplataforma  ✔  ✔ Permite llevar a cabo aplicaciones de manera práctica y rápida  ✔ Soporta programación imperativa, orientada a objetos y funcional  Su objetivo principal es aumentar la legibilidad y la facilidad de desarrollo. | punto caliente en este icono |
| **Punto caliente 5** | **Características del lenguaje *Python*.**   * Es Interpretado: el código del programador es traducido mediante un intérprete a medida que lo necesite. * Multiparadigma: soporta más de un paradigma. * Tipo dinámico: permite la transformación de variables. * Multiplataforma: se puede trabajar en cualquier sistema operativo. | punto caliente en este icono |
| **Punto caliente 6** | Ejemplo de empresas a nivel mundial que son un caso de éxito con el uso de *Python*:  Google, Netflix, YouTube, Dropbox, Spotify, Yahoo!, la NASA, PayPal, Prezi, Instagram y muchas más. | punto caliente en este icono |

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| ¡Importante!  Se debe tener en cuenta que *Python* no es adecuado para la programación de bajo nivel. | |

|  |  |
| --- | --- |
| Tipo de recurso | Acordeón tipo 1 |
| **Introducción** | Instalación de *Python* en Windows: para instalar *Python* en un equipo con el sistema operativo Windows se debe realizar los siguientes pasos: |
| **Imagen:** 228130\_i11 | |
| 1.Ir a la página de descargas para Windows del sitio oficial y descargar la versión adecuada del instalador ejecutable de *Python*. | |
| 2.Realizar la instalación haciendo doble clic sobre el archivo ejecutable descargado para lo cual es necesario tener permisos de Administrador del equipo. | |
| 3.Una vez iniciado el instalador, en la ventana *Install Python* se debe activar las casillas de las opciones: *Install launcher for all users* (*recommended) y Add Python to PATH*. Después, seleccione la opción *Customize installation*. | |
| 4.En la ventana *optional features* seleccione todas las opciones y hacer clic en el botón [*Next*]. | |
| 5.En la ventana *advanced options* activar todas las opciones, aceptar la ruta dada por defecto o escribir una ruta personalizada del directorio de instalación e iniciar la instalación haciendo clic en el botón *[Install*]. | |
| 6.Esperar unos segundos y comenzará el proceso de instalación de *Python*. | |
| 7. Al finalizar la instalación debe aparecer una ventana con el mensaje *setup was successful*. En esta ventana se puede acceder a un tutorial *online*, a la documentación oficial de *Python* y a información adicional de las novedades de la versión instalada. | |
| 8. Posteriormente se podrá instalar cualquier paquete adicional utilizando el instalador PIP. | |

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| **Un ejemplo introductorio desarrollado en modo interactivo.**  Para iniciar el aprendizaje del lenguaje *Python,* en esta sección se desarrolla un ejemplo en forma interactiva en la ventana principal o *shell.* Se recomienda que el aprendiz realice esta práctica en la computadora.  Solución en modo interactivo en la ventana principal o *shell de Python*.  Para iniciar el entorno interactivo de *Python* se debe escribir *Python* desde la línea de comando o símbolo del sistema, en el cual puede escribir instrucciones de *Python*. Al inicio de las siguientes líneas se muestra un aviso para escribir cada instrucción. Este aviso son tres ángulos como estos: >>>  En esta ventana escriba cada instrucción a la derecha del símbolo >>>  Al final de cada línea presione la tecla *Ente*r. Siga el ejemplo que se muestra a continuación:  >>>*print*(“¡Hola Mundo!”)  >>>¡Hola Mundo!  >>>*print*(“¡Mis primeras líneas de código en *Python*!”)  >>>¡Mis primeras líneas de código en *Python*!  En el anterior ejemplo, se ordena visualizar primero un mensaje con las palabras: ¡Hola Mundo! y en la siguiente línea también se visualiza el mensaje: ¡mis primeras líneas de código en *Python*!  El mismo ejemplo se debe resolver posteriormente escribiendo un programa en una ventana de edición de *Python*. Esta práctica permitirá resaltar las diferencias entre el modo interactivo y el modo de programación. | |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Carrusel de tarjetas | |
| **Introducción** | **Operadores de *Python.***  Son los símbolos utilizados para expresar las operaciones básicas en los programas, por ejemplo, hacer los cálculos matemáticos y la comparación de diferentes expresiones del código. | |
| **Imagen:** 228130\_i12 | | |
| Los operadores se clasifican en:  a. Operadores aritméticos  b. Operadores de comparación  c. Operadores lógicos  d. Operadores de asignación | | **Imagen:** 228130\_i13 |
| **Operadores aritméticos*.***  Son los símbolos que se utilizan para escribir ecuaciones aritméticas. También se pueden usar los paréntesis para definir el orden de las operaciones. El resultado de estas operaciones siempre será numérico.  (+): es el operador para realizar la suma de dos o más valores.  (-): es el operador para realizar la resta de dos o más valores.  (\*): este es el operador para realizar la multiplicación de dos o más valores en la ecuación.  (/): este es el operador para realizar la división de dos o más valores. | | **Imagen:** 228130\_i14 |
| **Operadores relacionales o de comparación**  Estos símbolos son utilizados para escribir expresiones de comparación entre dos o más valores. El resultado siempre será un valor lógico o booleano (*True o False*).  (==): evaluar si dos valores son iguales  (! =): evaluar si dos valores son diferentes  (<): evaluar si el valor del lado izquierdo de la expresión es menor que el valor que está en el lado derecho.  (>): evaluar si el valor del lado izquierdo de la expresión es mayor que el valor que está en el lado derecho.  (<=): evaluar si el valor del lado izquierdo de la expresión es menor o igual que el valor que está en el lado derecho.  (> =): evaluar si el valor del lado izquierdo de la expresión es mayor o igual que el valor que está en el lado derecho. | | **Imagen:** 228130\_i15 |
| **Los operadores lógicos**  Estos operadores se utilizan para evaluar el resultado de dos o más expresiones. El resultado será siempre un valor booleano (*True o False*).  Los tres principales operadores lógicos son: ***And, Or y Not***.  *And*: se utiliza entre dos o más expresiones y su resultado será verdadero si sus expresiones son verdaderas.  *Or:* se utiliza entre dos o más expresiones y su resultado será verdadero si una de sus expresiones es verdadera.  *Not*: es la negación de una expresión. | | **Imagen:** 228130\_i16 |
| **Operadores de Asignación**  Estos operadores permiten realizar una operación y guardar su resultado en la misma variable. | | **Imagen:** 228130\_i17 |
| **Ejemplos de operadores de asignación:** | | **Figura 2**  *Ejemplo de operadores de asignación*    **Imagen:** 228130\_i18 |
| **Operadores especiales**  Existen otros tipos de operadores, a continuación, los 2 más usados de esta categoría:  Operador de inclusión *in o not in.*  Este operador evalúa si un valor está incluido o no en una colección de datos. Ejemplo:  *'o' in 'Python' #Resultado: True*  *'z' in 'Python' #Resultado: False* | | **Imagen:** 228130\_i19 |
| **Operador de concatenación**  El operador + se puede usar para concatenar cadenas de texto o *strings*. Cuando se concatenan cadenas, el resultado es una cadena compuesta por ambas cadenas. Ejemplo:  a='*Python* '  b='es un lenguaje de programación'  c=a + b  print(c) #Resultado: '*Python* es un lenguaje de programación' | | **Imagen:** 228130\_i20 |

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| **Precedencia de operadores**  Si en una expresión hay operadores de diferente tipo, primero se evalúan las operaciones aritméticas, luego las operaciones relacionales y finalmente las lógicas.  Los paréntesis se pueden usar para definir con claridad la precedencia de las operaciones. | |

|  |  |
| --- | --- |
| **Tipo de recurso** | Infografía estática |
| **Texto introductorio** | A continuación, las expresiones más comunes dentro de *Python*: |
| **Figura 3**  *Expresiones más comunes de Python*    Se debe elaborar una infografía similar o igual a la que se presenta en la imagen, se anexa la imagen trabajada en canva a la carpeta como CF16\_228130\_anexo2 | |
| **Código de la imagen** | 228130\_i21 |

**2.1. Estructuras de control**

|  |
| --- |
| Cuadro de texto |
| Las estructuras de control son las encargadas de direccionar la secuencia o flujo de ejecución de las instrucciones de un algoritmo. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Rutas /Pasos Horizontal | |
| **Introducción** | Las estructuras de control se dividen en tres categorías en función del orden de ejecución: | |
| **Paso 1** | a. **De secuencia**: como su nombre lo indica, se ejecutan las instrucciones una seguida de la otra.  b. **De selección o control condicional:** controlan si una sentencia o secuencia de sentencias se ejecutan.  c. **De iteración o de control de ciclos:** se refiere a las instrucciones que se repiten. | **Imagen:** 228130\_i22 |
| **Paso 2** | Las estructuras de secuencias son la forma de solución más sencilla de resolver un problema en un programa, ya que las instrucciones son ejecutadas en un orden secuencial de arriba hacia abajo.  Por lo general, el proceso de una estructura secuencial tiene un comienzo, seguido por la sentencia número uno, luego la sentencia número dos, posteriormente la sentencia número tres, y así sucesivamente hasta llegar a la finalización del proceso. | **Imagen:** 228130\_i23 |
| **Paso 3** | Toda estructura secuencial de un programa se basa en 5 fases:  a. **Declaración de variables:** para utilizar una variable tanto en pseudocódigo como en diagrama de flujo, es necesario siempre declararla, es decir, indicarle al algoritmo que va a utilizar una variable.  b. **Inicialización de variables:** es asignarle un valor a cada una de las variables dependiendo de su tipo de dato.  c. **Lectura de datos:** es el ingreso de datos que se realiza por medio de un dispositivo de entrada.  d. **Cálculos:** son las diferentes operaciones que se realizan con los datos de entrada.  e. **Salida:** es el resultado final de la estructura secuencial. | **Imagen:** 228130\_i24 |
| **Paso 4** | Ventajas de las estructuras secuenciales  Con este tipo de estructura, se tienen tenemos las siguientes ventajas:  a. Los programas son más fáciles de crear  b. Ayuda a entender visualmente la secuencia ordenada  c. Es fácil identificar y corregir posibles errores  d. Disminución de pruebas  e. Mantiene condiciones específicas  f. Permite la rápida corrección y modificación de las sentencias  g. Reducción de tareas de mantenimiento  h. Aumenta la producción del sistema | **Imagen:** 228130\_i25 |
| **Paso 5** | Las estructuras de control condicional  Son las que manejan diferentes opciones de selección dependiendo del resultado de una operación booleana (siempre será un valor verdadero o falso), por ejemplo, una comparación (x = y). Según la expresión sea cierta o falsa, se lleva a cabo un código u otro.  En el lenguaje *Python* existen los siguientes tipos de estructuras de control condicional: | **Imagen:** 228130\_i26 |
| **Paso 6** | **Estructura condicional *if***  Son las que permiten evaluar si una o más condiciones se cumplen, para decidir qué instrucciones se van a llevar a cabo. El resultado de la condición arroja un valor de solo dos posibilidades: verdadero (*True*) o falso (*False*).  El condicional *if* en el lenguaje de programación *Pytho*n maneja las siguientes estructuras:  a. La sintaxis de una sentencia *if* es la siguiente:  *if* condición:  ejecutar código si la condición es *True* | **Figura 4**  *Ejemplo de una estructura de control condicional if*  if (x==0):  print "x tiene un valor de 0" #Resultado: x tiene un valor de 0  **Imagen:** 228130\_i27 |
| **Paso 7** | b. La sintaxis de una sentencia *if* - else es la siguiente:  *if* condición:  ejecutar código si la condición es *True*  *else:*  ejecutar código si la condición es *False* | **Figura 5**  *Ejemplo de una estructura de control condicional if- else*  if (x==0):  print "x tiene un valor de 0"  else:  print "x tiene un valor diferente a 0"  **Nota.** Creación propia  **Imagen:** 228130\_i28 |
| **Paso 8** | c. La sintaxis de una sentencia *if - elif - else* es la siguiente:  *if* primera\_condicion:  ejecutar sentencia  *elif* segunda\_condicion:  ejecutar sentencia  *else:*  ejecutar sentencia alternativa si todas las condiciones previas son evaluadas como *False* | **Figura** 6  *Ejemplo de una estructura de control condicional if - elif - else*  if (x==0):  print "x tiene un valor de 0"  elif (x<=10):  print("x tiene un valor menor que 10")  else:  print "x tiene un valor mayor que 10"  **Imagen:** 228130\_i29 |
| **Paso 9** | Las estructuras de control usan los siguientes tipos de datos:  a. **Datos simples como son:** numéricos, alfanuméricos y lógicos  b. **Datos estructurados como son:** arreglos, registros, archivos y apuntadores  Las estructuras de control también manejan constantes y variables, las cuales pueden ser numéricas, alfanuméricas y lógicas, y por su trabajo, en acumulares y contadores. | **Imagen:** 228130\_i30 |

**2.2. Ciclos**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| Los ciclos también llamados bucles, son estructuras de control iterativas. Tal como su nombre lo indica, consiste en una forma de trabajar que permite llevar a cabo un pedazo del código varias veces. | |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | A continuación, se presentan los tipos de ciclos o bucles: |
| **Imagen:** 228130\_i31 | |
| **Ciclo *for* (para)**  El ciclo *for* o bucle es una sentencia que agrupa un conjunto de instrucciones y es controlado por una variable normalmente llamada i (pero puede ser identificada con cualquier otra letra), de tipo entero cuyo valor va cambiando en cada iteración.  El lenguaje *Python* usa ciclos *for* para iterar sobre una lista de elementos, en otras palabras, los ciclos *for* iteran sobre estructuras de datos basadas en colecciones como listas, tuplas y/o diccionarios. | |
| La sintaxis básica de un ciclo *for* es la siguiente:  *For* valor in lista\_de\_valores:  # puedes usar la variable valor dentro de este bloque de código  #*for, in* -->> son palabras reservadas del bucle *for*  #se puede leer de la siguiente forma:  #por cada <valor> en la <lista\_de\_valores> | |
| Ejemplo de una estructura de un ciclo *for*:  *for i in range(5):*  *print(i)*  El resultado de este ciclo es lo siguiente:  *range*(5)  >  0  1  2  3  4  > | |
| **Ciclo *while* (mientras)**  El ciclo *while* también es una sentencia que agrupa un conjunto de instrucciones, las cuales se repiten mientras se cumpla la respectiva condición booleana (es decir, una expresión que su evaluación sea verdadera o falsa). En el ciclo *while* difícilmente se puede saber con anterioridad el número de veces que se repetirán las instrucciones. | |
| La sintaxis básica de un ciclo *while* es el siguiente:  *while* (condición):  # Ejecuta este bloque de código  # Siempre que condición sea verdadera | |
| Ejemplo de una estructura de un ciclo *while*:    numero = 0  nombres = ['Uno', 'Dos', 'Tres', 'Cuatro', 'Cinco']  *while* numero < 5:  *print* ("Número" + nombres[numero])  numero + = 1  El resultado de este ciclo es lo siguiente:  Número Uno  Número Dos  Número Tres  Número Cuatro  Número Cinco | |

**2.3. Estructura de datos**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | | Pestañas o tabs Verticales |
| **Introducción** | | A continuación, se presenta la manera en que están divididas las estructuras de datos: |
| **Imagen:** 228130\_i32 | | |
| **Imagen:** 228130\_i33 | El lenguaje de programación *Python* maneja diferentes tipos de estructura de datos, las cuales permiten el procesamiento de los datos que utilizan los programas de acuerdo con la respectiva necesidad.  Estas estructuras se encuentran divididas en: | |
| **Imagen:** 228130\_i34 | *a.* Tipos de datos básicos  Los tipos de datos básicos más usados son los siguientes:   * *Entero o int*   En *Python,* los enteros o también conocidos como *int,* son un tipo de datos que permite representar números enteros, es decir, positivos y negativos sin decimales y sin importar la longitud del número almacenado.  Ejemplo:  x=12  *print* (x)  Resultado:  12  Se puede usar la función *type* () para determinar el tipo de dato de la variable indicada entre paréntesis. Ejemplo:  *print(type*(x))  Resultado: <*class 'int'*> | |
| **Imagen:** 228130\_i35 | * Reales o números de punto flotante (*float*)   Son un tipo de datos que permite representar números con punto decimal o expresados en notación de potencias de 10. Ejemplos:  6.52  - 0.5  4.3 e-5 es el número 4.3 multiplicado por 10 elevado a la - 5 | |
| **Imagen:** 228130\_i36 | * Cadena de caracteres o *str*   Son un tipo de datos que permite representar textos o cadenas de caracteres alfanuméricos encerradas entre comillas simples o comillas dobles.  Ejemplos:  s1 = '*Pyhton* es un lenguaje de programación'  s2 = "*Pyhton* es un lenguaje de programación"  *print*(s1)  *print*(s2)  *print(type*(s1))  *print(type*(s2))  Resultado:    *Pyhton* es un lenguaje de programación  *Pyhton* es un lenguaje de programación  <*class 'str'*>  <*class 'str'*> | |
| **Imagen:** 228130\_i37 | * Boolean   En *Python* existe el tipo de dato booleano, el cual es un tipo de dato que permite almacenar dos valores: *True* o *False.*  Ejemplo:  X =*True*  *print* (x)  Resultado:  *True* | |
| **Imagen:** 228130\_i38 | *b.* Tipos de datos complejos  *Python,* posee además de los tipos de datos vistos anteriormente, tres tipos más complejos que pueden almacenar colecciones de datos de diversos tipos y se diferencian por su sintaxis, la manera en la que los datos son gestionados y que admiten una colección de datos.  Los tipos de datos complejos son los siguientes:   * Listas * Tuplas * Diccionarios | |
| **Imagen:** 228130\_i39 | * Listas: es un tipo de colección ordenada   Las listas son estructuras de datos que permiten guardar datos de cualquier tipo, son dinámicas y mutables porque permiten que se pueda agregar, cambiar o eliminar sus elementos.  Las listas son el equivalente a lo que en otros lenguajes se les llama como *arrays,* o vectores y pueden almacenar cualquier tipo de dato: numéricos, *strings,* booleanos e incluso otras listas.  Una lista se crea sencillamente indicando sus valores entre corchetes y separados por comas. | |
| **Imagen:** 228130\_i40 | Ejemplo:  lista1 = [94, “cadena de texto”, *True*, 'otra cadena de texto', 4.8, [5, 10]]  A las listas se accede a sus valores por su número de índice de la siguiente forma:  *print* lista1[1] #Resultado: 94  *print* lista1[2:4] #Resultado: ["cadena de texto", *True,* 'otra cadena de texto']  *print* lista1[-4] #Resultado: *True* | |
| **Imagen:** 228130\_i41 | Las listas permiten modificar cada uno de los datos después de que es creada de la siguiente manera:  lista1[5] = 7.1 #el quinto elemento de la lista se cambia a 7.1  lista1[2] = "primera cadena de texto" #el segundo elemento de la lista se cambia a "primera cadena de texto" | |
| **Imagen:** 228130\_i42 | También se puede recorrer una tupla de la igual forma que con las listas.  Ejemplo:  Lista = [1, 3, 5]  *for* j *in* Lista:  *print*(j) #Resultado 1, 3, 5  Las listas permiten agregar nuevos valores de la siguiente forma:  lista1.*append*('Nuevo elemento de la lista'). | |
| **Imagen:** 228130\_i43 | * Tuplas   El concepto de tupla es muy similar al de lista. Aunque hay algunas diferencias menores, lo fundamental es que, mientras una lista es mutable y se puede modificar, una tupla no admite cambios y, por lo tanto, es inmutable.  Son un tipo de datos que permite almacenar varios datos que no se pueden modificar después de que son declaradas. Una tupla es similar a una lista con la diferencia fundamental de que no permite modificar los datos una vez creados. | |
| **Imagen:** 228130\_i44 | Las tuplas son declaradas utilizando paréntesis en lugar de corchetes.  Ejemplo:  tupla1 = (23, 'cadena de texto', *False*, 'otra cadena de texto', 7.5)  A las tuplas se accede a sus valores por su número de índice de la siguiente forma, siendo 0 (cero), el índice del primer dato:  *print(type*(tupla1)) #Resultado: <*class 'tuple'*>  *print tupla*1[0] #Resultado: 23 | |
| **Imagen:** 228130\_i45 | También se puede obtener una parte de la tupla, indicando el rango de los índices separados por el carácter: en el cual algunos de los dos índices pueden ser opcional. Ejemplos:  print tupla1[1:3] #Resultado: ["cadena de texto", False, 'otra cadena de texto']  print tupla1[3:] #Resultado: ('otra cadena de texto', 7.5)  print tupla1[:2] #Resultado: (23, 'cadena de texto', False) | |
| **Imagen:** 228130\_i46 | Otra forma de acceder a la tupla es de atrás hacia adelante, usando un índice negativo. Ejemplos:  *print tupla1*[-1] #Resultado: 7.5  *print tupla*1[-2] #Resultado: otra cadena de texto  Las tuplas también pueden declararse sin los paréntesis, separando con la coma (,) todos sus elementos.  tupla = 1, 2, 3  *prin*t(tupla) #Resultado: (1, 2, 3) | |
| **Imagen:** 228130\_i47 | Al igual que las listas, las tuplas también pueden ser anidadas.  Ejemplo:  *tupla* = 1, 3, ('x', 'y'), 5  *print*(tupla) #Resultado: (1, 3, ('x', 'y'), 5)  *print*(tupla[2][0]) #Resultado: x | |
| **Imagen:** 228130\_i48 | También se puede convertir una lista en una tupla por medio de la función *tuple*().  Ejemplo:  lista = [1, 3, 5]  tupla = *tuple*(lista)  *print(type*(tupla)) #Resultado: <class 'tuple'>  *prin*t(tupla) #Resultado: (1, 3, 5) | |
| **Imagen:** 228130\_i49 | También se puede asignar el valor de una tupla con varios datos a n variables.  Ejemplo:  lista = (2, 4, 6)  x, y, z = lista  *print*(x, y, z) #Resultado: 2 4 6 | |
| **Imagen:** 228130\_i50 | También es posible declarar una tupla de un solo elemento. Para lo cual se debe usar la coma antes del paréntesis, con el fin de que el dato sea interpretado como un valor entero.  Ejemplo:  tupla = (2,)  *print(type*(tupla)) #<*class 'tuple'*> | |
| **Imagen:**  228130\_i51 | También se puede recorrer una tupla de la igual forma que con las listas.  Ejemplo:  tupla = (1, 3, 5)  *for* j *in* tupla:  *print(*j) #Resultado: 1, 3, 5 | |
| **Imagen:** 228130\_i52 | Tuplas vs Listas  Aunque puedan parecer estructuras de datos muy similares, las tuplas no cuentan con ciertas operaciones, especialmente las que tienen que ver con la modificación de sus valores, ya que son inmodificables. Si las listas son más potentes y flexibles, ¿por qué́ se necesitan las tuplas? Las siguientes son las ventajas potenciales del uso de tuplas con respecto a las listas:  1. Las tuplas ocupan menos espacio en memoria  2. Debido a que las tuplas son inmutables, tienen protección frente a cambios no deseados  3. Las tuplas se pueden usar como claves de diccionarios | |
| **Imagen:** 228130\_i53 | * Diccionarios   Van Rossum (2017) indica “Lo mejor es pensar en un diccionario como un conjunto no ordenado de pares clave: valor, con el requerimiento de que las claves sean únicas (dentro de un diccionario en particular). Un par de llaves crean un diccionario vacío: {}”. | |
| **Imagen:** 228130\_i54 | Son una colección de datos que permite guardar su contenido por medio de unas palabras conocidas como llaves o claves y cada una tiene asignado un valor. Mientras que en las estructuras de listas y tuplas se obtienen sus datos solo y únicamente por un número de índice, los diccionarios permiten usar una clave para declarar y acceder a un valor de acuerdo con la siguiente sintaxis:  nombre\_diccionario = {'llave1': valor1, 'llave2': valor2, 'llave3': valor3}  Ejemplo:  diccionario1 = {"Nombre": "Ana", "Edad": 23, "Documento": 12345678}  print(diccionario1) #Resultado: {'Nombre': 'Ana', 'Edad': 23, 'Documento': 12345678} | |
| **Imagen:** 228130\_i55 | Otra forma de declarar un diccionario en *Python* es usando *dic*t() e colocando los pares llave y valor entre paréntesis. Ejemplo:  diccionario1 = *dict*([        ('Nombre', 'Ana'),        ('Edad', 23),        ('Documento', 12345678),  ])  *print*(diccionario1) #Resultado: {'Nombre': 'Ana', 'Edad': 23, 'Documento': 12345678} | |
| **Imagen:** 228130\_i56 | Los diccionarios en *Python* tienen las siguientes *características*:   * Las claves deben ser únicas y pueden ser de cualquier tipo de datos inmutables: cadenas de texto, enteros, flotantes, tuplas, etc. * Son **mutables**, por lo que permiten añadir, borrar y modificar sus elementos. * Mantienen el orden en el que se insertan las claves. * Tienen un **acceso muy rápido** a sus elementos, debido a la forma en la que están implementados internamente. | |
| **Imagen:** 228130\_i57 | Algunas propiedades de los diccionarios en *Python* son las siguientes:  Son dinámicos, pueden crecer o decrecer ya que se pueden añadir o eliminar elementos.  Son indexados, los elementos del diccionario son accesibles a través de la respectiva llave.  Y son anidados, un diccionario puede contener a otro diccionario en su campo valor. | |
| **Imagen:** 228130\_i58 | Un diccionario permite eliminar cualquier entrada de la siguiente forma:  del(diccionario['llave2'])  Al igual que las listas, el diccionario permite modificar los valores  diccionario['clave\_1'] = 'Nuevo Valor' | |

|  |
| --- |
| Cuadro de texto |
| Otro tipo de estructura de datos:  **Matrices**  Las matrices no son una estructura propia de *Python*. Simplemente, una matriz es una lista de listas que se interpreta desde el punto de vista matemático.  Cuando se habla de matrices en *Python*, se está refiriendo a una matriz rectangular bidimensional especializada de datos, los cuales están almacenados en filas y columnas. Dentro de esta matriz puede haber datos en forma de números, cadenas, símbolos, expresiones, etc. La matriz es una de las estructuras de datos más importantes que se pueden utilizar en cálculos matemáticos y científicos.  Las **matrices** son una estructura de datos bidimensional donde los elementos se organizan en **filas** y **columnas**.  Ejemplo de esto:  **Tabla 1**  *Matrices*   |  |  |  |  | | --- | --- | --- | --- | | 2 | -5 | -11 | 0 | | -9 | 4 | 6 | 13 | | 4 | 7 | 12 | -2 |   Esta **matriz** es una matriz de 3 x 4 porque tiene 3 filas y 4 columnas. |

**3. Métodos y funciones**

|  |
| --- |
| Cuadro de texto |
| *Python* dispone de algunos objetos denominados clases que contienen funciones definidas para ciertas aplicaciones. Estas funciones se definen como métodos y la metodología se denomina programación orientada a objetos.  Para manejo de listas, *Python* tiene una clase denominada *lis*t. Esta clase es residente en la librería estándar y no se necesita importar para usarla. Al crearse una variable u objeto de tipo lista se tiene acceso a las funciones o métodos definidos en la clase y se usa con la siguiente notación:  objeto.método  Algunos métodos disponibles en la clase *list* son los siguientes:  Sean **o** un objeto de tipo lista y **e** un elemento de la lista.  **Tabla 2**  *Métodos disponibles en la clase list*   |  |  | | --- | --- | | **Método** | **Resultado** | | o.*append(*e) | Agrega a la lista **o** el elemento **e** | | o.*insert*(i,e) | Inserta el elemento **e** en la posición i de la lista **o** | | o.*count*(e) | Cuenta las instancias del elemento **e** en la lista **o** | | o.*remove*(e) | Elimina el primer elemento **e** de la lista **o** | | o.*index*(e) | Devuelve la posición del primer elemento **e** en la lista **o** | | o.*sort*() | Ordena la lista **o** ascendentemente | | o.*reverse*() | Invierte la lista **o** | | o.*clear*() | Vacía la lista **o** |   Algunas de estas funciones generan un error *ValueError* si no está el elemento indicado, por lo cual es recomendable, hacer una validación previa de la existencia del elemento. |

|  |
| --- |
| Cuadro de texto |
| **¡Felicitaciones!** Ha finalizado el estudio de este componente formativo, a través del cual ha aprendido sobre la construcción de algoritmos en un entorno de desarrollo (IDE). Ahora, es importante revisar y estudiar la síntesis con la que podrá recordar de forma general los saberes que se han desarrollado; realizar la actividad de aprendizaje donde podrá poner a prueba lo aprendido; consultar el material complementario que le servirá como refuerzo y profundización y a estudiar el glosario que le dará un repaso sobre los distintos conceptos |

**Síntesis**

|  |  |
| --- | --- |
| **Tipo de recurso** | Síntesis |
| Tecnólogo en implementación y gestión de base datos  Síntesis: construcción de algoritmos en un entorno de desarrollo (IDE). | |
| **Introducción** | En este estudio de creación de algoritmos en un entorno de desarrollo (IDE), el aprendiz se puede sumergir en una variedad de conceptos y temas muy interesantes que permiten obtener la información suficiente dentro del componente formativo. Podrá diseñar, programar y desarrollar su propio pensamiento lógico basado en la construcción de algoritmos en un entorno de desarrollo, nombre y misión de esta componente. Los temas que son tratados en todo este ciclo son de gran ayuda para resolver tipos de problemas con características tanto simples como complejas, lo que conlleva a un reto constante, debido a que cada día se indagan nuevas tecnologías para la automatización de procesos, con una mayor funcionalidad y potencia; todo esto enfocado en los lenguajes de programación. Como aspecto final de este recurso, los invitamos a observar el siguiente gráfico, que contempla de manera resumida la temática estudiada en este componente formativo: |
| **Figura 7**  *Construcción de algoritmos en un entorno de desarrollo (IDE)*    **Imagen:** 228130\_i59 | |

**Actividad didáctica**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Actividad didáctica. Opción múltiple | |
| Apreciado aprendiz, a continuación, encontrará unos enunciados relacionados con la experiencia de aprendizaje del presente componente. Escoja la respuesta correcta de acuerdo con lo aprendido.  ¡Adelante! | | **Imagen 1:** 228130\_i60 |
| 1. Proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales. | | **imagen 2:** 228130\_i61 |
| a) Programación (Correcta) | | b) Compilación |
| c) Hardware | | d) *Software* |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 1. Lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. | | **Imagen 3:** 228130\_i62 |
| a) Variables | | c) Lenguaje de programación. (Correcta) |
| b) Storyboard | | d) Lenguaje de diseño y visualización |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 1. Cuál de las siguientes opciones **NO** se relaciona con la clasificación de los lenguajes de programación: | | **Imagen 4:** 228130\_i63 |
| a) Lenguaje de máquina | | b) Lenguajes de programación de bajo nivel |
| c) Lenguajes de programación *PSeInt* (Correcta) | |  |
| **Retroalimentación**   * **Respuesta correcta**   **¡Excelente!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 1. Conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. | | **Imagen 5:** 228130\_i64 |
| a) Código fuente (Correcta) | | b) Histograma |
| c) Diagrama de flujo | | d) Pseudocódigos |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 5. Conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. | | **Imagen 6:** 228130\_i65 |
| a) Compilación | | b) Algoritmo (Correcta) |
| c) Ciclo de actividades | | d) Ninguna de las anteriores |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 6. ¿Qué es un Diagrama de flujo de datos? | | **Imagen 7:** 228130\_i66 |
| a) Es una serie de pasos para dar solución a un problema. | | b) Es el orden de los pasos para un sistema de información. |
| c) Es el estudio de las ordenes creadas por un programador. | | d) Es la representación simbólica de un algoritmo. (Correcta) |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 7. ¿Cuál de los siguientes tipos de lenguaje de programación **NO** pertenece a los de propósitos generales? | | **Imagen 8:** 228130\_i67 |
| a) C | | b) *Java* |
| c) C++ | | d) *Python* (Correcta) |
| **Retroalimentación**   * **Respuesta correcta**   **¡Excelente!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 8. ¿Cuál de los siguientes tipos de lenguaje de programación **NO** pertenece a los de aplicaciones web? | | **Imagen 9:** 228130\_i68 |
| a) Python | | b) C# (Correcta) |
| c) PHP | | d) *Ruby on Rails* |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 9. Mezcla de lenguaje natural con algunas convenciones sintácticas propias de lenguajes de programación | | **Imagen 10:** 228130\_i69 |
| a) *Python* | | b) Pseudocódigo (Correcta) |
| c) SQL *server* | | d) PHP |
| **Retroalimentación**   * **Respuesta correcta**   **¡Muy bien!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| 10. ¿Qué es un entorno de desarrollo integrado (IDE)? | | **Imagen 11:** 228130\_i70 |
| a) Es una aplicación de *software* que ayuda a los programadores a desarrollar código de *software* de manera eficiente. (Correcta) | | b) Es un lenguaje de programación ampliamente utilizado en las aplicaciones web. |
| c) Un conjunto de instrucciones que conducen a la solución de un problema determinado. | | d) Es un *software* que enseña a programar desde cero a experto. |
| **Retroalimentación**   * **Respuesta correcta**   **¡Excelente!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Actividad didáctica. Verdadero y falso | |
| Apreciado aprendiz, el objetivo principal de la presente actividad es verificar los saberes más importantes del material correspondiente: | | **Imagen 1:** 228130\_i71 |
| Pseudocódigo se crea mediante un lenguaje natural que nos lleve a una sintaxis para interpretar los lenguajes de programación. | | **imagen 2:** 228130\_i72 |
| Verdadero (correcto) | | Falso |
| **Retroalimentación**   * **Respuesta correcta**   **¡Excelente!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |
| El lenguaje de programación de bajo nivel es el más primitivo y se basa en la numeración binaria, 0 y 1. Este lenguaje es utilizado directamente por máquinas o computadoras. | | **Imagen 3:** 228130\_i73 |
| Verdadero | | Falso (correcto) |
| **Retroalimentación**   * **Respuesta correcta**   **¡Excelente!** Has seleccionado la respuesta correcta.  **Respuesta incorrecta**  **¡Vamos, inténtelo de nuevo!** Lo invitamos a retomar el estudio de los saberes desarrollados en este componente formativo. | | |

**Material complementario**

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de recurso | Material complementario | | |
| Tema | Referencia APA del material | tipo | Enlace |
| Estructura secuencial | GSL Industrias. (7 de junio de 2021). *Estructura secuencial.* [Web log post]. *Blogspot.* | Blogs | <https://industriasgsl.com/blogs/automatizacion/que-es-la-estructura-secuencial> |
| Estructuras de control | El libro de Python (2022). *Estructura de control*. | Página web | <https://ellibrodepython.com/estructuras-control-python> |
| Estructura de datos | Rootstack. (29 de diciembre de 2021)Cómo funcionan las matrices en Python y cómo usarlas*.* Web log post]. *Blogspot.* | Blogs | <https://rootstack.com/es/blog/python> |
| Estructura de datos | Python *Software* Foundation. (23 de septiembre de 2022). *El tutorial de Python.* | Página web | <https://docs.python.org/es/3/tutorial/index.html> |
| Entornos de desarrollos | Laca, M. (12 de septiembre de 2018). *Programación estructurada. Lenguaje Python.* | Página web | <https://pythones.net/que-es-python-y-sus-caracteristicas/#Caracteristicas_de_Python> |
| Estructura de datos | Delgado, S. (17 de abril de 2022). Aprende *Python.* *Estructura de datos.* | Libro PDF | <https://aprendepython.es/_downloads/907b5202c1466977a8d6bd3a2641453f/aprendepython.pdf> |
| Ciclos | Moreno, A. & Muñoz, S. (2020). Bucles. *Python práctico.* | Libro PDF | <https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed>= |
| Operadores y tipos de datos | Trejos, O. & Muñoz, L. (2020). Operadores y tipos de datos. *Introducción a la programación con Python.* | Libro PDF | <https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed>= |

**Glosario**

|  |  |
| --- | --- |
| **Tipo de recurso** | Glosario |
| Código de un programa: | la computadora lo usa para generar una versión del programa que es capaz de procesar |
| Depurar: | tarea o proceso especializado para corregir y eliminar errores en un código fuente durante la edición |
| *Frameworks*: | herramienta cuyo objetivo es facilitar el desarrollo y la organización de un *software* o aplicación. |
| Interfaz: | es un programa para que [objetos](https://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)) no relacionados se comuniquen entre sí |
| Librerías o bibliotecas: | es un conjunto de códigos desarrollados por un programador para ser utilizados posteriormente por él mismo o por otros programadores |
| Lenguaje de programación: | es el lenguaje utilizado para realizar el código fuente de todo programa |
| Operadores: | son elementos que relacionan de forma diferente, los valores de una o más variables y/o constantes. Es decir, los operadores nos permiten manipular valores |
| Programación: | es el proceso que se lleva a cabo tanto en el diseño como en el desarrollo de programas y aplicaciones |
| *Script*: | instrucciones de un lenguaje de programación escritas en un archivo de texto |
| Variable: | es un espacio en la memoria para almacenar un valor que puede ser modificado a lo largo de la ejecución |

**Referencias bibliográficas**

|  |  |
| --- | --- |
| **Tipo de recurso** | Bibliografía |
| Amos, D., Bader, A., Jablonski, J. & Heisler, F. 2020) *Python Basics: A Practical Introduction to Python 3.* Real Python. | |
| Joyanes, L. (2020). *Fundamentos de programación*. McGraw-Hill. <https://www.ingebook.com/ib/NPcd/IB_Escritorio_Visualizar?cod_primaria=1000193&libro=10211> | |
| Lewis, R.M. (2017). *Tutorial fácil de seguir para aprender la programación de Python en menos de una semana*. Studoku. <https://www.studocu.com/es/document/universidad-nebrija/programacion-ii/tutorial-facil-de-seguir-para-aprender-la-programacion-de-python-en-menos-de-una-semana-by-r-m-lewis-z-liborg/13284599> | |
| Rossum, G. (2017). *El tutorial de Python*. Python *Software* Foundation. | |