IU Digital de Antioquia

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

Desarrollo de contenido

Unidad 2
Algoritmos Codificados

Fundamentos de programación

Ingeniería mecatrónica

ORD No 74 de 2017 · VIGILADA MinEducación

www.iudigital.edu.co

Introducción a la Unidad 2

En la unidad dos de fundamentos de programación el estudiante desarrollará una estructura lógica de programación que le permitirá codificar algoritmos y diagramas de flujo mediante el uso de estructuras algorítmicas secuenciales, condicionales y cíclicas .Una vez el estudiante comprenda la temática, estará capacitado para proponer soluciones lógicas a las problemáticas identificando las entradas y salidas del proceso, los flujos y las estructuras de control necesarias para llevar a cabo dicho proceso. Por ende, el estudiante estará en capacidad de optimizar código, distribuir y segregar funciones utilizando los diferentes tipos de datos, operadores y controladores de flujo que forman parte de un algoritmo.

Objetivos de aprendizaje de la Unidad 2

- Comprender y utilizar los conceptos básicos informáticos en la codificación de aplicaciones.
- Solucionar problemas básicos de programación identificando correctamente las salidas y entradas del problema inicial.
- Interpretar un formato de pseudocódigo o diagrama de flujo para su posterior codificación.
- Realizar correctamente la codificación de un pseudocódigo o diagrama de flujo utilizando los diferentes tipos de datos, operadores y controladores de flujo que forman parte de un algoritmo.

Cronograma de actividades Unidad 2

Cronograma de actividades Unidad 2			
Actividad de aprendizaje*	**Evidencia de aprendizaje****	**Semana*****	**Ponderación**
AA2. Algoritmos codificados	EA2. Proyecto integrador parte dos.	Semana 5	25%
Total			**25%**

Unidad 2: Algoritmos codificados

Inicie el desarrollo de los temas de la Actividad de Aprendizaje

En esta Actividad de Aprendizaje abordarás los conceptos de variables, estructuras de control y selección, funciones, recursividad de orden superior. De esta manera tendrás los elementos necesarios para realizar codificación de algoritmos en el lenguaje de programación Python, para resolver los problemas presentados.

Los temas que estudiarás son los siguientes:

1. Entorno de desarrollo del lenguaje de programación
- Intérprete
- Instalación de Integrated Development Environment (IDE)
- Librerías de Python

2. Sintaxis del lenguaje
- (Datos, operadores, expresiones y estructuras de control, selección e iterativas)

3. Codificación de algoritmos en el lenguaje de programación y ejercicios prácticos

Al finalizar el estudio de los temas propuestos estarás en capacidad de trabajar en el proyecto integrador parte dos desde un IDE

1. ENTORNO DE DESARROLLO DEL LENGUAJE DE PROGRAMACIÓN

1.1. Intérpretes

Los entornos de desarrollo integrado o en inglés Integrated Development Environment (IDE) fueron desarrollados por la necesidad de interactuar directamente con un sistema operativo dado que estos no siempre podían soportar una terminal que realizará dicha tarea. Los IDE fueron diseñados para optimizar la productividad de los desarrolladores de software generando entornos de desarrollo con funcionalidades y componentes iguales que pudieran estandarizar el trabajo. En la mayoría de los casos los IDE tienen funcionalidades tales como autocompletado, depuración, creación, modificación y compilación.

Uno de los objetivos esenciales de los entornos integrados de desarrollos es darle la posibilidad al desarrollador de reducir las configuraciones necesarias para iniciar a codificar en un lenguaje de programación. En el siguiente video encontrarás información relevante sobre la definición de IDE o entorno de desarrollo integrado, esta te permitirá seleccionar una herramienta de trabajo para codificar.

Video

- **Autor:** Originpath Academy
- **Título:** Diccionario Coder - ¿Qué es un IDE?
- **URL:** https://www.youtube.com/watch?v=r9PqL-tw_QE

Sabías que el primer IDE fue creado por la empresa Softlab Múnich en 1975 y su nombre era “Maestro 1”.

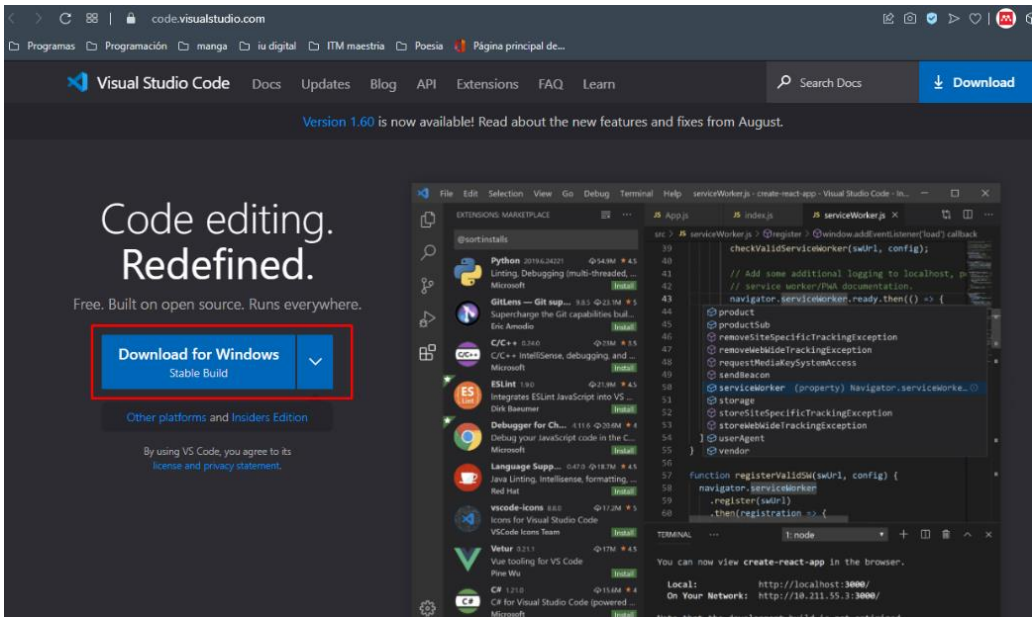
En el video anterior obtuviste conocimiento sobre la definición de entorno de desarrollo integrado, el cual será tu herramienta fundamental para la codificación de los programas que creas para dar solución a un problema específico.

1.2. Instalación del IDE

A continuación, mostraremos el paso a paso de la instalación de visual studio code:

1. Ingrese a la URL: https://code.visualstudio.com.
2. De clic en el botón de descarga como se muestra en la figura 1:

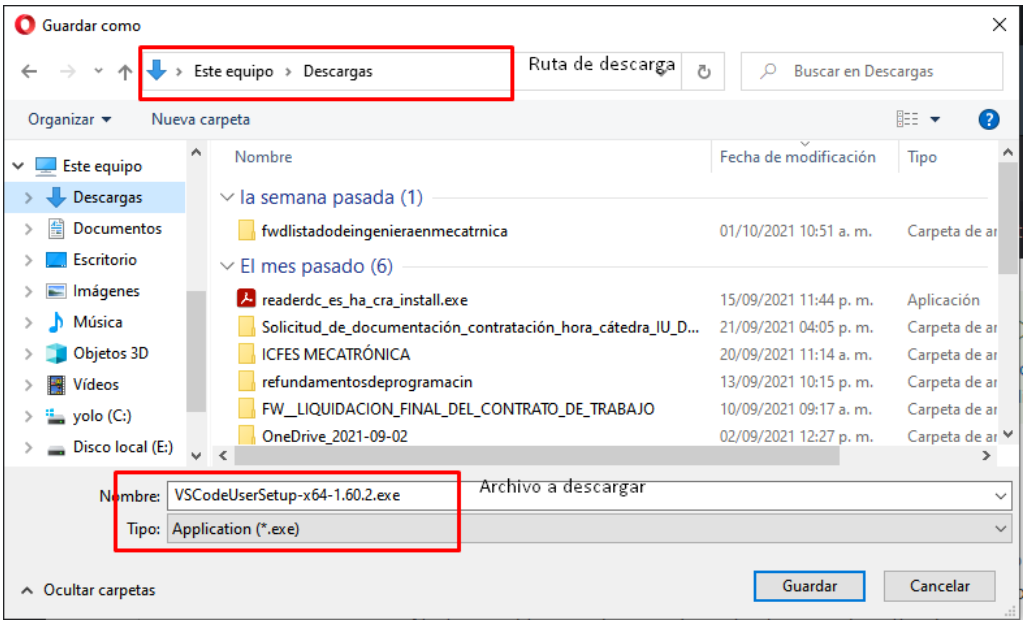
Figura 1. Descargar para Windows



Fuente: Elaboración propia.

3. Luego de dar clic, te mostrará un mensaje pidiéndote seleccionar la ubicación donde deseas descargar el archivo de instalación como se muestra en la figura 2:

Figura 2. Guardar como

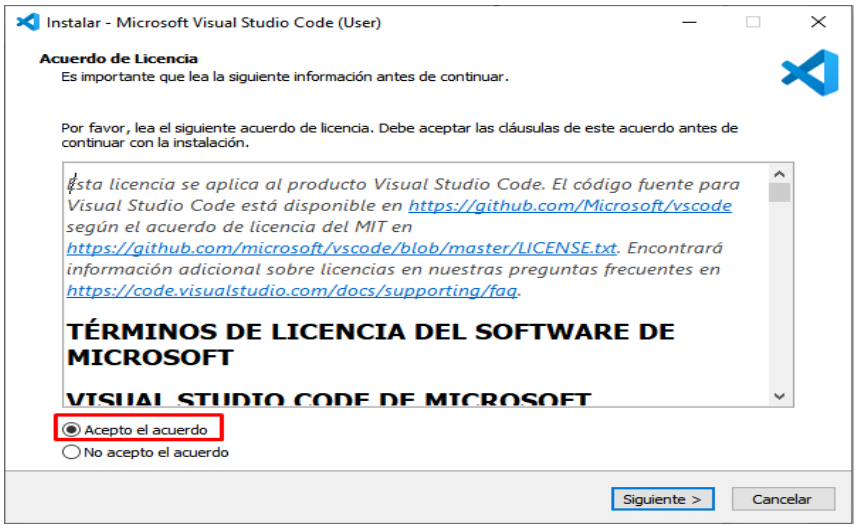


Fuente: Elaboración propia.

4. Selecciona el archivo de la ruta de descarga y ejecútalo.

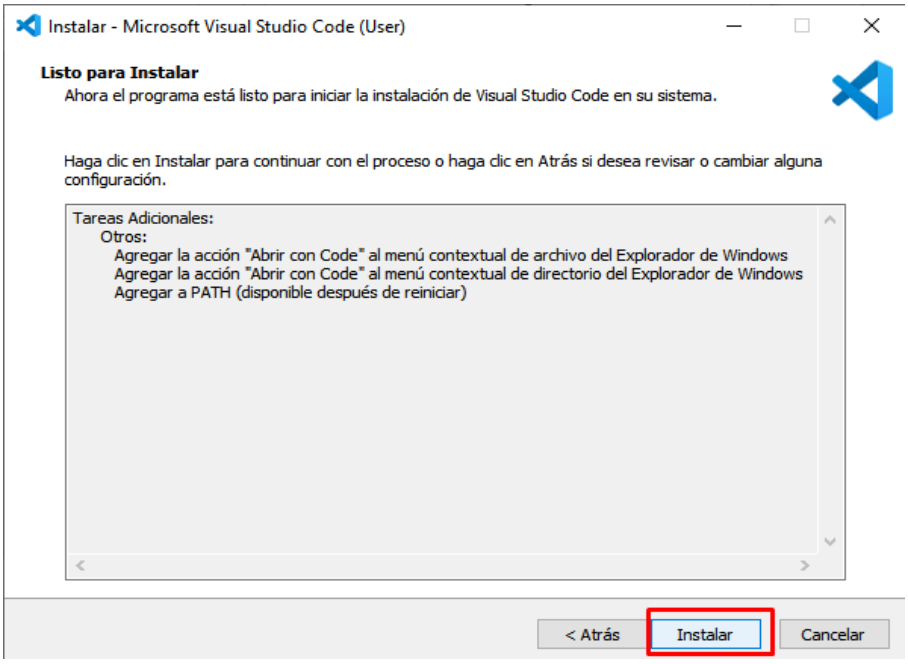
5. Selecciona las opciones subrayadas con rojo como se muestra en las figuras 3,4,5:

Figura 3. Aceptar el acuerdo



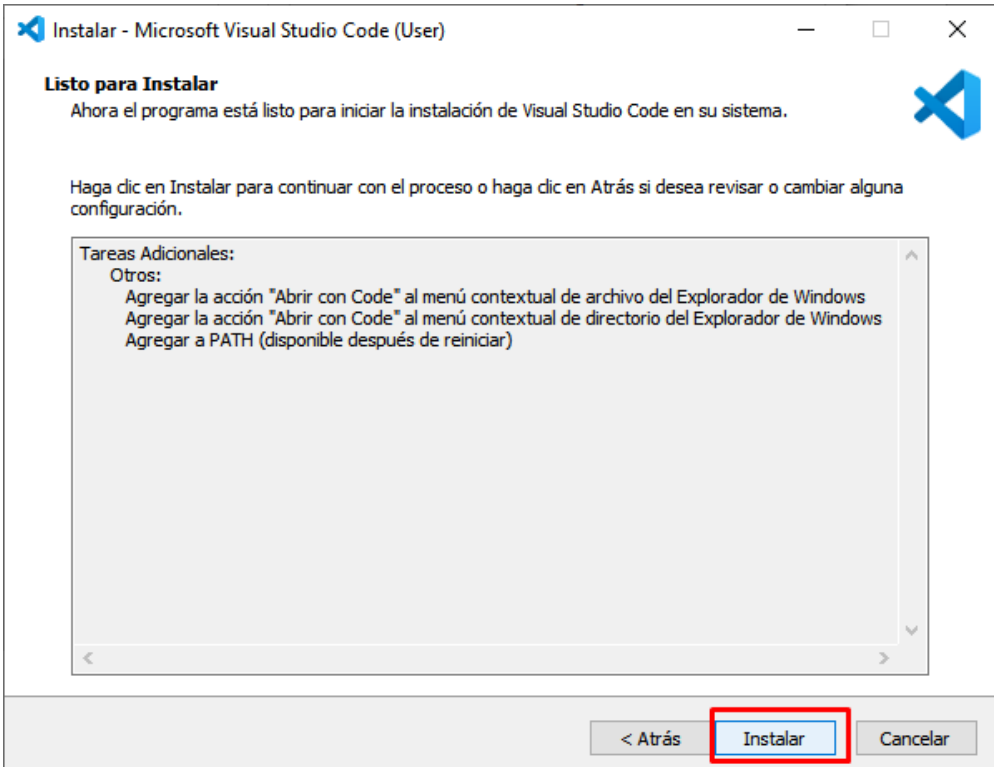
Fuente: Elaboración propia.

Figura 4. Instalar



Fuente: Elaboración propia.

Figura 5. Nuevamente instalar



Fuente: Elaboración propia

Para más información sobre la instalación del IDE remitirse al siguiente video:

Video

- **Autor:** Fazt
- **Título:** Visual Studio Code | Instalación de Visual Studio Code en Windows 10
- **URL:** https://www.youtube.com/watch?v=66r9Gqu7GWY

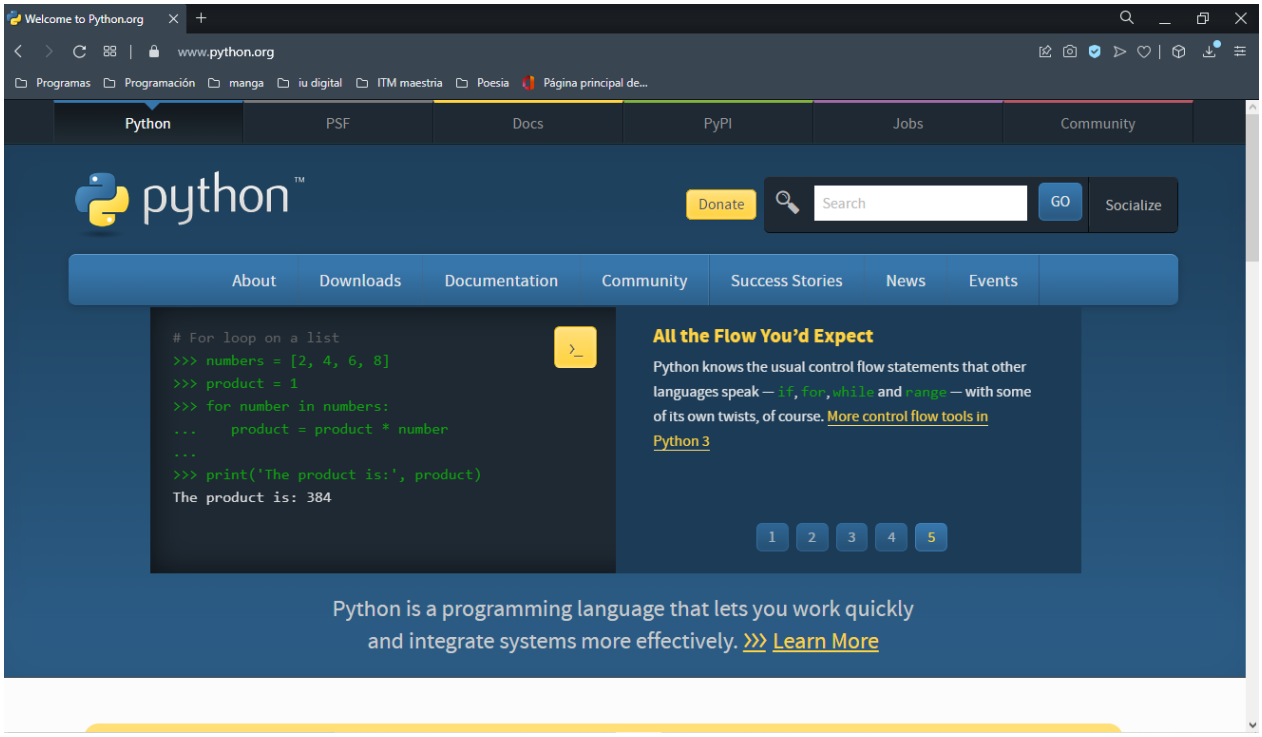
En el video anterior obtuvimos el conocimiento para realizar de la manera apropiada la instalación del software visual studio code que será el software principal en el que desarrollaremos el curso, esto nos será útil para la siguiente instalación de Python.

1.3. Instalación Python

Para realizar la instalación de Python lo primero que debemos hacer es ingresar desde el explorador de nuestra preferencia a la siguiente página web:

www.python.org como se muestra en la figura 6.

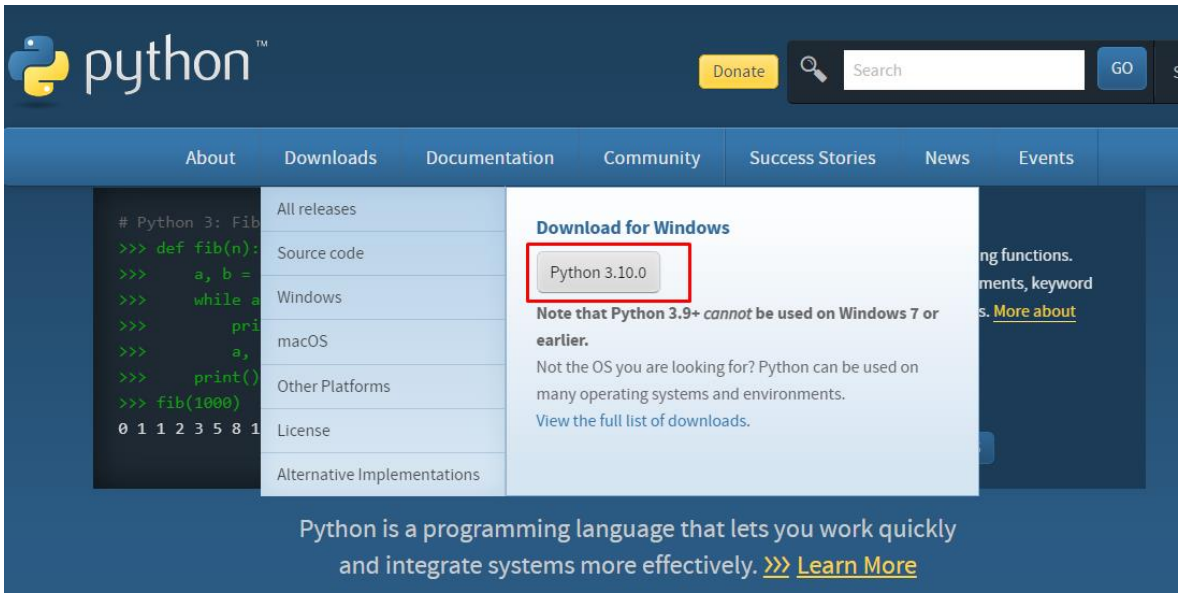
Figura 6. Interfaz de Phyton



Fuente: Elaboración propia.

Procedemos a colocar el clic sobre el botón “downloads” y le damos clic en el botón resaltado con el rectángulo rojo como se ve en la figura 7.

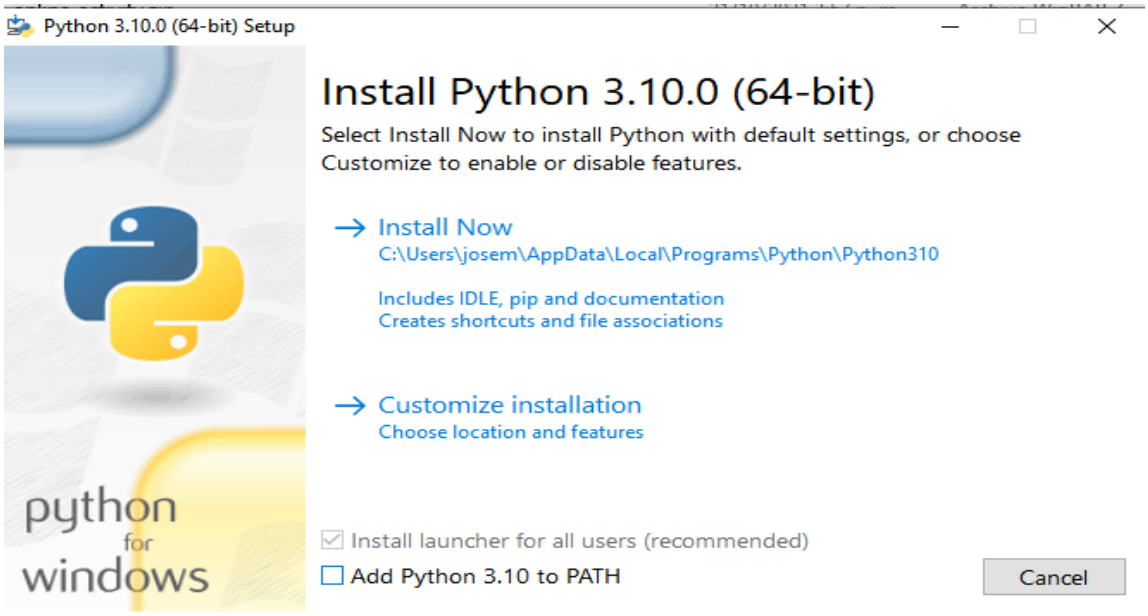
Figura 7. Descargar Phyton para Windows



Fuente: Elaboración propia.

Y posteriormente nos dirigiremos a nuestra carpeta de descargas y ejecutaremos el instalador dándole doble clic, y veremos el siguiente mensaje de la figura 8.

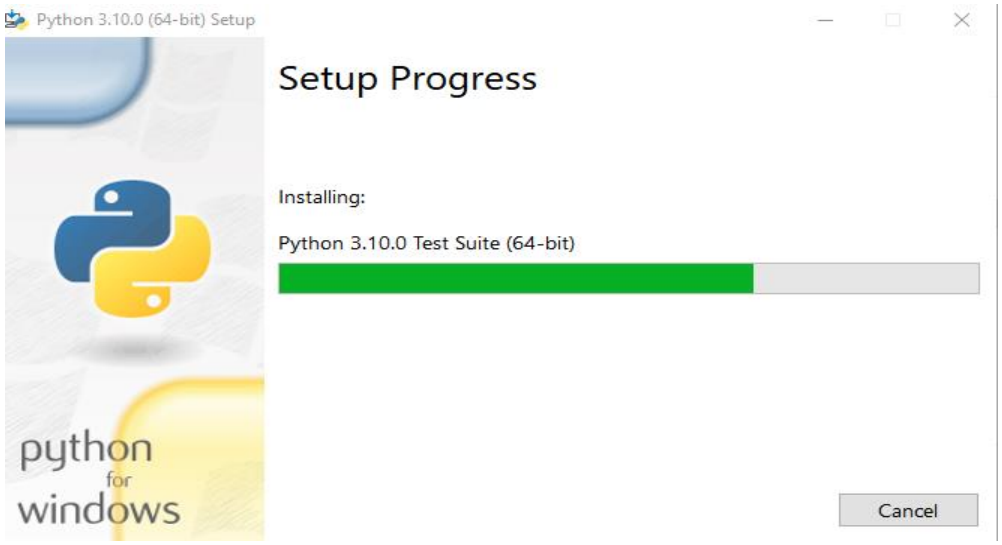
Figura 8. Instalar Phyton 3.10.0



Fuente: Elaboración propia.

Luego daremos clic en instalar ahora y visualizamos el mensaje de la figura 9.

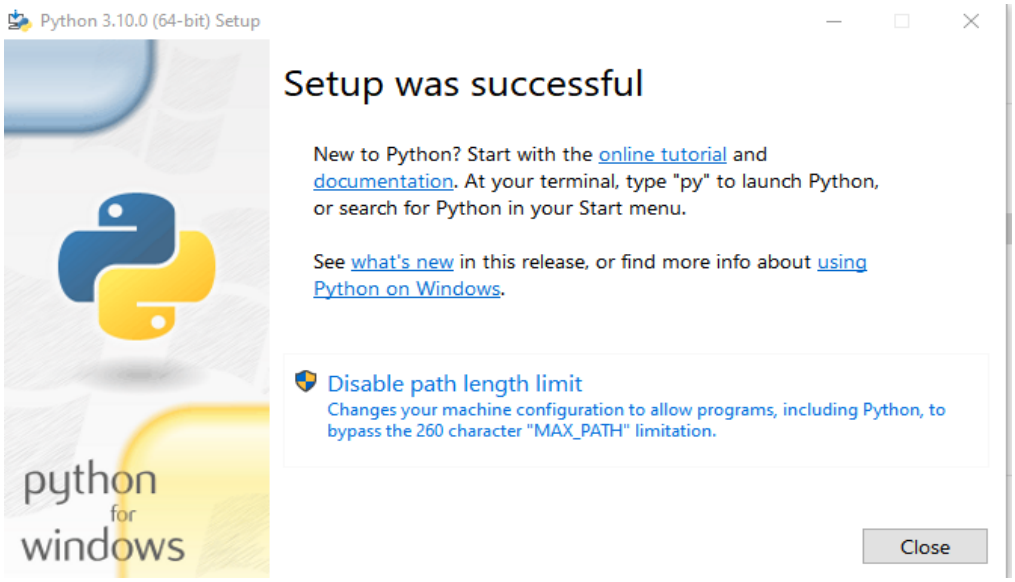
Figura 9. Setup progress



Fuente: Elaboración propia.

Finalmente, esperamos a que finalice el proceso de carga de archivos y veremos el siguiente mensaje de la figura 10 cuando el proceso haya finalizado.

Figura 10. Finalizar proceso de carga de archivos



Fuente: Elaboración propia.

Al llegar este punto el estudiante contará con los conocimientos esenciales para realizar la instalación del ide visual studio code y las librerías de python necesarias para poder utilizar dicho lenguaje.

En el siguiente video encontraremos un resumen de la configuración esencial de visual studio code y python con sus respectivas configuraciones, lo cual nos permitirá realizar la inicialización del entorno y la elaboración de programas codificados en dicho lenguaje.

Video

- **Autor:** Jose Mena Palomeque, Jaime Ramírez Ospina
- **Título:** Visual Studio Code | Instalación de Visual Studio Code en Windows 10 e instalacion de python 3
- **URL:** Incluir enlace cuando se suba al canal institucional. Ver diap 19

El vídeo da cuenta de la configuración de nuestro entorno y nos enseña a validar el correcto funcionamiento de python y de nuestro editor de código.

2. SINTAXIS DEL LENGUAJE

En nuestro día a día, tenemos que realizar múltiples tareas, y su ejecución parte de la decisión de priorizar y posterior planeación para ejecutar las cosas. La planeación implica también un orden, que se vuelve lógico y coherente en la medida que vamos conociendo los actos y las consecuencias asociadas a los mismos, con el fin de llegar a un objetivo concreto.

Vamos a mirarlo de otra manera:

Un mecánico automotriz, debe reparar un carro. Para esta reparación, cuenta con todas las herramientas necesarias: gato hidráulico, cruceta, juego de llaves, etc. No tiene sentido que el mecánico comience su reparación sin antes conocer cuál es el daño que presenta el vehículo. Las herramientas por sí solas, tampoco están en capacidad de realizar alguna acción. Conociendo la falla y teniendo los implementos necesarios, es la persona la que debe ejecutar la acción.

En este caso el daño es una llanta desinflada y debe montar el repuesto. El señor mecánico debe identificar cuál es la llanta para intervenir y luego hacer acciones en orden para lograr el objetivo de manera satisfactoria, sin sufrir accidentes. Ubicar el carro en una superficie plana, sin inclinación, accionar el freno de mano, aflojar un poco los pernos de anclaje de la llanta, sin quitarlos del todo, usar el gato hidráulico para levantar el carro, validar que el carro haya quedado a la altura necesaria, que le permita espacio para retirar la llanta. Terminar de quitar los pernos, bajar la llanta, sacar la llanta de repuesto del lugar donde está guardada, ubicarla, poner los pernos para sostenerla, accionar el gato para bajar el carro, apretar los pernos, recoger la herramienta, guardarla, etc.

Así, puedes darle el orden que consideres pertinente a la ejecución de la tarea, pero siempre conservando un orden lógico, en este caso, por ejemplo, no puedes retirar en su totalidad los pernos, sin antes haber levantado el carro, pues vas a incurrir en un accidente y la acción no se realizará de manera eficiente.

Como mencionamos al comienzo, aplica para todo en tú día a día, la planeación y orden lógico de las diferentes tareas.

De igual manera sucede con los programas que realizas en un computador. Este es un dispositivo, que es capaz de realizar una tarea, cuando de manera muy detallada y en un lenguaje que él entienda, le dices qué debe hacer.

Para realizar programas, debemos tener el mismo cuidado, conocer y definir el objetivo al cual queremos llegar y posterior comenzar a dar un orden secuencial y lógico, que nos permitirá ejecutar la acción específica que necesitamos.

Realizar dicha tarea, de manera clara y concreta, recibe el nombre de algoritmo, lo cual nos ayuda a trazar el camino que tomaremos para llegar a la ejecución de alguna acción específica.

Posteriormente, llevar ese algoritmo a un lenguaje que el computador comprenda, recibe el nombre de codificación. Es diferente al algoritmo como tal, ya que este no lo comprende el computador.

Seguramente en ocasiones hemos deseado tener la inteligencia que tiene un computador. A diferencia de lo que algunas personas creen, un computador no es inteligente. Solo tiene la capacidad de procesar información a alta velocidad. Cabe resaltar que el computador requiere de unas órdenes que él entienda, unas órdenes claras, lógicas, que finalmente lleven al objetivo que estamos buscando, como lo hablábamos anteriormente.

Las herramientas que se usan para hablarle al computador pueden llamarse también intérpretes y son aquellos que nos permiten lograr la comunicación e interacción con la máquina. Recordemos que, en este curso, el lenguaje de programación que vamos a emplear es Python.

En la unidad anterior, estuvimos analizando temas como los sistemas de numeración, los tipos de datos, operadores lógicos, operaciones lógicas, estructuras de control, condicionales, entre otros temas más. Es muy importante recordar y tener presentes los apuntes de la primera unidad.

En esta parte de tu curso, profundizaremos más en las instrucciones de programación, en funciones, para que logres mejores habilidades como programador y mayor versatilidad al momento de crear tus programas.

Para iniciar, es muy importante tener en cuenta, que todo el tiempo tenemos muchas cosas en la cabeza, temas familiares, laborales, personales, en fin, un mundo de cosas que están constantemente en nuestros pensamientos. Al momento de realizar un programa, debemos saber que la extensión del mismo es algo que no podemos controlar, cada programa tendrá la dimensión pertinente para cada necesidad, lo que nos dice que vamos a crear programas cortos y fáciles de entender al momento de analizarlos posteriormente, como también vamos a tener programas extensos, con múltiples funciones, lo que nos llevará a que el proceso para entenderlo o modificarlo, tiempo después de haberlo creado, sea más tedioso. Por lo anterior debes tener en cuenta siempre la documentación de tu programa, para tu futura comprensión o incluso el análisis de otro programador.

Esta documentación, que también podemos llamar comentarios, aparece solamente en el entorno de programación, y nos ayudan a documentar y explicar la razón de ser y el paso a paso de cada creación que hagamos.

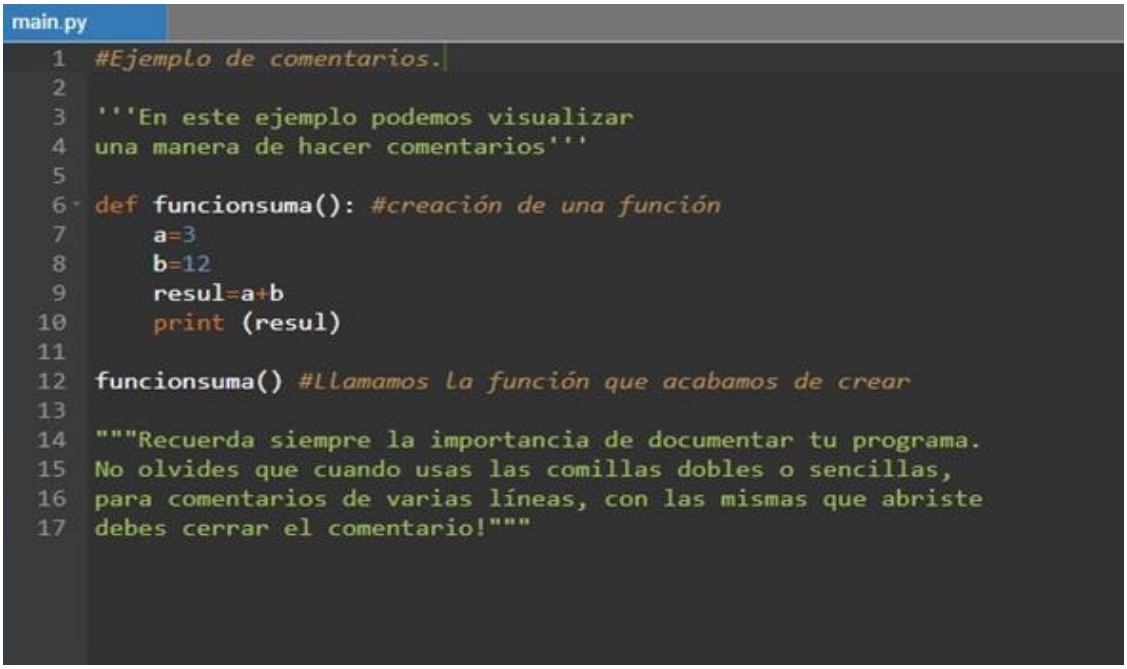
Para efectuar esta documentación, podemos hacerlo en una sola línea, o en varias líneas, teniendo para cada opción una manera de hacerlo.

Cuando vamos a hacer un comentario de una sola línea, abrimos el comentario con el signo *#,* en este caso no es necesario cerrar, pues deja de ser comentario al pasar a la siguiente línea de código.

Cuando vamos a hacer un comentario de varias líneas, usaremos tres veces ya sea la comilla doble, o comilla sencilla para abrir el comentario y tres veces la misma comilla para cerrarlo.

Los comentarios no tienen ninguna incidencia sobre la ejecución del programa. Miremos:

Figura 11.
Ejemplo de comentarios en Python.



Fuente: Elaboración propia.

3. CODIFICACIÓN Y SINTAXIS EN EL LENGUAJE DE PROGRAMACIÓN

En un contexto de programación, son varios los elementos que se deben tener en cuenta. Dentro de estos cabe resaltar la codificación y la sintaxis. La codificación consiste en traducir el algoritmo propiamente al lenguaje de programación y la sintaxis es la manera como se deben emplear y utilizar cada una de las instrucciones propias de cada lenguaje. Es importante conocer esto, para no incurrir en errores al momento de codificar nuestros programas.

Recordemos que los programas se hacen para dar soluciones a problemas específicos y durante su creación vemos la necesidad de ser amigables, claros y dar instrucciones a la persona que está en la ejecución de nuestro programa.

Nosotros lo conocemos, pero nosotros no seremos el usuario final. Por ende, es importante guiar paso a paso a cualquier persona que ejecute nuestro programa.

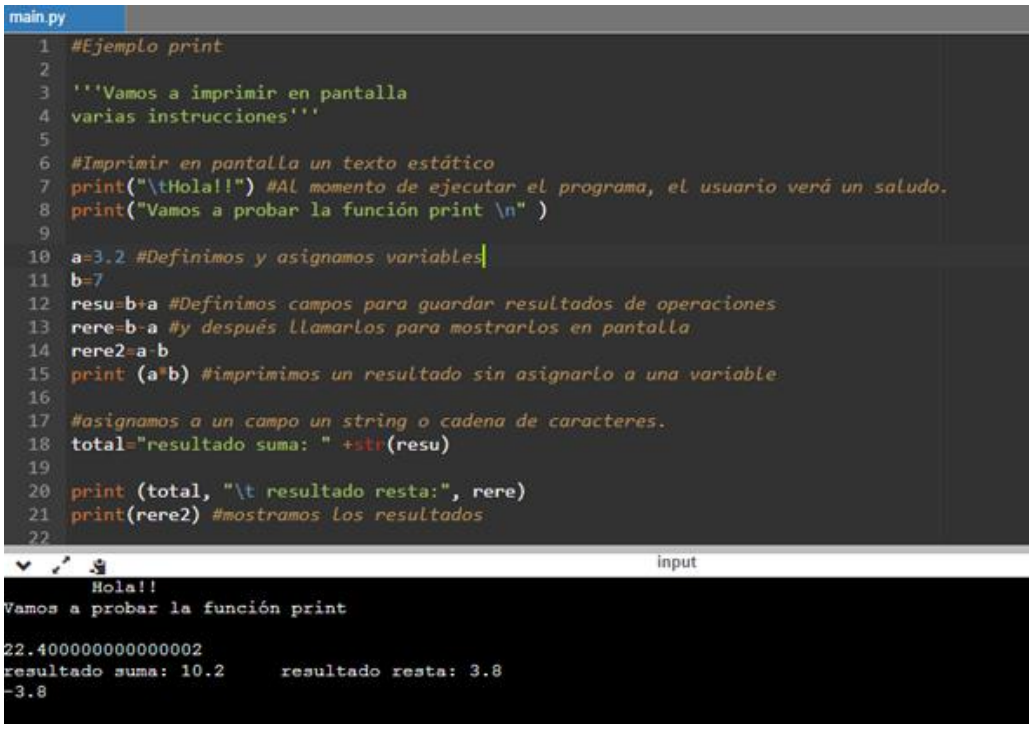
Dar instrucciones de uso. Así ayudamos a conseguir de mejor manera la solución al problema propuesto. De acuerdo con lo anterior, en algunas situaciones vamos a tener que mostrar un texto informativo o de otra índole a la persona que está interactuando con nuestro programa, por lo tanto, debemos imprimir o mostrar en pantalla un texto que supla lo que se requiere en el momento.

Para esto contamos con la función *print,* que como mencionamos anteriormente, se encarga de imprimir en pantalla todo aquello que debamos mostrar.

Recuerda que el entorno que tú ves como programador, no es el mismo que ve la persona que ejecute el programa.

Para usar la función *print,* y cualquier función, debes tener en cuenta su sintaxis, para evitar errores. Vamos a analizar un ejemplo:

Figura 12.
Ejemplo función print



Fuente: Elaboración propia.

En la figura 12, vemos diferentes formas de usar la función *print*, donde imprimimos solo texto, imprimimos una operación matemática, una cadena de caracteres, que incluye texto y dos campos variables y finalmente se imprime un campo, que anteriormente había sido asignado con el resultado de una operación matemática.

La sintaxis de una función es la manera cómo la usamos, según sean sus parámetros de entrada y de salida.

La función *print*, por ejemplo, la usamos así:

Print *("lo que vamos a mostrar, cuando es una cadena de caracteres o texto")*
Print *(campo), cuando vamos a mostrar un valor o un dato que está almacenado en un campo, previamente definido.*

En la parte inferior de la imagen, vemos el resultado de la ejecución del programa. Podemos ver que el texto inicial que dice "¡Hola!!", está desplazado hacia la derecha, y hay un salto de línea entre el texto "Vamos a probar la función print" y el resultado de la multiplicación "22.4". Esto lo logramos gracias a los apuntadores **\t** y **\n.** Donde \t funciona como un tab y \n funciona como un salto de línea, estas acciones nos sirven para dar orden a la pantalla con la cual interactúa el usuario.

Al momento de emplear ***print,*** y necesitar las instrucciones vistas para dar orden a la pantalla de usuario, lo hacemos dentro de las comillas propias de la sintaxis de la función ***print***.

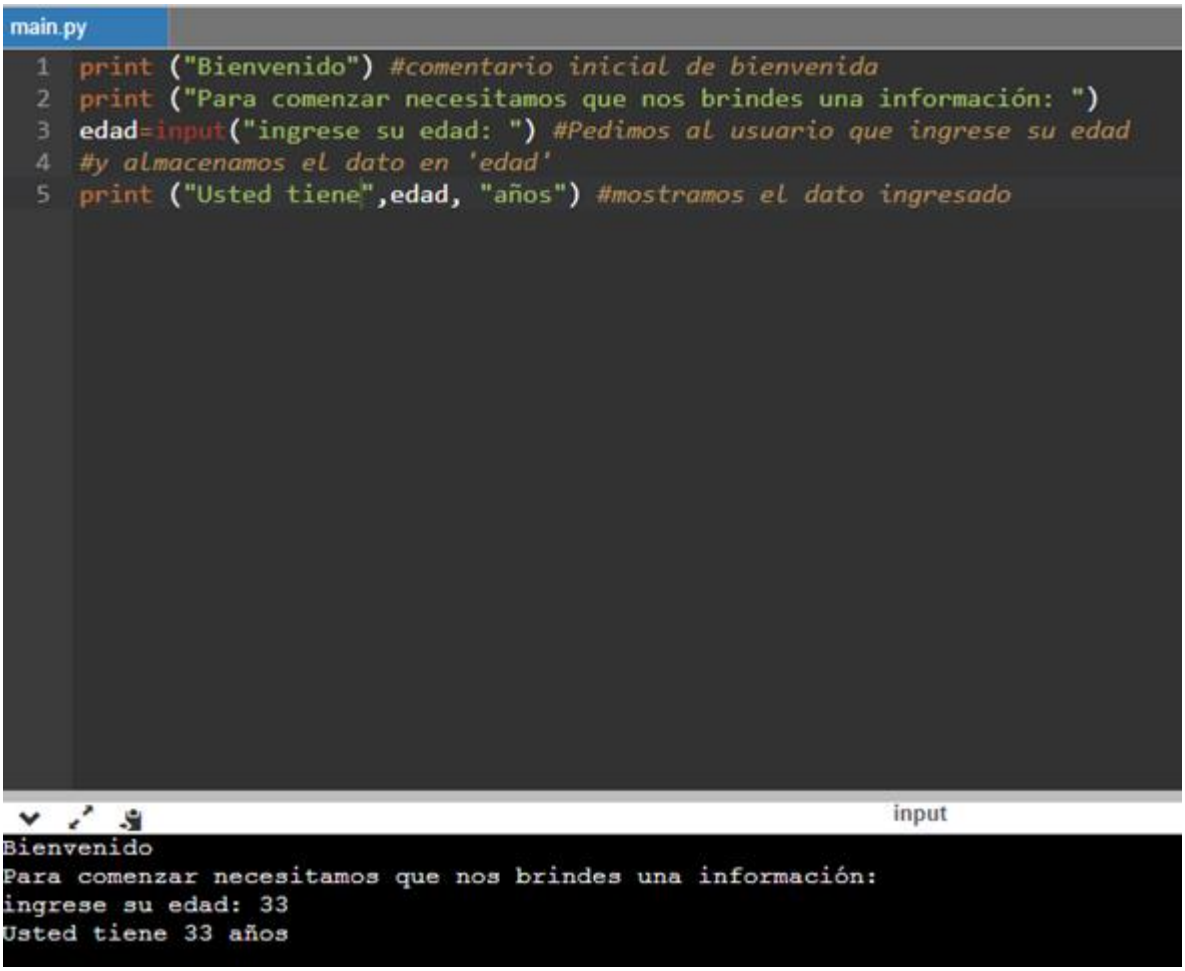
Durante esta interacción con el usuario de tu programa, en la cual lo vas llevando y le vas explicando qué debe hacer, para encontrar el uso y el sentido de tu elaboración, también llegará el momento en que este usuario, debe ingresar un dato, por ejemplo, su edad, debe seleccionar una opción de un menú, algo en lo que deba introducir un dato mediante el teclado. En este caso, deben suceder dos cosas.

Primero se solicita que ingrese un dato y el usuario lo ingresa. Segundo, el programador debe tomar o leer ese dato, para poder realizar la acción siguiente.

Cuando en programación decimos leer un dato, hacemos referencia a coger el dato y guardarlo en un campo determinado, para su posterior uso.

En la figura 13, vamos a ver un ejemplo, donde recordamos la documentación o los comentarios del programa, saludamos al usuario y además le pedimos que ingrese su edad. Posterior, mostramos el dato que ingresó, acompañado de un texto.

Figura 13.
Uso de la función input



Fuente: Elaboración propia.

Como mencionamos anteriormente, esta función es importante para almacenar un dato ingresado y usarlo luego para realizar alguna acción. El usuario, luego al digitar el dato, debe presionar la tecla ***enter***.

Teniendo en cuenta los temas vistos anteriormente, podemos entrar a elaborar programas un poco más complejos, que incluyan condicionales, y otro tipo de instrucciones que nos lleven a ir creciendo en nuestras habilidades y destrezas como programadores.

A continuación, vamos a ver el condicional ***if,*** el cual podemos analizar de la siguiente manera:

Si (x > y)
Imprima (x es mayor que y)
Si no imprima (x No es mayor que y)

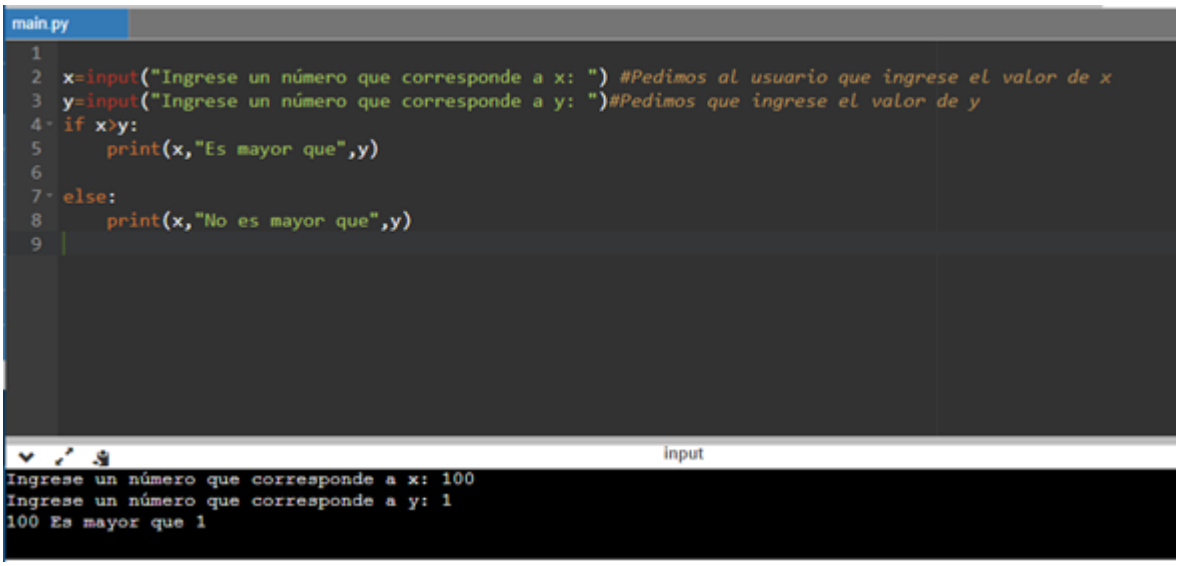
En el algoritmo anterior, tenemos una condición x > y.

En el programa que vamos a construir, x, y, son valores ingresados por el usuario. Nuestra función es averiguar si x es mayor que y. Si esta condición se cumple, o sea que es verdadera, entra al condicional y va a mostrar en pantalla un mensaje que muestra el valor de x y el de y, acompañado de un mensaje que afirma que x es mayor que y.

Si esta condición no se cumple, o sea que es falsa, no entra al interior del condicional, sino que salta a la línea siguiente, donde nos encontramos una afirmación que nos dice que x no es mayor que y.

En la figura 14, podemos ver de manera codificada lo que acabamos de enunciar. Miremos:

Figura 14.
Uso del condicional *if*



Fuente: Elaboración propia.

Recuerda siempre tener en cuenta la sintaxis de las funciones que vayas a utilizar.

Al finalizar las instrucciones de la línea principal del *if,* vemos que hay elementos nuevos como los dos puntos. Estos nos indican que se acaba la línea principal de la condición y pasamos a estructurar las acciones que debe realizar nuestro programa, siempre y cuando la condición sea positiva o verdadera. Cuando esta condición es falsa sigue derecho.

El *if* en programación, es comparable con la toma de decisiones en la vida real, donde cada decisión tiene una consecuencia, y estas consecuencias se viven siempre y cuando hayamos tomado la decisión que nos lleva a ellas.

¿Sabías qué?

En Python todos los valores diferentes de 0 y las cadenas de caracteres no vacías, se consideran Verdaderos.
Lo único Falso es 0.

Los programas que vamos creando, se van haciendo más complejos en la medida que empleamos nuevas herramientas y funciones de programación. No necesariamente un programa largo es un programa complejo y bien estructurado.

En ocasiones nos podemos encontrar con programas muy extensos, con un costo computacional alto, que no sean complejos, puede ser una constante repetición de instrucciones, lo que hace que la construcción sea ineficiente y de alguna manera desordenada.

¿Sabías qué?

Las funciones o subprogramas son fragmentos de código que puedes llamar y utilizar en varias ocasiones y en diferentes partes de tu programa, con el fin de hacerlo más eficiente.

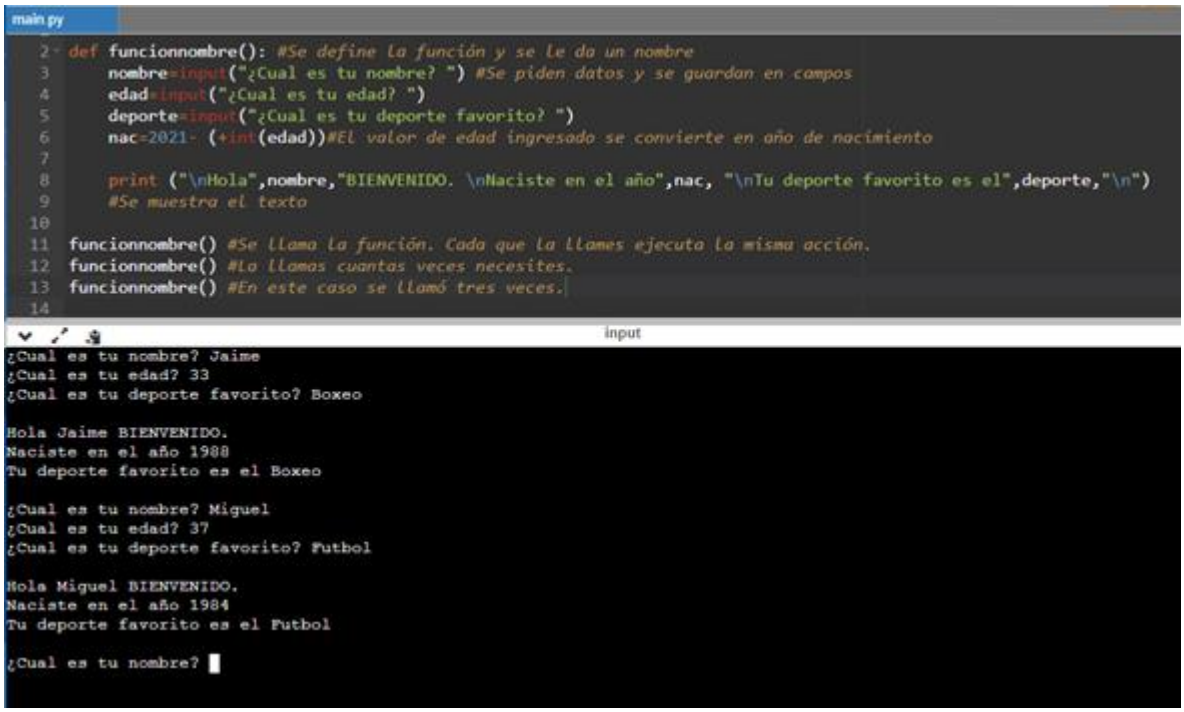
Por lo anterior, surge la necesidad de crear funciones. Por ejemplo, cuando vas a usar la función *print*, no tienes que escribir toda la subrutina que hizo el creador de la función, sino simplemente llamarla y usarla teniendo en cuenta su sintaxis.

Analicemos el siguiente ejemplo:

Necesitamos conocer el nombre, la edad y el deporte favorito de un grupo de personas. Para no hacer un código diferente para cada persona, donde le pidamos que ingrese los datos que requerimos, diseñamos una función local que pida el nombre y lo almacene, pida la edad y la convierta en el año de nacimiento y pida que mencione el deporte favorito, para finalizar, muestra en un texto todo lo anterior, además le da la bienvenida a la persona.

En la figura 15, observamos una solución para la ejecución de esta acción mediante el uso de una función o subprograma creados por nosotros.

Figura 15.
Ejemplo de función en Python.



Fuente: Elaboración propia.

Una función o subprograma, debe comenzar a definirse primero con la palabra *def,* luego el nombre de la función y por último en paréntesis los parámetros. Estas son las variables

en las funciones y puede tener valores diferentes, de acuerdo a lo que necesites cada que llames tu función. Cada parámetro debe ser separado por coma.

Luego de esto, en la línea siguiente de código, se empieza a estructurar el cuerpo de la función, nos damos cuenta de que estamos escribiendo dentro de nuestra función, porque cada línea va con una sangría diferente a la primera, donde definiste tu función. Dentro del cuerpo de la función, puedes usar cualquiera de las instrucciones que vimos, e incluso muchas más, combinarlas y así crear subprogramas cada vez más eficientes.

Los datos y las variables que están dentro de la función son propios de ella, por lo tanto, no puedes usarlos fuera de la misma. Seguramente será una necesidad a la que vas a llegar. Usar datos y resultados que están dentro de la función.

¿Cómo lo hacemos?

Para mostrar en la pantalla del usuario usamos la instrucción ***print,*** la cual solo imprime en pantalla los resultados, pero no realiza ningún movimiento con los datos, ni realiza ningún tipo de asignación.

Para suplir esta necesidad y poder usar una variable interna, propia de la función, fuera de ella, lo hacemos usando la instrucción ***return.***

Por favor, debes observar y analizar el siguiente video, que nos mostrará el tema de la sangría en las funciones, la diferencia y uso de la instrucción ***print,*** la instrucción ***return,*** el ciclo ***for,*** que vimos en la unidad 1.

Video

- **Autor:** ChelinTutorials
- **Título:** Programación Python 3: Return vs Print y ciclo definido for
- **URL:** https://www.youtube.com/watch?v=4FYHeA0HEFs

Revisando de nuevo el tema de las funciones, y hablando específicamente de nuestro ejemplo visto en la Figura 15, vemos que, en la parte inferior de la imagen, aparece la ejecución del programa, nuestra función se ejecuta tres veces. Resolvemos dos y una más queda lista para diligenciar los datos.

Los programadores debemos adelantarnos a una mala escritura o un error en la ejecución o diligenciamiento de los datos que debe ingresar el usuario. Por lo tanto, siempre debemos cuidarnos de esto.

Por ejemplo: estás llenando un formulario en una página de internet y hay uno de los espacios que te pide que ingreses tu edad, y tu número fijo de teléfono. Ambos espacios deben recibir únicamente datos numéricos, pues la edad y el teléfono te los piden así.

Te equivocaste y en el espacio del número fijo, pusiste tu nombre. Teniendo en cuenta que esto puede pasar, el programador desde un principio tuvo que diseñar una solución para evitarlo y se asegura que en cada espacio que el usuario debe diligenciar, digite la información que debe ser. Por ejemplo, en el espacio de la edad, que solo se puedan ingresar datos numéricos, coherentes con la edad de una persona, de lo contrario sale un mensaje de advertencia, diciendo que debes corregir esa información. Nosotros también podemos hacerlo.

Actividad

Se le propone al estudiante que para poner en práctica los conocimientos adquiridos, digite el programa de la Figura 15 y adicional, haga una función que valide que el dato ingresado para la edad, si sea un número y este en el rango de 7 a 99 años.

Si no cumple con estas dos condiciones, debe mostrar un mensaje diciendo:

El dato ingresado debe ser un número entre 7 y 99 años.

En un programa puedes crear y llamar cuantas funciones consideres necesarias., lo importante es tener en cuenta la sintaxis, los parámetros que necesitas y el modo de uso de las variables, ya sea un manejo interno o propio de la función (local), o un manejo más global, o sea que podamos usar y manipular los valores que arroja la función, en cualquier punto del programa, fuera de la función.

Recuerda que en la medida que más practiques, vas a mejorar notablemente tus habilidades en programación, que serán necesarias a lo largo del desarrollo de tu carrera, como en el ejercicio de tu vida profesional.

La sintaxis de programación puede variar un poco de intérprete a intérprete, incluso en dispositivos programables como microcontroladores o autómatas (PLC).

La clave fundamental para abordar nuevos desafíos en programación, independiente del software o del dispositivo, es crear, mejorar y fortalecer tu lógica de programación.

Para lograr esta habilidad, debes partir de un análisis, que te llevará a conocer el problema, delimitarlo de acuerdo a su utilidad y requerimientos, y de esta manera puedes definir el camino que vas a tomar, para llegar al objetivo.

Como sabemos, para volvernos expertos, debemos practicar, seguramente equivocarnos, mejoramos el aprendizaje y así llegaremos a una aplicación y uso final.

Todas las herramientas que hemos estudiado en este curso serán de gran utilidad para el desarrollo de tus habilidades y tus competencias como programador.

Ya vimos el condicional, que nos sirve para ejecutar una acción específica en nuestro programa, siempre y cuando se cumpla una condición que nosotros definamos.

Por ejemplo, pedimos al usuario que ingrese dos números, en nuestro programa debemos definir si son pares o impares. Leemos ambos números y damos paso a nuestra condición. Un número puede ser par o impar, en este caso no hay otra opción.

Evaluamos si el primer número ingresado es par, en caso de que si, se cumple una primera condición que entrará al ***if,*** y ejecutará la acción de mostrar en pantalla un mensaje que indique que el primer número es par.

En caso de no ser par, será impar. Al evaluar los dos números, se mostrarán los mensajes correspondientes y la ejecución del programa ha finalizado.

Esto también lo conocemos como estructuras de control condicionales.

Para pensar

Supongamos que estamos en el proceso de creación de un perfil en una red social. En medio de este proceso, llegamos a un punto donde nos piden ingresar nuestro correo electrónico. Acto seguido nos piden validar o escribir de nuevo el correo. Ambos deben coincidir, de lo contrario no podemos avanzar en la creación de nuestro perfil.

En este caso los desarrolladores de la aplicación para inscripción y creación de nuevos perfiles tuvieron que crear dos estructuras, donde se almacenan ambos correos, luego los comparan y si son iguales, o sea que se cumple la condición, nos permite continuar, de lo contrario no.

A diferencia de lo que pasaría en la estructura condicional, ahí finalizaría el programa, pero en este caso, para poder continuar, necesitamos que coincidan ambos correos. Esa es nuestra condición. En ese orden de ideas, entra un concepto nuevo que son estructuras de control iterativas, que pueden compararse con ciclos, donde el programa se quedará preguntando si ya se cumple la condición, si no se cumple, seguramente nos mostrará un mensaje que diga que los correos ingresados no son iguales, y se quedará preguntado hasta que sean iguales.

Cuando decimos que se queda preguntando, significa que se quedará de manera cíclica validando la condición y cuando esta sea verdadera, procede a realizar una acción diferente, mientras no lo sea, se quedará ahí hasta que ocurra lo contrario.

Estructuras de control iterativas

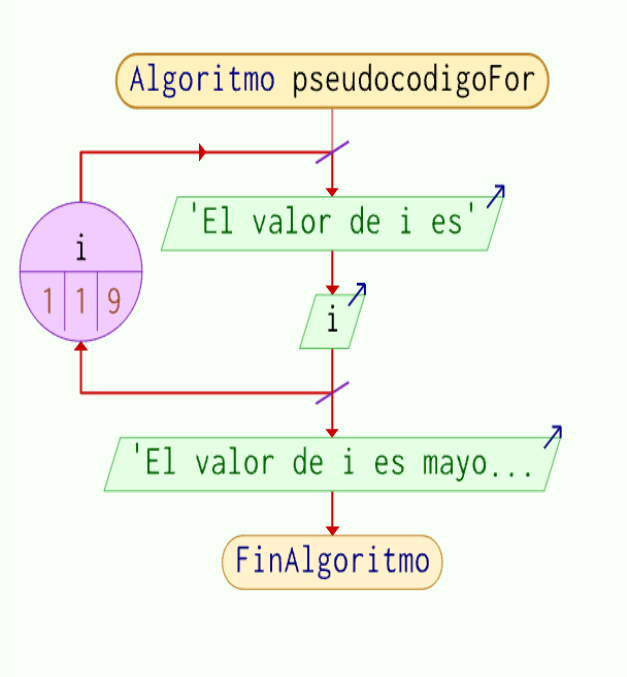
Las estructuras de control iterativas también llamadas bucles o cíclicas, son aquellas que nos permiten ejecutar el mismo código varias veces, siempre y cuando se cumpla una condición especifica.

En este recurso abordaremos las siguientes estructuras cíclicas:

- Estructura While
- Estructura For

Bucle For

La estructura For es un bucle que nos permite ejecutar una seguidilla de instrucciones un numero predeterminado de veces como se muestra en el siguiente seudocódigo y diagrama.




```
1  Algoritmo pseudocodigoFor
2      Para i←1 Hasta 9 Con Paso 1 Hacer
3          Imprimir "El valor de i es"
4          Imprimir i
5      FinPara
6      Imprimir "El valor de i es mayor o igual a 9"
7  FinAlgoritmo
```

¿Cuándo se utiliza?

El bucle For se utiliza cuando sabemos cuántas veces se va iterar en el ciclo en Python contamos con los siguientes objetos iterables:

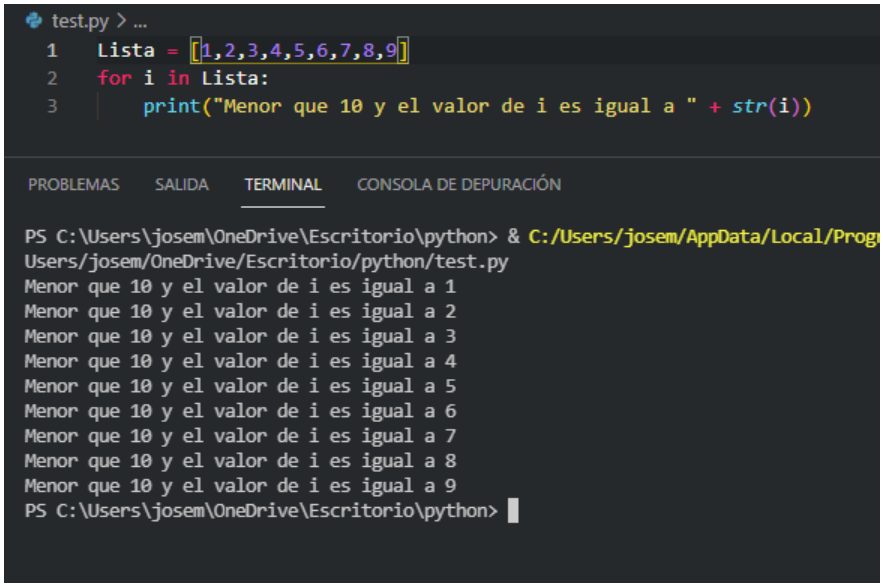
- Duplas
- Diccionarios
- String
- Listas

La sintaxis para utilizar el ciclo For es la siguiente:
For ValorORangoDeLaIteracion in Lista o Rango :

Instrucciones
Instrucciones

Ejemplos del bucle for:

Realice la codificación de un algoritmo que imprima los números de un rango de 0 a 10 hasta que el número actual del rango sea menor a 10.



Bucle While

La estructura While es un bucle que nos permite ejecutar una seguidilla de instrucciones mientras se cumpla una condición.

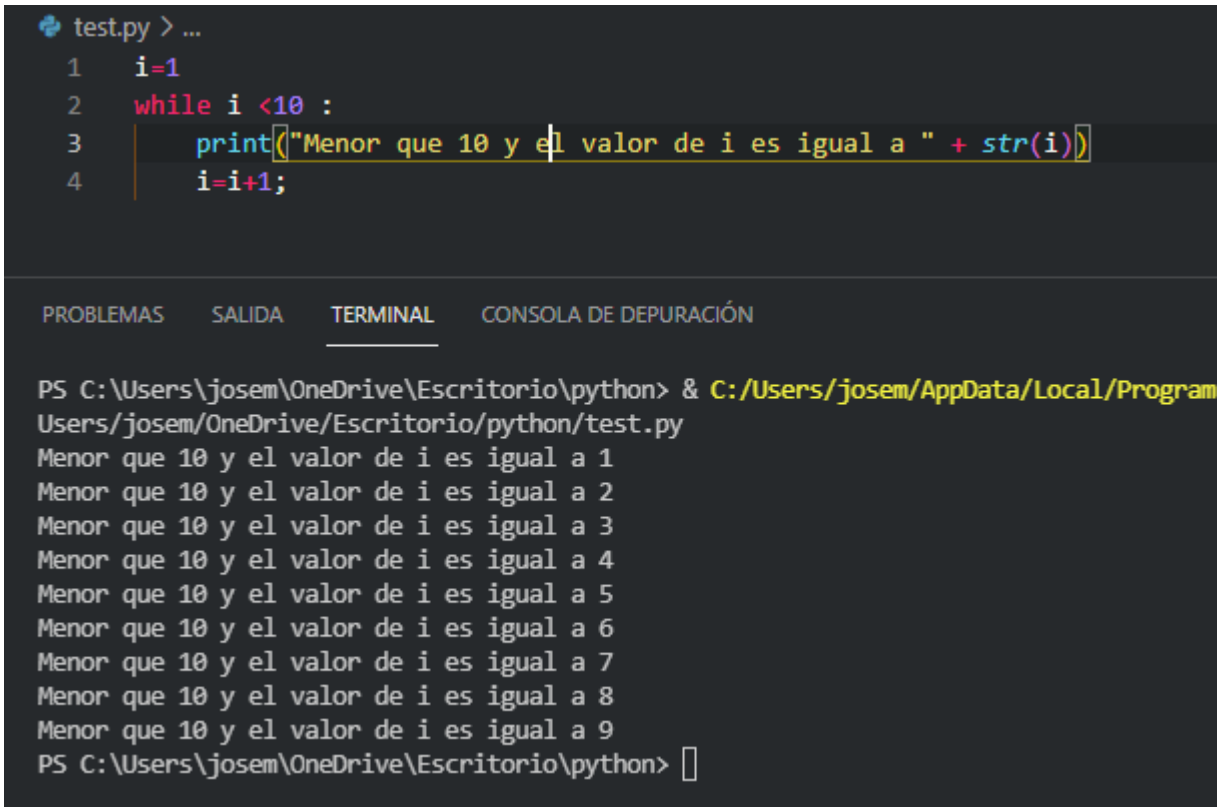
La sintaxis para utilizar el ciclo While es la siguiente:

While condición:

instrucción
instrucción

Ejemplos código while

Realice la codificación de un algoritmo que imprima los números de un rango de 0 a 10 hasta que el número actual del rango sea menor a 10.



En la codificación de algoritmos usando estructuras iterativas se debe tener en cuenta los siguientes aspectos:

En resumen, hemos aprendido que para realizar la codificación de algoritmos utilizando ciclos iterativos se debe tener en cuenta la correcta escritura de la sintaxis de cada ciclo evaluar si requerimos condicionar con base a un rango predeterminado de datos o con base a una condición.

Esto nos permitirá selección la estructura de control adecuada para cada caso de codificación.

Importante

Las estructuras de control iterativas nos ayudan a crear ciclos definidos, que nos sirven para evaluar y recorrer tuplas, listas, validar que se cumpla la condición que necesitamos y no continuar hasta que así sea y dar diferentes órdenes e instrucciones a nuestros programas.

Dentro del desarrollo de las habilidades de programación, te darás cuenta de que una manera amigable con la que puedes interactuar con los usuarios es mediante la creación y el uso de interfaz de usuario, mediante la cual puedes guiar a la persona que interactúe con tu programa a través de él.

Python ofrece una muy buena solución para la creación de la interfaz de usuario que requieras, según sea tu necesidad.

Recuerda que siempre debemos ser lo más claros que se pueda y amigables, para que el usuario sienta las ganas de continuar en nuestro programa y lo prefiera, aunque existan otros que también puedan suplir la necesidad.

A continuación, vamos a ver los videos 42 y 43 de Pildorasinformaticas, los cuales de manera muy detallada ayudarán a descubrir y darán luz para el uso de esta herramienta y así poder continuar con la creación de conocimiento en programación.

Esto nunca para, pero vamos por buen camino.

Video

- **Autor:** Pildorasinformaticas
- **Título:** Curso Python. Interfaces gráficas I. Video 42 y video 43
- **URL:** https://www.youtube.com/watch?v=hTUJC8HsC2I

Vamos a practicar la creación de una interfaz gráfica.

- **Ejercicios de prácticos**

- Hacer un programa que le pida al usuario que ingrese dos números. Luego de ingresarlos, muestra la opción de multiplicarlos, sumarlos, restarlos o dividirlos y mostrar el resultado correspondiente a la opción seleccionada.
- Vamos a tener en cuenta que debemos hacer una función que se llame validarnumero (), que retorne un mensaje de error si el dato ingresado no es un número.
- Hacer un programa que muestre en pantalla la portada para entregar un trabajo.
- El nombre del trabajo será: Portada unidad 2.
- Debe incluir el nombre del trabajo, el nombre del estudiante que la realiza, nombre del docente, nombre de la asignatura, institución, facultad, programa, semestre y año.
- La información debe estar bien distribuida, la portada debe ser estética.
- Hacer un programa que muestre las edades, los nombres y el promedio de edades de los estudiantes de la asignatura de fundamentos de programación, incluyendo el docente.

Ejercicios guiados

En los siguientes vídeos realizaremos la codificación de dos problemas algorítmicos con el objetivo de desarrollar y practicar nuestra lógica, en ellos repasamos los conceptos de ciclos y operadores lógicos.

Video

- **Autor:** Jose Mena Palomeque, Jaime Ramírez Ospina
- **Título:** Ejercicio 1: ciclo definido for
- **URL:** **Incluir enlace cuando se suba al canal institucional. Ver diap 49**

Video

- **Autor:** Jose Mena Palomeque, Jaime Ramírez Ospina
- **Título:** Ejercicio 2: Ejercicio práctico invertir números
- **URL:** **Incluir enlace cuando se suba al canal institucional. Ver diap 49**

En los videos anteriores practicamos conceptos básicos de estructuras, ciclos y operadores lógicos que nos permitirán avanzar de una manera más eficiente en la elaboración, gestión y solución de problemas algorítmicos, construyendo inicialmente un algoritmo, generando su diagrama de flujo y elaborando su codificación final.

Para finalizar, revisemos el siguiente recurso, encontrarás información sobre la manera correcta de codificar una solución algorítmica y los posibles errores que te encontrarás en el camino, esto te será útil a la hora de realizar modificaciones en las estructuras de tus flujos y procesos, y te dará las bases para obtener una noción básica de buenas prácticas de programación.

Para aprender más

Si deseas conocer más sobre la comunicación en medios virtuales, lee el siguiente material:

- **Libro electrónico**: Breve manual de programación en Python.
- **Sección a consultar:** Buenas prácticas de programación y errores. De la página 10 a la 12
- **URL:** https://www.fing.edu.uy/~darosa/manualFinal.pdf

Luego de realizar la lectura del texto nombrado anteriormente, obtuvimos el conocimiento necesario para realizar modificaciones a nuestro código teniendo en cuenta ciertas nociones básicas de buenas prácticas de programación y para la corrección de errores comunes en nuestro código.

Cierre

Una vez terminada la unidad el estudiante estará en capacidad de codificar pseudocódigos y diagramas de flujo, entenderá por qué se da el uso de los entornos de desarrollo integrado, y conocerá la manera apropiada de instalar librerías de un lenguaje de programación en este caso python, lo que le permitirá realizar codificaciones un poco más complejas en la siguiente unidad.

Lecturas
y Material Complementario

Lecturas:

- **Título:** Breve manual de programación en Python
- **Autor:**
- **Sección para consultar:** Buenas prácticas de programación y errores. De la página 10 a la 12
- **URL:** https://www.fing.edu.uy/~darosa/manualFinal.pdf

Videos

- **Autor:** Originpath Academy
- **Título:** Diccionario Coder - ¿Qué es un IDE?
- **URL:** https://www.youtube.com/watch?v=r9PqL-tw_QE

- **Autor:** Fazt
- **Título:** Visual Studio Code | Instalación de Visual Studio Code en Windows 10
- **URL:** https://www.youtube.com/watch?v=66r9Gqu7GWY

- **Autor:** ChelinTutorials
- **Título:** Programación Python 3: Return vs Print y ciclo definido for
- **URL:** https://www.youtube.com/watch?v=4FYHeA0HEFs

- **Autor:** Pildorasinformaticas
- **Título:** Curso Python. Interfaces gráficas I. Video 42 y video 43
- **URL:** https://www.youtube.com/watch?v=hTUJC8HsC2I

Bibliografía

- Cadavid, S. R. (2016). Manejo de operadores en programación. Colombia aprende. Recuperado 25 de octubre de 2021, de https://aprende.colombiaaprende.edu.co/stes/default/files/naspublic/curriculos_ex/n1g10_fproy/nivel1/programacion/unidad3/leccion1.html

IU Digital de Antioquia

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

ORD No 74 de 2017 · VIGILADA MinEducación

www.iudigital.edu.co