

**Este curso se asienta sobre la premisa de que los participantes han adquirido previamente conocimientos fundamentales en Python. Su objetivo radica en consolidar prácticas avanzadas en los siguientes conceptos:**

**En Python, la conceptualización de objetos atraviesa todas las instancias. En Python todo son objetos.**

1. **Las variables son objetos que actúan como elementos para el resguardo de información. Otros objetos incluyen las funciones integradas (built-in) y las definidas por el usuario, que son bloques de código reutilizables, así como los operadores, que realizan operaciones en objetos y valores.**
2. **Existen variables como objetos capaces de contener un único dato, como tipos numéricos (int, float, complex, oct, hex, bin), strings y booleanos, que coexisten con colecciones más complejas, tales como listas (list), conjuntos (set) y diccionarios (dict), caracterizadas por su estructura clave-valor.**
3. **En Python, solo hay un puñado de constantes preestablecidas en el intérprete, y el programador no puede crear nuevas constantes de manera estricta. Lo más cercano al concepto de constantes son las tuplas o los frozensets, pero estas son colecciones (estructuras que admiten múltiples datos).**

**Aunque no hay un concepto estricto de "constantes" en el sentido tradicional, la comunidad de programadores Python establecio poner en mayusculas el nombre del objeto para indicar que ciertas variables deben tratarse como constantes, (valores si bien pueden, no deben modificarse durante la ejecución del programa). Ver convención de estilo de Python.**

1. **Los objetos son representaciones o instancias de una clase y poseen tanto métodos como atributos. Los atributos pueden entenderse como características o propiedades que describen el estado de un objeto, mientras que los métodos son funciones asociadas a la clase que actúan sobre esos atributos o realizan operaciones específicas.**

**Principales funciones y estructuras de control de flujo en Python**

* **print: La función print() se utiliza para mostrar información en la consola. Puedes imprimir valores, variables o mensajes.**
* **if, else, elif: La estructura de control if, else, y elif se utiliza para tomar decisiones basadas en condiciones. Se ejecuta un bloque de código si la condición es verdadera y, en caso contrario, se ejecuta otro bloque.**
* **input: La función input() se utiliza para obtener la entrada de datos del usuario desde la consola. Esta captura siempre devuelve un string (cadena de texto).**
* **while: La estructura de control while se utiliza para repetir un bloque de código mientras una condición sea verdadera.**
* **for: La estructura de control for se utiliza para iterar sobre una secuencia de elementos como listas, tuplas, cadenas, string entre objetos iterables.**

**Funciones propias**

* **def y return: La palabra clave def se utiliza para definir una función en Python. La función puede tener parámetros y realiza operaciones específicas dentro del bloque de código indentado. La palabra clave return se utiliza para devolver un resultado desde la función.**

**Algunas funciones incorporadas en Python que trabajan con secuencias de datos.**

* **min(): Devuelve el valor mínimo de una secuencia.**
* **max(): Devuelve el valor máximo de una secuencia.**
* **sum(): Devuelve la suma de todos los elementos en una secuencia.**
* **len(): Devuelve la longitud (número de elementos) de una secuencia.**

**El propósito de este curso es revisar y ampliar los conocimientos previos desde un enfoque más profesional en programación. Abordaremos la manipulación de grandes cantidades de información mediante el uso de bases de datos y exploraremos una introducción al manejo de interfaces gráficas de usuario (GUI- Graphical User Interface)**

**Las funciones de orden superior son aquellas funciones que toman una o más funciones como argumentos o devuelven funciones como resultados. Aquí te presento algunas funciones de orden superior en Python y sus descripciones:**

* **map(func, iterable, ...): Aplica la función dada a cada elemento del iterable (o varios iterables) y retorna un iterable con los resultados.**
* **filter(func, iterable): Filtra los elementos de un iterable basándose en si la función devuelve True o False. Retorna un iterable con los elementos que cumplen la condición.**
* **reduce(func, iterable[, initializer]): Aplica la función de manera acumulativa a los elementos del iterable, de izquierda a derecha, reduciendo el iterable a un solo valor. El argumento opcional initializer proporciona un valor inicial. from functools import reduce**

**Funciones built in:**

* **any(iterable): Retorna True si al menos un elemento del iterable es verdadero. Si el iterable está vacío, retorna False.**
* **all(iterable): Retorna True si todos los elementos del iterable son verdaderos (o si el iterable está vacío).**
* **sorted(iterable, key=None, reverse=False): Retorna una nueva lista ordenada de los elementos de un iterable. El argumento key puede especificar una función de comparación, y reverse controla el orden ascendente o descendente.**
* **reversed(sequence): Retorna un objeto iterable que produce los elementos de la secuencia en orden inverso.**
* **zip(iterable1, iterable2, ...): Combina elementos de varios iterables en tuplas. Retorna un iterable de tuplas, donde la i-ésima tupla contiene el i-ésimo elemento de cada iterable.**
* **enumerate(iterable, start=0): Retorna pares (índice, elemento) para cada elemento en el iterable, comenzando desde el índice proporcionado (por defecto, desde 0).**

**Funciones útiles durante la depuración y la exploración de objetos en un programa Python.**

* **type(object): La función type se utiliza para obtener el tipo de un objeto. Puede devolver el tipo de cualquier objeto, ya sea un tipo incorporado o un tipo definido por el usuario.**
* **dir([object]): La función dir se utiliza para obtener una lista de nombres de atributos válidos para un objeto. Si se proporciona un objeto, la función devuelve los atributos del objeto; de lo contrario, devuelve los nombres de los atributos en el ámbito local.**
* **id(object): La función id devuelve el identificador único (identificador de objeto) de un objeto. Este identificador es único durante el tiempo de vida del objeto y se genera internamente por CPython, el intérprete de Python de referencia.**

**Funciones de casting o funciones de conversión. Se utilizan para cambiar el tipo de datos de un objeto a otro en Python.**

* **int(x [,base]): Convierte x a un entero. Si x es una cadena, puede incluir un segundo argumento opcional base que representa la base del sistema numérico (por defecto es 10).**
* **float(x): Convierte x a un número de punto flotante.**
* **str(x): Convierte x a una cadena.**
* **bool(x): Convierte x a un valor booleano. Devuelve False para 0, None, secuencias o colecciones vacías, y True para cualquier otro valor.**
* **list(iterable): Convierte un iterable (como una tupla, cadena, o conjunto) a una lista.**
* **tuple(iterable): Convierte un iterable a una tupla.**
* **set(iterable): Convierte un iterable a un conjunto.**

**Manejo de excepciones:**

* **Los bloques try, except, finally, else son utilizados en Python para gestionar excepciones, que son situaciones excepcionales o errores que pueden ocurrir durante la ejecución de un programa.**
* **try : Este bloque contiene el código que puede lanzar una excepción. El intérprete de Python intentará ejecutar este código, y si se produce una excepción, saltará a la sección except correspondiente.**
* **except: Este bloque se ejecuta cuando se produce una excepción en el bloque try. Puedes tener múltiples bloques except para manejar diferentes tipos de excepciones.**
* **else: Este bloque se ejecuta si no se produce ninguna excepción en el bloque try. Proporciona un código que se ejecutará solo si el bloque try se ejecuta sin errores.**
* **finally: Este bloque se ejecuta siempre, ya sea que se haya producido una excepción o no en el bloque try. Se utiliza comúnmente para realizar acciones de limpieza o liberar recursos, independientemente de si se maneja la excepción o no.**

**Manejo de datos con bases de datos es un aspecto fundamental en el desarrollo de aplicaciones y sistemas de información.**

* **Diseño de la Base de Datos.**
* **Modelo de Datos (Define la estructura y las relaciones entre los datos con MySQL o SQLite3)**
* **Entidades, Atributos y Relaciones (enlazar entidades entre sí esta pueden ser uno a uno, uno a muchos o muchos a muchos).**

**Creación y Conexión:**

* **Crear la Base de Datos: Utiliza un sistema de gestión de bases de datos (DBMS) para crear la base de datos según el diseño previamente establecido.**
* **Establecer Conexión: Desde tu aplicación, establece una conexión con la base de datos utilizando un conector o un controlador específico del DBMS.**

**Operaciones CRUD:**

* **Crear: Inserta nuevos datos en la base de datos.**
* **Leer: Recupera datos de la base de datos mediante consultas (queries).**
* **Actualizar: Modifica datos existentes en la base de datos.**
* **Eliminar: Elimina datos de la base de datos.**

**Lenguaje SQL:**

* **Structured Query Language (SQL): Es el lenguaje estándar para realizar operaciones en bases de datos relacionales. Incluye instrucciones como SELECT, INSERT, UPDATE, DELETE, y CREATE.**

**Transacciones:**

* **Transacciones: Agrupan operaciones en una unidad lógica. Garantizan la consistencia de los datos, permitiendo confirmar o deshacer un conjunto de operaciones en bloque.**

**Indices y Optimización:**

* **Indices: Mejoran el rendimiento de las consultas al proporcionar un acceso más rápido a los datos. Se crean en columnas específicas.**
* **Optimización de Consultas: Asegura que las consultas sean eficientes y se ejecuten en un tiempo razonable. Puede implicar el uso de índices, ajuste de consultas o revisión del diseño de la base de datos.**

**CRUD: Es un acrónimo que representa las cuatro operaciones fundamentales en la gestión de datos, forman la base de las interacciones de manipulación de datos en muchas aplicaciones y sistemas:**

* **Create: La operación de creación implica la inserción de nuevos datos en una base de datos o sistema de almacenamiento.**
* **Read: La operación de lectura implica la recuperación de datos existentes de una base de datos o sistema de almacenamiento.**
* **Update: La operación de actualización implica la modificación de datos existentes en una base de datos o sistema de almacenamiento.**
* **Delete: La operación de eliminación implica la remoción de datos existentes en una base de datos o sistema de almacenamiento.**

**ABM: Es un acrónimo que se utiliza en el contexto de sistemas de información y representa las operaciones de Alta, Baja, y Modificación principalmente de datos:**

* **Alta: Similar a la operación de "Create" en CRUD, el alta implica la inserción de nuevos datos o la creación de nuevos registros.**
* **Ejemplo: Registrar un nuevo cliente en un sistema.**
* **Baja: Similar a la operación de "Delete" en CRUD, la baja implica la eliminación de datos o registros existentes.**
* **Modificación: Similar a la operación de "Update" en CRUD, la modificación implica la alteración de datos existentes.**

**Diferencias y Relación:**

**CRUD es más amplio y general, abarcando la creación, lectura, actualización y eliminación de datos, campos o estructuras de las bases de datos.**

**ABM se centra específicamente en las operaciones de alta, baja y modificación, que son un subconjunto de las operaciones CRUD mas especifico cuando se quiere enfocar en las operaciones fundamentales de manipulación de datos en sistemas.**

**Manejo de interfaces gráficas de usuario (GUI- Graphical User Interface).**

**Una GUI es un tipo de interfaz que permite la interacción entre usuarios y sistemas informáticos a través de elementos gráficos como iconos, botones, ventanas y menús, en contraste con interfaces de línea de comandos (CLI) que utilizan texto. Estas hacen que las interacciones con los sistemas informáticos sean más intuitivas y visuales, facilitando la tarea para los usuario. Estas interfaces son comunes en sistemas operativos, aplicaciones de software, aplicaciones móviles y muchos otros contextos informáticos.**

**Nuevamente en Python todo son objetos. Los componentes visuales relacionedos a la pantalla gráfica interactiva se denominan “widget” y permiten a los usuarios interactuar con una aplicación o programa mediante acciones como hacer clic, arrastrar, o ingresar datos.**

**Widgets comunes:**

**Ventanas: Contenedores para organizar y mostrar información o aplicaciones.**

**Botones: Elementos que los usuarios pueden presionar para activar una acción.**

**Cuadros de Texto: Áreas donde los usuarios pueden ingresar o editar texto.**

**Etiquetas: Componentes para mostrar información estática.**

**Listas y Cuadros Combinados: Permiten a los usuarios seleccionar entre opciones.**

**Casillas de Verificación y Botones de Opción:Para seleccionar opciones en formularios.**

**Inicio 1:**

* Estadísticas
* Información del mercadoIT
* Mundo IT
* Recursos
* comunidad Python
* Python en su estructuras
* Python en mas empresas de software
* Python en los sistemas operativos
* Python en Backend
* Python en Frontend
* Instalación

**Porque Python**

* Sintaxis
* Amplia comunidad y recursos
* Comunidad y cultura de colaboración
* Versatilidad y aplicaciones prácticas
* Amplia biblioteca estándar
* Fácil integración
* Portabilidad
* Enfoque en la legibilidad del código
* Amplia adopción y demanda laboral
* Aprendizaje gradual y progresivo
* Desarrollo rápido de prototipos y productividad

**Aprendizaje de conceptos de programación fundamentales**

* Desarrollo web
* Ciencia de los datos
* Aprendizaje automático
* Multimedia
* Desarrollo de software
* Marcos web como Django
* Aplicación GUI

**Conceptos clave relacionados con los objetos en Python:**

* Atributos
* Métodos

**interprete Python**

* Instalación del IDE
* Python en linea
* anaconda y jupyter

**Concepto de funciones Built-in**

**Uso de comillas en Python**

**Uso de secuencias de escape mas usadas en Python**

**Módulo 1:**

Lenguaje humano: Facultad del ser humano de expresarse y comunicarse con los demás a través del sonido articulado o de otros sistemas de signos. (<https://dle.rae.es/lenguaje?m=form>)

En resumen es un medio, una herramienta de comunicación. Los lenguajes de programación son métodos de comunicación con un procesador CPU y/o GPU

Un circuito electrónico permite el traspaso de información en estados digitales binarios donde cargado eléctricamente es 1 o True y sin carga 0 o false. También analíticamente se puede trasmitirse información es valores de escala entre un valor base y un máximo.

Por ejemplo si la base es 0 y el máximo es 255 y se envía un 128 estaría mandando un estado de 50% la escala establecida.

Entre la comunicación necesaria para estos manejos de compuertas, datos y demás procesos electrónicos básicos y el de comunicación habitual entre los humanos hay una gran diferencias.  
Estas diferencias se sanjan con distintos lenguajes que permiten la comunicación entres el humano y los microprocesadores

Los lenguajes cercanos a los procesos electrónicos se denominan de bajo nivel y aquellos que estan mas cerca del lenguaje humano se denomina lenguajes de alto nivel

La comunicación en un lenguaje de bajo nivel es muy rápida ya que es leida por el hardware y ejecutada sin intermediarios, también son realmente muy complicadas para ser escritas y desarrollados por un humano.

Por otro lado si escribimos en un lenguaje de alto nivel, cercano al humano, se debe procesar mucho esta comunicación para poder ser ejecutada por un microprocesador lo que es mucho mas lento en la ejecución.

Para podes comunicar dos entidades (humanos, animales, computadoras,etc.) se requieren que ambas puedan comprender un conjunto de sonidos o símbolos para intercambiar información. Si es escrito tenemos un alfabeto como en ingles, español, kanji en japonés, hans (Hansi - 汉斯 – Hànsī) chinos, números indo arábigos, romanos, 0 y 1 en binarios.

Una vez establecido el alfabeto en común para que las entidades se entiendan se necesitan de un léxico o un diccionario que establezca el significado de un conjunto de sonidos o símbolos como ideas, ordenes, palabras, oraciones, etc.

Posteriormente al tener un alfabeto y un léxico en común debemos establecer una sintaxis, reglas para que la sucesión de palabras se haga de forma que se interprete el significado. Este conjunto de reglas utilizadas para determinar si una sucesión de palabras forma, o no, una oración válida.

Por ultimo hay un conjunto de reglas semánticas para que las entidades puedan dar el mismo significado a una frase o linea de comando.

Si una entidad recibe una frase o sentencia y no puede interpretarla o lo hace en otro sentido al que la emitió la comunicación no fue exitosa. No se realizo la comunicación, alguien emitió un información u ordenes y el receptor no las entendió por lo que no las ejecuto

Debajo encontraras información del importantísimo rol del lenguaje Python en programación y porque este curso se dicta en el.

Iniciaremos con un paradigma funcional y luego nos introduciremos en el paradigma de programación orientada a objetos (POO) .

**Porque Python:**

|  |
| --- |
| **No hay que pagar licencias ni actualizaciones!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**  **Sintaxis** sencilla y legible: Python se destaca por su sintaxis simple y legible, lo que facilita su aprendizaje y comprensión. Su estructura de indentación obligatoria ayuda a mantener un código limpio y organizado.  **Amplia comunidad y recursos:** Python cuenta con una gran comunidad de desarrolladores que brindan soporte, comparten conocimientos y crean bibliotecas y frameworks útiles. Además, existen numerosos recursos en línea, como documentación, tutoriales y cursos, que facilitan el aprendizaje y el desarrollo en Python.  **Comunidad y cultura de colaboración:** La comunidad de Python es conocida por su actitud de apertura, colaboración y ayuda mutua. Los desarrolladores de Python están dispuestos a compartir conocimientos, colaborar en proyectos y ayudar a resolver problemas. Ser parte de esta comunidad puede ser enriquecedor y brindarte oportunidades de crecimiento personal y profesional.  **Versatilidad y aplicaciones prácticas:** Python se utiliza en una amplia variedad de campos, como desarrollo web, análisis de datos, inteligencia artificial, aprendizaje automático, automatización de tareas, scripting y más. Esto significa que aprender Python puede abrirte puertas en diferentes industrias y roles laborales.  **Amplia biblioteca estándar:** Python cuenta con una biblioteca estándar muy completa, que incluye módulos para realizar tareas comunes, como manipulación de archivos, acceso a bases de datos, networking, procesamiento de texto, entre otros. Esto te permite ahorrar tiempo y esfuerzo al aprovechar estas funcionalidades listas para usar.  **Fácil integración:** Python es conocido por su capacidad de integración con otros lenguajes y sistemas. Puede interactuar con código escrito en C, C++, Java y otros lenguajes, lo que facilita la combinación de diferentes tecnologías en un proyecto.  **Portabilidad:** Python es un lenguaje multiplataforma, lo que significa que puedes escribir un código en Python y ejecutarlo en diferentes sistemas operativos, como Windows, macOS y Linux, sin realizar modificaciones significativas.  **Enfoque en la legibilidad del código:** La filosofía de diseño de Python se basa en la legibilidad del código, lo que fomenta buenas prácticas de programación y facilita la colaboración en proyectos de equipo. Esto también ayuda a reducir errores y a mantener el código a largo plazo.  **Amplia adopción y demanda laboral:** Python ha experimentado un crecimiento significativo en los últimos años y se ha convertido en uno de los lenguajes de programación más populares. Esto ha llevado a una alta demanda de profesionales con habilidades en Python en el mercado laboral. Aprender Python puede aumentar tus oportunidades de empleo y mejorar tus perspectivas.  **Aprendizaje gradual y progresivo:** Python es un lenguaje amigable para principiantes, lo que lo hace ideal para aquellos que están comenzando en la programación. Su sintaxis clara y legible, junto con su enfoque en la simplicidad, permite un aprendizaje gradual y progresivo, lo que ayuda a construir una base sólida en conceptos de programación.  **Desarrollo rápido de prototipos y productividad:** Python es conocido por su enfoque en la productividad y el desarrollo rápido de prototipos. La combinación de su sintaxis sencilla y sus bibliotecas de alto nivel permite a los desarrolladores escribir código de manera eficiente y rápida, lo que acelera el proceso de desarrollo de software.  **Aprendizaje de conceptos de programación fundamentales:** Al aprender Python, adquirirás conocimientos y habilidades fundamentales de programación que son transferibles a otros lenguajes. Aprenderás sobre estructuras de datos, control de flujo, funciones, objetos y módulos, lo que te brinda una base sólida para abordar otros lenguajes y paradigmas de programación.  La biblioteca de Python se puede utilizar para los siguientes propósitos:   * desarrollo web * Ciencia de los datos * Aprendizaje automático * multimedia * Desarrollo de software * Marcos web como Django * Aplicación GUI |

·

|  |
| --- |
| Antes que nada debemos aclarar algo para los que vienen de otro lenguaje.  **En Python, todo se considera un objeto.** Un objeto es una entidad que combina datos (**atributos**) y funciones (**métodos**) relacionadas que operan en esos datos. Es la base del paradigma de programación orientada a objetos (POO) en Python. |

·

|  |
| --- |
| **Conceptos clave relacionados con los objetos en Python:**  **Atributos:**  Los atributos son las variables que pertenecen a un objeto. Pueden ser variables de instancia (definidas dentro de un método) o variables de Módulo (definidas en la Módulo pero fuera de los métodos). Cada instancia de una Módulo puede tener diferentes valores para los atributos.  **Métodos:**  Los métodos son las funciones asociadas a una módulo. Pueden realizar operaciones en los datos del objeto y pueden acceder y modificar los atributos. Los métodos pueden ser llamados en las instancias de la Módulo y actúan en contexto a los datos específicos de la instancia. |

|  |
| --- |
| En adelante utilizaremos y reforzaremos el termino objeto. Ej Objeto variable tipo string.  Los objetos son las entidades fundamentales que encapsulan datos y funcionalidad relacionada. Las módulos definen la estructura y el comportamiento de los objetos, mientras que las instancias son objetos específicos creados a partir de una módulo. La programación orientada a objetos en Python facilita la organización, la reutilización y la modularidad del código. |

El curso al desarrollarse en Python requiere que se instale el interprete y un IDE.

Tenga en cuenta que cada alumno puede estar en linux, en alguna de sus varias distribuciones, windows 32 o 64 y en versiones 7,8,10 u 11 o MacOS con procesador Intel o ARM.

|  |
| --- |
| **interprete Python**  Link de descargas para:  windows 8.1 en adelante  Linux  MacOs 10 en adelante (intel y AMR)  <https://www.python.org/downloads/> |
| Tener en cuenta que windows 7 solo es compatible hasta la versión Python 3.10.8  Link de descarga [Windows 7 installer (32-bit)](https://www.python.org/ftp/python/3.10.8/python-3.10.8.exe)  <https://www.python.org/ftp/python/3.10.8/python-3.10.8.exe>  Link de descarga [Windows 7 installer (64-bit)](https://www.python.org/ftp/python/3.10.8/python-3.10.8-amd64.exe)  <https://www.python.org/ftp/python/3.10.8/python-3.10.8-amd64.exe> |

* Instalación del IDE

|  |  |
| --- | --- |
| **IDEs**  Links de descargas | |
| Download Visual Studio Code | <https://code.visualstudio.com/download> |
| Eclipse | <https://www.eclipse.org/downloads/> |
| pyzo | <https://pyzo.org/start.html> |
| Eric | <https://eric-ide.python-projects.org/eric-download.html> |
| PyCharm | <https://www.jetbrains.com/pycharm/download/> |
| Spyder | <https://www.spyder-ide.org/> |
| Geany | <https://www.geany.org/> |
| Disponibles para  windows 8.1 en adelante  Linux  MacOs 10 en adelante | |

Mas adelante , para ciencia de datos, trabajaremos con anaconda y jupyter

En linea (si bien mas lento no requiere instalación)

## ****Python.org**** https://www.Python.org/

**El intérprete de Python.org es el sitio oficial del lenguaje. En él, se consigue todo tipo de información relacionada con el lenguaje. Los instaladores para las distintas plataformas, noticias, eventos y demás. También se cuenta con acceso a una shell Python que puede ser utilizada de forma anónima. No se pueden cargar archivos ni instalar módulos que no pertenezcan a la librería estándar, pero cuenta con las ultimas actualizaciones disponibles y se accede de forma muy rápida y sencilla. Es muy útil para probar estructuras como ciclos y funciones. Python.org**

## ****REPL.it**** <https://replit.com/> **(uno de los mejores)**

Es un intérprete para Python desarrollado de manera abierta y disponible en Github. Es un compilador online que emula un terminal en el cual se ejecuta todo el código escrito. Además de los lenguajes ya citados ofrece soporte para escribir en otros 16 lenguajes [variados] incluido Javascript y Coffee Script.

Algunas características

* Desarrollado en js y python: Replit ofrece una interfaz minimalista.
* Se puede acceder desde cualquier dispositivo.
* Codificación directa y libre de distracción.
* Provee de ejemplos necesarios para cada lenguaje.
* Permite Guardar y compartir nuestro código.

## ****PythonAnywhere**** <https://www.pythonanywhere.com/>

Es un ambiente de desarrollo integrado (IDE) online para Python y servicio de alojamiento en linea. Tiene 512 MB disponibles para almacenamiento de archivos en las cuentas gratuitas, terminal bash, Python, MySQL, PyPy, IPython, y terminales personalizadas. También cuenta con posibilidad de ejecutar aplicaciones web de Python y Jupyter Notebooks para cuentas no gratuitas. Se pueden programar tareas y manejar bases de datos Postgres y MySQL. Al contar con una terminal bash, PythonAnywhere permite utilizar el gestor de paquetes de Python, pip, para instalar módulos que no pertenezcan a la librería estándar de Python.

## ****Ideone.com**** [https://www.ideone.com/](https://www.pythonanywhere.com/)

Es un interprete Python y compilador e IDE online para Python, C/C++, Java, Haskell, PHP, y muchos otros lenguajes de programación. Cuenta con una sencilla pero muy útil interfaz en la que se puede copiar el código de un programa para luego compilarlo o ejecutarlo. Es una buena alternativa para alguien que necesite ejecutar o probar un programa de forma rápida, anónima y sencilla. Tiene opciones para cambiar la entrada estándar, cambiar la privacidad del código e ingresar notas alternativas al código del programa.

## ****Codeanywhere**** [https://www.codeanywhere.com/](https://www.pythonanywhere.com/)

Es un IDE multilenguaje basado en la nube. Su interfaz cuenta con un editor de texto y una terminal que varía dependiendo del entorno con el cual se defina el contenedor, el cual es el lugar en donde se guardarán los archivos que se redacten o carguen. Cuenta con ambientes de Ubuntu y CentOS. Tiene posibilidades de instalar paquetes personalizados en los ambientes con el gestor de paquetes apt, en el entorno Ubuntu, y yum, en el entorno CentOS, y cuenta también con el gestor de paquetes de Python, pip. Inicialmente se cuenta con una prueba gratuita del sistema con la cual se tiene acceso por 7 días y se puede generar un contenedor y una conexión remota.

y hay varios mas en la web.

|  |
| --- |
| Tenga en cuenta que normalmente cada alumno deberá registrar para poder grabar los scripts.  En caso de uso en linea y que el alumno no desee registrarse una vez finalizada el script y ejecutado podrá copiarse y pegarse en un block de notas para que el alumno tenga los scripts guardados  En caso de instalación en la pc del alumno recuerde que probablemente python en su primer pantalla de instalación marcar add Python to Path asi se agrega a las variables de entorno en windows. Esta información paso a paso se encuentra en la guiá de instalación que acompaña este archivo.  **Por favor confirme que cada alumno pueda ejecutar el siguiente código.** |

·

* Concepto de funciones Built-in

|  |  |  |  |
| --- | --- | --- | --- |
| <https://docs.python.org/es/3.12/library/functions.html>  Clic sobre cada función para acceder a una descripción a python.org | | | |
| **A**  [abs()](https://docs.python.org/es/3.12/library/functions.html" \l "abs)  [aiter()](https://docs.python.org/es/3.12/library/functions.html" \l "aiter)  [all()](https://docs.python.org/es/3.12/library/functions.html" \l "all)  [any()](https://docs.python.org/es/3.12/library/functions.html" \l "any)  [anext()](https://docs.python.org/es/3.12/library/functions.html" \l "anext)  [ascii()](https://docs.python.org/es/3.12/library/functions.html" \l "ascii)  **B**  [bin()](https://docs.python.org/es/3.12/library/functions.html" \l "bin)  [bool()](https://docs.python.org/es/3.12/library/functions.html" \l "bool)  [breakpoint()](https://docs.python.org/es/3.12/library/functions.html" \l "breakpoint)  [bytearray()](https://docs.python.org/es/3.12/library/functions.html" \l "func-bytearray)  [bytes()](https://docs.python.org/es/3.12/library/functions.html" \l "func-bytes)  **C**  [callable()](https://docs.python.org/es/3.12/library/functions.html" \l "callable)  [chr()](https://docs.python.org/es/3.12/library/functions.html" \l "chr)  [classmethod()](https://docs.python.org/es/3.12/library/functions.html" \l "classmethod)  [compile()](https://docs.python.org/es/3.12/library/functions.html" \l "compile)  [complex()](https://docs.python.org/es/3.12/library/functions.html" \l "complex)  **D**  [delattr()](https://docs.python.org/es/3.12/library/functions.html" \l "delattr)  [dict()](https://docs.python.org/es/3.12/library/functions.html" \l "func-dict)  [dir()](https://docs.python.org/es/3.12/library/functions.html" \l "dir)  [divmod()](https://docs.python.org/es/3.12/library/functions.html" \l "divmod) | **E**  [enumerate()](https://docs.python.org/es/3.12/library/functions.html" \l "enumerate)  [eval()](https://docs.python.org/es/3.12/library/functions.html" \l "eval)  [exec()](https://docs.python.org/es/3.12/library/functions.html" \l "exec)  **F**  [filter()](https://docs.python.org/es/3.12/library/functions.html" \l "filter)  [float()](https://docs.python.org/es/3.12/library/functions.html" \l "float)  [format()](https://docs.python.org/es/3.12/library/functions.html" \l "format)  [frozenset()](https://docs.python.org/es/3.12/library/functions.html" \l "func-frozenset)  **G**  [getattr()](https://docs.python.org/es/3.12/library/functions.html" \l "getattr)  [globals()](https://docs.python.org/es/3.12/library/functions.html" \l "globals)  **H**  [hasattr()](https://docs.python.org/es/3.12/library/functions.html" \l "hasattr)  [hash()](https://docs.python.org/es/3.12/library/functions.html" \l "hash)  [help()](https://docs.python.org/es/3.12/library/functions.html" \l "help)  [hex()](https://docs.python.org/es/3.12/library/functions.html" \l "hex)  **I**  [id()](https://docs.python.org/es/3.12/library/functions.html" \l "id)  [input()](https://docs.python.org/es/3.12/library/functions.html" \l "input)  [int()](https://docs.python.org/es/3.12/library/functions.html" \l "int)  [isinstance()](https://docs.python.org/es/3.12/library/functions.html" \l "isinstance)  [issubclass()](https://docs.python.org/es/3.12/library/functions.html" \l "issubclass)  [iter()](https://docs.python.org/es/3.12/library/functions.html" \l "iter) | **L**  [len()](https://docs.python.org/es/3.12/library/functions.html" \l "len)  [list()](https://docs.python.org/es/3.12/library/functions.html" \l "func-list)  [locals()](https://docs.python.org/es/3.12/library/functions.html" \l "locals)  **M**  [map()](https://docs.python.org/es/3.12/library/functions.html" \l "map)  [max()](https://docs.python.org/es/3.12/library/functions.html" \l "max)  [memoryview()](https://docs.python.org/es/3.12/library/functions.html" \l "func-memoryview)  [min()](https://docs.python.org/es/3.12/library/functions.html" \l "min)  **N**  [next()](https://docs.python.org/es/3.12/library/functions.html" \l "next)  **O**  [object()](https://docs.python.org/es/3.12/library/functions.html" \l "object)  [oct()](https://docs.python.org/es/3.12/library/functions.html" \l "oct)  [open()](https://docs.python.org/es/3.12/library/functions.html" \l "open)  [ord()](https://docs.python.org/es/3.12/library/functions.html" \l "ord)  **P**  [pow()](https://docs.python.org/es/3.12/library/functions.html" \l "pow)  [print()](https://docs.python.org/es/3.12/library/functions.html" \l "print)  [property()](https://docs.python.org/es/3.12/library/functions.html" \l "property) | **R**  [range()](https://docs.python.org/es/3.12/library/functions.html" \l "func-range)  [repr()](https://docs.python.org/es/3.12/library/functions.html" \l "repr)  [reversed()](https://docs.python.org/es/3.12/library/functions.html" \l "reversed)  [round()](https://docs.python.org/es/3.12/library/functions.html" \l "round)  **S**  [set()](https://docs.python.org/es/3.12/library/functions.html" \l "func-set)  [setattr()](https://docs.python.org/es/3.12/library/functions.html" \l "setattr)  [slice()](https://docs.python.org/es/3.12/library/functions.html" \l "slice)  [sorted()](https://docs.python.org/es/3.12/library/functions.html" \l "sorted)  [staticmethod()](https://docs.python.org/es/3.12/library/functions.html" \l "staticmethod)  [str()](https://docs.python.org/es/3.12/library/functions.html" \l "func-str)  [sum()](https://docs.python.org/es/3.12/library/functions.html" \l "sum)  [super()](https://docs.python.org/es/3.12/library/functions.html" \l "super)  **T**  [tuple()](https://docs.python.org/es/3.12/library/functions.html" \l "func-tuple)  [type()](https://docs.python.org/es/3.12/library/functions.html" \l "type)  **V**  [vars()](https://docs.python.org/es/3.12/library/functions.html" \l "vars)  **Z**  [zip()](https://docs.python.org/es/3.12/library/functions.html" \l "zip)  **\_**  [\_\_import\_\_()](https://docs.python.org/es/3.12/library/functions.html" \l "import__) |

·

|  |
| --- |
| print ("hola Curso Python") |

Si se instalo en la pc se podrá encontrar es necesario compilar o construir y luego ejecutar el script

Depende del IDE emerge consola o se ve el resultado en la barra de powershell

En linea veremos o particiones de la pantalla o emergentes o demás salidas pero en todas se verán

|  |
| --- |
| hola Curso Python |

|  |
| --- |
| Siguiente paso es que podamos escribir varios literales y ver la salida del código.  Se deberá dar una explicación del proceso interpretación. |

**Uso de comillas en Python**

' Comillas simple

" Comillas doble

Al inicio y final de una cadena de texto (string)

Son utilizables uno u otra, pero si se abre con un tipo se debe cerrar con el mismo

' Hola mundo IT curso Python'

"Hola mundo IT curso Python"

\*\* \*\*

inicio fin

Se pueden usar uno dentro de otra

'Hola "mundo IT" curso Python'

"Hola 'mundo IT' curso Python"

\*\* \*\* internas \*\* \*\*

inicio fin

Tenga en cuenta que en ingles las comillas simples se suelen usar muy habitualmente como apostrofe

'Tom's cat'

\*\* \*\* \*\*

inicio fin inicio…………..

"I don't want eat this candy"

\*\* \*\* interna \*\*

inicio fin

**' ' '**  Comillas simple x 3

**" " "** Comillas doble x 3

**Esto habilita usar strings de mas de una linea**

""" esto es

un

texto

de varias

lineas **"""**

**Uso de secuencias de escape mas usadas en Python**

\n newline (salto de linea)

\t tab (tabulación)

\\ Backslash (barra)

\' Comillas simple

\" Comillas doble

|  |
| --- |
| # \n para generar una nueva linea entre el texto anterior y el posterior en la salida  print ("hola Curso \n Python") |

Salida esperada

|  |
| --- |
| hola Curso  Python |

·

|  |
| --- |
| # \n para generar una nueva linea entre el texto anterior y el posterior en la salida  print ("""hola Curso  Python""") |

Salida esperada

|  |
| --- |
| hola Curso  Python |

|  |
| --- |
| # \t para tabular (de 4 a 8 espacios) entre el texto anterior y el posterior en la salida  print ("hola Curso \t Python") |

Salida esperada

|  |
| --- |
| hola Curso Python |

·

|  |
| --- |
| # \t para tabular (de 4 a 8 espacios) entre el texto anterior y el posterior en la salida  print ("hola Curso Python") |

Salida esperada por consola

|  |
| --- |
| hola Curso Python |

·

|  |
| --- |
| # \" imprimir comillas  # \' imprimir comillas  print ("hola Curso \" Python\" ") |

Salida esperada por consola

|  |
| --- |
| hola Curso "Python" |

·

|  |
| --- |
| # \" imprimir comillas  # \' imprimir comillas  print ('hola Curso " Python" ')  print ("hola Curso 'Python' ") |

Salida esperada por consola

|  |
| --- |
| hola Curso "Python"  hola Curso 'Python' |

·

|  |
| --- |
| \a que el alumno investigue que salida tiene, bell un sonido |

La primer función en invocar sera para poder sacar información desde el programa al usuario

|  |
| --- |
| Conceptos de compilador, lenguajes, niveles, consola, salida de información. No dejamos mas que la idea de trabajar son lo echo sin avanzar en nuevos temas para esta practica.  Si quieren esbozar los conceptos variables y constantes como lugares en memoria donde se guardan datos. |

Para poder entender a Python en un contexto historico les dejo el siguiente articulo

**Fuente:** https://es.wikipedia.org/wiki/IBM\_Personal\_Computer\_XT

El IBM Personal Computer XT, normalmente abreviado como IBM XT o simplemente XT, fue el sucesor de IBM al IBM PC original. Fue puesto a la venta como IBM número de producto 5160 el 8 de marzo de 1983. XT son las siglas de eXtended Technology (Tecnología eXtendida). Estaba basado esencialmente en la misma arquitectura que el PC original, únicamente añadiendo algunas mejoras: se añadió un disco duro, 8 slots de expansión en vez de 5, más memoria en la tarjeta madre, una fuente de alimentación de mayor potencia y se le quitó la interfaz para casetes del IBM PC original. Se convirtió en un estándar. El sistema estaba concebido para usuarios de negocios y un 3270 PC correspondiente implementaba la emulación de la terminal IBM 3270 que se comercializaría después, en octubre de 1983. Posteriormente, con el AT llegaría una nueva arquitectura de bus de 16 bits.

Características

El estándar XT traía de serie 128KB de memoria RAM en la tarjeta madre, una disquetera 5 1/4" de doble cara, doble densidad, de 360KB de tamaño completo, un disco duro Seagate de 10MB Seagate ST-412 y un adaptador asíncrono serial (RS 232) una fuente de alimentación de 130w suministraba energía eléctrica a todos los componentes. La placa base tenía ocho ranuras de expansión ISA de 8 bits, y al igual que el IBM PC, un microprocesador Intel 8088 corriendo a 4,77 MHz y un zócalo para coprocesador matemático Intel 8087. El sistema operativo con el que se solía vender fue el PC-DOS 2.0 y superior. Las ocho ranuras de expansión eran un aumento de las cinco del IBM PC, aunque tres de ellas eran utilizadas por el adaptador de la unidad de disquetes, el adaptador del disco duro y la tarjeta de video. Pronto se actualizó la especificación básica para estandarizarla a 256KB de memoria RAM en la tarjeta madre.

Había dos versiones de la placa base del XT. La original podía admitir hasta 256kB en la misma placa (en cuatro bancos de chips de 64kB), con un máximo de 640kB alcanzados usando tarjetas de expansión. La segunda revisión de la placa, introducida en 1986, podía admitir los 640kB enteros en la placa base, en dos bancos de chips de 256kb y dos de 64kB. Las placas más recientes podían ser adaptadas para las últimas especificaciones después de un par de modificaciones menores. La segunda revisión de la placa tenía además un IBM BIOS revisado, que incluía soporte para el teclado expandido y reducía el tiempo de arranque a la mitad.

Como pueden ver la personal computer (pc) xt original podía admitir hasta 256kB en la misma placa en 1983/86

Trabajamos con potencia de 2

2\*\*n = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192,16384, 32768, 65536

1 byte = 8 bits

La Comisión Electrotécnica Internaciona (IEEE) recomienda que se use k minúscula para decimal y K mayúscula para binario.

1024 bytes = 1 Kb (kilo byte)

Nuestro uso habitual en matemáticas es en base 10 o potencias de 10:

10\*\*1 = 10

10\*\*2 = 10\*10 = 100

10\*\*3 = 10\*10\*10 = 1,000

10\*\*6 = 1,000,000

En computación usamos base 2:

2\*\*1 = 2

2\*\*2 = 2\*2 = 4

2\*\*3 = 2\*2\*2 = 8

2\*\*10 = 1,024

2\*\*20 = 1,048,576

·

Unidad Equivalentes

1 kilobyte (KB) 1KB 1,024 bytes

1 megabyte (MB) 1024 KB 1,048,576 bytes

1 gigabyte (GB) 1024 MB 1,073,741,824 bytes

1 terabyte (TB) 1024 GB 1,099,511,627,776 bytes

1 petabyte (PB) 1024 TB 1,125,899,906,842,624 bytes

|  |
| --- |
| Regla de tres  Si una pc tenia 640kb o 1mb  Cuanta memoria tienen los alumnos en su celular.  Calcular cuantos una disquetes 5 1/4" de doble cara, doble densidad, de 360KB  necesitan para cargar una imagen de 10 megapixeles  Calcular que cantidad de pc se deberían conformar un centro de cómputos con pc xt (8086) o 286 (80286) para tener la misma capacidad que la memoria de su celular.  Relacionar el tamaño de un archivo .txt o .py con los sistemas de respaldos de Microsoft, AWS, Google, IBM, etc (exabyte, zettabytes o yottabytes) |



**Inicio 1:**

* Estadísticas
* Información del mercadoIT
* Mundo IT
* Recursos
* comunidad Python
* Python en su estructuras
* Python en mas empresas de software
* Python en los sistemas operativos
* Python en Backend
* Python en Frontend
* Instalación

**Porque Python**

* Sintaxis
* Amplia comunidad y recursos
* Comunidad y cultura de colaboración
* Versatilidad y aplicaciones prácticas
* Amplia biblioteca estándar
* Fácil integración
* Portabilidad
* Enfoque en la legibilidad del código
* Amplia adopción y demanda laboral
* Aprendizaje gradual y progresivo
* Desarrollo rápido de prototipos y productividad

**Aprendizaje de conceptos de programación fundamentales**

* Desarrollo web
* Ciencia de los datos
* Aprendizaje automático
* Multimedia
* Desarrollo de software
* Marcos web como Django
* Aplicación GUI

**Conceptos clave relacionados con los objetos en Python:**

* Atributos
* Métodos

**interprete Python**

* Instalación del IDE
* Python en linea
* anaconda y jupyter

**Concepto de funciones Built-in**

**Uso de comillas en Python**

**Uso de secuencias de escape mas usadas en Python**