
Universidad
Tecmilenio®

Universidad Tecmilenio

Operaciones y Desarrollo: DevOps

Fundamentos de Linux y scripting en Bash. Parte 1

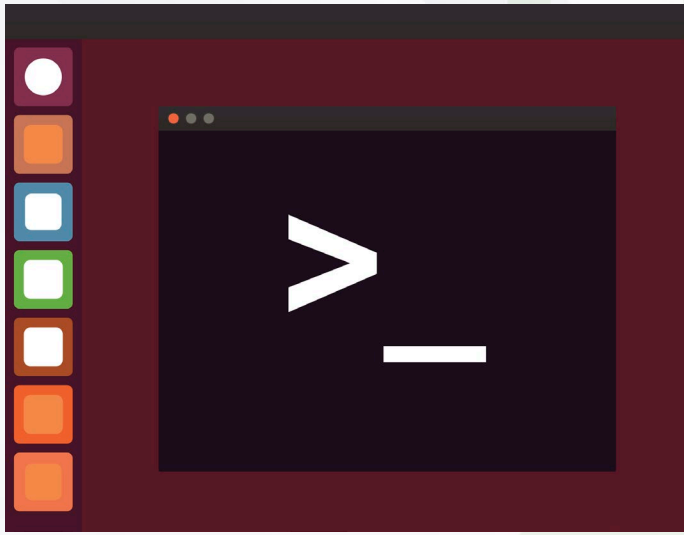
Semana 2

Introducción

Linux® es un sistema operativo de código abierto. Una versión alternativa y gratuita del sistema operativo Minix que, a su vez, se basa en los principios y diseño de Unix.

Linux es el sistema operativo más utilizado en los servidores de Internet disponibles públicamente y el único sistema operativo usado en las 450 supercomputadoras más rápidas.

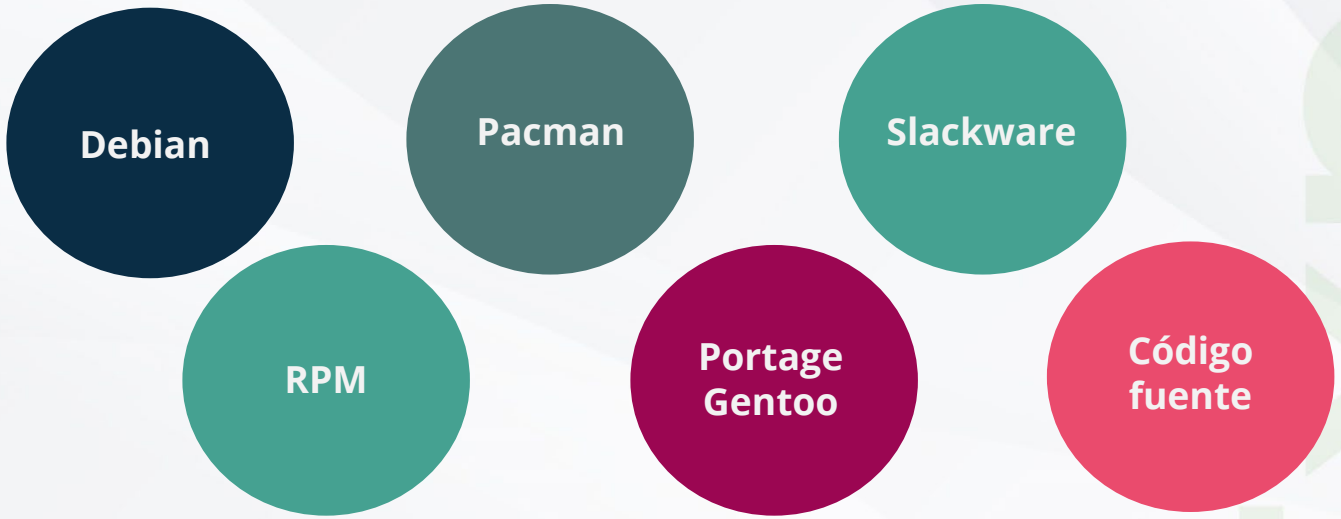
Cualquiera puede ejecutar, estudiar, compartir y modificar el software. El código modificado también se puede redistribuir e incluso vender, pero debe hacerse bajo la misma licencia.



Explicación

Linux maneja distribuciones que son sistemas operativos basados en el kernel de Linux y a menudo en un sistema de administrador de paquetes.

Los sistemas de manejo de paquetes más usuales son los siguientes:



Explicación

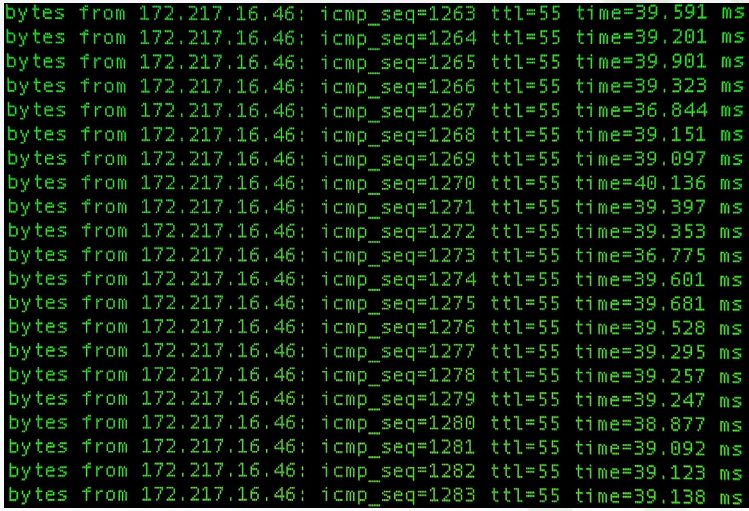
Programación en Shell

Por lo general, los usuarios de Linux trabajan dentro del sistema operativo por medio de comandos de texto dentro del Shell, pues cuando Linux empezaba, todo se manejaba por comandos.

Ahora es un poco más intuitivo, pero para sacar el mejor provecho los usuarios siguen optando por trabajar por medio de comandos. Un Shell es un programa que sirve de interfaz entre el usuario y el sistema operativo (Pedamkar, s.f.).

Tipos de Shell

- Bourne Shell
- C Shell
- Korn Shell
- GNU/Bourne Again Shell
- Zshell



Explicación

Secure Shell (SSH) es un protocolo utilizado para establecer conexiones entre dos computadoras de forma segura y en la mayoría de las distribuciones de Linux incluyen la versión OpenSSH (Geeks For Geeks, 2019).

Open SSH es un grupo de herramientas de conectividad que usan el protocolo SSH. Encripta todo el tráfico entre dos terminales de cómputo con el objetivo de eliminar escuchas no permitidas, el secuestro de conexiones o algún otro tipo de ataques.

La forma más sencilla de conectarse a un servidor vía SSH es con la instrucción que se muestra a continuación:

```
localhost:~# ssh localhost
```

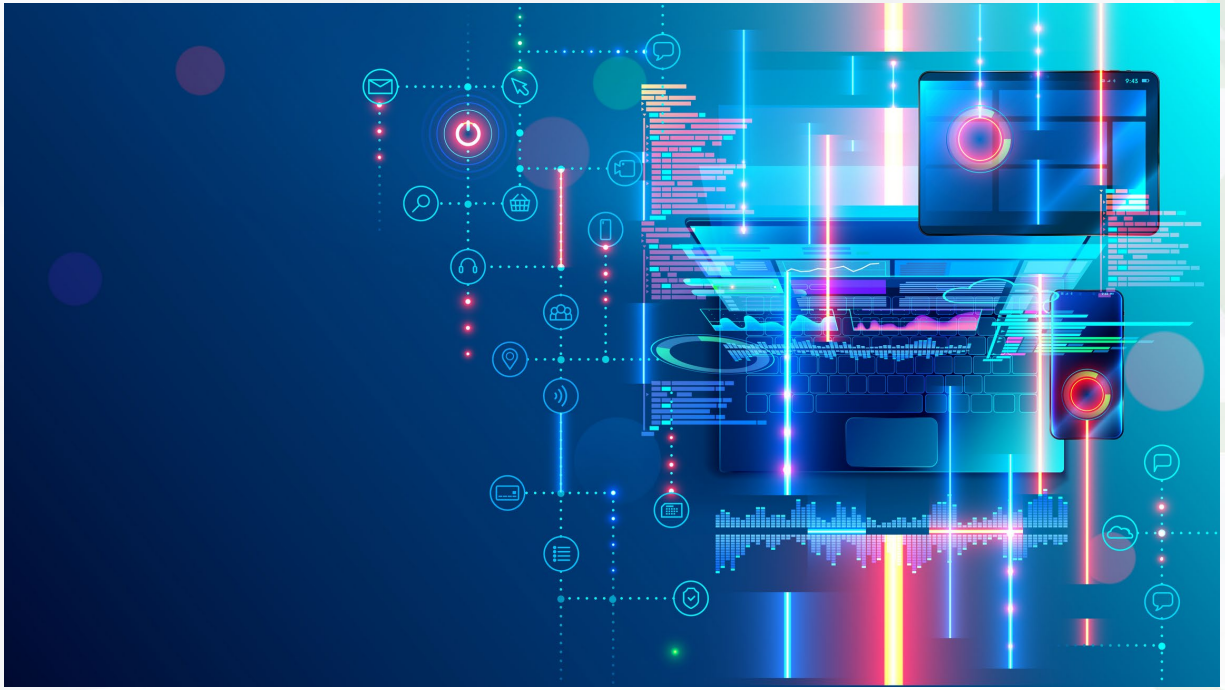
Ejercicio

1. Crea un directorio con el nombre “Tecmilenio actividad 4”.
2. Crea un archivo con tu nombre, que contenga una lista con tus aptitudes y pasiones, utilizando nano como Shell.
3. Mueve el archivo de la carpeta a otra llamada Tecmilenio actividad 4.1.
4. Crea un archivo de eventos donde especifiques tus tareas diarias en la segunda carpeta.
5. Crea un script que automatice el procesamiento y separe las tareas en “tareas de hogar y tareas de universidad” en la segunda carpeta.

Cierre

Utilizar scripts en Shell para interactuar con Linux o cualquier sistema operativo da la ventaja de manejar la información tal y como se desea.

Así que para sacar el mayor provecho de Linux es importante aprender a interactuar por medio de comandos en el Shell.



Bibliografía

Geeks For Geeks. (2019). *Different Shells in Linux*. Recuperado de https://www.geeksforgeeks.org/different-shells-in-Linux

Pedamkar, P. (s.f.). *Shell Script Parameters*. Recuperado de https://www.educba.com/shell-script-parameters/

Universidad Tecmilenio

Operaciones y Desarrollo: DevOps

Fundamentos de Linux y scripting en Bash. Parte 2

Semana 2

Las distribuciones de Linux utilizan programas o grupos de programas precompilados que se llaman paquetes, los cuales están listos para instalarse en esa distribución.

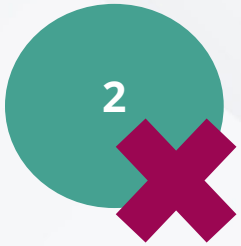
Bajo esta premisa se entiende que, para poder instalar, parchar y eliminar paquetes es importante conocer cómo se gestionan y cómo se les da seguimiento a las actualizaciones de dicho software en repositorios específicos de Linux.

Explicación

En la antigüedad de Linux se utilizaban archivos llamados Tarball, que se encargaban de recopilar todos los archivos necesarios para la gestión de un programa, así que el usuario manualmente se encargaba de descomprimir el tarballl, para utilizar la aplicación.

Esto desencadenaba dos desventajas:

Mala gestión de versiones



Dependencias con otras aplicaciones.

Explicación

Gestor de paquetes RPM

Los archivos RPM resuelven ambas desventajas, al encapsular metadatos con los archivos del software, los cuales incluyen:

- Número de versión.
- Lista de archivos dentro del paquete.
- Descripción del paquete.
- Información del empaquetador.

Cada paquete RPM tiene tres componentes:

- Archivos.
- Información asociada al paquete.
- Scripts.

Explicación

Comandos principales para los paquetes RPM (Bytemind, 2019):

- Instala y gestiona paquetes RPM.
- Instala y gestiona paquetes con herramientas gráficas.
- Para hacer consultas.
- Muestra la lista de archivos incluidos.
- Lista de archivos de configuración.
- Muestra la documentación del paquete.
- Instalación de paquetes.

Explicación

Editor VIM

Vim es un editor de texto modal basado en el editor de texto vi. Tiene distintas formas de insertar texto o manipularlo y está enfocado en los usuarios que utilizan la terminal para hacer varias tareas (Carles, 2021).

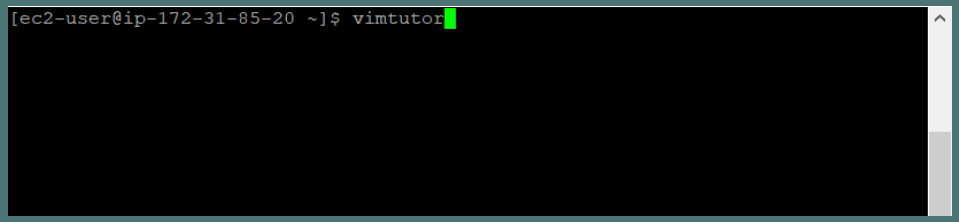
Modos de operación:

- Normal
- Inserción
- Reemplazo
- Visual
- Línea de comando

Ejercicio

Utilizando la línea de comandos de Linux, realiza el tutorial de VIM.

1. Abre la línea de comando en Linux.
2. Abre el tutorial vimtutor, escribiendo el comando vimtutor en la línea de comandos, como se muestra en la pantalla siguiente:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

3. En caso de no tener VIM, instálalo según la distribución de Linux que tengas.
4. Realiza las siete lecciones del tutorial.

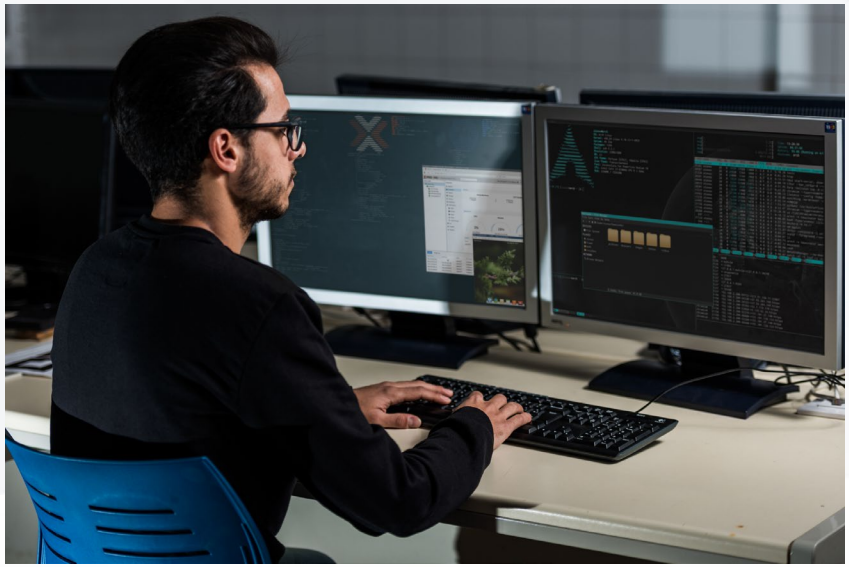
Universidad Tecmilenio

Cierre

Linux tiene diferentes herramientas para la gestión del sistema operativo, como RPM, que da las ventajas de instalar, eliminar, actualizar y verificar paquetes de software. A su vez, puede utilizar una base de datos de paquetes instalados para verificarlos y obtener información sobre ellos.

Otra herramienta importante es VIM, no solo por la flexibilidad de manipular documentos, sino porque puede trabajar sin necesidad de usar otra cosa dentro del teclado.

Puede usarse para construir y editar archivos de texto y de código.



Bibliografía

Bytemind. (2019). *Comando RPM - 15 ejemplos para manejar paquetes RPM.* Recuperado de https://byte-mind.net/comando-rpm-15-ejemplos-para-manejar-paquetes-rpm/

Carles, J. (2021). *Atajos de teclado y comandos para usar VIM eficientemente*. Recuperado de https://geekland.eu/atajos-de-teclado-y-comandos-para-usar-vim-eficientemente/

Universidad Tecmilenio

Operaciones y Desarrollo: DevOps

Lenguajes de programación (scripting). Parte 1

Semana 2

Python es uno de los lenguajes que ha tomado protagonismo en los últimos años, ya que está orientado a objetos, con él se pueden hacer muchas cosas, como *front end* y *back end* de una aplicación o página, local o en línea, análisis de datos, análisis numéricos, investigaciones científicas, inteligencia artificial, videojuegos, etc.

Así que dominar este lenguaje podría convertir a cualquier desarrollador en alguien experto en todo tipo de programación.

Explicación

Las características más importantes de Python son las siguientes:

- Es fuente abierta (open source).
- Incrementa la productividad.
- Integra trabajos con otros lenguajes.
- Tiene librerías extensas.
- *Machine learning* y *data science.*

Explicación

Las estructuras de control son bloques que permiten agrupar instrucciones de manera controlada.

Estructuras de control condicionales.	Permiten evaluar si una o más condiciones se cumplen para decir qué acción vas a ejecutar (verdadero o falso). Operadores relacionales. Operadores lógicos.
Estructuras de control iterativas o de repetición.	Permiten ejecutar un mismo código de manera repetida, mientras se cumpla una condición, ya sea verdadera o falsa. Bucle while. Bucle for.

Explicación

Los conceptos de Python se manejan diferente a otros lenguajes como C++ o C#, o incluso Java.

Variables

Las variables pueden guardar diferentes tipos, como cadenas de caracteres (*strings*), números enteros, números de punto flotante y booleanos.

Una de las ventajas de este lenguaje es que el tipo de variable se asigna automáticamente dependiendo del dato que se va a guardar y puede cambiar a lo largo de la codificación (El Libro de Python, 2022).

```
entero = 10
flotante = 3.1416
nombre ='Python'

print(entero," " , flotante , " " , nombre)
```

Funciones

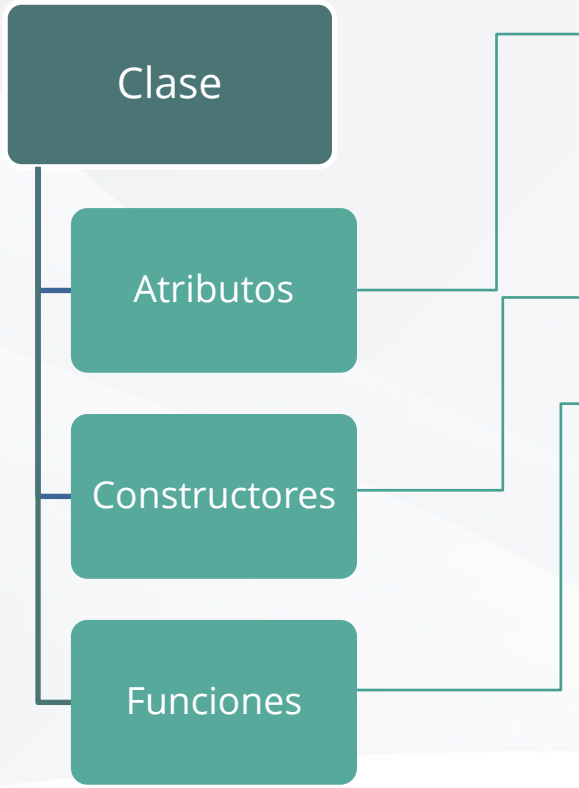
Una función es un bloque de código que se ejecuta solo cuando se le llama. A estas funciones se les pueden pasar datos a los que se les conoce como parámetros. Para crear una función, tienes que utilizar la palabra reservada “def”.

```
def Mi_Funcion(nombre):
    print(nombre,"un saludo")


Mi_Funcion("Luis") #Llamado de la función
Mi_Funcion("Paco") #Llamado de la función
```

Universidad Tecmilenio

Explicación

Python es un lenguaje que se utiliza en la programación orientada a objetos y se puede decir que todo en Python es un objeto con sus propiedades y métodos. Las clases se crean para poder crear objetos.




```python
class Empleado:
#atributos
    cedula = "L12345"
    nombre = "Pedro"
    apellido = "Hernandez"
    sexo = "M"
```

```python
class Enemigo:

    def __init__(self):
        self.nivel = 0

    def getNivel(self): #getter
        return self.nivel

    def setNivel(self,nivel): #setter
        self.nivel = nivel

caballero = Enemigo()
bruja = Enemigo()

caballero.setNivel(5)
bruja.setNivel(10)

print(caballero.getNivel())
print(bruja.getNivel())
```

Explicación

Herencia

La herencia es uno de los pilares de la programación orientada a objetos. Esto significa que una clase puede heredar propiedades y funciones de otra clase.

La clase padre o clase base es la clase de la cual se está heredando y la clase hijo o clase derivada es la clase que está heredando.

La herencia evita que se repitan propiedades o funciones que ya se tienen en otras clases (Lemonaki, (2022).

```
class Enemigo:

    def nombreEnemigo(self):
        print("no tiene nombre!!!")

class Caballero(Enemigo): #Herencia

    def getNombre(self):
        print("es un caballero")

enemigo1 = Caballero()
enemigo1.nombreEnemigo() #función heredada de la clase Enemigo
enemigo1.getNombre()
```

Universidad Tecmilenio

Explicación

Arreglos

Un arreglo es una estructura de datos que almacena múltiples valores del mismo tipo de objeto.

Es un contenedor ordenado que aloja un cierto número de objetos del mismo tipo, por lo que se utiliza para almacenar varios valores al mismo tiempo y los asigna a una sola variable

```
import numpy as np


# Create a numpy array from a list of numbers
arr = np.array([11, 19, 18, 14, 15, 11, 19, 21
, 46, 29, 21, 19])


result = np.where((arr > 15) & (arr < 21))


print(result)
```

Diccionarios

Un diccionario es una estructura de datos que asocia o mapea un conjunto de llaves con sus valores correspondientes. Es un conjunto desordenado de pares llave-valor.

Para crear un diccionario, se requiere de una lista de pares llave-valor separados por comas y encerrados en corchetes.

```
from itertools import chain
from collections import defaultdict

lista_lunes = {
    'manzanas': 1,
    'naranjas': 2,
    'kiwis': 2,
    'salmón': 0.5,
    'berenjena': 5,
    'guisantes': 0.5
    }


lista_martes = {
    'manzanas': 1,
    'naranjas': 0.5,
    'salmón': 1,
    'guisantes': 2,
    'alcachofas': 1.5,
    }
```

Ejercicio

Utilizando el archivo de la actividad Act6.py, practica los diferentes comandos que se mencionan.

1. Abre el archivo Act6.py.
2. Ejecuta el contenido y, al finalizar, verás el mensaje Fin de la actividad 6.
3. Identifica las líneas de código que trabajan con diferentes tipos de variables; cópialas y ejecútalas en tu editor.
4. Identifica las líneas de código que trabajan con impresión de valores en pantalla; cópialas y ejecútalas en tu editor.
5. Identifica las líneas de código que trabajan funciones; cópialas y ejecútalas en tu editor.
6. Identifica las líneas de código que trabajan con elementos de control de flujo; cópialas y ejecútalas en tu editor.
7. Identifica las líneas de código que trabajan con tuplas; cópialas y ejecútalas en tu editor.
8. Identifica las líneas de código que trabajan con listas; cópialas y ejecútalas en tu editor.

Cierre

Python es un lenguaje que admite la programación orientada a objetos y la programación funcional. Tiene características de administración de memoria dinámica que pueden hacer uso de los recursos computacionales de manera eficiente.

También es compatible con todas las plataformas populares y los sistemas operativos. Esta herramienta, por su flexibilidad, puede ser aceptada universalmente por todos los programadores.

Bibliografía

El Libro De Python. (2022). *Hash en Python*. Recuperado de https://ellibrodepython.com/hash-python

Lemonaki, D. (2022). *Python array tutorial – define, index, methods*. Recuperado de https://www.freecodecamp.org/news/python-array-tutorial-define-index-methods/

Universidad Tecmilenio

Operaciones y Desarrollo: DevOps

Lenguajes de programación (scripting). Parte 2

Semana 2

Antes de lanzar un programa o una aplicación, existe una fase muy importante en donde se realizan las pruebas, ya que esto determinará si un programa es estable y cumple con las funciones por las cuales se creó. Para ello, en Python hay distintas herramientas con las que se pueden realizar pruebas unitarias que permiten probar unidades independientes de código, ya sea un método, una clase o un módulo.

Explicación

Clase abstracta

En Python existe una clase particular llamada clase abstracta, la cual contiene métodos abstractos, que tienen una declaración, pero no una implementación en la clase base, que sirve para crear estructuras o interfaces que se pasan a todas las clases derivadas por medio de la herencia.

Python cuenta con un módulo para definirlas llamado ABC. Estas clases no pueden ser instanciadas por la creación de objetos (Python basics, s.f.).

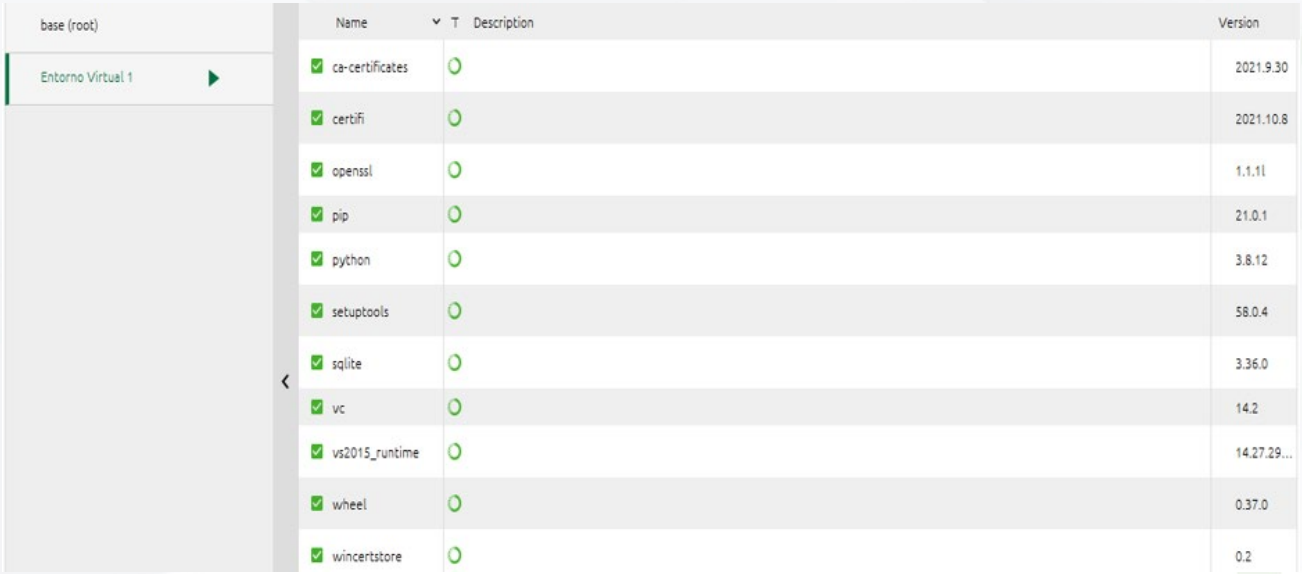
```
from abc import ABC, abstractmethod

class Polygono(ABC):

    @abstractmethod
    def NumeroDeLados(self):
        pass

class Triangulo(Polygono):

    # overriding abstract method
    def NumeroDeLados(self):
        print("Tiene 3 lados")

class Pentagono(Polygono):

    # overriding abstract method
    def NumeroDeLados(self):
        print("Tiene 5 lados")

class Hexagono(Polygono):

    # overriding abstract method
    def NumeroDeLados(self):
        print("Tiene 6 lados")

class Cuadrilatero(Polygono):

    # overriding abstract method
    def NumeroDeLados(self):
        print("Tiene 4 lados")

# Ejecución del código
figura = Triangulo()
figura.NumeroDeLados()

figura2 = Cuadrilatero()
figura2.NumeroDeLados()

figura = Pentagono()
figura.NumeroDeLados()

figura2 = Hexagono()
figura2.NumeroDeLados()
```

Explicación

Entornos virtuales

Los entornos virtuales son la herramienta perfecta para evitar las incompatibilidades, es decir, si se desea trabajar en paralelo con distintas versiones de Python se logra con entornos virtuales, ya que todo es aislado e independiente (librerías, recursos, etc.).

Existen varias herramientas para crear entornos virtuales, sin embargo, Anaconda es una de las mejores para trabajar con Python.



Explicación

Las pruebas unitarias sirven para determinar la correctitud de una unidad de código de forma aislada, ya sea clase, conjuntos de clases, método, módulo o componente.

Estas pruebas son para reducir los *bugs* en un software y en Python existen estas herramientas para realizarlas.

- Unittest
- Doctest
- Pytest
- Nose
- Hypothesis
- Mimesis
- Testify

Explicación

Web scraping

En la programación actual, el *web scraping* juega un papel muy importante, pues es una práctica muy particular en la que se busca información específica de un servidor web, donde se aloja la información requerida; toma únicamente los datos que se necesitan como información y la muestra con un formato configurado. Todo esto en periodos de tiempo muy cortos.

Solicitud de contenido.

Descarga la información.

Identifica lo que se desea extraer.

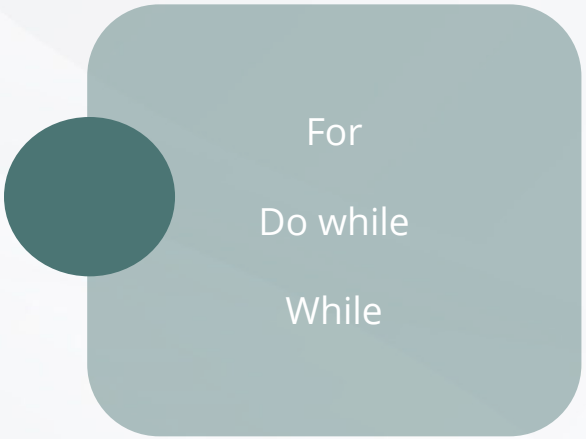
Extraer y dar formato para la visualización.

Explicación

Declaración de iteración

Los scripts en bash y Python son dos alternativas para programar y automatizar un sistema operativo, pero en ambas se deben declarar esos comandos que el sistema operativo o el programa deben realizar por medio de un control de flujos.

En Python, el control de flujos permite elegir entre diferentes alternativas a partir de una condicional.



Ejercicio

Realiza lo que se pide utilizando el entorno de trabajo Jupyter Notebook y la información contenida en el sitio web:

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.
https://www.commonsensemedia.org/movie-reviews

1. Extrae la siguiente información de las primeras 50 películas utilizando BeautifulSoup: título, calificación, breve descripción de la película y el año de publicación.
2. Guarda la información en un archivo csv.
3. Identifica una oportunidad en dónde utilizar web scraping y escribe el algoritmo para resolverlo usando una librería distinta a BeautifulSoup.
4. Almacena los resultados obtenidos del paso anterior en un archivo csv.

Cierre

Las pruebas unitarias garantizan que el código cumpla con los estándares de calidad antes de implementarse. Esto proporciona un entorno de ingeniería confiable, donde la calidad es primordial.

El web scraping se ha convertido en una herramienta invaluable para recopilar datos en la industria, ya que permite tener acceso automático a información crucial, como contactos, productos, comparación de precios, análisis de sentimientos, correos electrónicos, opiniones, revisiones, entre otros.

Bibliografía

Python basics. (s.f.). *Learn Python Programming*. Recuperado de https://pythonbasics.org/