DataDosis

# Programacion Python: Curso Completo

Por: Eric Alexander

# Curso Completo de Python 3:
## Que Haremos?

DataDosis

- Aprenderas Programacion **Python desde Cero a Experto**
- Crearemos **Paginas Web** con Python
- Aplicaciones con **Inteligencia Artificial**
- **API**’s con Python
- **Hacking Etico** con Python

# Curso Completo de Python 3:
## Que Aprenderas?

DataDosis

- **Al final podras:**
  - Crear **tu propia empresa** con Python
  - **Conseguir el Empleo** Mas Deseado en Programacion
  - Saber **Resolver** Cualquier **problema con Codigo Python**
  - **Crear Paginas Web**
  - **Crear Inteligencia Artificial**
  - Alimentar **Informacion** a Travez de un **API**
  - Crear **Software** para **Obtener Contrasenas** de Usuarios

# Curso Completo de Python 3:
## Que Obtendras?

DataDosis

- **Que Obtendras con el Curso?**
  - Este Material de Descarga
  - Mas de 100 videos
  - Mas de 20+ Horas de Contenido
  - Acceso Exclusivo a Preguntas y Comunidad

# Curso Completo de Python 3:

## Porque Python

DataDosis

- Porque usar Python?
  - Disenado para ser claro, con logica facil de leer
  - Existen miles de librerias escritas en python que nos permiten crear aplicaciones complejas sin tener que esforzarnos mucho
- Python esta enfocado en optimizer el tiempo de Desarrollo, no esta disenado para optimizer la computadora (por eso no usamos python para video juegos)
- Buena Documentacion online: docs.python.org/3

# Curso Completo de Python 3:
## Porque Python

DataDosis

- Que puedo lograr con Python?
  - Primero aprenderemos una base solida del lenguaje
  - Luego pasaremos a crear Aplicaciones con logica compleja, hackear contrasenas y haremos inteligencia artificial!
- Python es muy versatil, enfocado mas que nada para ciencia de datos

# Curso Completo de Python 3:
## Porque Python

DataDosis

- Usamos Python para Automatizar tareas simples
  - Buscar y editar archivos
  - Buscar informacion de internet
  - Leer y editar archivos excel
  - Trabajar con PDF’s
  - Automatizar Emails y Mensajes de Texto
  - Rellenar Formularios

# Curso Completo de Python 3:
## Porque Python

DataDosis

- Usamos Python para Ciencia de Datos y Machine Learning
  - Analiza Grandes Datos (Big Data)
  - Crea Visualizaciones
  - Tareas de Machine Learning
  - Crea algoritmos de prediccion

# Curso Completo de Python 3:
## Porque Python

DataDosis

- Usamos Python para Desarrollo Web
  - Crea Paginas Web
    - Usa Django o Flask para manejar el backend de tu sitiio web muy facilmente
    - Crea dashboards interactivos
- Desarrolla aplicaciones avanzadas con API para intercomunicar muchas aplicaciones

# Curso Completo de Python 3:

## Linea de Comandos

DataDosis

- Antes de trabajar con python debemos aprender a usar la linea de commandos
- Es aqui donde vamos a corer nuestros archivos que escribiremos con codigo python

- La linea de commandos es muy importante, te recomiendo pasar un buen rato jugando con el cmd

# Curso Completo de Python 3:

## Instalando Python (Local)

DataDosis

- Python debe ser instalado para poder correr los archivos escritos en .py

- Usamos python.org para poder descargar python, hoy en dia Windows, Mac y Linux traen Python pre instalado

# Curso Completo de Python 3:

## Instalando Python (Ambiente Virtual)

DataDosis

- Usamos los ambientes virtuales cuando deseamos crear aplicaciones

- Las apps que creamos usan librerias y versions de diferentes tipos. Un app puede tenerr version 1.x y otra app que creamos puede ser version 2.x, esto crea conflict con nuestras dependencias

- Debido a ello es que usamos Ambientes Virtuales para programar correctamente

- La manera mas facil de instalar un ambiente de python es con Anaconda

# Curso Completo de Python 3:
## IDEs para Python

DataDosis

- Visual Studio Code

- Spyder

- Jupyter Notebook

DataDosis

# Estructuras de Datos

- Cubriremos tipos de datos basicos
- La programacion se basa en matematicas y logica, necesitamos guardar la informacion
- Estos “Tipos de Datos” son la base de codigo mas Avanzado

- Veamos rapidamente todos los tipos de datos que temenos en python, luego Podemos ver uno por uno

# Curso Completo de Python 3:
## Estructuras de Datos

DataDosis

| Nombre | Tipo | Descripcion |
|---|---|---|
| Enteros | Int | 3  500  50000 |
| Punto Flotante | Float | 2.5  6.76  56.856 |
| Cadenas de Texto | Str | “hola” ‘eric’ “20,000” |
| Listas | List | [10,”hola”, 200.4] |
| Diccionarios | Dict | {“palabra”:”significado”, “clave”:”valor”} |
| Tuplas | Tup | Orden Inmutable de Objetos: (10,”hola”,200.4) |
| Sets | Set | Coleccion de Objetos no ordenados {“a”,”b”} |
| Booleanos (Verdadero o Falso) | bool | True o False |

# Curso Completo de Python 3:
## Variables

DataDosis

- Acabamos de trabajar con numeros, pero es dificil saber quue representan estos numeros si no les asignamos una variable

- Seria Bueno asignar a estos tipos de datos un nombre par poder reconocerlos

- **Por ejemplo:**
  - *Mis_casas = 2*

# Curso Completo de Python 3:
## Variables

DataDosis

- Reglas para nombrar Variables
  - *Se considera Buena practica usar nombres en minuscula*
  - *Evita usar palabras con significado especial como “list” o “str”*

- Python usa nomenclatura Dinamica
  - Significa que puedes re-asignar variables a otros tipos de datos
  - Esto hace python bastante flexible al momento de asignar variables

```
Mis_casas = 2
Mis_casas = [“casa1”,”casa2”]
```

Esto es correcto en python!

# Curso Completo de Python 3:
## Variables

DataDosis

- Pros de Nomenclatura Dinamica
  - Facil de Trabajar
  - Rapido de desarrollar

- Cons
  - Resulta en errrores de datos inesperados
  - Cuidado con type()

- A veces debes especificar el tipo de variable con la que trabajaras

# Curso Completo de Python 3:

## Cadenas de Texto (Strings)

DataDosis

- Las cadenas son secuencias de caracteres, usan sintaxis con comillas singulars o doble comillas
  - ‘hola’
  - “hola”
  - Caracter: “ cadena de texto con muchos caracteres “
  - Indice: 0 1 2 3 4 5
  - Indice Inversio: 0 -5 -4 -3 -2 -1

- Debido a que las cadenas son **secuencias ordenadas** significa que Podemos usar **indexado** y **slicing** para agarrar sub-secciones de una cadena
- **Notacion de Indexado:** [], asignado despues de la cadena
- **Indexado** te permite agarrar un character singular de una cadena de texto

# Curso Completo de Python 3:

## Cadenas de Texto (Strings)

DataDosis

- Slicing permite agarrar una subseccion de multipes caracteres, un “slice” de una cadena.
- Tiene la siguiente sintaxis:
  - [start:stop:step] -> [0:4:2]
- **Start** es un indice numeric para el slice iniciar
- **Stop** es el numero donde paramos el slice
- **Step** son los pasos que damos

# Curso Completo de Python 3:

## Formato de Impresion de Cadenas de Texto

DataDosis

- Usualmente queremos “inyectar” una variable en una cadena para imprimir.
- Por ejemplo:
  - mi_nombre = “Eric”
  - Print(“Hola “ + mi_nombre)

- Hay multiples maneras de formatear cadenas para imprimir variables en ellas
- A esto se le cice “interpolacion de cadenas”

# Curso Completo de Python 3:

## Formato de Impresion de Cadenas de Texto

DataDosis

- Exploremos dos metodos para hacer esta interpolacion:
  - .format() *metodo*
  - F-strings *literales de cadena formateado*

# Curso Completo de Python 3:
## Listas

DataDosis

- Listas son secuencias ordenadas que guardan una una variedad de tiipos de objetos
- Usan [] braquets y commas para separar objetos en una lista
  - **[1,2,3,4,5]**

- **Listas soportan Indices y Slicing**
- Puedes tener listas adentro de listas y pueden guardar metodos que pueden ser llamados

# Curso Completo de Python 3:
## Diccionarios

DataDosis

- Mapeos desordenados para guardar objetos
- Previamente vimos como las listas guardan objetos en una secuancia ordenada,
- Los diccionarios utilizan un orden basado en par Clave/Valor

- Este par Clave/Valor permite al usuario agarrar objetos sin necesidad de saber la locacion (numero) del indice

# Curso Completo de Python 3:

## Diccionarios

DataDosis

- Diccionarios usan brackets {} y : para simbolizar las llaves y su valor asociado

- Cuando deberiamos **escoger una lista o un diccionario?**

- **Dicccionarios**: Objetos retornados por llave
  - Desordenado yy no se guarda, Bueno para cuando no sabes donde se encuentra algo

- **Listas**: Objetos retornados por locacion
  - Puede ser Indice o Slicing

# Curso Completo de Python 3:

## Tuplas

DataDosis

- Tuplas son similar a las listas. Sin embargo, tienen una diferencia – **immutabilidad**.

- Una vez que un element se encuentra en una tupla, este no puede ser re-asignado

- Las tuplas usan parentesis: **(1,2,3)**

# Curso Completo de Python 3:
## Sets

DataDosis

- Coleccion Unica y Desordenada de Elementos

- **Solamente puede haber UNA representacion del mismo objeto**

DataDosis

# Comparadores de Operacion

# Curso Completo de Python 3:
## Comparadores de Operacion

DataDosis

| Operador | Descripcion | Ejemplo |
|---|---|---|
| == | Si los valores de dos operandos son iguales, la condicion es verdadera | (a == b) NO es verdadero |
| != | Valores de dos operandos no son iguales, condicion es igual | (a != b) es Verdadero |
| > | Si los valores del operando izquierdo son mayores que el de la derecha entonce la condicion es verdadero | (1 > 2) no es verdadero |
| < | Si el valor de operando izquierdo es menor que el derecho, es verdadero | ( 1 < 2) verdadero |
| >= | Si el valor del operando izquierdo es mayor o igual queel de la derecho, entonces es verdadero | ( a >= b ) no es verdadero |
| <= | Si el operando izquiedo es menor que el derecho entonces condicion es verdadera | ( a <= b ) es verdadero |

# Curso Completo de Python 3:

## Encadenando Comparadores de Operacion

DataDosis

- Podemos usar operadores logicos para combiner comparasiones:
  - And
  - Or
  - Not

DataDosis

# Declaraciones

# Curso Completo de Python 3:

## Declaracion If, Elif y Else

DataDosis

- Usamos las declaraciones para controlar el flujo de nuestra aplicacion
- Usualmente solo queremos que cierto codigo sea ejecutado ccuando una condicion particular ocurra

- Por ejemplo, IF mi perro tiene hambre (Aplicar logica),  ELSE (apliicar logica si perro no tiene hambre)

# Curso Completo de Python 3:

## Declaracion If, Elif y Else

DataDosis

- Para controlar este flujo de logica usamos palabras clave:
  - If
  - Elif
  - else

# Curso Completo de Python 3:

## Declaracion If, Elif y Else

DataDosis

- Python usa un Sistema de indentacion al momento que declaramos funciones y logica con IF.

- Se llama “Control Flow Syntax” y usamos : e Indentacioon (espacios en blanco)

- Este Sistema de Indentacion es muy importante y es lo que separa a python del resto de lenguajes

# Curso Completo de Python 3:

## Declaracion If, Elif y Else

DataDosis

- Sintaxis de una declaracion **IF**

```
If alguna_condicion:
    # Ejecutamos codigo
```

- Sintaxis de una declaracion **IF/ELSE**

```
If alguna_condicion:
    # Ejecutamos codigo
else:
    # Aplicar algo mas
```

# Curso Completo de Python 3:
## Declaracion If, Elif y Else

DataDosis

- Sintaxis de una declaracion **ELIF**

```
If alguna_condicion:
    # Ejecutamos codigo
elif alguna_otra_condicion:
    # algo distinto
else:
    # hacer algo mas
```

# Curso Completo de Python 3:
## Ciclos For

DataDosis

- Varios objetos en python son “iterables”, significa que Podemos iterar sobre cada element en el objeto

- Podemos iterar a travez de una Lista o Todos los caracteres en una Cadena de Texto

- Podemos usar Ciclos FOR para ejecutarr un bloquue de codigo en cada iteracion

# Curso Completo de Python 3:
## Ciclos For

DataDosis

- El termino iterable significa que puedes “**iterar**” sobre cada objeto.

- Por ejemplo: Podemos iterar a travez de cada character en una **cadena de texto**, iterar sobre todos los items de una **lista**, iterar sobre todas las llaves de un **diccionario**

# Curso Completo de Python 3:
## Ciclos For

DataDosis

- Sintaxis para un Ciclo FOR

```
lista_iterable = [1,2,3]

For nombre_item in lista_iterable:
    print(nombre_item)
```

# Curso Completo de Python 3:
## Ciclos WHILE

DataDosis

- Los ciclos while van a continuar ejecutando un bloque de codigo **while (mientras)** una condiciion ssiga siendo verdadera

- Por ejemplo, **while** mi carro no tenga gasolia, sigue hechando gas

- O **while** tenga hambre, comer alimentos

# Curso Completo de Python 3:
## Ciclos WHILE

DataDosis

- Sintaxis para ciclo WHILE

```
while condicion_booleana:
    # hacer algo
```

- Puedes combinar con else:

```
while condicion_booleana:
    # hacer algo
else:
    # hacer algo distinto
```

# Curso Completo de Python 3:
## Ciclos WHILE

DataDosis

- Sintaxis para ciclo WHILE

```
while condicion_booleana:
    # hacer algo
```

- Puedes combinar con else:

```
while condicion_booleana:
    # hacer algo
else:
    # hacer algo distinto
```

# Curso Completo de Python 3:
## Comprehension de Listas

DataDosis

- Manera unica de crear una lista de Python rapidamente

- Si te encuentras usando un ciclo for con .append() para crear una lista, puedes usar una comprehension en su lugar

DataDosis

# Metodos y Documentacion de Python

# Curso Completo de Python 3:
## Metodos

- Objetos creados en python con una variedad de metodos que Podemos usar

Exploremos con mas detalle como Podemos encontrar estos metodos

# Curso Completo de Python 3:

## Funciones

DataDosis

- Crear codigo limpio, ordenado y repetible es muy importante para nosotros ser efectivos programando

- **Funciones** nos permiten crearr bloques de codigo que Podemos ejecutar varias veces, sin necesidad de reescribir codigo redundante

# Curso Completo de Python 3:
## Funciones

DataDosis

- Las funciones son un gran salto en tu Carrera como programador python

- Esto significa que los problemas se pondran mas dificiles!

- Combinaremos todo lo que sabemos con funciones

# Curso Completo de Python 3:
## Funciones (Sintaxis)

DataDosis

- Crear funciones require de una sintaxis especial, empezamos con **def**

- **Veamos una function:**

```
def nombre_de_function(nombre):
    # Corremos codigo
    print("Las funciones retornan algo" + nombre)
```

# Curso Completo de Python 3:

## Funciones (Sintaxis)

DataDosis

- Tipicamente usamos la palabra clave return para retornar el valor de una function

- Return nos permite asignar un output de la function

- **Funcion de Suma:**

```
def suma(num1,num2):
    return num1+num2
```

# Curso Completo de Python 3:
## Argumentos (*args y **kwargs)

DataDosis

- Arguments y Keyword Arguments

DataDosis

# Expresiones Lambda, Mapas y Filtros

DataDosis

# Programacion Orientada a Objetos

# Curso Completo de Python 3:

## Programacion Orientada a Objetos

DataDosis

- Permite a los programadores crear sus propios objetos que tienen metodos y atributos

- Podemos llamar distintos metodos que se encuentran en una clase.

- Los metodos actuan como funciones que usan informacion sobre el objeto para poder retornar resultados

# Curso Completo de Python 3:
## Programacion Orientada a Objetos

DataDosis

- Podemos crear nuestros propios objetos

- Podemos crear codigo repetible y organizado

# Curso Completo de Python 3:

## Programacion Orientada a Objetos

DataDosis

- Para los libretos grandes de python necesitamos algo mas que solo funciones para organizar todo

- Tareas repetidas son definidas en clases para evitar la redundancia de codigo.

- Veamos un ejemplo

# Curso Completo de Python 3:
## Programacion Orientada a Objetos

DataDosis

```
Class NombreDeClase():
        def __init__(self,param1,param2):
                self.param1 = param1
                self.param2 = param2

        def otraFuncion(self):
                #accion
                print(self.param1)
```

DataDosis

# Modulos y Paquetes

# Curso Completo de Python 3:

## Pip install y PyPi

DataDosis

- Hasta el momento hemos usado librerias internas de python

- Hay varias librerias externas disponibles para descargar usando un sitio web como **PyPi**

- Tambien usamos el commando **pip install** en la linea de commandos para instalar estos paquetes

# Curso Completo de Python 3:

## Pip install y PyPi

DataDosis

- Cuando instalamos python a travez de python.org Tambien trae con este **pip**

- Si te encuentras en Mac o Linux y deseas instalar pip puedes corer el commando:
  - Sudo apt-get python-pip

- Pip instala directo del repositorio de **PyPi**

# Curso Completo de Python 3:
## Pip install y PyPi

DataDosis

- Existen paquetes creado para casi todos los usos que te imaginas

    - **Desarrollo Web**
        - Flask
        - Django
    - **Ciencia de Datos**
        - Tensorflow
        - Matplotlib
        - Pandas
        - Numpy
    - **Hacking Etico**
        - Scapy

# Curso Completo de Python 3:
## Pip install y PyPi

DataDosis

- Aprendamos ahora a instalar paquetes usando Pip

- Tambien Podemos instalar paquetes usando Conda

- https://pypi.org/

# Curso Completo de Python 3:
## Modulos y Paquetes

DataDosis

- Ahora que entendemos como instalar paquetes externos, exploremos como crear nuestros propios modulos y paquetes

- **Modulos** son simples archivos .py que llamamos desde otro archive

- **Paquetes** son una coleccion de modulos

- Usamos esto bastante en Programacion Web y en cualquier App con logica mas compleja

# Curso Completo de Python 3:
## __name__ y “__main__”

DataDosis

- Cuando corremos codigo Avanzado descargado del internet muchas veces vemos esta linea de codigo en la parte de abajo
  - If __name__ == “__main__:

- Cuando importamos un modulo queremos saber si las funciones usadas estan siendo usadas como import o si estas el original del archive .py del modulo

- Exploremos esto en codigo para entenderlo mejor!

DataDosis

# Manejo de Errores y Excepciones

# Curso Completo de Python 3:

## Errores y Excepciones

DataDosis

- Eventualmente algo en nuestro codigo se va a romper, sobretodo si le damos el codigo a alguien mas para que use el programa y no pre-vemos el possible escenario de uso

- Podemos usar manejo de errores para poder planear posibles casos de uso donde ocurra un error

# Curso Completo de Python 3:
## Errores y Excepciones

DataDosis

- Usamos las palabras clave:

- **Try**: Esto bloquea el codigo de ser corrido (puede dar error)

- **Except**: Bloque de codigo es ejecutado en caso de haberr un error en el bloque Try

- **Finally**: bloque de codigo ejecutado finalmente sin importar el error