Guia Ágil

TechGuide - Alura

Python

Nivel 1

- [ ] **Python - Fundamentos**:

- Python es un lenguaje de programación de alto nivel de uso general, ampliamente utilizado en aplicaciones web, desarrollo de software, ciencia de datos y aprendizaje automático. Su filosofía de diseño se centra en la legibilidad del código mediante el uso de sangría significativa. Python es de tipado dinámico y cuenta con un recolector de basura.
- Los tipos de datos primitivos.
- Declarar variables, teniendo en cuenta los diferentes tipos.
- Utilizar estructuras condicionales ('if', 'else').
- Conocer los operadores de asignación y comparación.
- Usar estructuras de repetición y bucles ('while', 'for').
- Utilizar funciones, pasando parámetros y argumentos.
- Manipular métodos.
- Manipular arrays y listas.
- Obtener datos de una API.
- Crear constructores.
- Utilizar funciones anónimas.

- [ ] **Conceptos de Orientación a Objetos**:
  - La Programación Orientada a Objetos es un paradigma de programación de software basado en la composición e interacción entre diversas unidades llamadas 'objetos' y las clases, que contienen una identidad, propiedades y métodos. Se basa en cuatro componentes de la programación - abstracción digital, encapsulación, herencia y polimorfismo.
  - Cómo funcionan los objetos
  - Crear y utilizar constructores
  - Qué son las clases
  - Crear y utilizar métodos
  - Cómo funciona la encapsulación
  - Qué es la herencia
  - Qué es el polimorfismo
  - Cómo funcionan las interfaces
  - Qué son las abstracciones
- [ ] **Estructura de Datos**:
  - En el contexto de los ordenadores, una estructura de datos es una forma específica de almacenar y organizar los datos en la memoria del ordenador para que esos datos puedan ser fácilmente recuperados y utilizados de forma eficiente cuando sea necesario posteriormente.
  - Conocer las principales estructuras de datos
  - Implementar las principales estructuras de datos
- [ ] **Python - Colecciones**:
  - Una colección representa un grupo de objetos, conocidos como sus elementos. Son como contenedores que agrupan varios elementos en una única unidad. Algunas colecciones permiten la duplicación de elementos y otras no. Algunas están ordenadas y otras no lo están.
  - Aprender a usar listas y tuplas
  - Aprovechar el polimorfismo en las colecciones

- Utilizar conjuntos y diccionarios.

- [ ] **Python - Pruebas**:

- Las pruebas de software son el proceso de evaluación y verificación de que un software realmente hace lo que debería hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento
- Pruebas unitarias
- Pruebas de integración
- Pruebas de comportamiento (behavior)
- Mocks

- [ ] **Python - Comunicación con APIs**:

- Una API es una interfaz que los desarrolladores de software utilizan para programar la interacción con componentes o recursos de software fuera de su propio código. Una definición aún más simple es que una API es la parte de un componente de software que es accesible para otros componentes.
- Comprender qué es una API REST
- Conocer los comandos básicos de comunicación HTTP
- Entender qué es una API REST
- Saber cómo hacer solicitudes autenticadas
- Convertir objetos a JSON y viceversa
- Saber cómo utilizar las herramientas del paquete Requests.

Nivel 2

- [ ] **Flask**:

- Flask es un pequeño framework web escrito en Python. Se clasifica como un microframework porque no requiere herramientas o bibliotecas particulares, manteniendo un núcleo simple pero extensible. No posee una capa de abstracción de base de datos, validación de formularios u otros componentes, ya que bibliotecas de terceros proporcionan funciones

comunes. Sin embargo, Flask ofrece soporte a extensiones que pueden agregar recursos a la aplicación como si fueran implementados en Flask mismo

- Crear aplicaciones web
- Definir rutas, redirecciones y plantillas
- Validar formularios

- [ ] **Python - Programación Orientada a Objetos Avanzada**:

- Mixin es una clase que ofrece implementación de métodos para ser reutilizados por múltiples clases hijas relacionadas.
- Sobrecarga de operador significa dar un significado extendido además de su significado operacional predefinido

- [ ] **Django**:

- Django es un framework web de alto nivel en Python que permite el rápido desarrollo de sitios web seguros y de fácil mantenimiento.
- Crear una aplicación web
- Comprender la arquitectura de una aplicación desarrollada con Django
- Crear el panel de administración de una página
- Utilizar plantillas y rutas
- Crear formularios

- [ ] **Django Rest Framework**:

- Django REST Framework es un conjunto de herramientas poderosas y flexibles para la construcción de APIs.
- Desarrollar APIs
- Trabajar con modelos, serializadores y vistas
- Incluir filtros, búsquedas y ordenación
- Limitar el número de solicitudes

- [ ] **Python - MVC y MTV (o MVT)**:

- MVC y MTV son dos padrones de proyecto (desing patterns) utilizados para implementar interfaces y aplicaciones web.
- Comprender el patrón MVC
- Comprender el patrón MTV
- Comprender la diferencia entre los patrones MVC y MTV

- [ ] **Python - Lambdas y Closures**:

- Las funciones lambda son funciones anónimas. Mientras que las funciones tradicionales pueden ser creadas utilizando "def" como prefijo, las funciones lambda se crean utilizando "lambda".
- Un cierre (closure) en Python es un objeto de función interna, es decir, una función que se comporta como un objeto, que recuerda y tiene acceso a las variables en el ámbito local en el que fue creado, incluso después de que la función externa haya terminado de ejecutarse. También puede ser definido como un mecanismo para conectar datos a una función sin la necesidad de pasar parámetros.

Nivel 3

- [ ] **Arquitectura de Microservicios**:

- Los microservicios son un enfoque de arquitectura en el que el software consiste en pequeños servicios independientes que se comunican entre sí y se organizan de acuerdo con sus dominios de negocio.
- Aprender el concepto de arquitectura diseñada para microservicios
- Realizar la comunicación mediante API
- Mejorar la escalabilidad de un sistema

- [ ] **Contenedores**:

- Los contenedores son paquetes de software que contienen todos los elementos necesarios para ejecutarse en cualquier entorno. La gestión de contenedores es un área crucial en la computación en nube y DevOps, que implica el uso de tecnologías para automatizar el proceso de creación, implementación, escalado y monitoreo de contenedores. Los contenedores

son unidades de software estandarizadas que permiten a los desarrolladores empaquetar todas las dependencias de una aplicación (código, bibliotecas, configuraciones, etc.) en un solo paquete. Esto permite que la aplicación se ejecute de forma consistente en cualquier entorno de infraestructura.

- La tecnología de contenedores, como ejemplifica Docker, proporciona un entorno coherente y portátil para el desarrollo, las pruebas y la implementación de aplicaciones, lo que es vital para el trabajo eficiente de la ingeniería de datos. Además, Kubernetes, un sistema de organización de contenedores, permite la gestión, automatización y escalabilidad de aplicaciones basadas en contenedores en entornos de producción. Dominar estos conceptos y tecnologías permite a los ingenieros de datos construir y mantener canalizaciones de datos eficientes y confiables.
- Kubernetes (también conocido como k8s o Kube) es una plataforma de orquestación de contenedores de código abierto que automatiza gran parte de los procesos manuales necesarios para implementar, gestionar y escalar aplicaciones en contenedores.
- Aislar el software para que funcione independientemente
- Implementación de software en clústeres
- Modularizar su sistema en paquetes más pequeños
- Conocer la plataforma Docker
- Conocer Kubernetes

- [ ] **Python - Tipado Estático**:

- Python es un lenguaje de tipado dinámico, lo que significa que no es necesario pensar en tipos de datos. Los lenguajes de tipado estático (como C o Java) realizan verificaciones de tipos durante la compilación. Puede parecer más seguro, ya que puede especificar inmediatamente el tipo de parámetro de cada función.
- Conocer el "type hinting"

- [ ] **Python - Generadores**:

- Los generadores permiten declarar una función que se comporta como un iterador, por ejemplo, se puede usar en un ciclo 'for'.
- Crear objetos iteradores
- Utilizar evaluación perezosa
- Realizar tareas simultáneas
- Uso de la palabra clave 'yield'

- [ ] **Python - Asíncrono**:

- En la programación asincrónica, las funciones no se ejecutan en orden. Con la asincronía, podemos interrumpir el código para obtener alguna otra información necesaria para continuar la ejecución. Esto significa que el código espera a otra parte del código y, mientras espera, ejecuta las demás partes.
- Aprender acerca de las corutinas
- Las corutinas son generalizaciones de subrutinas. Se utilizan para la multitarea cooperativa, donde un proceso cede el control de manera voluntaria, periódica o cuando está inactivo, para permitir que múltiples aplicaciones se ejecuten simultáneamente.
- Manejar la concurrencia
- Conocer el concepto de objetos esperables
- Crear tareas concurrentes
- Conocer la biblioteca 'asyncio'

- [ ] **Python - args & kwargs**:

- Las variables mágicas *args y **kwargs se utilizan comúnmente en la definición de una función y sirven para pasar un número desconocido de argumentos a una función.
- Comprender la diferencia entre *args y **kwargs

- [ ] **Python - Métodos especiales (dunder)**:

- Los métodos especiales, o métodos mágicos, en Python son métodos predefinidos en todos los objetos, con invocación automática en

circunstancias especiales. Normalmente, no se llaman directamente por el usuario, pero pueden ser sobrecargados (sobrescritos y modificados). Sus nombres comienzan y terminan con guiones bajos dobles llamados "dunder" (una expresión derivada de double underscore).

- Comprender el concepto de métodos especiales (o mágicos)
- Conocer los principales métodos mágicos y cómo usarlos

☐ **Python - Metaprogramación**:

- Metaprogramación es una técnica de programación en la que los programas tienen la capacidad de tratar a otros programas como sus datos. Esto significa que un programa puede estar diseñado para leer, generar, analizar o transformar otros programas, e incluso modificarse a sí mismo durante la ejecución.
- Escribir un programa que manipula otros programas
- Usar metaclasses

☐ **Python - Multiprocesamiento**:

- En Python, el módulo de multiprocesamiento incluye una API muy simple e intuitiva para dividir el trabajo entre varios procesos.
- Ejecutar procesos en paralelo
- Conocer la clase Pool

☐ **Reflection y atributos**:

- Los objetos de Reflection (reflexión) se utilizan para obtener información del tipo en tiempo de ejecución. Las clases que dan acceso a los metadatos de un programa en ejecución se encuentran en el espacio de nombres System.Reflection.
- Escribir código que lee la información y metadatos de objetos en tiempo de ejecución
- Obtener nombres de clases en tiempo de ejecución y crear objetos de una clase

Habilidade Auxiliar: Infraestructura

- [ ] **Git y GitHub - Fundamentos**:
  - Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia.
  - GitHub es un servicio de hosting para el desarrollo de software y el control de versiones mediante Git.
  - Crear un repositorio
  - Clonar un repositorio
  - Comprometerse, empujar y tirar hacia y desde el repositorio
  - Revertir un commit
  - Crear de ramas y Pull requests
  - Manejar fusiones y conflictos
- [ ] **HTTP - Fundamentos**:
  - HTTP significa Protocolo de transferencia de hipertexto. La comunicación entre las computadoras cliente y los servidores web se realiza mediante el envío de solicitudes HTTP y la recepción de respuestas HTTP.
  - Comprender la diferencia entre los verbos HTTP
  - Probar solicitudes y verificar los códigos de estado en el navegador
  - Aprendiendo a hacer una solicitud HTTP en la línea de comando con WGET
  - Descargar una imagen con WGET
  - Realización de una POST
- [ ] **JSON**:
  - JSON significa Notación de objetos de JavaScript. Es un formato de texto para almacenar y transportar datos.
  - Crear un objeto
  - Transformar un objeto en una cadena

- Transformar una cadena en un objeto
- Manipular un objeto

- [ ] **Línea de Comando - Fundamentos**:

- CLI es un programa de línea de comandos que acepta la entrada de texto para ejecutar funciones del sistema operativo.
- Conocer los comandos más importantes

- [ ] **Cloud - Fundamentos**:

- La computación en nube, o cloud computing, es la distribución de servicios informáticos a través de Internet mediante un modelo de tarificación de pago por uso. Una nube se compone de varios recursos informatizados, desde los propios ordenadores (o instancias, en terminología de nube) hasta las redes, el almacenamiento, las bases de datos y todo lo que les rodea. En otras palabras, todo lo que normalmente se necesita para montar el equivalente a una sala de servidores, o incluso un centro de datos completo, estará listo para usar, configurar y ejecutar.
- Conocer la diferencia entre IaaS, PaaS y SaaS
- Conocer los mayores proveedores de nube
- Especializarse en un proveedor específico de su preferencia

- [ ] **SQL - Fundamentos**:

- Conocer los comandos más comunes de SQL
- Usar SELECT para consultar una tabla
- Usar INSERT para insertar datos en una tabla
- Usar UPDATE para actualizar una tabla
- Usar DELETE para eliminar datos de una tabla
- Usar JOIN para conectar los datos de múltiples tablas
- Conocer las cláusulas (FROM, ORDER BY, etc.)

Habilidade Auxiliar: Buenas prácticas y herramientas

- [ ] **SOLID**:
  - Solid tiene cinco principios considerados como buenas prácticas en el desarrollo de software que ayudan a los programadores a escribir los códigos más limpios, dividiendo las responsabilidades, disminuyendo los acoplamientos, facilitando la refactorización y estimulando el reaprovechamiento del código. Propuesto por Robert C. Martin, SOLID propicia el desarrollo de un código limpio, legible y comprobable.
- [ ] **Clean Architecture**:
  - Clean Architecture (Arquitectura Limpia) es una forma de desarrollar software, de tal forma que solo mirando el código fuente de un programa, debes ser capaz de decir lo que el programa hace.
- [ ] **Design Patterns**:
  - En ingeniería de software, un "patrón de diseño" (Design Pattern en inglés) es una solución general y reutilizable para un problema que ocurre normalmente dentro de un determinado contexto de diseño de software.
  - Conocer y aplicar los principales patrones de diseño.
- [ ] **Jupyter y Colab**:
  - Jupyter Notebook y Google Colaboratory son portátiles que permiten la creación de bloques de texto y bloques de código
  - Los Notebooks facilitan la elaboración de proyectos de Data Science por ser posible visualizar el resultado de la ejecución luego del trecho de código
  - Google Colaboratory le permite escribir y ejecutar códigos Python directamente en el navegador, sin ninguna o pocas configuraciones necesarias
  - Facilitan el intercambio de proyectos entre el equipo
- [ ] **Extracción y tratamiento de datos**:

- Obtener los datos que se analizarán
- Tratar los datos obtenidos, transformándolos, alterando su estructura y valores a fin de dejar la base de datos más coherente y garantizar que los datos que serán trabajados estén en las mejores condiciones para ser analizados

---