Python

TechGuide - Alura, FIAP e PM3

Python

Nivel 1

- [ ] **Lógica de Programação**:
  - Aprender lógica de programação, fundamental para o desenvolvimento de software
  - Conhecer as bases para se criar, analisar e resolver problemas computacionais de forma estruturada e eficiente
  - Entender o que são tipos de dados
  - Declarar variáveis, considerando os diferentes tipos
  - Conhecer os operadores de atribuição e comparação
  - Usar estruturas condicionais
  - Usar estruturas de repetição e laços
  - Usar funções, passando parâmetros e argumentos
- [ ] **Python - Fundamentos**:
  - Python é uma linguagem de programação de alto nível, de uso geral, amplamente utilizada em aplicações web, desenvolvimento de software, ciência de dados e Machine Learning. Sua filosofia de projeto enfatiza a legibilidade do código com o uso de indentação significativa. Python é dinamicamente tipada e tem um garbage collector.
  - Conhecer os tipos primitivos

- Declarar variáveis, considerando os diferentes tipos
- Usar estruturas condicionais ('if', 'else')
- Conhecer os operadores de atribuição e comparação
- Usar estruturas de repetição e laços ('while', 'for')
- Usar funções, passando parâmetros e argumentos
- Manipular métodos
- Manipular arrays e listas
- Obter dados de uma API
- Criar construtores
- Funções anônimas

☐ **Conceitos de Orientação a Objetos**:

- A Programação Orientada a Objetos é um paradigma de programação de software baseado na composição e interação entre diversas unidades chamadas de 'objetos' e as classes, que contêm uma identidade, propriedades e métodos). Ela é baseada em quatro componentes da programação: abstração digital, encapsulamento, herança e polimorfismo.
- Como funcionam objetos
- Criar e utilizar construtores
- O que são classes
- Criar e utilizar métodos
- Como funciona encapsulamento
- O que é herança
- O que é polimorfismo
- Como funcionam interfaces
- O que são abstrações

☐ **Estruturas de Dados**:

- No contexto dos computadores, uma estrutura de dados é uma forma específica de armazenar e organizar os dados na memória do computador

para que esses dados possam ser facilmente recuperados e utilizados de forma eficiente quando necessário posteriormente.

- Conhecer as principais estruturas de dados
- Implementar as principais estruturas de dados

- [ ] **Python - Coleções**:

- Uma coleção representa um grupo de objetos, conhecidos como seus elementos. Eles são como recipientes que agrupam vários itens em uma única unidade. Algumas coleções permitem a duplicação de elementos e outras não. Algumas são ordenadas e outras não ordenadas.
- Utilizar listas e tuplas
- Utilizar polimorfismo nas coleções
- Utilizar conjuntos e dicionários

- [ ] **Python - Testes**:

- O teste de software é o processo de avaliação e verificação de que um software realmente faz o que deveria fazer. Os benefícios dos testes incluem a prevenção de bugs, a redução dos custos de desenvolvimento e a melhoria do desempenho.
- Usar testes unitários
- Usar testes de integração
- Usar testes de comportamento (behavior)
- Usar mocks

- [ ] **Python - Comunicação com APIs**:

- Uma API é uma interface que desenvolvedores de software utilizam para programar a interação com componentes ou recursos de software fora de seu próprio código. Uma definição ainda mais simples é que uma API é a parte de um componente de software que é acessível a outros componentes.
- Entender o que é uma API REST
- Conhecer os comandos básicos de comunicação HTTP

- Entender o que é uma API REST
- Saber fazer requisições autenticadas
- Converter objetos para JSON e vice-versa
- Saber usar as ferramentas do pacote Requests

Nivel 2

- [ ] **Flask**:

- Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível. Não possui camada de abstração de banco de dados, validação de formulário ou quaisquer outros componentes onde bibliotecas de terceiros pré-existentes fornecem funções comuns. No entanto, o Flask oferece suporte a extensões que podem adicionar recursos do aplicativo como se fossem implementados no próprio Flask.
- Criar aplicações web
- Definir rotas, redirecionamentos e templates
- Validar formulários

- [ ] **Python - Orientação a Objetos Avançada**:

- Mixin é uma classe que fornece implementações de métodos para reutilização por múltiplas classes filhas relacionadas.
- Sobrecarga do operador significa dar significado estendido além de seu significado operacional predefinido.

- [ ] **Django**:

- Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção.
- Criar aplicações web
- Entender a arquitetura de uma aplicação feita com Django
- Criar o admin de uma página

- Utilizar templates e rotas
- Criar formulários

- [ ] **Django Rest Framework**:

- O Django REST Framework é um conjunto de ferramentas poderosas e flexíveis para a construção de APIs.
- Desenvolver APIs
- Trabalhar com modelos, serializers e views
- Incluir filtros, buscas e ordenação
- Limitar o número de requisições

- [ ] **Python - MVC e MTV**:

- MVC e MTV são dois padrões de projeto (design patterns) utilizados para implementar interfaces e aplicações web.
- Entender o padrão MVC
- Entender o padrão MTV
- Compreender a diferença entre os padrões MVC e MTV

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

- Funções lambda nada mais são do que funções anônimas. Enquanto funções normais podem ser criada utilizando def como prefixo, as funções lambda são criadas utilizando lambda.
- Uma closure em Python é um objeto de função interna, uma função que se comporta como um objeto, que se lembra e tem acesso a variáveis no escopo local em que foi criado, mesmo depois que a função externa tenha terminado de ser executada. Também pode ser definida como um meio de ligar dados a uma função sem passá-los como parâmetro.

Nivel 3

- [ ] **Arquitetura de Microsserviços**:

- Microsserviços são uma abordagem de arquitetura na qual o software consiste de pequenos serviços independentes que se comunicam entre si e são organizados de acordo com seus domínios de negócio.
- Aprender o conceito de arquitetura planejada para microsserviços
- Realizar a comunicação usando APIs
- Melhorar a escalabilidade de um sistema

☐ **Contêineres**:

- Os contêineres são pacotes de software que contêm todos os elementos necessários para serem executados em qualquer ambiente. Gerenciamento de contêineres é uma área crucial na computação em nuvem e DevOps, que envolve o uso de tecnologias para automatizar o processo de criação, implantação, escalonamento e monitoramento de contêineres. Contêineres são unidades de software padronizadas que permitem aos desenvolvedores empacotar todas as dependências de um aplicativo (código, bibliotecas, configurações, etc.) em um único pacote. Isso permite que o aplicativo seja executado de forma consistente em qualquer ambiente de infraestrutura.
- A tecnologia de contêineres, como exemplificada pelo Docker, fornece um ambiente consistente e portátil para desenvolvimento, teste e implantação de aplicativos, o que é vital para o trabalho eficiente de engenharia de dados. Além disso, o Kubernetes, um sistema de orquestração de contêineres, permite o gerenciamento, a automação e a escalabilidade de aplicações baseadas em contêineres em ambientes de produção. Dominar esses conceitos e tecnologias possibilita a engenheiros de dados construir e manter pipelines de dados eficientes e confiáveis.
- O Kubernetes (também conhecido como k8s ou kube) é uma plataforma de orquestração de containers open source que automatiza grande parte dos processos manuais necessários para implantar, gerenciar e escalar aplicações em containers.
- Isolar seu software para funcionar independentemente
- Implantar software em clusters
- Modularizar seu sistema em pacotes menores
- Conhecer a plataforma Docker

- Conhecer Kubernetes

- [ ] **Python - Tipagem estática**:

- Python é uma linguagem dinamicamente tipada, ou seja, não há necessidade de pensar em tipos de dados. Linguagens estaticamente tipadas (tais como C ou Java) realizam verificações de tipo durante a compilação. Pode parecer mais seguro, pois você pode dizer imediatamente o tipo de parâmetro de cada função.
- Conhecer type hinting

- [ ] **Python - Geradores**:

- Geradores permitem declarar uma função que se comporta como um iterador, por exemplo, ela pode ser usada em um loop 'for'.
- Criar objetos iteradores
- Usar avaliação preguiçosa
- Executar tarefas simultâneas
- Uso da palavra reservada 'yield'

- [ ] **Python - Assíncrono**:

- Na programação assíncrona as funções não são executadas em ordem. Com o assincronismo, podemos interromper do código para conseguirmos alguma outra informação necessária para a continuar a execução. Isso significa que o código espera por uma outra parte do código e, enquanto espera, executa as demais partes.
- Aprender sobre corrotinas
- Coroutinas são generalizações de sub-rotinas. Elas são utilizadas para multitarefas cooperativas onde um processo cede controle voluntariamente, periodicamente ou quando ocioso, a fim de permitir que múltiplas aplicações sejam executadas simultaneamente.
- Lidar com concorrência
- Conhecer o conceito de objetos aguardáveis
- Criar tarefas concorrentemente

- Conhecer a biblioteca 'asyncio'

- [ ] **Python - args & kwargs**:
  - As variáveis mágicas *args e **kwargs são normalmente usadas na definições de uma função, e são usadas para passar um número desconhecido de argumentos para uma função.
  - Entender a diferença entre *args e **kwargs
- [ ] **Python - Métodos especiais (dunder)**:
  - Métodos especiais, ou métodos mágicos, em Python são métodos predefinidos em todos os objetos, com invocação automática sob circunstâncias especiais. Eles normalmente não são chamados diretamente pelo usuário mas podem ser overloaded (sobrescritos e alterados). Seus nomes começam e terminam com sublinhados duplos chamados de dunder (uma expressão derivada de double underscore).
  - Entender o conceito de métodos especiais (ou mágicos)
  - Conhecer os principais métodos mágicos e como usá-los
- [ ] **Python - Metaprogramação**:
  - Metaprogramação é uma técnica de programação com a qual programas têm a capacidade de tratar outros programas como seus dados. Isso significa que um programa pode ser projetado para ler, gerar, analisar ou transformar outros programas, e até mesmo modificar-se mesmo durante a execução.
  - Escrever um programa que manipula outros programas
  - Usar metaclasses
- [ ] **Python - Multiprocessamento**:
  - Em Python, o módulo de multiprocessamento inclui uma API muito simples e intuitiva para dividir o trabalho entre vários processos.
  - Executar processos em paralelo
  - Conhecer a classe Pool

- [ ] **Reflection e atributos**:
    - Os objetos de Reflection (reflexão) são usados para obter informações do tipo em tempo de execução. As classes que dão acesso aos metadados de um programa em execução estão no namespace System.Reflection.
    - Escrever código que lê as informações e metadados de objetos em tempo de execução
    - Obter nomes de classes em tempo de execução e criar objetos de uma classe

Habilidade Auxiliar: Infraestrutura

- [ ] **Git e GitHub - Fundamentos**:
    - Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com tudo, desde projetos pequenos a muito grandes com velocidade e eficiência.
    - GitHub é um serviço de hospedagem para desenvolvimento de software e controle de versão usando Git.
    - Criar um repositório
    - Clonar um repositório
    - Fazer commit, push e pull de e para o repositório
    - Reverter um commit
    - Criar branches e pul requests
    - Lidar com merge e conflitos
- [ ] **HTTP - Fundamentos**:
    - HTTP significa Hyper Text Transfer Protocol. A comunicação entre computadores cliente e servidores web é feita enviando solicitações HTTP e recebendo respostas HTTP.
    - Entender a diferença dos verbos HTTP
    - Testar os requests e ver os status codes no navegador
    - Saber fazer uma requisição HTTP na linha de comando com WGET

- Baixar uma imagem com WGET
- Fazer um post

☐ **JSON**:

- JSON significa JavaScript Object Notation (notação de objeto JavaScript). É um formato de texto para armazenar e transmitir dados.
- Criar um objeto
- Transformar um objeto em uma string
- Transformar uma string em objeto
- Manipular um objeto

☐ **Linha de comando - Fundamentos**:

- CLI é um programa de linha de comando que aceita entradas de texto para executar funções do sistema operacional.
- Conhecer os principais comandos

☐ **Cloud - Fundamentos**:

- Cloud, ou computação em nuvem é a distribuição de serviços de computação pela Internet usando um modelo de preço pago conforme o uso. Uma nuvem é composta de vários recursos de computação, que abrangem desde os próprios computadores (ou instâncias, na terminologia de nuvem) até redes, armazenamento, bancos de dados e o que estiver em torno deles. Ou seja, tudo o que normalmente é necessário para montar o equivalente a uma sala de servidores, ou mesmo um data center completo, estará pronto para ser utilizado, configurado e executado.
- Conhecer a diferença entre IaaS, PaaS e SaaS
- Conhecer os maiores provedores de cloud
- Especializar-se em algum provedor

☐ **SQL - Fundamentos**:

- SQL (Structured Query Language, traduzindo, Linguagem de Consulta Estruturada) é uma linguagem de programação padronizada que é usada

para gerenciar bancos de dados relacionais e realizar várias operações sobre os dados neles contidos.

- Conhecer os comandos mais comuns do SQL
- Usar SELECT para consultar uma tabela
- Usar INSERT para inserir dados em uma tabela
- Usar UPDATE para atualizar uma tabela
- Usar DELETE para remover dados de uma tabela
- Usar JOIN para conectar os dados de múltiplas tabelas
- Conhecer as cláusulas (FROM, ORDER BY, etc)

Habilidade Auxiliar: Boas práticas e ferramentas

- [ ] **SOLID**:
  - O Solid possui cinco princípios considerados como boas práticas no desenvolvimento de software que ajudam os programadores a escrever os códigos mais limpos, separando as responsabilidades, diminuindo acoplamentos, facilitando na refatoração e estimulando o reaproveitamento do código.
- [ ] **Clean Architecture**:
  - A Clean Architecture (Arquitetura Limpa) é uma forma de desenvolver software, de tal forma que apenas olhando para o código fonte de um programa, você deve ser capaz de dizer o que o programa faz.
- [ ] **Design Patterns**:
  - Na engenharia de software, um "padrão de projeto" (Design Pattern em inglês) é uma solução geral e reutilizável para um problema que ocorre normalmente dentro de um determinado contexto de projeto de software.
  - Conhecer e aplicar os principais Design Patterns
- [ ] **Jupyter & Colab notebooks**:

- Jupyter Notebook e Google Colaboratory são Notebooks que permitem a criação de blocos de texto e blocos de código
- Os Notebooks facilitam a elaboração de projetos de Data Science por ser possível visualizar o resultado da execução logo após o trecho de código
- O Google Colaboratory permite escrever e executar códigos Python diretamente no navegador, sem nenhuma ou poucas configurações necessárias
- Essas ferramentas facilitam o compartilhamento de projetos entre o time

☐ **Extração e Tratamento de Dados**:

- A extração de dados é o processo de coleta ou recuperação de tipos diferentes de dados de uma variedade de fontes, muitos dos quais podem estar mal organizados ou completamente desestruturados.
- Obter os dados que serão analisados
- Tratar os dados obtidos, transformando-os, alterando sua estrutura e valores a fim de deixar a base de dados mais coerente e garantir que os dados que serão trabalhados estejam nas melhores condições para serem analisados
- Utilizar as bibliotecas Pandas e Scikit-learn para tratar os dados

---