INTRODUÇÃO AO PYTHON - MÓDULO 01
SECRETARIA DE ESTADO DE SEGURANÇA PÚBLICA DO PARANÁ
CENTRO INTEGRADO DE COMANDO E CONTROLE EM CRISE REGIONAL

NIVELAMENTO EM *PYTHON*[1]

[1]*EQUIPE TÉCNICA CICCR*

29 de maio de 2023

1 Objetivos do Módulo

O Nivelamento de ***Conceitos de Programação - Introdução ao Python*** tem como objetivo introduzir os participantes à linguagem de programação *Python*, fornecendo as habilidades básicas para escrever programas funcionais, compreender conceitos fundamentais e promover a resolução de problemas. Ao final do nivelamento, os alunos estarão preparados para aplicar seus conhecimentos em projetos pessoais ou profissionais.

Ao fim deste módulo, o aluno deverá dominar os seguintes tópicos:

- **Do que se trata programar**: Programar é o ato de escrever código de computador para instruir um computador a executar determinadas tarefas. Envolve a criação de algoritmos, o uso de linguagens de programação e a aplicação de lógica e resolução de problemas para desenvolver soluções computacionais.
- **O que significa o termo algoritmo**: Um algoritmo é uma sequência de passos ou instruções lógicas bem definidas para resolver um problema ou realizar uma tarefa. É um conjunto de regras que descreve uma série de etapas para obter um resultado desejado.

- **Onde encontrar informações sobre como instalar o *Python***: Para obter informações sobre como instalar o ***Python***, você pode visitar o site oficial do ***Python*** em *python.org*. Lá você encontrará documentação detalhada, tutoriais e guias de instalação para diferentes plataformas. Para um guia de consulta rápida, acesso **Guia de Instalação do *Python***.

- **Sintaxe e Estrutura de Dados**: A sintaxe em programação se refere às regras e convenções que determinam como o código fonte deve ser escrito em uma determinada linguagem de programação. A estrutura de dados se refere às formas de organizar e armazenar dados em um programa, como listas, tuplas, conjuntos, dicionários, entre outros.

- **Funções em *Python***: Em ***Python***, as funções são blocos de código reutilizáveis que realizam uma tarefa específica. Elas permitem dividir um programa em partes menores e mais gerenciáveis, facilitando a organização e a reutilização de código. As funções podem receber argumentos, executar um conjunto de instruções e retornar um resultado, se necessário.

- **Primeira Aplicação em Python**: A primeira aplicação em Python refere-se à criação do seu primeiro programa usando a linguagem ***Python***. Geralmente, é um programa simples que mostra uma saída na tela ou realiza uma tarefa básica para familiarizá-lo com a sintaxe e as estruturas básicas da linguagem.

- **Comentários na sua primeira aplicação**: Comentários são trechos de texto adicionados ao código fonte para fornecer explicações e informações adicionais sobre o código. Na sua primeira aplicação em ***Python***, os comentários podem ser usados para descrever o propósito do programa, fornecer instruções ou explicar partes específicas do código. Eles são ignorados pelo interpretador ***Python*** durante a execução do programa.

2 Introdução Ao Python

2.1 Programação e Algoritmos

O que é programar Programar é o ato de escrever um conjunto de instruções ou algoritmos que um computador pode executar para realizar uma tarefa específica. Envolve a criação de código em uma linguagem de programação que segue uma sintaxe e estrutura definidas. A programação permite que os desenvolvedores criem software, aplicativos, sites e sistemas que automatizam processos, resolvam problemas e forneçam funcionalidades úteis. Programar requer habilidades lógicas e analíticas para que as instruções sejam escritas de forma clara e precisa, permitindo que o computador execute as

tarefas conforme especificado. É uma habilidade essencial no mundo da tecnologia e desempenha um papel fundamental no avanço da computação e da inovação tecnológica.

Programar refere-se ao processo de criar um conjunto de instruções ou algoritmos para serem executados por um computador ou outro dispositivo eletrônico. É a arte de escrever código, que consiste em uma série de comandos precisos e lógicos, para que o computador execute tarefas específicas.

A programação permite que os desenvolvedores criem programas de software, aplicativos móveis, sites, jogos e uma ampla gama de soluções tecnológicas. É por meio da programação que os computadores são capazes de realizar diversas tarefas, desde cálculos complexos até a execução de operações de processamento de dados.

Ao programar, os desenvolvedores utilizam linguagens de programação, como Python, Java, C++, JavaScript, entre outras. Essas linguagens fornecem uma sintaxe específica e um conjunto de regras que permitem aos programadores expressar suas instruções de forma clara e precisa para que o computador as execute corretamente.

A programação requer habilidades lógicas, criatividade e resolução de problemas. Os programadores devem entender o problema que desejam resolver, projetar uma solução eficiente e implementar o código necessário para alcançar o resultado desejado. Além disso, a depuração e o teste do código são partes essenciais do processo de programação para garantir que o programa funcione corretamente e produza os resultados esperados.

A programação é uma área em constante evolução, impulsionada pelo avanço da tecnologia e das demandas da sociedade. A capacidade de programar é uma habilidade valiosa em muitos setores, como desenvolvimento de software, ciência de dados, inteligência artificial, engenharia de software e muitos outros. Ela permite que as pessoas criem soluções inovadoras e automatizem tarefas para melhorar a eficiência e a produtividade em diversos campos de atuação.

O que é uma linguagem de Programação Uma linguagem de programação é uma forma de comunicação entre um programador e um computador. É um conjunto de regras e símbolos que permitem escrever instruções para que o computador execute tarefas específicas. Essas linguagens são projetadas para serem compreensíveis tanto para os seres humanos quanto para as máquinas.

As linguagens de programação fornecem um conjunto de palavras-chave, símbolos, estruturas gramaticais e regras semânticas que permitem aos programadores expressar suas instruções de forma estruturada e lógica. Elas permitem que os programadores escrevam algoritmos, que são sequências de passos que o computador deve seguir para resolver um determinado problema.

Existem muitas linguagens de programação disponíveis, cada uma com suas próprias características e finalidades. Algumas linguagens são de uso geral, como Python, Java, C++ e JavaScript, e

podem ser usadas para uma ampla variedade de aplicações. Outras são mais especializadas e foram projetadas para fins específicos, como linguagens para desenvolvimento web, análise de dados, inteligência artificial, entre outros.

As linguagens de programação podem ser classificadas em diferentes níveis de abstração. Linguagens de baixo nível, como ***Assembly***, estão mais próximas da linguagem de máquina e requerem um conhecimento mais detalhado do hardware do computador. Por outro lado, linguagens de alto nível, como ***Python***, permitem que os programadores expressem suas instruções de maneira mais próxima da linguagem humana, sendo menos dependentes dos detalhes de implementação.

Cada linguagem de programação tem suas próprias regras e características, mas todas compartilham o objetivo de permitir que os programadores desenvolvam soluções e criem programas que possam ser executados em um computador. O conhecimento de diferentes linguagens de programação permite aos programadores escolher a linguagem mais adequada para o problema em questão e expandir suas habilidades no desenvolvimento de software.

Estrutura de uma linguagem de programação A estrutura de uma linguagem de programação é composta por diversos elementos que definem como o código deve ser organizado e como as instruções devem ser escritas. Os principais elementos de estrutura de uma linguagem de programação são:

- ***Sintaxe:*** A sintaxe é o conjunto de regras que define a estrutura correta das instruções na linguagem. Ela determina como as palavras-chave, operadores, variáveis e outros elementos devem ser combinados para formar instruções válidas. A sintaxe define a ordem e a forma correta de escrever o código.

- ***Palavras-chave:*** As palavras-chave são termos reservados pela linguagem de programação que possuem um significado específico. Elas são usadas para definir estruturas de controle, declaração de variáveis, operações matemáticas e outras funcionalidades da linguagem. Exemplos de palavras-chave incluem "if", "for", "while", "function", "class", entre outras.

- ***Tipos de dados:*** As linguagens de programação possuem diferentes tipos de dados, como números, texto, booleanos, arrays, objetos, entre outros. Os tipos de dados definem o tipo de valor que uma variável pode armazenar e as operações que podem ser realizadas com esses valores.

- ***Variáveis:*** As variáveis são utilizadas para armazenar valores na memória durante a execução do programa. Elas possuem um nome e um tipo de dados associado. As variáveis podem ser usadas para armazenar informações temporárias, realizar cálculos e representar dados em geral.

- ***Estruturas de controle:*** As estruturas de controle permitem que o fluxo de execução do programa seja controlado. Elas incluem estruturas condicionais (como "if"e "else") que permitem executar diferentes blocos de código dependendo de uma condição, e estruturas de repetição (como "for"e "while") que permitem executar um bloco de código várias vezes.

- ***Funções e procedimentos:*** As funções e procedimentos são blocos de código que podem ser definidos e chamados em diferentes partes do programa. Eles permitem agrupar um conjunto de instruções em uma unidade lógica e reutilizável. As funções podem receber parâmetros e retornar valores.

- ***Bibliotecas e módulos:*** Muitas linguagens de programação possuem bibliotecas ou módulos que são conjuntos de código pré-existente que fornecem funcionalidades adicionais. Essas bibliotecas podem conter funções, classes e outras estruturas que podem ser importadas e utilizadas em um programa. Elas permitem estender as capacidades da linguagem de programação e facilitam o desenvolvimento de soluções complexas.

- ***Estruturas de dados:*** As estruturas de dados são formas organizadas de armazenar e manipular conjuntos de dados. Elas incluem *arrays*, listas, conjuntos, dicionários, entre outros. Cada estrutura de dados tem suas próprias características e métodos para realizar operações específicas, como adicionar, remover, buscar ou modificar elementos.

- ***Orientação a objetos:*** Muitas linguagens de programação são orientadas a objetos, o que significa que elas permitem a criação de classes e objetos. A programação orientada a objetos organiza o código em torno de objetos que possuem atributos (dados) e métodos (ações). Essa abordagem permite encapsular a lógica do programa e promover a reutilização de código.

- ***Tratamento de erros:*** As linguagens de programação têm mecanismos para lidar com erros e exceções que podem ocorrer durante a execução do programa. Isso permite que o desenvolvedor capture e trate essas situações inesperadas, evitando falhas ou comportamentos indesejados do programa.

Paradigmas de Programação Um paradigma, no contexto da programação de computadores, refere-se a um conjunto de conceitos, princípios e abordagens que definem a forma como os programas são estruturados, organizados e desenvolvidos. É uma forma de pensar e abordar a resolução de problemas na programação.

Cada paradigma de programação possui suas próprias regras, diretrizes e estilo de escrita de código. Ele define a estrutura básica do programa, as técnicas de solução de problemas e os padrões de

design a serem seguidos.

Os paradigmas de programação fornecem diferentes formas de pensar sobre a estrutura do programa, o fluxo de controle, o gerenciamento de estado e a interação entre os componentes do sistema. Eles influenciam a maneira como os programadores abordam a resolução de problemas e as ferramentas e técnicas que eles utilizam.

É importante mencionar que não existe um único paradigma ***"melhor"*** ou ***"correto"***. Cada paradigma tem suas vantagens e desvantagens, e a escolha do paradigma adequado depende do tipo de problema a ser resolvido, das restrições e das preferências pessoais. Além disso, muitas linguagens de programação permitem a combinação de múltiplos paradigmas, permitindo maior flexibilidade na forma como os programas são escritos.

Os paradigmas de programação mais comuns incluem a programação imperativa, orientada a objetos, funcional, lógica, estruturada, entre outros. Cada um deles oferece uma abordagem única para a construção de programas e tem suas próprias técnicas e conceitos específicos.

Alguns paradigmas de programação são:

- ***Programação Imperativa:*** É o paradigma mais tradicional e amplamente utilizado. Nesse paradigma, os programas são estruturados em sequências de instruções que modificam o estado do programa. Ele se baseia na ideia de que um programa é uma série de comandos que são executados em ordem.

- ***Programação Orientada a Objetos (POO)***: Nesse paradigma, os programas são organizados em torno de objetos, que são instâncias de classes. Os objetos possuem atributos (dados) e métodos (ações) que podem interagir uns com os outros. A POO enfatiza a reutilização de código, encapsulamento e modularidade.

- ***Programação Funcional:*** Nesse paradigma, o foco está nas funções. Os programas são escritos em termos de funções puras, que não possuem efeitos colaterais e retornam um valor com base em seus argumentos. A programação funcional enfatiza a imutabilidade dos dados e o uso de funções de ordem superior.

- ***Programação Lógica:*** Nesse paradigma, os programas são escritos em termos de regras lógicas. A programação lógica se baseia no uso de predicados e inferência lógica para resolver problemas. A linguagem de programação Prolog é um exemplo comum de programação lógica.

- ***Programação Estruturada:*** É um paradigma que enfatiza a organização do código em estruturas bem definidas, como sequências, laços e condicionais. A programação estruturada busca

evitar o uso de desvios incondicionais (como o "goto") e promover a legibilidade e a manutenibilidade do código.

Além desses, existem outros paradigmas de programação, como programação procedural, programação orientada a eventos, programação concorrente, entre outros. Cada paradigma tem suas próprias vantagens e é mais adequado para determinados tipos de problemas. Muitas linguagens de programação permitem a combinação de paradigmas, permitindo ao programador escolher a abordagem mais adequada para cada situação.

Programação Procedural

Programação Estruturada

Programação Orientada a Objetos

O que são algoritmos O estudo e uso de algoritmos remonta a milhares de anos. A palavra *"algoritmo"* deriva do nome de um matemático persa do século IX, *Al-Khwarizmi*, que foi um dos primeiros a sistematizar métodos de resolução de equações lineares e quadráticas. No entanto, a ideia de algoritmo e seu uso prático são anteriores a *Al-Khwarizmi*.

Antes do advento dos computadores, os algoritmos eram resolvidos manualmente por matemáticos e cientistas. Grandes avanços foram feitos por matemáticos notáveis, como ***Euclides***, que desenvolveu o algoritmo para encontrar o maior divisor comum de dois números (Algoritmo de Euclides), e ***Isaac Newton***, que desenvolveu algoritmos para cálculo diferencial e integral.

Com o avanço da tecnologia e a invenção dos computadores, o estudo e desenvolvimento de algoritmos expandiu-se significativamente. Durante a Segunda Guerra Mundial, os primeiros computadores foram construídos para auxiliar nos cálculos e criptografia. Esse período marcou o início da programação de computadores e do desenvolvimento de algoritmos para resolver problemas complexos.

Nos anos seguintes, a ciência da computação emergiu como uma disciplina acadêmica, e muitos pesquisadores contribuíram para o desenvolvimento de algoritmos eficientes e otimizados. Diversos algoritmos famosos foram criados nessa época, como o algoritmo de classificação rápida (*quicksort*), o algoritmo de busca binária e o algoritmo de *Dijkstra* para encontrar o caminho mais curto em um grafo.

À medida que os computadores se tornaram mais poderosos e acessíveis, o campo da ciência da computação se expandiu rapidamente, levando ao desenvolvimento de algoritmos mais avançados

e sofisticados. Algoritmos de aprendizado de máquina, algoritmos de criptografia, algoritmos de compressão de dados e algoritmos de otimização são apenas alguns exemplos das áreas em que os algoritmos têm sido amplamente utilizados.

Hoje em dia, os algoritmos estão presentes em quase todos os aspectos de nossas vidas, desde a pesquisa na web até as transações financeiras. O estudo contínuo e a evolução dos algoritmos são fundamentais para acompanhar os avanços tecnológicos e resolver problemas cada vez mais complexos.

Algoritmo é uma sequência de passos usada para resolver um problema. A sequência apresenta um método único de abordar uma questão, fornecendo uma solução específica. Um algoritmo não precisa representar conceitos matemáticos ou lógicos, embora as apresentações de algoritmos frequentemente se enquadrem nessa categoria. Algumas fórmulas especiais também são algoritmos, como a fórmula quadrática. Para que um processo represente um algoritmo, ele deve ser:

- ***Finito:*** O algoritmo deve eventualmente resolver o problema. Este livro discute problemas com uma solução conhecida para que você possa avaliar se um algoritmo resolve o problema corretamente.
- ***Bem-definido:*** A série de passos deve ser precisa e apresentar etapas compreensíveis. Especialmente porque os computadores estão envolvidos no uso de algoritmos, o computador deve ser capaz de entender as etapas para criar um algoritmo utilizável.
- ***Efetivo:*** Um algoritmo deve resolver todos os casos do problema para o qual alguém o definiu. Um algoritmo deve sempre resolver o problema que precisa ser resolvido. Embora se deva antecipar algumas falhas, a ocorrência de falhas é rara e ocorre apenas em situações aceitáveis para o uso pretendido do algoritmo.

Para criar um algoritmo, siga estas etapas:

- ***Compreenda o problema:*** Analise e compreenda claramente o problema que deseja resolver. Identifique os requisitos, restrições e objetivos do problema.
- ***Divida o problema em etapas menores:*** Quebre o problema em etapas menores e mais gerenciáveis. Isso ajuda a simplificar o problema e facilita a resolução passo a passo.
- ***Identifique as entradas e saídas:*** Determine quais informações são necessárias como entrada para o algoritmo e qual é a saída esperada após a execução do algoritmo.
- ***Projete a lógica do algoritmo:*** Desenvolva a lógica do algoritmo, definindo a sequência de passos que devem ser seguidos para resolver o problema. Use estruturas de controle, como loops e condicionais, para controlar o fluxo do algoritmo.

- ***Teste e revise o algoritmo:*** Teste o algoritmo com diferentes conjuntos de dados de entrada para garantir que ele esteja funcionando corretamente. Faça ajustes e revisões conforme necessário para melhorar a eficiência e corrigir erros.

- ***Documente o algoritmo:*** Escreva o algoritmo de forma clara e organizada. Utilize comentários para explicar o propósito de cada etapa e fornecer informações adicionais para facilitar a compreensão.

- ***Implemente o algoritmo:*** Traduza o algoritmo para a linguagem de programação escolhida. Escreva o código correspondente para cada etapa do algoritmo.

- ***Teste e depure o código:*** Execute o código implementado, fornecendo diferentes dados de entrada e verificando se a saída corresponde ao esperado. Identifique e corrija quaisquer erros ou problemas de execução.

- ***Otimize o algoritmo:*** Analise o desempenho do algoritmo e faça melhorias para torná-lo mais eficiente, reduzindo o tempo de execução ou a utilização de recursos.

Lembre-se de que a criação de um algoritmo requer prática e experiência. À medida que você ganha mais familiaridade com a programação e a resolução de problemas, se tornará mais fácil criar algoritmos eficazes e eficientes.

3 Linguagem Python

3.1 Funções

- **Modularidade:** As funções permitem dividir um programa em blocos de código independentes e reutilizáveis. Isso facilita a compreensão e organização do código, além de promover a reutilização de código em diferentes partes do programa.

- **Reutilização de código:** Ao definir uma função, você pode chamá-la quantas vezes for necessário em diferentes partes do programa. Isso evita a duplicação de código e torna as atualizações e correções mais fáceis, já que você só precisa fazer as alterações em um único lugar.

- **Legibilidade e manutenção:** Utilizar funções ajuda a melhorar a legibilidade do código, pois as funções podem ter nomes descritivos que indicam sua funcionalidade. Além disso, ao dividir o código em funções menores e mais focadas, é mais fácil entender, testar e corrigir problemas específicos.

- **Abstração:** Funções permitem abstrair detalhes de implementação complexos em um nível mais alto de abstração. Isso significa que você pode usar uma função sem precisar saber todos os detalhes internos de como ela funciona, tornando o código mais fácil de entender e usar.

- **Testabilidade:** Funções isoladas podem ser testadas de forma independente, o que facilita a identificação de erros e o desenvolvimento de testes automatizados. Isso também contribui para a qualidade e confiabilidade do código.

- **Encapsulamento:** As funções permitem encapsular um conjunto de instruções em um bloco único, fornecendo um contexto claro para a execução das ações contidas. Isso ajuda a evitar interferências indesejadas entre partes diferentes do código.

- **Modularidade do desenvolvimento em equipe:** Ao trabalhar em equipe, as funções facilitam a divisão de tarefas entre os membros e a integração posterior do trabalho realizado. Cada membro pode desenvolver funções independentes e, posteriormente, combiná-las para construir um programa completo.