Aula 1: Fundamentos da Programação

Curso de Introdução à Programação com Python

1 O que são Linguagens de Programação

As linguagens de programação são ferramentas que permitem aos seres humanos comunicarem instruções aos computadores. Elas funcionam como um intermediário entre a linguagem humana e a linguagem de máquina, traduzindo as intenções dos programadores em comandos que o computador pode executar.

1.1 Importância das linguagens de programação

As linguagens de programação são fundamentais para o desenvolvimento de tecnologia. Elas oferecem várias vantagens, como:

- Criação de Software Complexo: Permitem o desenvolvimento de sistemas operacionais, aplicativos, jogos e mais.
- Resolução de Problemas Computacionais: Facilitam a implementação de soluções para problemas diversos, desde cálculos matemáticos até processamento de dados.
- Automação de Tarefas: Tornam possível a automação de processos repetitivos, aumentando a eficiência.
- Interatividade: Algumas linguagens permitem criar interfaces interativas, melhorando a experiência do usuário.
- Análise de Dados: Linguagens como Python são amplamente utilizadas para análise e visualização de dados.

1.2 Exemplo de linguagem de programação

Aqui está um exemplo simples em Python, uma linguagem conhecida por sua legibilidade e simplicidade:

```
print("Ola, mundo!")
```

Este código instrui o computador a exibir a mensagem "Olá, mundo!" na tela, um clássico exemplo de um primeiro programa.

1.3 Exemplo adicional: Calculando a soma de dois números

Outro exemplo prático em Python:

```
a = 5
b = 3
soma = a + b
print(f"A soma de {a} e {b} e: {soma}")
```

Este código calcula e exibe a soma de dois números.

2 O que são Algoritmos

Algoritmos são sequências de passos lógicos e bem definidos para resolver um problema ou realizar uma tarefa. Eles são a base da programação, pois descrevem a lógica que o código deve seguir.

2.1 Uso de algoritmos na programação

Antes de escrever qualquer código, é essencial pensar no algoritmo que vai resolver o problema em questão. Um bom algoritmo pode ser implementado em várias linguagens de programação.

2.2 Exemplo de algoritmo do cotidiano

Algoritmo para fazer um sanduíche:

1. Pegar duas fatias de pão.
2. Passar manteiga em uma face de cada fatia.
3. Colocar uma fatia de queijo sobre uma fatia de pão.
4. Colocar uma fatia de presunto sobre o queijo.
5. Cobrir com a outra fatia de pão.
6. Cortar o sanduíche ao meio (opcional).

Este exemplo ilustra como algoritmos podem ser utilizados em atividades cotidianas.

2.3 Exemplo de algoritmo em Python

Aqui está um algoritmo simples que calcula a média de dois números:

```
def calcular_media(a, b):
    soma = a + b
    media = soma / 2
    return media
```

```
# Uso do algoritmo
resultado = calcular_media(10, 20)
print(f"A media e: {resultado}")
```

Neste exemplo, a função ‘calcular_media‘ recebe dois números, calcula a soma e retorna a média.

2.4 Exemplo adicional: Verificando se um número é par ou ímpar

Aqui está um algoritmo que determina se um número é par ou ímpar:

```
def verificar_par_impar(numero):
    if numero % 2 == 0:
        return "Par"
    else:
        return "Impar"

# Uso do algoritmo
resultado = verificar_par_impar(7)
print(f"O numero 7 e: {resultado}")
```

Este código verifica se o número 7 é par ou ímpar e exibe o resultado.

3 História do Python

Python foi criado por Guido van Rossum e lançado pela primeira vez em 1991. Desde então, tornou-se uma das linguagens de programação mais populares do mundo, sendo amplamente utilizada em diversas áreas.

3.1 Origem e Desenvolvimento

A ideia de Python surgiu no final da década de 1980, quando Guido van Rossum começou a trabalhar no projeto como um hobby durante o Natal. Ele queria criar uma linguagem que fosse fácil de usar e que mantivesse a legibilidade, inspirando-se em linguagens como ABC.

3.2 Lançamentos Importantes

- Python 1.0 (1994): A primeira versão oficial, que incluiu recursos como funções, módulos e tipos de dados básicos.
- Python 2.0 (2000): Introduziu listas de compreensão, suporte a Unicode e um sistema de gerenciamento de memória que melhorou a eficiência.
- Python 3.0 (2008): Uma versão que não era retrocompatível com Python 2, foi projetada para corrigir falhas de design e melhorar a linguagem. Essa versão trouxe mudanças significativas, como a função ‘print‘ se tornando uma função em vez de uma declaração.

Figure 1: Guido Van Rossum

3.3 Crescimento e Popularidade

A popularidade do Python cresceu exponencialmente devido à sua simplicidade e versatilidade. Ele é amplamente utilizado em áreas como:

- Desenvolvimento Web: Frameworks como Django e Flask permitem a criação de aplicações robustas.
- Ciência de Dados: Bibliotecas como NumPy, Pandas e Matplotlib são ferramentas essenciais para análise e visualização de dados.
- Inteligência Artificial: Python se tornou a linguagem preferida para aprendizado de máquina e inteligência artificial, com bibliotecas como TensorFlow e scikit-learn.
- Automação de Tarefas: A linguagem é frequentemente usada para scripts de automação, facilitando a execução de tarefas repetitivas.

3.4 Comunidade e Contribuições

A comunidade Python é uma das suas maiores forças. Com uma grande base de desenvolvedores e colaboradores, muitos projetos de código aberto são mantidos. Eventos como a PyCon reúnem desenvolvedores de todo o mundo para compartilhar conhecimento e experiências.

3.5 O Futuro do Python

Python continua a evoluir, com novas versões sendo lançadas regularmente. O foco em eficiência, legibilidade e suporte a novas tecnologias garante que a

linguagem permaneça relevante e amplamente adotada no futuro.

3.6 Principais filosofias do Python

As filosofias do Python são guiadas por princípios que incentivam boas práticas de programação, como:

- Legibilidade do Código: O código deve ser fácil de ler e entender.
- Simplicidade: Soluções simples são preferíveis a soluções complexas.
- "Há apenas uma maneira óbvia de fazer algo": Isso promove a consistência e a clareza.
- Comunidade Ativa: A comunidade Python é acolhedora e colaborativa.

3.7 O Zen do Python

O Zen do Python, escrito por Tim Peters, resume a filosofia da linguagem. Você pode vê-lo digitando o seguinte no interpretador Python:

```
import this
```

Alguns princípios do Zen do Python incluem:

- Bonito é melhor que feio: A estética do código é importante.
- Explícito é melhor que implícito: A clareza é preferida a suposições.
- Simples é melhor que complexo: A simplicidade deve ser a meta.
- Legibilidade conta: O código deve ser facilmente compreensível.