Guia de Introdução ao Python para Análise de Dados

1. O que é Python?

Python é uma linguagem de programação moderna que se destaca pela simplicidade e legibilidade do código. Desenvolvida para ser fácil de aprender e usar, Python permite que programadores escrevam instruções de forma clara e organizada.

Uma das grandes vantagens do Python é sua versatilidade - ela pode ser usada para desenvolver desde aplicações web até sistemas complexos de inteligência artificial. Para nós, analistas de dados, Python é especialmente valiosa devido às suas bibliotecas especializadas que facilitam o trabalho com grandes volumes de informação, cálculos matemáticos e visualização de dados.

A linguagem é "interpretada", o que significa que não precisamos compilar o código antes de executá-lo, tornando o processo de desenvolvimento mais ágil e ideal para experimentação e análise exploratória.

2. Ferramentas que Usaremos

Python

O Python é o coração do nosso trabalho. É a linguagem propriamente dita que instalaremos em nossos computadores.

Verificação da instalação:

```
bash
python --version
```

VS Code (Visual Studio Code)

O VS Code será nosso editor de código principal. Pense nele como um processador de texto sofisticado, mas especificamente desenhado para programação.

Principais recursos:

- Destaque de sintaxe (código colorido)
- Autocompletar inteligente
- Identificação de erros em tempo real
- Sistema de extensões

Extensões Necessárias

As extensões Python e Pylance transformam o VS Code em um ambiente de desenvolvimento profissional, oferecendo ferramentas de depuração, gerenciamento de ambientes e execução de código.

3. Primeiros Passos: Hello World

O "Hello World" é uma tradição na programação - é o primeiro programa que a maioria dos desenvolvedores cria quando aprende uma nova linguagem.

Código básico:

```
python
```

```
print("Hello, World!")
```

Execução:

- Salve o arquivo com extensão `.py`
- Execute com F5 ou pelo terminal: `python nome_do_arquivo.py`

Este pequeno programa valida que todo o ambiente está configurado corretamente e nos dá confiança para avançar para conceitos mais complexos.

4. Uso do Comando Print e Formatação de Texto

O comando `print` é uma das ferramentas mais fundamentais em Python. Ele funciona como nossa "voz" do programa - é através dele que mostramos informações para o usuário.

Print básico:

```
python
print("Bem-vindos ao curso de Analista de Dados!")
```

Múltiplos argumentos:

```
python
nome = "Maria"
idade = 25
print("Nome:", nome, "Idade:", idade)
Formatação com f-strings (recomendado):
python
nome = "Carlos"
nota = 8.5
print(f"Aluno {nome} tirou {nota} na prova")
Outros métodos de formatação:
python
# Método .format()
print("Aluno {} tirou {} na prova".format(nome, nota))

# Formatação com % (mais antigo)
print("Aluno %s tirou %.2f na prova" % (nome, nota))
```

A formatação de texto é crucial para apresentar dados de forma clara e profissional. As f-strings nos permitem criar mensagens dinâmicas onde valores de variáveis são inseridos diretamente no texto, sendo extremamente útil para criar relatórios automáticos ou exibir resultados de cálculos.

5. Uso de Variáveis

Variáveis são como "recipientes" que usamos para armazenar informações na memória do computador. Imagine-as como etiquetas que colamos em caixas de armazenamento - cada etiqueta tem um nome e a caixa contém um valor.

Declaração de variáveis:

```python
# Variáveis simples
nome = "Ana"
idade = 30
altura = 1.65
estudante = True

# Reatribuição
idade = 31

# Múltiplas atribuições
x, y, z = 10, 20, 30
```

Regras para nomes de variáveis:

- Podem conter letras, números e _
- Não podem começar com número
- Case sensitive (idade ≠ Idade ≠ IDADE)
- Use nomes descritivos

Exemplos práticos para análise de dados:

```python
# Para análise de dados
quantidade_vendas = 1500
valor_total = 45000.75
produto = "Notebook"
media_vendas = valor_total / quantidade_vendas

print(f"Produto: {produto}")
print(f"Média de valor por venda: R$ {media_vendas:.2f}")
```

A beleza das variáveis em Python está na sua simplicidade: não precisamos declarar antecipadamente que tipo de dados elas vão armazenar. O Python descobre automaticamente se estamos guardando texto, números ou valores verdadeiro/falso.

6. Uso de Input

A função `input` transforma nossos programas de monólogos em diálogos. Ela permite que o programa "espere" e "escutE" a entrada do usuário antes de continuar a execução.

Input básico:

```
python
nome = input("Digite seu nome: ")
print(f"Olá, {nome}!")
Convertendo input (importante!):
python
# Input sempre retorna string
idade = int(input("Digite sua idade: "))
altura = float(input("Digite sua altura: "))

print(f"Daqui a 5 anos você terá {idade + 5} anos")
```

Exemplo prático para análise:

```
python
# Coletando dados do usuário
vendas_dia1 = float(input("Vendas do dia 1: R$ "))
vendas_dia2 = float(input("Vendas do dia 2: R$ "))

total_vendas = vendas_dia1 + vendas_dia2
media_diaria = total_vendas / 2

print(f"Total de vendas: R$ {total_vendas:.2f}")
print(f"Média diária: R$ {media_diaria:.2f}")
```

Um ponto crítico que frequentemente causa confusão para iniciantes é que o `input` sempre retorna texto (string), mesmo quando o usuário digita números. Por isso, a conversão usando `int()` ou `float()` é necessária para realizar operações matemáticas.

7. Tipos de Dados em Python e Conversão

Assim como na matemática temos números inteiros, decimais, e na linguagem temos textos, na programação também categorizamos os dados em tipos específicos. Cada tipo de dado tem características e capacidades próprias.

Tipos principais:

Tipo	Exemplo	Uso

`int`	`idade = 25`	Números inteiros
`float`	`preco = 19.99`	Números decimais
`str`	`nome = "João"`	Texto
`bool`	`ativo = True`	Verdadeiro/Falso
`list`	`numeros = [1, 2, 3]`	Lista de elementos
`dict`	`pessoa = {"nome": "Ana"}`	Dicionário chave-valor

Verificando tipos:

```python
x = 10
y = "Python"
z = 3.14

print(type(x))  # <class 'int'>
print(type(y))  # <class 'str'>
print(type(z))  # <class 'float'>
```

Conversão entre tipos:

```python
# String para número
texto = "123"
numero = int(texto)
decimal = float("45.67")

# Número para string
idade = 25
texto_idade = str(idade)

# Para boolean
valor = 10
bool_valor = bool(valor)  # True (qualquer número ≠ 0 é True)
```

```
# Exemplos práticos
preco_str = "29.90"
preco_float = float(preco_str)
quantidade = 3
total = preco_float * quantidade

print(f"Total: R$ {total:.2f}")
```

Exemplo completo com conversão:

```
python
# Coletando e processando dados
nome_produto = input("Nome do produto: ")
preco_str = input("Preço do produto: R$ ")
quantidade_str = input("Quantidade vendida: ")

# Conversão para tipos numéricos
preco = float(preco_str)
quantidade = int(quantidade_str)

# Cálculos
total_venda = preco * quantidade

# Exibindo resultados
print("\n" + "="*30)
print("RELATÓRIO DE VENDA")
print("="*30)
print(f"Produto: {nome_produto}")
print(f"Preço unitário: R$ {preco:.2f}")
print(f"Quantidade: {quantidade}")
print(f"Total da venda: R$ {total_venda:.2f}")
```

A conversão entre tipos é como uma "tradução" que permite que dados de um tipo sejam entendidos como outro tipo quando necessário. Isso é especialmente importante quando trabalhamos com dados externos, como planilhas ou entradas de usuário, onde os números frequentemente chegam como texto e precisam ser convertidos para permitir cálculos.

🚀 Próximos Passos

Os conceitos apresentados neste guia formam a fundação sobre a qual construiremos todo o nosso conhecimento em Python para análise de dados.

Dominar estes fundamentos é como aprender o vocabulário básico antes de formar frases complexas em um novo idioma.

Nos próximos módulos, evoluiremos destes conceitos básicos para estruturas que nos permitirão organizar dados de forma mais complexa (listas e dicionários), tomar decisões automáticas (estruturas de controle) e criar operações reutilizáveis (funções).

A prática consistente é a chave para a proficiência em programação. Cada conceito novo deve ser experimentado e testado através de pequenos programas próprios. Não tenha medo de errar - os erros são oportunidades de aprendizado e parte natural do processo de desenvolvimento.

Dica importante: Comece criando pequenos programas que resolvem problemas do seu dia a dia como analista de dados. Por exemplo, um programa que calcula médias, converte unidades ou gera relatórios simples.

Bom curso! 🐍