UNIFAVIP
wyden

Apostila

PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO EM PYTHON

Prof°: Sebastião Rogério

Av. Adjar da Silva Casé, 800

Indianópolis Caruaru – Pernambuco – PE

CEP 55024-740 | Telefone 0800 771-5001

Introdução ao Python

Python é amplamente utilizado em áreas como desenvolvimento web, ciência de dados, inteligência artificial e automação de tarefas. Sua simplicidade e legibilidade fazem dele uma linguagem favorita tanto para iniciantes quanto para desenvolvedores experientes.

Manipulação de Variáveis e Strings

- **Variáveis**: Armazenam dados e não precisam ter o tipo declarado. Devem começar com letras e podem incluir números e sublinhado (_). Entrada de dados: Use input() para receber dados do usuário, que por padrão é uma string.
- **Conversão de tipos:** Embora o input() sempre retorne uma string, você pode converter facilmente para outros tipos como int(), float() e bool().

Exemplo:

idade = int(input("Digite sua idade: "))

- **Fatiamento de strings:** Python permite acessar partes específicas de uma string usando índices.

Exemplo:

texto = "Python"

print(texto[0:3]) # Saída: Pyt

Mais métodos de manipulação de strings

Método	Descrição	Exemplo
upper()	Converte para maiúsculas	"python".upper() ➞ 'PYTHON'
lower()	Converte para minúsculas	"PYTHON".lower() ➞ 'python'
replace()	Substitui partes da string	"banana".replace("a", "o") ➞ 'bonono'
strip()	Remove espaços em branco no início e fim	" teste ".strip() ➞ 'teste'

Curiosidade: Strings são imutáveis, ou seja, não podem ser alteradas diretamente.

Dicionários em Python

- Dicionários armazenam dados como pares de chave:valor, sendo acessados através de suas chaves.

Exemplo de iteração sobre dicionários:

```
dados = {"Nome": "Ana", "Idade": 25}
for chave, valor in dados.items():
    print(f"{chave}: {valor}")
```

Dica: Dicionários são ideais para representar objetos do mundo real e estruturas de dados complexas.

Estruturas de Decisão

- **Operadores Lógicos:** Python possui operadores lógicos que são frequentemente utilizados em estruturas condicionais: and, or, not.

Exemplo prático:

```
idade = 20
nome = "Ana"
if idade >= 18 and nome == "Ana":
    print("Maior de idade chamada Ana")
```

- **Operadores de Comparação:** são utilizados para comparar dois valores e retornar um valor booleano, que pode ser True (verdadeiro) ou False (falso). Eles são essenciais para estruturas de controle, como condicionais (if, elif, else) e loops (while), permitindo a tomada de decisões com base em comparações.

Símbolo	Definição
==	Igual
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que

- Em Python, as instruções if, else e elif são usadas para criar estruturas de controle de fluxo que permitem a tomada de decisões com base em condições. Elas formam a base para implementar lógica condicional em um programa, permitindo que diferentes blocos de código sejam executados dependendo do resultado de uma ou mais condições.

- **IF (Se):** A instrução if verifica se uma condição é verdadeira. Se for, o bloco de código dentro do if é executado. Se a condição for falsa, o programa continua para o próximo bloco de código.

```
idade = 18
if idade >= 18:
    print("Você é maior de idade.")
```

- **Else (Senão):** A instrução else é usada junto com o if para especificar um bloco de código a ser executado caso a condição do if seja falsa.

```
idade = 16
if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

- **Elif (Else If):** O elif é a abreviação de "else if" e é usado quando há múltiplas condições a serem verificadas. Ele permite verificar condições adicionais se a condição do if original for falsa. Você pode usar quantos elif forem necessários.

```
nota = 75

if nota >= 90:
    print("Aprovado com excelência!")
elif nota >= 70:
    print("Aprovado")
elif nota >= 50:
    print("Recuperação")
```

```
else:
    print("Reprovado")
```

Estruturas de Repetição

- Além do while, temos a estrutura for, que é muito utilizada para iterar sobre sequências.

Exemplo de for loop:

```
nomes = ["Ana", "Carlos", "Beatriz"]
for nome in nomes:
    print(nome)
```

Range é muito útil em loops:

```
for i in range(5):
    print(i)  # Imprime de 0 a 4
```

- Além do for, também temos a estrutura while, que é usada para repetir um bloco de código enquanto uma condição for verdadeira.

Exemplo de while loop:

```
contador = 0
while contador < 5:
    print(contador)
    contador += 1
# Incrementa o valor de 'contador' para evitar um loop infinito
```

Neste exemplo, o while continuará executando o bloco de código enquanto a variável contadora for menor que 5. Assim que contador atingir 5, o loop será encerrado.

- A principal diferença entre for e while é que o for é usado quando sabemos o número de iterações, enquanto o while é usado quando não temos certeza de quantas vezes a condição será verdadeira.

Trabalhando com Funções

- As funções permitem reutilizar código e organizá-lo de forma lógica.

Definição de uma função:

```
def saudacao(nome):
    print(f"Olá, {nome}!")
```

Importando Módulos e Bibliotecas

- Python possui uma vasta biblioteca padrão e também permite a importação de pacotes externos para expandir suas funcionalidades.

Exemplo de importação:

```
import math
print(math.sqrt(16))  # Saída: 4.0
```

Manipulação de Arquivos

- Python facilita a leitura e escrita de arquivos.

Exemplo de leitura de arquivos:

```
with open('dados.txt', 'r') as arquivo:
    conteudo = arquivo.read()
    print(conteudo)
```