SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula veremos os fundamentos do Python, explorando sua sintaxe básica, tipos de dados, estruturas de controle e funções e manipulação de arquivos. Estes tópicos são essenciais para qualquer programador(a) que deseja iniciar com Python, fornecendo uma base sólida para desenvolvimentos mais avançados.

HANDS ON

Nesta aula prática, apresentaremos alguns pontos fundamentais de quando trabalhamos com o Python.

SAIBA MAIS

O Python é uma linguagem de programação altamente versátil e poderosa, utilizada em uma ampla variedade de aplicações. Nesta aula, exploraremos seus fundamentos, complementando o que aprendemos anteriormente. Para isso, forneceremos exemplos práticos para consolidar o seu aprendizado.

Vamos iniciar pela arranjo básico do Python, que oferece várias estruturas de dados embutidas que são fundamentais para uma programação eficaz. Vamos explorar algumas delas:

- **Listas** são coleções ordenadas e mutáveis de itens, elas são definidas usando colchetes.

```
frutas = ['maçã', 'banana', 'cereja']
frutas.append('laranja')
print(frutas)  # ['maçã', 'banana', 'cereja', 'laranja']
```

- **Tuplas** são coleções ordenadas e imutáveis de itens:

```
numeros = (1, 2, 3)
print(numeros[1])  # 2
```

- **Conjuntos**: são uma estrutura de dados que permite armazenar elementos únicos, sem repetição.

```
conjunto = {1, 2, 3, 4, 4}
print(conjunto)  # {1, 2, 3, 4}
```

- **Dicionários**: são coleções de pares chave-valor.

```
aluno = {'nome': 'João', 'idade': 20, 'curso': 'Engenharia'}
print(aluno['nome'])  # João
```

Controle de Fluxo

O Python possui várias estruturas de controle de fluxo que permitem a execução condicional de código e a repetição de blocos de código.

- Condicionais (if, elif, else):

```
idade = 18
if idade >= 18:
```

```
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

- Laços de Repetição (for, while):

```
for i in range(5):
    print(i)  # 0 1 2 3 4

contagem = 0
while contagem < 5:
    print(contagem)
    contagem += 1
```

Funções

No Python, funções permitem a modularização e a reutilização de código. Elas são definidas usando a palavra-chave `def`.

- Definindo e Chamando Funções

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

print(saudacao("Agnes"))  # Olá, Agnes!
```

- Funções com Argumentos Padrão

```
def saudacao(nome="mundo"):
    return f"Olá, {nome}!"

print(saudacao())  # Olá, mundo!
```

- Funções Lambda

```
dobro = lambda x: x * 2
print(dobro(5))  # 10
```

Manipulação de Arquivos

O Python facilita a manipulação de arquivos, permitindo a leitura e a escrita de dados de maneira simples.

- Escrevendo em Arquivos

```
with open('arquivo.txt', 'w') as file:
```

```
    file.write("Olá, mundo!")
```

- Lendo Arquivos

```
with open('arquivo.txt', 'r') as file:
    conteudo = file.read()
    print(conteudo) # Olá, mundo!
```

Módulos e Pacotes

- Importando Módulos

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

- Criando e Utilizando Módulos

```
# em meu_modulo.py
def soma(a, b):
    return a + b

# em outro arquivo
from meu_modulo import soma
print(soma(3, 4))  # 7
```

Exceções

O tratamento de exceções no Python permite que o programa lide com erros de maneira controlada.

- Capturando Exceções:

```
try:
    resultado = 10 / 0
except ZeroDivisionError:
    print("Erro: divisão por zero.")
```

- Lançando Exceções

```
def dividir(a, b):
    if b == 0:
        raise ValueError("O divisor não pode ser zero.")
    return a / b
```

```
try:
    print(dividir(10, 0))
except ValueError as e:
    print(e)
```

Compreensão de Listas

Compreensão de listas é uma maneira concisa de criar listas no Python.

- Exemplo de Compreensão de Listas:

Caso esse seja o seu primeiro contato com o list comprehension, ele nada mais é do que uma maneira de construir listas no Python de forma mais rápida, em uma linha de código.

A seguir, nós temos um exemplo:

```
quadrados = [x ** 2 for x in range(10)]
print(quadrados)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Decoradores

Decoradores são uma ferramenta poderosa em Python para modificar o comportamento de funções ou métodos.

- Exemplo de Decorador

```
def decorador_saudacao(func):
    def wrapper(*args, **kwargs):
        print("Saudação!")
        return func(*args, **kwargs)
    return wrapper

@decorador_saudacao
def ola(nome):
    print(f"Olá, {nome}!")

ola("Mundo")  # Saudação! Olá, Mundo!
```

Context Managers

Context managers permitem a alocação e a liberação de recursos de forma eficiente, utilizando a palavra-chave `with`.

- Exemplo de Context Manager

```
class GerenciadorDeContexto:
    def __enter__(self):
        print("Entrando no contexto.")
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        print("Saindo do contexto.")

with GerenciadorDeContexto():
    print("Dentro do bloco with.")
```

Agora que vimos estes fundamentos, te convido para reassistir nossa videoaula para que você possa entender alguns dos pontos básicos do Python na prática.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, vimos os fundamentos do Python, explorando sua sintaxe básica, tipos de dados, estruturas de controle, funções e manipulação de arquivos. Esses tópicos são essenciais para qualquer programador(a) que deseja iniciar com o Python, fornecendo uma base sólida para desenvolvimentos mais avançados.

REFERÊNCIAS

AWARI. **Fundamentos Do Python**: Aprenda As Bases Da Linguagem De Programação Mais Utilizada No Mercado De Tecnologia2023. Disponível em: <https://awari.com.br/fundamentos-do-python-aprenda-as-bases-da-linguagem-de-programacao-mais-utilizada-no-mercado-de-tecnologia>. Acesso em: 14 ago. 2024.

HASHTAG. **O que é list comprehension no python e como usá-lo**. 2021. Disponível em: <https://www.hashtagtreinamentos.com/list-comprehension-python?gad_source=1&gclid=CjwKCAjwps-zBhAiEiwALwsVYQ0JYnnNNinmVCZzMbCdHHKH_0opM3a3JvDJizHGJ4BTNdFI613U-hoCRkUQAvD_BwE>. Acesso em: 14 ago. 2024.

PYTHON. **Python 3.12.5 documentation**. 2024. Disponível em: <https://docs.python.org/3/>. Acesso em: 14 ago. 2024.

PALAVRAS-CHAVE

Palavras-chave: Python. Functions. Estruturas de controle.