O PRÍNCIPE DAS SERPENTES:

Dominando Python

Vitória Emilly

INTRODUÇÃO AO MUNDO MÁGICO DO PYTHON

Os principais comandos Python

Bem-vindo, jovem bruxo ou bruxa, ao universo encantado da linguagem de programação Python! Prepare sua varinha (ou melhor, seu teclado) e embarque nesta jornada de descoberta e aprendizado. Neste eBook, iremos desvendar os principais encantamentos (ou comandos) que o Python oferece, com exemplos práticos para que você possa dominar esta arte da programação.

01

Feitiços Básicos: Variáveis e Tipos de Dados

Em Python, podemos criar feitiços poderosos usando variáveis para armazenar informações.

VARIÁVEIS E TIPOS DE DADOS

As variáveis são como recipientes que armazenam dados em Python. Elas podem conter diferentes tipos de dados, como números, strings (texto), booleanos (True ou False) e outros. No exemplo fornecido, criamos variáveis para armazenar o nome, idade e altura de uma pessoa.

basic.py

```python
# Atribuindo valores a variáveis
nome = "Harry"
idade = 11
altura = 1.45
```

02

Portal de Entrada e Saída: Interagindo com o Mundo Exterior

Para interagir com o mundo exterior, utilizamos feitiços de entrada e saída de dados em Python.

ENTRADA E SAÍDA DE DADOS

Este tópico trata da interação entre o programa Python e o usuário ou outros sistemas. O comando "input()" permite que o programa solicite dados ao usuário, enquanto o "print()" exibe informações na tela. Isso é útil para criar programas interativos, como jogos, sistemas de cadastro, entre outros.

basic.py

```python
# Entrada de dados do usuário
nome = input("Digite seu nome: ")
print("Olá,", nome, "! Bem-vindo(a) ao mundo da magia.")

# Saída de dados
idade = 13
print("Sua idade é:", idade)
```

03

Encantamentos Matemáticos e Lógicos: Operadores Mágicos

Os operadores matemáticos e lógicos são como as fórmulas dos feitiços em Python.

OPERADORES MATEMÁTICOS E LÓGICOS

Aqui, falamos sobre os operadores matemáticos, como adição (+), subtração (-), multiplicação (*) e divisão (/), que são utilizados para realizar operações aritméticas em Python. Além disso, abordamos os operadores lógicos, como and, or e not, que são usados para combinar condições lógicas em expressões mais complexas.

basic.py

```python
# Operadores matemáticos
a = 10
b = 5
soma = a + b
subtracao = a - b
multiplicacao = a * b
divisao = a / b
```

basic.py

```python
# Operadores lógicos
x = True
y = False
resultado_and = x and y
resultado_or = x or y
resultado_not = not x
```

04

Poções de Controle: Estruturas de Decisão

Com as estruturas de controle, podemos direcionar o fluxo de nossos feitiços.

ESTRUTURAS DE DECISÃO

As estruturas de controle, como "if", "elif" e "else", permitem que o programa tome decisões com base em condições específicas. Isso é fundamental para direcionar o fluxo do programa de acordo com diferentes cenários. No exemplo fornecido, usamos uma estrutura "if-elif-else" para determinar a mensagem com base na idade.

basic.py

```python
# Estrutura de decisão
idade = 13
if idade < 11:
    print("Você é uma criança.")
elif idade < 18:
    print("Você é um jovem bruxo.")
else:
    print("Você é um bruxo adulto.")
```

05

Encantamento de Listas: Manipulação de Dados em Conjuntos

As listas são como os baús mágicos de Hogwarts, onde podemos guardar diversos itens.

MANIPULAÇÃO DE DADOS EM CONJUNTOS

Listas são coleções ordenadas de itens em Python. Podemos adicionar, remover e acessar itens em uma lista usando índices. No exemplo dado, criamos uma lista de feitiços e demonstramos como adicionar um novo feitiço a ela.

basic.py

```python
# Criando uma lista de feitiços
feitiços = ["Expelliarmus", "Lumos", "Alohomora"]

# Acessando elementos da lista
print(feitiços[0])  # Saída: Expelliarmus

# Adicionando um novo feitiço
feitiços.append("Expecto Patronum")
print(feitiços)  # Saída: ["Expelliarmus", "Lumos", "Alohomora", "Expecto Patronum"]
```

06

Teias de Laços: Explorando a Estrutura de Repetição *While*

O feitiço *while* permite repetir uma ação enquanto uma condição for verdadeira.

LAÇOS DE REPETIÇÃO (WHILE)

O comando "while" cria um laço de repetição que executa um bloco de código enquanto uma condição especificada for verdadeira. É útil quando não sabemos quantas vezes o bloco de código precisará ser repetido. No exemplo dado, o código repete a impressão de "Lançando feitiço..." cinco vezes.

basic.py

```python
# Estrutura de repetição while
contador = 0
while contador < 5:
    print("Lançando feitiço...")
    contador += 1
```

07

Força dos Feitiços: Desvendando a Estrutura de Repetição *For*

O feitiço *for* permite percorrer uma sequência de elementos.

LAÇOS DE REPETIÇÃO (FOR)

O comando "for" também cria um laço de repetição, mas é usado principalmente para iterar sobre uma sequência de elementos, como listas, strings ou intervalos numéricos. No exemplo dado, o código percorre a lista de feitiços e imprime cada feitiço na tela.

basic.py

```
# Estrutura de repetição for
feitiços = ["Expelliarmus", "Lumos", "Alohomora"]
for feitiço in feitiços:
    print("Lançando o feitiço:", feitiço)
```

08

Magia Avançada: Bibliotecas e Módulos

Além dos feitiços básicos, Python oferece uma vasta biblioteca de magias avançadas.

BIBLIOTECAS E MÓDULOS

Python possui uma vasta biblioteca padrão que fornece uma ampla gama de funcionalidades adicionais, como manipulação de data e hora, acesso a bancos de dados, processamento de texto, entre outros. Para utilizar essas funcionalidades, precisamos importar os módulos correspondentes usando a palavra-chave "import". No exemplo dado, importamos o módulo "random" e usamos sua função "randint()" para gerar um número aleatório.

basic.py

```python
# Importando uma biblioteca
import random

# Utilizando uma função da biblioteca
numero_aleatorio = random.randint(1, 100)
print("Seu número da sorte é:", numero_aleatorio)
```

09

Encantando Funções: Criando seus Próprios Feitiços

Nada como criar seus próprios feitiços em Python, certo? Veja como criar uma função:

FUNÇÕES

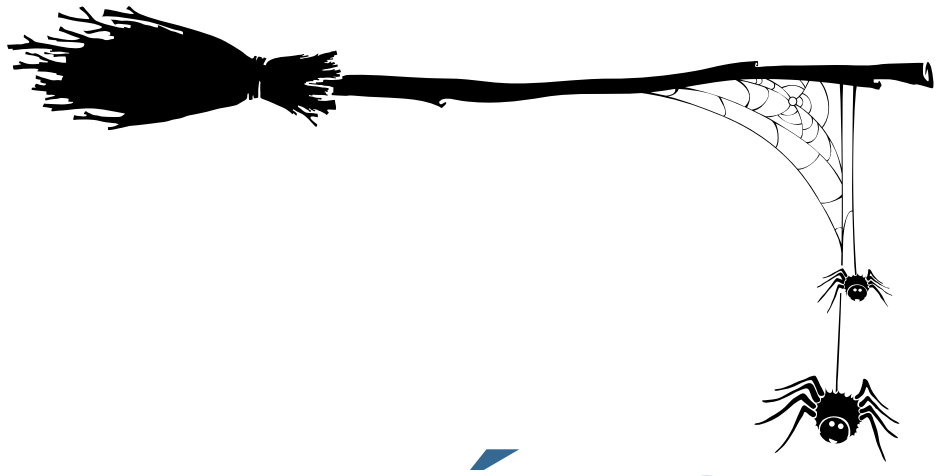
As funções em Python são blocos de código que realizam uma tarefa específica e podem ser reutilizadas em diferentes partes do programa. Elas são definidas usando a palavra-chave "def" seguida pelo nome da função e seus parâmetros. No exemplo dado, criamos uma função de saudação que recebe o nome de uma pessoa como parâmetro e retorna uma mensagem personalizada.

basic.py

```python
# Definindo uma função
def saudacao(nome):
    return "Olá, " + nome + "! Bem-vindo(a) ao mundo da magia."

# Chamando a função
mensagem = saudacao("Hermione")
print(mensagem)  # Saída: Olá, Hermione! Bem-vindo(a) ao mundo da magia.
```

AGRADECIMENTOS



OBRIGADA POR LER ATÉ AQUI.

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.

https://github.com/vitsantos/prompts-recipe-to-create-a-ebook/

Vitória Emilly

GitHub | LinkedIn | Instagram