Introdução à Programação em Python: Entrada e Saída Básica

Mateus de Oliveira Silveira

1 A Função print

A função integrada `print` em Python é uma ferramenta essencial para exibir informações ao usuário. Ela pode receber múltiplos argumentos e oferece várias opções de formatação. Vamos explorar suas funcionalidades em detalhes.

1.1 Uso Básico

O uso mais simples da função `print` é passar uma string como argumento:

```
print('Bem-vindo ao fascinante mundo do Python!')
```

1.2 Strings com Aspas Simples e Duplas

Em Python, você pode usar aspas simples (') ou aspas duplas (") para delimitar strings. Ambas produzem o mesmo resultado:

```
print('Python é uma linguagem versátil.')
print("Python é uma linguagem poderosa.")
```

A escolha entre aspas simples e duplas é uma questão de preferência pessoal, mas usar aspas duplas pode ser útil quando a string contém apóstrofos:

```
print("A sintaxe de python e facil de aprender.")
```

1.3 Imprimindo Múltiplos Itens

A função `print` pode receber múltiplos argumentos separados por vírgula. Por padrão, esses argumentos são separados por um espaço na saída:

```
print('Python', 'é', 'incrível!')
nome = 'Alice'
idade = 30
print('Meu nome é', nome, 'e tenho', idade, 'anos.')
```

1.4 Sequências de Escape

Python usa a barra invertida (\) como caractere de escape para representar caracteres especiais em strings. Aqui estão algumas sequências de escape comuns:

- `\n`: Nova linha
- `\t`: Tabulação
- `\\`: Barra invertida
- `\"`: Aspas duplas
- `\'`: Aspas simples

Exemplo:

```
print('Linha 1\nLinha 2\n\tLinha 3 (com tabulação)')
print('Ele disse: \"Python é incrível!\"')
```

1.5 Parâmetros da Função print

A função `print` aceita vários parâmetros opcionais que permitem personalizar a saída:

- `sep`: Especifica o separador entre os argumentos (padrão é espaço).
- `end`: Especifica o que imprimir no final (padrão é nova linha).

Exemplo:

```
print('Python', 'é', 'incrível', sep='-', end='!\n')
print('Contagem:', 1, 2, 3, sep=', ', end='... ')
print('Go!')
```

2 Strings com Aspas Triplas

Strings com aspas triplas em Python são delimitadas por três aspas duplas (`"""`) ou três aspas simples (`'''`). Elas são particularmente úteis para strings multilinhas ou strings que contêm tanto aspas simples quanto duplas:

```
poema = """
Duas estradas divergiam em um bosque amarelo,
E lamentando não poder seguir ambas
E ser um só viajante, por muito tempo fiquei parado
E olhei para uma delas tão distante quanto pude
Até onde se perdia na mata;
"""
print(poema)

codigo = '''
def saudacao(nome):
```

```
    print(f"Olá, {nome}!")
    print('Bem-vindo ao Python!')
'''
print(codigo)
```

3 Formatação de Strings

Python oferece várias maneiras de formatar strings, permitindo que você insira valores de variáveis em strings de maneira eficiente e legível.

3.1 F-strings (Python 3.6+)

F-strings são uma maneira concisa e legível de incorporar expressões dentro de strings:

```
nome = 'Alice'
idade = 30
print(f'Meu nome é {nome} e tenho {idade} anos.')
print(f'Daqui a 5 anos, terei {idade + 5} anos.')
```

3.2 Método format()

O método `format()` é outra maneira de formatar strings:

```
nome = 'Bob'
idade = 25
altura = 1.75
print('Meu nome é {}, tenho {} anos e minha altura é {:.2f}m.'.
    format(nome, idade, altura))
```

4 Obtendo Entrada do Usuário

A função `input()` é usada para obter entrada do usuário. Ela exibe uma mensagem opcional (prompt) e retorna a entrada do usuário como uma string.

4.1 Uso Básico

```
nome = input("Qual é o seu nome? ")
print(f"Olá, {nome}! Bem-vindo ao Python.")
```

4.2 Convertendo Entrada para Números

Como `input()` sempre retorna uma string, é necessário converter a entrada para o tipo desejado quando se trabalha com números:

```
idade = int(input('Quantos anos você tem? '))
altura = float(input('Qual é a sua altura em metros? '))
print(f"Você tem {idade} anos e {altura:.2f}m de altura.")
print(f"Daqui a 10 anos, você terá {idade + 10} anos.")
```

5 Exemplo Prático: Calculadora Simples

Vamos criar uma calculadora simples que demonstra o uso de entrada e saída:

```python
print("Calculadora Simples")
num1 = float(input("Digite o primeiro número: "))
num2 = float(input("Digite o segundo número: "))

soma = num1 + num2
subtracao = num1 - num2
multiplicacao = num1 * num2
divisao = num1 / num2 if num2 != 0 else "Erro: Divisão por zero"

print(f"\nResultados:")
print(f"{num1} + {num2} = {soma}")
print(f"{num1} - {num2} = {subtracao}")
print(f"{num1} * {num2} = {multiplicacao}")
print(f"{num1} / {num2} = {divisao}")
```

6 Conclusão

Neste módulo, exploramos em detalhes:

- O uso avançado da função print() para exibir informações
- Diferentes tipos de strings e sequências de escape em Python
- Técnicas de formatação de strings, incluindo f-strings e o método format()
- Como usar a função input() para ler dados do usuário
- Conversão de tipos de dados para trabalhar com entradas numéricas
- Um exemplo prático combinando entrada e saída em uma calculadora simples

Estes conceitos formam a base para a interação com o usuário em programas Python e são fundamentais para o desenvolvimento de aplicações mais complexas.

7 Exercícios Práticos

1. Crie um programa que solicite o nome, idade e cidade do usuário, e então imprima uma mensagem formatada com essas informações.

2. Desenvolva uma calculadora de IMC (Índice de Massa Corporal). Peça ao usuário sua altura em metros e peso em quilogramas, calcule o IMC e imprima o resultado com duas casas decimais.

3. Escreva um programa que converta temperaturas entre Celsius e Fahrenheit. Peça ao usuário para escolher a direção da conversão e então realize o cálculo.

4. Crie um gerador de cumprimentos. Peça o nome do usuário, hora do dia (manhã, tarde ou noite) e gere um cumprimento personalizado.

5. Desenvolva um programa que peça ao usuário para inserir uma frase e então imprima: - A frase em maiúsculas - A frase em minúsculas - O número de caracteres na frase - A frase com cada palavra capitalizada