Fundamentos de Programação em Python

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis Mayr Lima Martins

Aula 4: Manipulação de Strings

Bem-vindo a mais uma aula de **Fundamentos de Programação em Python**!

As strings são um dos tipos de dados fundamentais em Python e representam sequências de caracteres. Elas são amplamente utilizadas na programação para armazenar, processar e formatar textos, sendo essenciais para tarefas como entrada de dados, exibição de informações e manipulação textual em arquivos, bancos de dados e aplicações web.

Em Python, uma string pode ser definida usando aspas simples ('...'), aspas duplas ("...") ou triplas ('''...''' ou """..."""") para textos de múltiplas linhas. Além disso, a linguagem fornece uma variedade de métodos embutidos e operações que permitem modificar, pesquisar e transformar strings de maneira eficiente.

** O que veremos hoje?**

- Como criar e acessar strings em Python.
- Como utilizar fatiamento (slicing) para extrair partes de uma string.
- Métodos úteis para modificar e formatar strings.
- Como utilizar operações com strings (concatenação, repetição, substituição).
- Como percorrer uma string com loops e realizar buscas com expressões regulares.

** Objetivos de Aprendizagem**

Ao final desta aula, você será capaz de:

- Compreender os conceitos fundamentais de strings em Python.

- Acessar e manipular substrings utilizando indexação, fatiamento e métodos de formatação.

-

Percorrer strings e buscar informações.

Conceitos Fundamentais

No contexto da programação, uma string é uma estrutura de dados fundamental utilizada para representar e manipular sequências de caracteres. Em Python, as strings são objetos da classe str e podem ser definidas utilizando aspas simples ('...'), aspas duplas ("...") ou triplas ('''...''' ou """..."""), permitindo a criação de textos de uma única linha ou múltiplas linhas

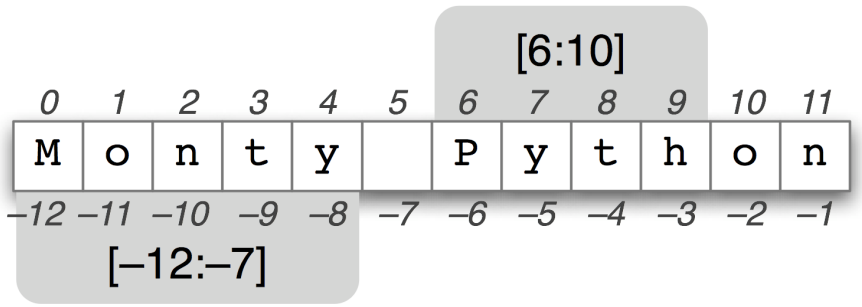
```
empty_string = ""
print(empty_string)

greetings = "Hi! My name is Snake. I have 15 years old."
print(greetings)

# Verifique o comprimento da string
```

Indexação de Strings em Python

Indexação é uma técnica fundamental ao lidar com strings em Python. Ela permite acessar ou modificar individualmente os caracteres dentro de uma string. No contexto das strings, o termo "index" se refere à posição do caractere dentro da sequência.

A sintaxe básica para indexação de strings em Python é: `string[index]`, onde `string` é a string que você deseja trabalhar e `index` é a posição do caractere que você gostaria de acessar. A contagem dos índices começa em 0, o que significa que o primeiro caractere da string tem um índice de 0.

```
comedy = "Monty Python"
```




```
# Exemplos de indexação
```

Slicing em Strings: Cortando e Manuseando Seções

O conceito de "slicing" (ou corte) é uma ferramenta poderosa em programação, especialmente quando lidamos com sequências de dados como strings. Em Python, a operação de slicing permite que você extraia ou modifique partes específicas da string sem ter que trabalhar com todos os caracteres.

A sintaxe básica para o corte de strings é:

`string[start:stop]`

Onde `start` e `stop` são índices, definindo a parte da string que você deseja extrair. Os seguintes valores podem ser usados:

- **start**: O índice do caractere inicial que deseja extrair. Se omitido, o corte inicia automaticamente na posição 0 (o primeiro caractere).
- **stop**: O índice do caractere final a ser incluído no corte. Se omitido, o corte continua até o fim da string.

```
# Exemplo de slicing
```

Exercício 1:

1. Obtenha os caracteres "y", "t", "h" da string "Python".
2. Use slicing para extrair apenas os últimos 4 caracteres de uma string passada pelo usuário.

```
# Adicione sua solução aqui
```

Métodos de Manipulação de Strings

As strings são imutáveis, o que significa que, uma vez criadas, seus valores não podem ser modificados diretamente. No entanto, Python oferece uma ampla variedade de métodos embutidos que possibilitam a manipulação eficiente dessas estruturas, como a conversão de maiúsculas para minúsculas, substituição de caracteres, extração de substrings, concatenação e formatação.

```
frase = "Python"
frase[0] = 'J'
print(frase)  # Saída: Jython
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[48], line 2
      1 frase = "Python"
----> 2 frase[0] = 'J'
      3 print(frase)

TypeError: 'str' object does not support item assignment
```

upper() e lower()

- **upper()**: Retorna a string em maiúsculas.
- **lower()**: Retorna a string em minúsculas.

```
print(greetings.lower())
```

```
print(greetings.upper())
```

```
print(greetings.capitalize())

print(greetings.title())
```

strip(), lstrip() e rstrip()

- **strip([chars])**: Remove caracteres de ambos os lados (padrão é espaço em branco).
- **lstrip([chars])**: Remove caracteres da esquerda.
- **rstrip([chars])**: Remove caracteres da direita.

```
frase = '   Minha frase   '
print(frase.strip())  # Minha frase
```

Operações com Strings

As operações básicas com strings são fundamentais para qualquer tipo de manipulação de texto em Python. Aqui estão duas dessas operações essenciais: concatenação e repetição.

Concatenação de Strings (+)

A concatenação é o processo de juntar duas ou mais strings para formar uma única string. Em Python, você pode usar o operador `+` para concatenar strings. Aqui está um exemplo simples:

```
nome = 'João'
sobrenome = 'Silva'

nome_completo = nome + ' ' + sobrenome

print(nome_completo)  # Saída: João Silva
```

Repetição de Strings (*)

Outra operação importante é a repetição de strings. Você pode usar o operador `*` para multiplicar uma string por um número inteiro. Aqui está como fazer isso:

```
frase = 'Python'
repeticao = frase * 3

print(repeticao)  # Saída: PythonPythonPython
```

Exercício 2: Faça um programa que concatene o nome, a idade, o RG, o CPF e a cidade do usuário. As informações do usuário devem ser obtidas pela função `input()`

```
# Adicione sua solução aqui
```

split() e join()

- **split([sep])**: Divide a string em um lista de sub-strings, baseado no separador (padrão é espaço em branco).
- **join(iterable)**: Combina uma lista de strings em uma única string.

```
nome = 'João,Silva'
print(nome.split(','))  # ['João', 'Silva']

# Split name from email address
username = 'user@mail.com'

nomes = ['João', 'Maria']
print(','.join(nomes))  # João,Maria

# Join nome e email
username = ['user', 'mail.com']

colors = ['red', 'green', 'blue', 'yellow', 'white', 'black']
```

splitlines() e joinlines()

- **splitlines()**: Divide a string em um lista, considerando cada linha como uma sub-string.
- **joinlines([sep])**: Combina uma lista de strings em uma única string.

```
frase = 'Linha 1\nLinha 2'
print(frase.splitlines())

linhas = ['Linha 1', 'Linha 2']
print('\n'.join(linhas))
```

replace()

- **replace(old, new)**: Substitui a ocorrência da string `old` por `new`.

Strings são imutáveis, o que significa que, uma vez criadas, seus valores não podem ser modificados diretamente.

```
frase = 'Eu não gosto de chocolate.'
print(frase.replace('chocolate', 'mel'))  # Eu não gosto de mel.
```

find()

O método `find()` é uma ferramenta poderosa em Python para localizar a primeira ocorrência de um caractere ou substring dentro de outra string.

```
frase = 'Python é incrível.'

# Encontrar a primeira ocorrência da letra "P"
```

```
posicao_p = frase.find('P')
if posicao_p != -1:
    print(f'A letra "P" foi encontrada na posição {posicao_p}.')
else:
    print('A letra "P" não foi encontrada.')
```

f-Strings

Em Python 3.6+, você pode usar a sintaxe chamada "f-string" ou "formatted string literal" para criar strings formatadas de forma fácil e elegante. Aqui está uma visão geral rápida desta ferramenta poderosa!

Para utilizar f-strings, você envolve o texto com aspas, segue por um caractere especial `f`, e então você pode usar a sintaxe de formato usual para especificar os valores dos dados.

```
nome = 'João'
idade = 25
print(f"Olá, meu nome é {nome} e tenho {idade} anos.")

pi = 3.14159265359
print(f'O valor aproximado de pi é {int(pi)}.')
```

Exercício 3: Refaça o exercício 2 usando f-strings.

```
# Adicione sua solução aqui
```

Percorrendo Strings com Loops

Você pode usar um loop `for` para percorrer cada caractere de uma string. Este método é útil quando precisamos manipular ou processar individualmente cada elemento da string.

- Iterando sobre uma string com for.
- Contando caracteres específicos.
- Construção de novas strings com loops.

```
frase = 'Python é incrível.'

for caractere in frase:
    print(caractere)
```

Processando strings: Se você quiser manipular ou processar os caracteres de alguma forma específica enquanto percorre a string, pode fazer isso dentro do bloco de código executado no loop. Aqui está um exemplo que capitaliza cada palavra:

```
frase = 'Python é incrível.'

nova_frase = ''
```

```python
for caractere in frase:
    if caractere == ' ':
        nova_frase += caractere
    else:
        nova_frase += caractere.upper()

print(nova_frase)
```

```python
# Exemplo (atividade do notebook anterior): loop `for` e um comando `if`
frase = 'Python é incrível.'

contagem_vogais = 0
contagem_consoantes = 0

for caractere in frase:
    if caractere.lower() == 'a':
        contagem_vogais += 1
    elif caractere.lower() == 'e':
        contagem_vogais += 1
    elif caractere.lower() == 'i':
        contagem_vogais += 1
    else:
        contagem_consoantes += 1

print(f'Quantidade de vogais: {contagem_vogais}')
print(f'Quantidade de consoantes: {contagem_consoantes}')
```

Exercícios para Praticar:

Nível Fácil: Escreva um programa que recebe uma string do usuário e verifica se a string é um palíndromo. Palíndromo é toda palavra ou frase que pode ser lida de trás para frente e que, independente da direção, mantém o seu sentido. Exemplos:

- Ada
- Ana
- Missa é assim.
- O lobo ama o bolo.

A verificação deve ignorar "case", ou seja, letras maiúsculas e minúsculas.

Nível Médio: Escreva um programa que recebe uma string do usuário e encontra a palavra mais frequente nessa string. Use apenas as funcionalidades básicas de strings e loops em Python, conforme visto até esta aula.

Conclusão

Parabéns por concluir esta aula!

O que aprendemos hoje?

Nesta aula, exploramos os principais conceitos e técnicas para manipulação de strings em Python. Aprendemos como criar, acessar e modificar strings utilizando diferentes métodos e abordagens, além de compreender a importância das strings na programação e em aplicações do mundo real.

Próximos Passos

- Resolva os problemas na seção "Exercícios para Praticar".
- Experimentar diferentes abordagens para formatação e substituição de strings (de entrada do usuário)

Parabéns pela dedicação! Nos vemos na próxima aula!