Lógica de Programação II

Tuplas e Strings

Tuplas

- Tuplas são sequências de valores, da mesma forma que listas
- Mas, existem diferenças...
  - Os valores de uma tupla, ao contrário de uma lista, são **imutáveis**
  - Tuplas usam parênteses enquanto listas usam colchetes

```
>>> lista = [1, 2, 3, 4]
>>> tupla = (1, 2, 3, 4)
```

Tuplas

- Tupla vazia

```
>>> tupla = ()
```

- Tupla com um único elemento (note a necessidade da vírgula, mesmo sendo um único elemento)

```
>>> tupla = (1,)
```

Acesso aos Elementos de uma Tupla

- Acesso é feito pela posição, da mesma forma que nas listas

```
>>> tupla = ("Maria", "Joao", "Carlos")
>>> tupla[0]
"Maria"
```

- Também é possível usar slices

```
>>> tupla = ("Maria", "Joao", "Carlos")
>>> tupla[0:2]
("Maria", "Joao")
```

Atualização de Tuplas

- Como são imutáveis, não é permitido atualizar os valores dentro de uma tupla

```
>>> tupla = ("Maria", "Joao", "Carlos")
>>> tupla[0] = "Ana"
TypeError: 'tuple' object does not support
   item assignment
```

Operadores Básicos sobre Tuplas

Expressão	Resultado	Descrição
len((1,2,3))	3	Número de elementos que a tupla contém
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenação
(1,) * 4	(1,1,1,1)	Repetição
3 in (1, 2, 3)	True	Pertencimento
for x in (1,2,3): print(x)	1 2 3	Iteração

Atribuição de tuplas

- Para trocar entre si os valores de duas variáveis

```
>>> temp = a
>>> a = b
>>> b = temp
```

- Python fornece uma forma de atribuição de tupla que resolve esse problema elegantemente:

```
>>> a, b = b, a
```

- O número de variáveis na esquerda e o número de valores na direita deve ser igual:

```
>>> a, b, c, d = 1, 2, 3
ValueError: unpack tuple of wrong size
```

Tuplas como valores de retorno

- Funções podem retornar tuplas como valor de retorno para poder ter mais de um valor de retorno

```
def troca(x, y):
    return y, x
```

- Exemplo, atribuir o valor de retorno para uma tupla com duas variáveis

```
a, b = troca(a, b)
```

- Versão incorreta

```
def troca(x, y):        # versao incorreta
    x, y = y, x
```

Strings

- Representam informação textual

```
nome = “Maria Silva”
nacionalidade = “brasileira”
nome_mae = “Ana Santos Silva”
nome_pai = “Jonas Nunes Silva”
```

Acesso a conteúdo das Strings

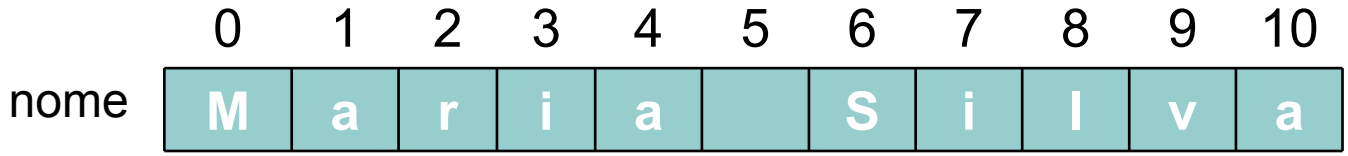
- Acesso pode ser feito pelo nome da variável que contém a string

```
nome = “Maria Silva”
print(nome)
```

Acesso a conteúdo das Strings

- String pode ser tratada como uma tupla
- Caracteres podem ser acessados pela sua posição dentro da String

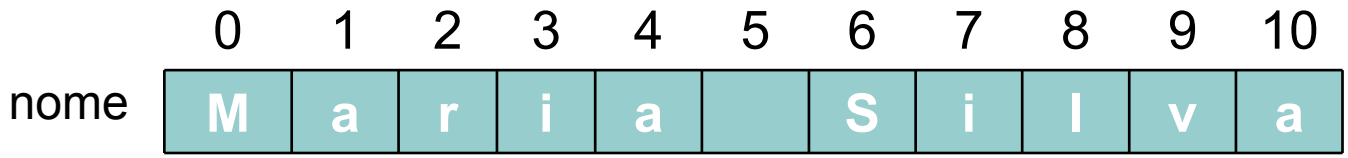
```
>>> nome = "Maria Silva"
>>> print(nome[0])
M
>>> print(nome[6])
S
```



Acesso a conteúdo das Strings

- Fatias também podem ser usadas

```
>>> nome = "Maria Silva"
>>> print(nome[:5])
Maria
>>> print(nome[6:])
Silva
```



Alteração

- Como as tuplas, o conteúdo das strings não pode ser alterado – são sequências imutáveis

```
>>> nome = “Maria Silva”
>>> nome[3] = “t”
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item
  assignment
```

Operadores

- Alguns operadores que atuam sobre sequências podem ser usados em strings
  - in
  - len
  - +
  - *

in

- letra **in** string
  - Retorna True ou False

```
>>> nome = “Maria Silva”
>>> “M” in nome
True
>>> “B” in nome
False
>>> “m” in nome
False
```

len

- len(string)
  - Retorna a quantidade de caracteres da string

```
>>> nome = “Maria”
>>> len(nome)
5
>>> nome = “Maria Silva”
>>> len(nome)
11
```

+ (Concatenação)

- string1 + string2
  - Concatena duas strings

```
>>> nome = “Maria” + “Silva”
>>> nome
MariaSilva
>>> nome = “Maria”
>>> sobrenome = “Silva”
>>> nome_completo = nome + sobrenome
>>> nome_completo
MariaSilva
```

* (Repetição)

- string * int
  - Repete a string **int** vezes

```
>>> nome = “Maria”
>>> nome_repetido = nome * 2
>>> nome_repetido
MariaMaria
```

Percorrendo uma String

- Os elementos de uma string podem ser acessados usando uma estrutura de repetição

```
nome = "Maria Silva"
for letra in nome:
        print(letra)
```

```
nome = "Maria Silva"
indice = 0
while indice < len(nome):
        print(nome[indice])
        indice +=1
```

Exercícios

1. Escreva uma função que recebe uma frase e uma palavra antiga e uma palavra nova. A função deve retornar uma string contendo a frase original, mas com a última ocorrência da palavra antiga substituída pela palavra nova. A entrada e saída de dados deve ser feita no programa principal.

   - Exemplo:
     - Frase: “Quem parte e reparte fica com a maior parte”
     - Palavra antiga: “parte”
     - Palavra nova: “parcela”
     - Resultado a ser impresso no programa principal: “Quem parte e reparte fica com a maior parcela”

Exercícios

2. Faça uma função que recebe uma string que representa uma cadeia de DNA e gera a cadeia complementar. A entrada e saída de dados deve ser feita pelo programa principal.

   - Exemplo:
     - Entrada: AATCTGCAC
     - Saída: TTAGACGTG

Referências

- Slides baseados no curso de Programação de Computadores I da Prof. Vanessa Braganholo