Aula 05

Programação Orientada a Objetos

String e conjuntos de dados

Yuri Max

Natal - RN

13 de setembro de 2022

Conteúdo

- String
- Lista
- Tupla
- Conjunto
- Dicionário
- Exercícios

String

- É uma coleção ordenada de caracteres;
- Cada caractere armazenado possui uma posição na memória;
- É possível acessar diretamente uma ou várias posições com o uso de [];
- A primeira posição é [0];
- Não é possível modificar um caractere diretamente;

```
s = 'abcdefghijklmnopqrstuvwxyz'

print(s[4])
print(s[4:10])
```

```
e
efghij
```

String

- Cada caractere possui um representante decimal, hexadecimal ou binário;
- Os primeiros 127 caracteres são compatíveis com ASCII;
- Podem ser percorridas com laços (while, for);
- Equivalente numérico obtido com ord();
- Equivalente caractere obtido com chr();

```
s = 'abcdefghijklmnopqrstuvwxyz'


for i in range(5):
    print(ord(s[i]))

for i in range(97,101+1):
    print(chr(i))
```

```
97
98
99
100
101
a
b
c
d
e
```

String

- Funções auxiliares:
  - len() obtém o número de caracteres;
  - .upper() coloca todos os caracteres em maiúsculo;
  - .lower() coloca todos os caracteres em minúsculo;
  - .count() retorna o número de repetições de um char;
  - .find() procura um char e retorna sua posição;

```
s = 'abcdefghijklmnopqrstuvwxyzA'

print(s.upper())
print(s.upper().lower())
print(s.upper().lower().count('a'))
print(s.upper().lower().find('a'))
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZA
abcdefghijklmnopqrstuvwxyza
2
0
```

String

- Funções auxiliares:
  - .capitalize() Primeira letra em maíusculo;
  - .isalnum() Retorna True se todos são num/letras;
  - .isalpha() Retorna True se todos são letras;
  - .isnumeric() Retorna True se todos são números
  - .strip() Remove espaços indesejados no inicio e no fim;

```
txt = '   oi123    '
txt2 = 'oi123'
print(txt.capitalize(), txt.strip(), txt2.isalnum(), txt2.isalpha(), txt2.isnumeric())
```

```
   oi123     oi123 True False False
```

String

- Funções auxiliares:
  - + adiciona uma string ao final da outra;
  - * multiplica a string por um número natural;

```
s = 'abcdefghijklmnopqrstuvwxyzA'
s += 'BC'
print(s)
s *= 2
print(s)
s = s + 2*'BC'
print(s)
```

```
abcdefghijklmnopqrstuvwxyzABC
abcdefghijklmnopqrstuvwxyzABCabcdefghijklmnopqrstuvwxyzABC
abcdefghijklmnopqrstuvwxyzABCabcdefghijklmnopqrstuvwxyzABCBCBC
```

String

- Funções auxiliares:
  - .format();
  - {:+} – Adiciona um sinal na frente do numero
  - {:,} – Separa o milhar com virgula
  - {:.Xf} – Limita a X casas decimais float
  - {:.X%} – Apresenta o número em porcentagem com X casas decimais

```
1    print('{:+,.2f}, {:.0%}'.format(-2000, 0.1))
```

```
-2,000.00, 10%
```

String

- Exemplo:

- ▶ Receba um texto do usuário, então imprima quantas palavras foram digitadas;
- ▶ Crie um programa que receba 10 palavras do usuário e as some em uma única string;

Lista

- Lista mutável de n entradas;
- Equivalente a um vetor ou pilha de elementos;
- Cada elemento pode ser de qualquer tipo (string, lista, etc.);
- Caracterizado por colchetes;
- Cada elemento pode ser acessado por colchetes e a posição;
- A posição inicial é 0;

```
lista = ['Hello World!', 'bc', [1, 2, 3, 4], 5+3j]

print(lista)
print(lista[0], type(lista[0]))
print(lista[3], type(lista[3]))
```

```
['Hello World!', 'bc', [1, 2, 3, 4], (5+3j)]
Hello World! <class 'str'>
(5+3j) <class 'complex'>
```

Lista

- Funções auxiliares
  - .append() adiciona um elemento ao final da lista;
  - list() transforma cada caractere de uma string em lista
  - .split() cria uma lista com os caracteres de uma string com a separação sendo o argumento;
  - .join() transforma uma lista em string com uma

```
lista = ['Hello World!', 'bc', 5+3j]

lista.append('TESTE')

print(lista)
print(list(lista[3]))
print((lista[0].split(' ')))
print(' '.join(lista[0].split(' ')))
```

```
['Hello World!', 'bc', (5+3j), 'TESTE']
['T', 'E', 'S', 'T', 'E']
['Hello', 'World!']
Hello World!
```

Lista

- Funções auxiliares
  - .reverse() inverte a posição dos elementos;
  - .sort() coloca os elementos em ordem crescente (int);
  - .pop() remove o último elemento;
  - .remove() remove um elemento igual ao argumento;

```
lista = [5, 4, 5, 2, 1]

lista.reverse()
print(lista)

lista.sort()
print(lista)

lista.pop()
print(lista)

lista.remove(2)
print(lista)
```

```
[1, 2, 5, 4, 5]
[1, 2, 4, 5, 5]
[1, 2, 4, 5]
[1, 4, 5]
```

Lista

- Funções auxiliares:
  - + adiciona uma lista ao final da outra;
  - * multiplica a lista por um número natural;
  - max() e min() retornam o valor máximo e mínimo da lista;
  - len() retorna a quantidade de elementos

```
lista = [5, 1]

lista += 2*lista

print(lista)
```

```
[5, 1, 5, 1, 5, 1]
```

Lista

- Exemplo:

- Crie uma lista que receba 10 números do usuário e os coloque em ordem;
- Crie um programa que remova o primeiro e o último dígito de uma lista de 5 posições com valores aleatórios;
- Crie 1000 números aleatórios e diga o maior e o menor entre eles;

Tupla

- Similar a lista;
- Possuem entradas imutáveis;
- Bom para proteger dados ou conjunto de dados com valor fixo;
- É caracterizada por parêntesis;
- Pode ser acessada como uma lista;
- Consomem menos recursos computacionais;
- Pode transformar uma lista em tupla com tuple()

```
tupla = ('Hello World!', 'bc', 5+3j)

print(tupla)
print(tupla[0])
```

```
('Hello World!', 'bc', (5+3j))
Hello World!
```

Tupla

- Funções auxiliares
  - sum() retorna a soma de todos os elementos;
  - any() retorna True se algum elemento é verdadeiro;
  - all() retorna True se todos os elementos são verdadeiros;
  - .index() procura um elemento e retorna sua posição

```
tupla = (1, 4, 3,False)

print(sum(tupla))
print(any(tupla))
print(all(tupla))
print(tupla.index(4))
```

```
8
True
False
1
```

Tupla

- Funções auxiliares
  - + retorna uma tupla concatenada;
  - sorted() retorna uma lista em ordem crescente;
  - .count() retorna o número de repetições de um elemento;
  - Aceita operadores lógicos (< > ==)
  - m

```
tupla = (1, 4, 3, False)
tupla2 = (2,)

print(tupla+tupla2)
print(sorted(tupla+tupla2))
print((tupla+tupla2).count(4))
print(tupla > tupla2)
```

```
(1, 4, 3, False, 2)
[False, 1, 2, 3, 4]
1
False
```

Tupla

- Exemplo:

► Crie uma tupla com os dias da semana. Imprima o dia da semana e a posição do dia da semana de acordo com o dia de hoje;

Conjunto

- Coleção não ordenada de elementos;
- Não admite elementos duplicados;
- Caracterizado por {};
- Pode-se usar a função set() para criar conjuntos mutáveis e frozenset() para conjuntos imutáveis;
- Aceita inteiros, strings, complexos e tuplas;

```
conjunto1 = {1, 2, 3}
conjunto2 = {'morango'}
conjunto3 = set()

print(conjunto1)
print(conjunto2)
print(conjunto3)
```

```
{1, 2, 3}
{'morango'}
set()
```

Conjunto

- Suporta algumas operações de conjuntos:
  - A - B retorna os elementos de A que não estão em B;
  - A | B faz a união dos elementos de A e B;
  - A & B retorna os elementos de A que estão em B
  - A ^ B retorna os elementos de A e B que não são compartilhados

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print(A - B)
print(A | B)
print(A & B)
print(A ^ B)
```

```
{1, 2}
{1, 2, 3, 4, 5, 6}
{3, 4}
{1, 2, 5, 6}
```

Conjunto

- Exemplo:

► Crie dois conjuntos com 5 entradas dadas pelo usuário e depois mostre os elementos iguais em ambos os conjuntos.

Dicionário

- Coleção de elementos não ordenados;
- Possuem chaves para acessar suas entradas;
- Chaves são imutáveis;
- Os valores são mutáveis;
- É possível adicionar novas chaves;
- Chaves podem ser inteiros, tuplas e strings.

```
A = {'A': ord('A'), 'B': ord('B')}

print(A)

A['C'] = ord('C')
print(A)

print(A['C'])
```

```
{'A': 65, 'B': 66}
{'A': 65, 'B': 66, 'C': 67}
67
```

Dicionário

- Funções complementares:
  - .items() retorna todas as chaves e valores de um dicionário;
  - .keys() retorna todas as chaves de um dicionário;
  - .values() retorna todos os valores de um dicionário;
  - dict() for onário;

```
A = dict(A = ord('A'), B = ord('B'), C = ord('C'))

print(A.items(), type(A.items()))

print(A.keys(), type(A.keys()))

print(A.values(), type(A.values()))
```

```
dict_items([('A', 65), ('B', 66), ('C', 67)]) <class 'dict_items'>
dict_keys(['A', 'B', 'C']) <class 'dict_keys'>
dict_values([65, 66, 67]) <class 'dict_values'>
```

Dicionário

- Exemplo:

- ▶ Crie um dicionário com todos os valores ASCII de a-z.
- ▶ Crie um dicionário com as chaves items, keys e values. Os valores correspondentes devem ser as explicações do que fazem em um dicionário. Depois imprima todos em um único print().

Exercícios

1. Escreva um programa que receba uma frase do usuário e separe por palavras. Depois diga quantas letras tem cada palavra e quantas palavras foram escritas.
2. Escreva um programa que lê uma frase do usuário e retorne em quais posições a letra ‘a’ apareceram.
3. Crie um programa que recebe do usuário uma string e só encerra quando todos os elementos digitados estiverem entre a-z.
4. Crie um programa armazene todos os números digitados pelo usuário, então coloque-os em ordem decrescente e imprima.
5. Faça um programa que diga quantos números negativos há em uma lista.
6. Faça um programa que una duas listas aleatórias de inteiros, com 100 entradas, e diga quais números são primos e em que posição eles aparecem.
7. Crie dois conjuntos aleatórios de 15 posições com valores aleatórios de a-z. Depois diga quais elementos são iguais e quais são diferentes entre esses dois conjuntos.
8. Crie uma tupla de 10 elementos aleatórios de a-z, então transforme em uma lista e depois em uma string com todos os caracteres em maiúsculo.
9. Receba uma string do usuário e crie uma nova string com os valores ascii de cada posição entre parêntesis, utilizando um dicionário cuja chaves são os caracteres de A-z e os valores são os valores correspondentes em ascii.
10. Crie um programa que só encerra quando o usuário digita uma única palavra.