TIPOS DE OBJETOS

Python é uma linguagem de programação orientada a objetos (POO). Um objeto possui métodos, que são atividades que o objeto pode executar. Vamos usar como exemplo o objeto cafeteira. A cafeteira, o objeto cafeteira, poderia ter os seguintes métodos: Fazer café, Ferver água, Verificar filtro, etc..

Os objetos em Python também possuem métodos. Vamos ver agora alguns dos objetos em Python e alguns de seus métodos. É importante observar que uma váriavel não tem um tipo, ela é apenas um nome. Somente o objeto ao qual ela se refere tem um tipo, ou seja, quando damos um comando *type* estamos vendo na verdade o tipo de objeto que a variavel armazena.

***Objeto int:** Esse tipo de objeto são os numeros inteiros, positivos e negativos.

Tipo int

```
1 a=3
```

```
1 type(a)
```

```
int
```

***Objeto float:** Esse tipo de objeto são os numeros com casas decimais, positivos e negativos. Vale lembrar que mesmo se a casa decimal for zero, ele ainda é tipo float

Tipo float

```
1 b=3.5
2 type(b)
```

```
float
```

```
1 c=3.0
2 type(c)
```

```
float
```

***Objeto Boolean:** Esse tipo de objeto armazena apenas dois valores. TRUE ou FALSE, seu resultado vem de uma comparação. E podemos usar os seguintes operadores para fazer essa comparação:

< Menor que
> Maior que
<= Menor igual que
>= Maior igual que
== Igual
!= Não igual

Tipo Boolean

```
1 a=3>2
2 type(a)
```

```
bool
```

```
1 a
```

```
True
```

```
1 b=5<2
2 b
```

```
False
```

* **Tipo String:** além dos tipos numericos e Boolenaos, o Python admite outros tipos de objetos, mais complexos. O tipo string é usado para manipular e representar textos, que são uma sequencia de caracteres, incluindo espaços, pontuação e diversos símbolos. Uma string é delimitada entre aspas simples.

String

```
1 string='hello world'
2 type(string)
```

```
str
```

Podemos usar os operadores de comparação:==,!=,< e > para comparar as strings. Veja alguns exemplos. Os operadores < e > usam a ordem do dicionário, esse conceito ainda não foi abordado.

String

```
1 string='hello world'
2 type(string)
```

```
str
```

```
1 string == 'hello world'
```

```
True
```

O operadores +, aplicado em strings irá **concatenar** duas strings formando uma nova string

```
1 s='Bem-vindo'
2 t='programador'
3 s+' '+t
```

```
'Bem-vindo programador'
```

Não é possível fazer a multiplicação entre duas strings. Porem é possível fazer a multiplicação de uma string por um inteiro, o resultado sera uma cópia da multiplicação da string pelo numero inteiro

```
1 3*'A'
```

```
'AAA'
```

Veja um quadro resumo desses e de outros usos com as strings

Uso	Explicação
X in s	Verdadeiro se x for substring da string s e falso caso contrário
X not in s	Falso se a string x for uma substring da string s, verdadeiro caso contrário
s+t	Concatenação da string s com a string t
s*n	Concatenação de n cópias de s
s[i]	Caractere da string s no indice i
len(s)	Comprimento da string s

Exemplo do operador indexação:

```
1 s='demontração de indexação'
2 s[0]
```

```
'd'
```

```
1 s[3]
```

```
'o'
```

Para ver os metodos do objeto string escreve o nome da string, coloque um ponto final e pressione tab. Aparecera uma lista com os metodos. Uma função muito util é o help. Veja um exemplo

```
1 help(s.capitalize)
```

```
Help on built-in function capitalize:

capitalize(...) method of builtins.str instance
    S.capitalize() -> str

    Return a capitalized version of S, i.e. make the first character
    have upper case and the rest lower case.
```

***Tipo Lista**: Lista são muito uteis para organizarmos os dados, por exemplo, lista de compras,lista de cursos, lista de contatos,etc... Em Python lista é uma sequencia de objetos separados por uma virgula, que podem ser de tipos diferentes!!!!

Lista

```
1 animais=['peixe','gato','cão']
```

```
1 coisas=['um',2,[3.4]]
```

Tambem é possível trabalhar com o operador indexação em lista, sendo que dessa vez o retorno sera o elemento da posição passada como indice. Vale lembrar que em Python os indices começam no zero.

```
1 animais[0]
```

```
'peixe'
```

```
1 coisas[2]
```

```
[3, 4]
```

Alguns métodos usados em listas: min retorna o menor valor dentro de uma lista, max retorna o maior valor dentro da lista, caso os itens sejam string, ele ira retornar a menor e maior string, respectivamente. O comando sum soma os itens de uma lista, se a lista for de string dara um erro.

```
1 lst=[23.99,19.99,34.5,120.99]
```

```
1 min(lst)
```

```
19.99
```

```
1 max(lst)
```

```
120.99
```

```
1 sum(lst)
```

```
199.46999999999997
```

```
1 sum(animais)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-1955084afb76> in <module>()
----> 1 sum(animais)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Listas são mutáveis, strings não. Isso quer dizer que é possivel atribuir um valor a uma posição especifica da lista, porem em uma string isso não é possível. Veja o teste abaixo

```
animais[1]='gato siamês'
animais
```

```
['peixe', 'gato siamês', 'cão']
```

```
s='bato siamês'
s[0]='g'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-40-e7c00e3ca04e> in <module>()
      1 s='bato siamês'
----> 2 s[0]='g'

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

As listas também possuem alguns métodos e pode ser útil usar o comando help, para saber como funciona os métodos, conforme vimos em strings.

```
help(animais.append)
```

```
Help on built-in function append:

append(...) method of builtins.list instance
    L.append(object) -> None -- append object to end
```

Todos os exemplos e textos foram retirados do livro:Introdução à programação usando Python: um foco no desenvolvimento de aplicações. Autor Ljubomir Perkovic, editora LTC.