**07 - Tipos de dados**

Já conhecemos as variáveis, e agora conversaremos sobre os **tipos de dados**. São eles que determinam como números e caracteres são armazenados e manipulados dentro de um programa Python. A linguagem Python possui os tipos básicos, que são so numéricos int e float (inteiro e ponto flutuante, respectivamente), além dos bool (*booleanos*, true para verdeiro e false para falso), as strings (sequências de caracteres que representam textos, como números, letras e caracteres especiais) e None, que representa a ausência de valor.

Dependendo do tipo de dados que estamos trabalhando, o programa se comportará de formas diferentes e específicas. Por exemplo, se estamos trabalhando com dados numéricos e utilizamos o operador de soma (+), simplesmente somaremos esses valores. Porém, trabalhando com strings, o operador de soma fará a concatenação dessas strings - ou seja, um comportamento completamente diferente. No futuro conheceremos também outros tipos mais específicos dentro do Python.

Já vinhamos trabalhando com dados numéricos, como quando especificamos a variável ano\_atual.

ano\_atual = 2019COPIAR CÓDIGO

Podemos verificar o tipo dessa variável com a função type(), uma das funções embutidas do Python (algo que chamamos de *built-in function*, assunto que será abordado na continuação deste treinamento).

type(ano\_atual)COPIAR CÓDIGO

int

Como retorno, obtivemos o tipo "int", que representa um numérico inteiro. Outra variável que utilizamos anteriormente foi a km\_total de um veículo.

km\_total = 44410.0COPIAR CÓDIGO

Quando utilizamos no valor um . seguido de algum número, estamos definindo que tal valor possui também uma parte decimal. Sendo assim, o Python armazenará essa informação como outro tipo de dado.

type(km\_total)COPIAR CÓDIGO

float

Para entendermos os valores booleanos, definiremos uma variável zero\_km que receberá o valor True, com a letra inicial maiúscula.

zero\_km = True

type(zero\_km)COPIAR CÓDIGO

bool

Se atribuíssemos False, o retorno da função type() seria o mesmo, já que ambos os valores são do tipo bool. Agora começaremos a trabalhar também com strings. Observe o que acontece quando atribuímos à variável nome um texto rodeado por aspas simples:

nome = 'Jetta Variant'

nomeCOPIAR CÓDIGO

'Jetta Variant'

Ao executarmos, recebemos como retorno o texto atribuído. Isso também funciona com aspas duplas.

nome = "Jetta Variant"

nomeCOPIAR CÓDIGO

"Jetta Variant"

Ou seja, tanto aspas simples quanto aspas duplas são utilizadas para definir valores do tipo string. Mas e se tivermos aspas dentro do nosso texto, como uma citação? Para isso usamos aspas de tipos diferentes. Por exemplo, se precisamos das aspas duplas dentro do texto, definiremos a string usando aspas simples.

nome = 'Jetta "Variant"'

nomeCOPIAR CÓDIGO

'Jetta "Variant"'

O contrário também se aplica:

nome = "Jetta 'Variant'"

nomeCOPIAR CÓDIGO

"Jetta 'Variant'"

Também é possível armazenar strings com mais de uma linha, algo que é feito utilizando três aspas simples ou duplas.

carro = '''

Nome

Idade

Nota

'''COPIAR CÓDIGO

'\n Nome\n Idade\n Nota\n'

Note que o \n nesse retorno significa que uma linha está sendo pulada. Todas essas variáveis que acabamos de criar são do tipo "str", ou seja, strings. O último tipo que queremos apresentar nesse vídeo é o None, a ausência de um valor. Para isso, criaremos uma variável quilometragem que receberá o valor None, com a qual estamos representando um veículo que nunca rodou.

quilometragem = None

quilometragemCOPIAR CÓDIGO

Essa execução não trará nenhum retorno, afinal o valor da variável quilometragem é vazio. Entretanto, ainda é possível verificar o seu tipo:

type(quilometragem)COPIAR CÓDIGO

NoneType

No próximo vídeo entenderemos como e quando é possível converter tipos de valores. Por exemplo, se usarmos o símbolo + para concatenar uma string e um valor numérico, podemos obter um erro. Sendo assim, tentaremos fazer transformações desses tipos de dados para trabalharmos com eles.