TIPOS DE DADOS EM PYTHON (http://excript.com/python/tipos-de-dados-python.html)

CURSO DE PYTHON (http://excript.com/python/tipos-de-dados-python.html)

Publicado em 18-07-2016 por Cláudio Rogério Carvalho Filho (http://excript.com/author/claudio-rogerio-carvalho-filho.html)

ÍNDICE ARTIGO VIDEOAULA

Nessa aula estudaremos os tipos de dados definidos pela linguagem Python, como também, as diferenças entre diferentes tipos. Por fim, veremos a diferença entre um o tipo do dado e o dado propriamente dito.

TIPOS

O **tipo** é uma forma de classificar as informação. As linguagens de programação normalmente trazem implementado o que é chamado de **tipos primitivos**, isto é, o tipo de dado mais genérico possível.

Toda informação que manipularemos será, por definição, de um tipo.

Na informática tudo é manipulado como sendo bits. Quando manipulamos letras estamos trabalhando também com bits, porém, numa camada acima. Então, vamos definir agora, que num primeiro momento, toda informação é de fato um

caractere, seja ele uma letra, um número, um simbolo ou então, um caractere especial.

Assim, definimos num primeiro momento que tudo são caracteres.

String

Na programação String representa um conjunto de caracteres disposto numa determinada ordem. A partir de agora, todas as vezes em que falarmos o termo String, estaremos nos referindo a um conjunto de caracteres.

Inteiro

Um segundo tipo de informação são os dados compostos por caracteres numéricos (algarismo). Os números são divididos em 2 partes:

- inteiros - chamados de **integer** ou **int**
- ponto flutuante - chamado de **float** ou **double**

Exemplos de informação

A título de exemplo, vamos citar alguns tipos de informação e em seguida, vamos definir o tipo que precisaríamos utilizar para armazenar um valor correspondente a informação.

- Nome - dado tipo String
- Idade - dado tipo Integer
- Conta bancária - dado composto por números, pontos e traços

As informações são classificadas devido ao fato de seguirem regras e estruturas iguais. Ou seja, um número de telefone possui uma regra para todo o território nacional, logo, é possível classificar esse tipo de dado, até porque, há uma regra que o define.

Atualmente, os números de telefones no estado de São Paulo contém 9 digitos, enquanto os demais estados e capitais utilizam somente 8 digitos. Essa é uma situação que já ocorreu há anos e voltou a acontecer. Por isso, devemos estar certos de que irá acontecer novamente no futuro. Assim, é prudente que nossos softwares sejam capazes de lidar com essa variação, seja para o momento atual, seja para estarem preparados para futuras modificações.

Um outros exemplo que possibilita a classificação é o número das contas bancárias que, geralmente, é composto por números e um digito verificador. O nome das pessoas, normalmente, é constituído somente por letras. A idade das pessoas é representado por números inteiros. Datas são a junção de números com alguns caracteres especiais e assim por diante.

Digito verificar é aquele número contido no final, geralmente após um traço ou então, após um ponto. A função deste digito é proporcionar um meio para chegar se os digitos anteriores estão corretos, até porque, o digito verificador é obtido através de uma fórmula matemática que utiliza todos os números e no final, gera um digito de confirmação. Por exemplo, o CPF é composto de 9 digitos, mais 2 digitos verificador, por exemplo: ``CPF: 123.456.789-09``.

A última parte do CPF acima, no caso, o número `09` é o digito verificador e o mesmo foi obtido através de uma fórmula que utilizou todos os 9 digitos e resultou no digito verificador `09`. Exemplos de informações que possuem digito verificador são: o número da conta corrente de alguns bancos, o CPF dentre outras informações.

Os tipos de dados são uma forma de classificação que facilita o processamento e a manipulação das informações.

TIPOS PRIMITIVOS

Tipo Primitivo são os tipos de dados mais simples, isto é, a informação em sua forma mais primitiva. Bons exemplos de valores primitivos são os caractere, os número, o valor **True** e o **False** e etc. A documentação do Python não trata os tipos de dados elementares (primitivos) com a nomenclatura normalmente encontrada na documentação da maioria das linguagens: **Tipo Primitivo**. Na documentação oficial, os tipos primitivos são chamados de tipos built-ins (https://docs.python.org/3.5/library/stdtypes.html) ou então, **tipos construídos**. Essa nomenclatura é utilizada para indicar que estamos utilizando informações que foram definidas, por padrão, através de classes dentro da Máquina Virtual do Python.

Nesse momento, chamaremos de **Tipos Primitivos** as informações em sua forma mais simples, porém, é importante saber que para o Python, não há tipo primitivos, mas sim, estruturas de dados que estão definidas, na maior parte das vezes, dentro da própria Máquina Virtual do Python.

É normal que as linguagens de programação tenham um conjunto de tipos chamados de: **tipos primitivos**. Devemos pensar nessa classificação como sendo os tipos primários de informações, como por exemplo, o tipo numérico. Como sabemos, todo número é constituído por algarismos. Dessa forma, o tipo numérico pode ser qualquer valor que seja composto por 1 ou mais caracteres numéricos.

Dessa forma, isto é, tendo a certeza de que uma variável declarada como sendo do tipo numérico inteiro sempre terá um valor numérico válido, somos capazes de desenvolver funções especificas que manipulam esse tipo de dado de maneira muito mais eficiente e sem a necessidade de verificação se o tipo da informação contida em determinada variável é válido.

Da mesma forma, temos o tipo de dado que representa conjuntos de caracteres, que na programação, é comumente chamado de *String* e o Python o chama de *str*. As *String* são capazes de armazenar conjuntos de caracteres que estão dispostos

numa determinada ordem. Todas as vezes que estivermos manipulando dados que contenha caracteres - o tipo mais primitivo, isto é, a maneira mais abstrata para representarmos caracteres - estaremos utilizando uma variável definida como sendo do tipo *str*.

O fato de o Python não trabalhar com tipo primitivos diretamente, deve-se ao fato de que em Python, **tudo são objetos**. Dessa forma, o que chamaríamos de primitivo é, em Python, representado como uma e toda informação será, um objeto propriamente dito. A seguir, temos a lista dos principais tipos built-ins da linguagem Python:

- int - para números inteiros
- str - para conjunto de caracteres
- bool - armazena True ou False
- list - para agrupar um conjunto de elementos
- tupla - semelhante ao tipo **list**, porém, imutável
- dic - para agrupar elementos que serão recuperados por uma **chave**

O Python fornece um conjunto de tipos básicos bastante amplo e que normalmente, atendem a maioria das necessidades. Cada tipo citado, possui um conjunto de funções e métodos que permitem manipularmos as informações, contidas na variável, de maneira bastante eficiente.

Sempre que formos criar um novo **tipo de dados**, acabaremos utilizando os tipos básicos da linguagem, como por exemplo, o tipo **`int`**, ou então, o tipo **`str`** e assim por diante.

DIFERENÇA ENTRE TIPO E VALOR

O **valor** é qualquer informação, seja um número, texto, música, vídeo e etc. O **tipo** por sua vez, é a estrutura da informação e a forma de classificarmos os dados.

Todo valor numérico deve ser capaz de ser somado a outro valor, ou então, subtraido. Da mesma forma que todo texto, deve ser capaz de ser concatenado a outro, isto é, ser juntado a outro conjunto de caracteres.

O **tipo** da informação deve ser pensado como uma forma de classificarmos as diferentes informações e assim, termos a disposição um conjunto de funções para tratarmos e modificarmos os valores.

É importante saber que somos capazes de criar novos tipos de dados a qualquer momento, e a programação orientada a objetos é, em sua definição mais primitiva, uma maneira de criarmos novos tipos abstratos e definirmos, na estrutura da classe, funções, métodos, verificações que busquem tratar valores que tenham uma mesma estrutura.

CONVERSÃO DE DADOS OU COERÇÃO DE TIPOS

Se as informações possuem tipos, logo, temos de ser capazes em converter um tipo de informação num outro tipo de dado. Essa ação de conversão é comumente chamada de **Coerção de Tipos**.

Linguagem tipada é aquela que permite a classificação das informações pelo uso de tipos de dados, por exemplo, o Python trata um conjunto de caracteres como sendo do tipo *String*, logo, o Python é uma linguagem tipada, no caso, uma linguagem dinamicamente tipada.

Se existem diferentes tipos de informação, temos de ser capazes de converter, por exemplo, um número para o tipo texto. Ou seja, a conversão de valores é essencial para que possamos trabalhar com informações tipadas, até porque, há diversas situações onde desejaremos manipular um número como sendo um texto.

Essa conversão é comumente chamada de **Conversão de Dados** ou então, **Coerção de Tipos**. É importante observar, que uma informação do tipo texto, pode ser constituida de letras e números, ou seja, o grupo de caracteres

alfanuméricos. Então, um número pode ser representado como um texto, mas o contrário, nem sempre será possível.

Pra convertermos, por exemplo, um texto para o tipo numérico, devemos especificar o tipo a ser convertido e passarmos o valor através de parêntesis, como podemos ver a seguir:

```
tipo_a_ser_convertido( informação )
```

A seguir, declararemos uma variável de nome *texto* e atribuimos um valor numérico a mesma. Em seguida, declaramos outra variável, de nome *num* e atribuimos a esta o resultado da Coerção de Tipos. Isto é, dissemos que o valor da variável *texto* deve ser convertido num tipo numérico *num = int(texto)*.

```
#coding: utf-8

texto = "10"
num = int(texto)

print( texto + str(10) )#o sinal de + concatena duas informaç
ões
print( num + 10 )#o sinal de + soma dois números

>>> 1010
>>> 20
```

No exemplo acima, podemos observar que a utilização do operador + funciona de maneira diferente conforme o tipo de dado que esteja sendo utilizado. Quando o sinal de adição estiver entre dois valores numéricos, estes serão somados. Quando o sinal adição estiver entre dois valores do tipo *String*, estes serão concatenados (juntados) e quando o sinal de adição estiver entre um valor do tipo numérico e do tipo *String*, uma exceção será levantada dizendo que não é possível utilizar o operador de adição entre tipos distintos.

PARA REFLETIR

Em Python, tudo é tratado como sendo um objeto, inclusive, o próprio código por nós escrito!

Linguagens como Java e C# também são fortemente Orientada a Objetos, porém, ambas possuem, tipos de dados primitivos, que como acabamos de estudar, não existe em Python.

Essas característica do Python deve-se ao fato de que o mesmo **realmente** trata tudo que possamos imaginar como sendo um objeto. Se tivermos 2 variáveis que contenham o número 10, para o Python, ambas variáveis estarão apontando para um objeto na memória, cujo valor é 10. Dizer que uma variável contém um valor é, por definição da linguagem Python, um erro, até porque, os valores não estão armazenados numa posição determinada pela variável. Em Python, os dados são associados a variáveis, logo, variáveis que contenham um mesmo valor, na verdade, internamente o que haverá, é uma informação associadas a várias variáveis. Há um excelente artigo falando sobre a forma que o Python trata os tipos e as informações publicado pelo Luciano Ramalho (http://pythonclub.com.br/tuplas-mutantes-em-python.html).

Por fim, temos 4 tipos para classificação para os tipos de informações.

- `Tipos simples` - constituidos por simples blocos, como `int()` e `float()`
- `Tipos de contêiner` - objetos capazes de conter outros objetos
- `Tipos de código` - objetos encapsuladores de elementos dos nosso programas
- `Tipos internos` - tipos que serão utilizados durante a execução do nosso programa

No link a seguir há mais informações sobre os tipos de dados do Python (https://www.ibm.com/developerworks/br/library/os-python1/).

LINKS EXTERNOS

- Tuplas Mutantes (http://pythonclub.com.br/tuplas-mutantes-em-python.html)

Tags curso (http://excript.com/tag/curso.html), python (http://excript.com/tag/python.html), tipos (http://excript.com/tag/tipos.html), tipo de dados (http://excript.com/tag/tipo-de-dados.html), int (http://excript.com/tag/int.html), float (http://excript.com/tag/float.html), string (http://excript.com/tag/string.html), date (http://excript.com/tag/date.html), bool (http://excript.com/tag/bool.html), str (http://excript.com/tag/str.html), blog (http://excript.com/tag/blog.html)

Comentários

Nobsin gamer • há 2 anos • edited

valeu cara, tenho 13 anos e estou aprendendo a programar, já to numa turma de robótica avançada e já sei o que eu amo na vida! Programar! Eu to aprendendo a programar pro arduino e python e pretendo ainda aprender a programar em muitas outras linguagens de programação! Valeu por me ajudar a fortalecer ainda mais meus laços com a programação!

1 • Responder • Partilhar ›

Cláudio Rogério Carvalho Filho Moderador ➚ Nobsin gamer • há 2 anos

Oi Nobsin. Fico muito feliz em saber que estou, positivamente contribuindo em sua formação!! Parabéns por já conhecer, aos 13 anos, como eu (mesma idade) o que desejas "brincar" quando na verdade o outros chamam trabalhar :D

1 • Responder • Partilhar ›

Sem Defesas • há 7 meses • edited

Olá! Tenho **11 anos**, e eu já programo coisas básicas em Python, e futuramente desejo aprender Java e C#.
Vou virar programador de coisas um pouco mais avançadas e trabalhar com isso, já que a code.org estima que em 10 anos, serão necessários 1 milhão e 400 mil programadores, e se o mundo continuar do jeito que está, só vai ter 400 mil programadores, ou seja, vai faltar 1 milhão de programadores, por esse e vários outros motivos irei virar programador.

• Responder • Partilhar ›

william jeferson • há um ano

como eu represento uma variável do tipo binaria no python.
por exemplo a string eu represento usando o %s, então como eu represento valores

binarios

• Responder • Partilhar ›

ConceituArt • há 2 anos

Até mesmo a escolha das palavras para compor o texto é maravilhosa. Muito obrigado.

• Responder • Partilhar ›

Imagem Este comentário foi apagado.

Cláudio Rogério Carvalho Filho Moderador ➚ Guest • há 2 anos

Isso mesmo! Isso acontece porque com o par de colchetes você tem o tipo de dado lista que sendo um tipo mutável, sempre terá uma ID diferente. Porém, se você utilizar parêntesis, terás um tipo de dado Tupla que não é mutável, logo, 2 tuplas contendo o mesmo valor terá o mesmo ID.

• Responder • Partilhar ›

Imagem Este comentário foi apagado.

Cláudio Rogério Carvalho Filho Moderador ➚ Guest • há 2 anos

aahhh que bom :D

• Responder • Partilhar ›

Balbino de Jesus • há 2 anos

Muito completo e bem explicado!
Muito obrigado, Cláudio!

• Responder • Partilhar ›

Cláudio Rogério Carvalho Filho Moderador ➚ Balbino de Jesus • há 2 anos

Grande Balbino!!

• Responder • Partilhar ›

TAMBÉM NO **EXCRIPT**

Pseudo-variável $this em PHP - eXcript

2 COMENTÁRIOS • há 3 anos

Imagem **Lucas Vieira** — Cara, que linda explicação, parabéns me ajudou mt :)

Operadores Relacionais do C# - eXcript

1 COMENTÁRIO • há 3 anos

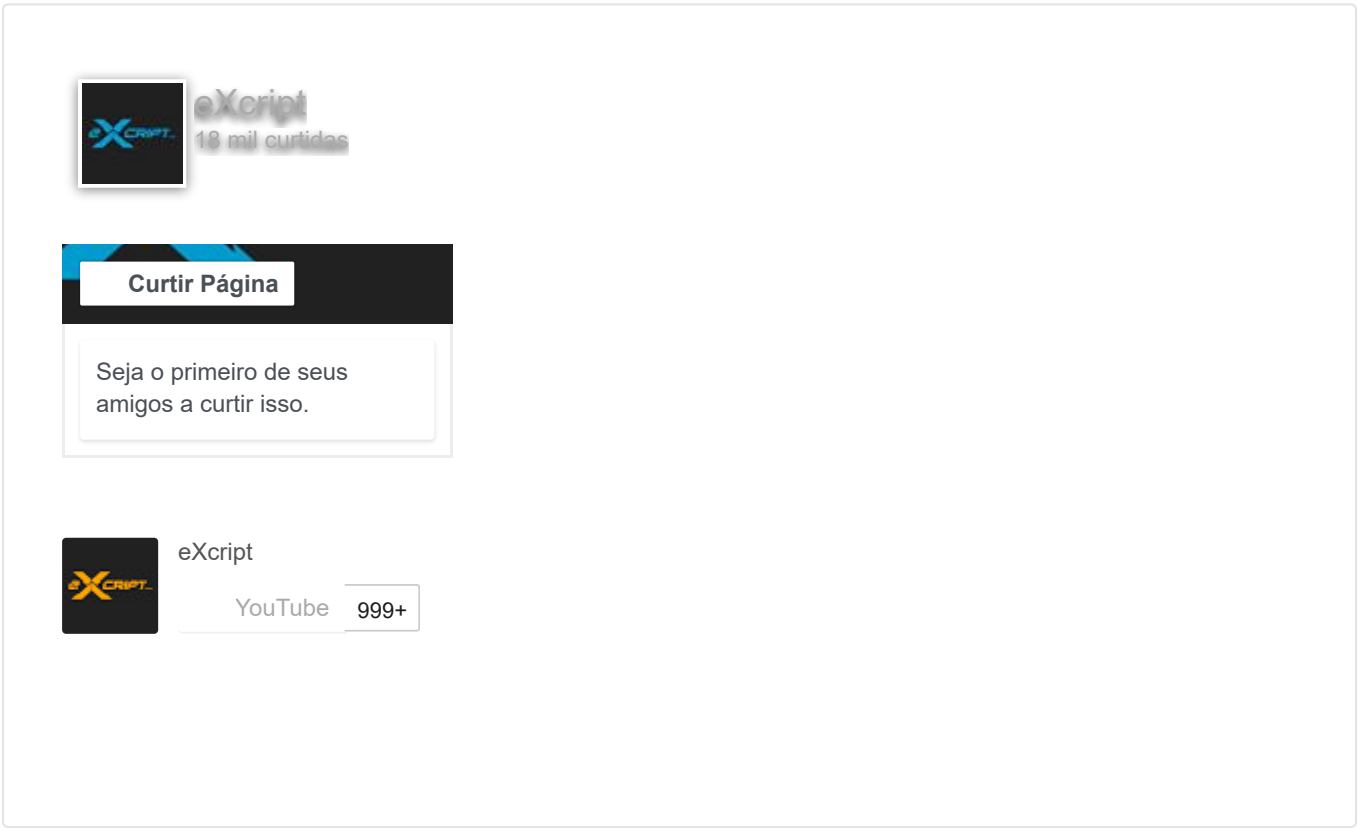
Imagem **Ronald Pinho** — Valeu!

Monitorando Solicitações WiFi - eXcript

2 COMENTÁRIOS • há 2 anos

O MercadoLivre me Penalizou - eXcript

2 COMENTÁRIOS • há 2 anos



eXcript

A eXcript é um projeto de produção audiovisual de conteúdo técnico com foco no autodidatismo.

CURSOS

- PYTHON (http://excript.com/curso-de-python.html)
- JAVA (http://excript.com/curso-de-java.html)
- PHP (http://excript.com/curso-php.html)
- C++ (http://excript.com/curso-cpp.html)
- C (http://excript.com/curso-c.html)
- ANDROID (http://excript.com/curso-de-android.html)

SOCIAL

- GOOGLE+ (https://plus.google.com/+excriptvideo)
- FACEBOOK (https://www.facebook.com/excript)
- TWITTER (https://twitter.com/eXcriptBrasil)