02

Introdução

Introdução

Bem-vindo ao treinamento de Python do Alura. Neste treinamento aprenderemos a linguagem Python no contexto de uma rede social. Sim, por mais que estejamos aprendendo o Python básico, nada nos impede de brincarmos com o tema, não? Tudo será feito através do terminal e do console do Python. A ideia é que no final o aluno tenha o conhecimento suficiente de Python para compreender o Django, um framework MVC totalmente feito nesta linguagem, assunto de outro treinamento que está por vir.

Aprenderemos a **versão 2.7** do Python. A versão 3.0 foi muito polêmica, quebrando a compatibilidade com versões anteriores, porém muitas novidades foram portadas para a versão 2.7.

Instalação

OSX (Mac) e o Linux já vêm com Python instalado por padrão, mas se você usa Windows precisará instalá-lo. O Instalador de Python para Windows pode ser baixado aqui (https://www.python.org/download). Apesar da instalação ser super fácil, ainda não será possível chamar o interpretador do Python pelo terminal. Será necessário adicionar o caminho `c:\Python27` no path do sistema operacional.

O console Python

Em alguns sistemas operacionais, a instalação do Python disponibiliza um console visual, porém a maneira tradicional de trabalharmos com esta linguagem é através da linha de comando. Como essa é a mais difundida e suportada por todos os sistemas operacionais, ficaremos com ela.

Tudo bem, por onde escreveremos nosso código em Python? Nossos primeiros passos com o Python será através de seu interpretador, chamado através do terminal pelo comando: `python` .

Por exemplo, no OSX, o comando `python` exibirá no console:

```
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Independente do sistema operacional que você esteja utilizando, o que nos interessa é o `>>>` . É nele que digitamos e executamos nosso código Python teclando ENTER no final.

Provavelmente você deve estar ansioso para interagir com o terminal, mas primeiro precisamos aprender algum recurso do Python. Por exemplo, uma rede social possui nomes, quantidade de convites e por aí vai. É por isso que toda linguagem que se preza trabalha com variáveis. Que tal começarmos por elas?

Declaração de variáveis

Não há declaração explícita de variáveis em Python, ou seja, ela só passa a existir quando atribuímos um valor:

```
>>> convites = 12
```

Convenção

A declaração do nome das variáveis seguem a convenção *snake case* (http://en.wikipedia.org/wiki/Snake_case) começando com minúscula e utilizando *underscore* para separar cada palavra:

```
>>> convites_ano_passado = 20
```

Se quisermos poupar tempo, podemos declarar variáveis e atribuir seus valores ao mesmo tempo:

```
>>> convites, convites_ano_passado = 12, 20
```

E para imprimir o valor da variável no console? Basta digitar seu nome seguido de ENTER:

```
>>> convites
12
>>> convites_ano_passado
20
```

Imagine se tivéssemos declarado mais variáveis. Não seria um pouco tedioso teclar ENTER para cada uma delas? Pensando nisso, o Python permite imprimir mais de uma variável por vez:

```
>>> convites, convites_ano_passado
(12, 20)
```

Tipos de variáveis

Aprendemos a criar variáveis seguindo a convenção do Python, mas todas guardavam números. E se agora quisermos armazenar um nome? Algo válido, não? Se tivéssemos vários nomes, eles poderiam vir dentro de uma lista, que tal? As necessidades podem ser ainda maiores. É por isso que o Python disponibiliza **cinco tipos básicos de variáveis**. Por enquanto, focaremos apenas os tipos **Number**, **String** e **List**.

No início do capítulo declaramos variáveis que armazenavam números. Sendo assim, nada mais justo do que começarmos a entender um pouco mais sobre o tipo **Number**.

Number

Escolha um número de um até cem. Escolheu? Agora, pense num salário que você gostaria de ganhar, mas que não ultrapasse R$ 50.000,00. Pensou? Não somos capazes de adivinhar esses dois números que estão em sua cabeça, mas com certeza podemos afirmar que o primeiro é um número inteiro e o segundo um número decimal como 50 e 7500.50 respectivamente. São números de características distintas, não? É por isso que existem os tipos numéricos **integer** e **float** em Python.

Já aprendemos declarar um Number do tipo integer:

```
>>> convites = 50
```

E agora, para criarmos um número com duas casas decimais?

```
>>> salario = 7500.50
```

Será que a maneira de declararmos o valor das variáveis são as únicas características que o diferem? Com certeza não. Por exemplo, um número do tipo integer em Python tem valor máximo de 9223372036854775807. Já o tipo float possui 1.7976931348623157e+308. Números difíceis de pronunciar, ainda bem que delimitamos nosso chute! Porém, se você estiver escrevendo um código muito específico (quem sabe para a NASA) que precise de números inteiros e decimais ainda maiores precisamos usar os tipos **long** e **complex** respectivamente:

```
>>> convites = 10L
>>> salario = 7500.50j
```

Veja que no exemplo anterior utilizamos o sufixo **L** ou **J**, porém o primeiro é recomendado para não confundirmos a letra com o número um. Este caracter especial serve para indicar ao interpretador Python que o tipo de número que a variável guarda não é um integer, mas um **long**. Fizemos a mesma coisa com a variável `salario`, mas utilizamos o sufixo **j**.

String

Até agora declaramos variáveis do tipo Number, mas muitas vezes queremos guardar um conjunto contínuo de caracteres, em Python chamado de **String**. Justamente o que precisamos para guardar o nome dos perfis de nossa rede social! Strings em Python podem ser declaradas utilizando aspas simples ou duplas. Vamos declarar duas variáveis utilizando o padrão *snake case*, uma com aspas duplas e outra com aspas simples.

```
>>> perfil_aspas_duplas = "Alura"
>>> perfil_aspas_simples = 'Alura'
```

Qual você prefere? Usar aspas simples nos evita de pressionar a tecla SHIFT, poupando um pouco nossa energia, energia que precisaremos utilizar ao longo do treinamento!

Agora, vejamos a seguinte String:

```
>>> convite = 'Flavio Henrique de Souza Almeida'
```

Imagine que essa variável seja utilizada na exibição de um convite de nossa rede social. Seu conteúdo é um pouco extenso, não? Será que há alguma maneira de fatiarmos apenas o primeiro e segundo nome, isto é, a string 'Flávio Henrique'? Claro que sim, através do operador **slice**. A palavra slice significa fatia ou também pode ser usado como o verbo fatiar. Vejamos em nosso código:

```
>>> convite[0:15]
'Flavio Henrique'
```

Neste exemplo usamos colchetes diretamente no nome da variável. Ele recebe como primeiro parâmetro a posição inicial e a posição final separados por dois pontos. As posições vão de 0 até o tamanho da `String` menos um. A posição final não é inclusa, isto é, o caracter dessa posição não entra na seleção.

Podemos guardar o resultado em outra variável:

```
>>> primeiro_segundo_nome = convite[0:15]
>>> primeiro_segundo_nome
'Flavio Henrique'
```

O tipo `String` possui uma série de recursos que tornam a vida do programador mais fácil, vimos apenas uma delas nessa parte introdutória.

Concatenação

Muito bem, aprendemos a trabalhar com os tipos Number e String. Vamos recapitular:

```
>>> convite = 'Rayson Shall'
>>> idade = 20
```

Que tal imprimirmos a mensagem "Convite do Rayson Shall"?

```
>>> 'Convite do ' + convite
'Convite do Rayson Shall'
```

Funciona! Neste exemplo, concatenamos um string com outra. Agora, adicionaremos a informação da idade:

```
>>> 'Convite do ' + convite + ' ' + idade +  ' anos'
TypeError: cannot concatenate 'str' and 'int' objects
```

Ops! Parece que o Python não permite concatenar uma string com um número, ou seja, ele não realiza uma conversão implícita de Number para String. E agora? Podemos pedir ajuda à função **print**, que funciona assim: passamos um string com lacunas que precisam ser preenchidas. Para cada lacuna, usamos um curinga especial para cada tipo. Usamos `%s` para tratar o resultado final como uma String:

```
>>> print 'Convite do %s idade %s'
```

Será que funciona? Ainda não, precisamos indicar os valores que entrarão nas lacunas `%s` e `%s`. Fazemos isso adicionando o sinal `%` imediatamente após a String e em seguida colocando as duas variáveis entre parênteses e separadas por vírgula. A ordem é importante: o valor da primeira variável substituirá o valor do primeiro `%s` da String, o valor da segunda o segundo `%s` e por aí vai.

```
>>> print 'Convite do %s idade %s'  % (convite, idade)
Convite do Rayson Shall idade 20
```

Funcionou!