Python-Completo-UDEMY (/github/rtadewald/Python-Completo-UDEMY/tree/master)
/ Notebooks Traduzidos (/github/rtadewald/Python-Completo-UDEMY/tree/master/Notebooks Traduzidos)

Formatação de impressão

Nesta aula, abordaremos brevemente as várias maneiras de formatar suas declarações impressas. À medida que você codifica mais e mais, você provavelmente deseja ter declarações de impressão que possam incluir uma variável em uma declaração de string impressa.

O exemplo mais básico de uma declaração de impressão é:

In [2]:
```
print('This is a string')
```

```
This is a string
```

Strings

Você pode usar o %s para formatar strings em suas instruções de impressão.

In [5]:
```
s = 'STRING'
print('Place another string with a mod and s: %s' %(s))
```

```
Place another string with a mod and s: STRING
```

Números de ponto flutuante

Os números de ponto flutuante usam o formato %n1.n2f onde o n1 é o número mínimo total de dígitos que a cadeia deve conter (estes podem ser preenchidos com espaço em branco se o número inteiro não tiver esses muitos dígitos. O espaço reservado n2 significa quantos números para mostrar após o ponto decimal. Vamos ver alguns exemplos:

In [3]:
```
print('Floating point numbers: %1.2f' %(13.144))
```
```
Floating point numbers: 13.14
```

In [4]:
```
print('Floating point numbers: %1.0f' %(13.144))
```
```
Floating point numbers: 13
```

In [5]:
```
print('Floating point numbers: %1.5f' %(13.144))
```
```
Floating point numbers: 13.14400
```

In [6]:
```
print('Floating point numbers: %10.2f' %(13.144))
```
```
Floating point numbers:      13.14
```

In [7]:
```
print('Floating point numbers: %25.2f' %(13.144))
```
```
Floating point numbers:                     13.14
```

Métodos de formato de conversão.

Deve notar-se que dois métodos% s e %r realmente convertem qualquer objeto python em uma string usando dois métodos separados: str() e repr(). Aprenderemos mais sobre essas funções mais tarde no curso, mas você deve notar que você pode realmente passar quase qualquer objeto Python com esses dois métodos e funcionará:

In [8]:
```
print('Here is a number: %s. Here is a string: %s' %(123.1,'hi'))
```
```
Here is a number: 123.1. Here is a string: hi
```

In [9]:
```
print('Here is a number: %r. Here is a string: %r' %(123.1,'hi'))
```

```
Here is a number: 123.1. Here is a string: 'hi'
```

Formatação múltipla

Passe uma tupla para junto com o símbolo do módulo para colocar vários formatos nas suas declarações de impressão:

In [10]:
```
print('First: %s, Second: %1.2f, Third: %r' %('hi!',3.14,22))
```

```
First: hi!, Second: 3.14, Third: 22
```

Usando o método string.format ()

A melhor maneira de formatar objetos em suas strings para instruções de impressão é usar o método format(). A sintaxe é:

'String aqui {var1} e também {var2}'.format(var1 = 'something1', var2 = 'something2')
Vamos ver alguns exemplos:

In [27]:
```
print('This is a string with an {p}'.format(p='insert'))
```

```
This is a string with an insert
```

In [28]:
```
print('One: {p}, Two: {p}, Three: {p}'.format(p='Hi!'))
```

```
One: Hi!, Two: Hi!, Three: Hi!
```

In [29]:
```
print('Object 1: {a}, Object 2: {b}, Object 3: {c}'.format(a=1,b='tw
```

```
Object 1: 1, Object 2: two, Object 3: 12.3
```

Esse é o básico da formatação de seqüência de caracteres!

In [ ]: