02

Dados da conta

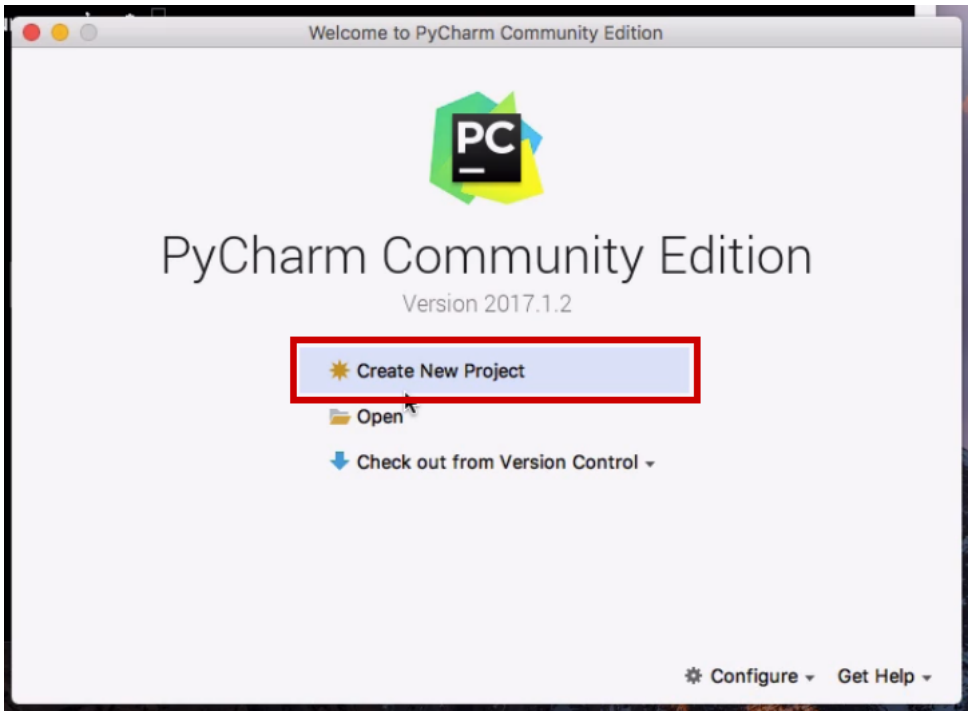
Transcrição

Neste curso, você vai precisar ter instalado o Python 3. Feito isto, iremos executá-lo na linha de comando:

```
~ nico$ python3 --version
```

Veremos que a versão foi `3.6.0`. Além disso, utilizaremos PyCharm, que usamos nos outros cursos de Python. Esta IDE nos ajudará a escrever o código Python, mas você pode fazer o mesmo com outro editor de texto de sua preferência (como Atom ou Sublime).

Após inicializarmos o PyCharm, selecionaremos a opção "Create New Project".



Iremos nomear o projeto como `oo`.

> Certifique-se no momento da criação do arquivo, ele será salvo na versão `3.6.0`.

Minha sugestão é que você não comece a escrever ainda o código. Primeiramente, siga a nossa explicação.

Até o momento, utilizamos pouco a Orientação a Objetos. Nós vimos superficialmente em cursos anteriores, mas não ressaltamos o uso de OO.

Trabalhamos de uma forma válida, mas sem abordar diretamente este paradigma. Nós escrevemos funções seguindo o paradigma **Procedural**. Nós criamos funções, que recebem parâmetros, realizam algo e dão um retorno. Utilizamos a

programação procedural para escrever o nosso programa — e funcionou bem.

Existem programas complexos feitos assim, o próprio Linux segue esse paradigma.

Iremos trabalhar com o exemplo de uma conta de banco. Imagine que você tem uma conta e ela possuirá algumas características: saldo, número, agência, titular e um limite. Estas características são comuns às contas bancárias, representá-las no Python Console será nosso objetivo. Começaremos adicionando `numero`:

```
>>> numero = 123
>>> titular = "Nico"
>>> saldo = 55.0
>>> limite = 1000.0
```

Trabalharemos com essas quatro características, mas poderíamos querer representar um sistema com diversas contas. Como representaríamos a segunda conta? Criar diversos números de contas não parece uma boa solução, na verdade, já vimos a forma correta de criar um conjunto de dados.

Nós queremos representar uma conta que possui um saldo, titular, cada um tendo um valor associado. Trabalharemos com um dicionário, que possui um número. No Console, abaixo de `limite`, adicionaremos `conta = {"numero"}`. O `numero` será seguido do valor `123`.

```
conta = {"numero": 123}
```

Depois, incluiremos as chaves `titular`, `saldo` e `limite`.

```
>>> numero = 123
>>> titular = "Nico"
>>> saldo = 55.0
>>> limite = 1000.0
>>> conta = {"numero": 123, "titular": "Nico", "saldo": 55.0, "limite": 1000.0}
```

Nós conseguimos agrupar todos os dados, que foram representados com a criação da variável `conta`. Agora, se solicitarmos `conta["numero"]`, o retorno será `123`. Ao definirmos qual é a chave, o console irá imprimir o valor referente.

```
>>> numero = 123
>>> titular = "Nico"
>>> saldo = 55.0
>>> limite = 1000.0
>>> conta = {"numero": 123, "titular": "Nico", "saldo": 55.0, "limite": 1000.0}
>>> conta["numero"]
123
>>> conta["saldo"]
55.0
```

Agora, se quisermos criar uma segunda conta, basta usar a linha com a variável `conta` como base.

```
>>> conta2 = {"numero": 321, "titular": "Marco", "saldo": 100.0, "limite": 1000.0}
```

Definimos os valores da segunda conta, já agrupados. Porém, precisaremos repetir seguidamente essa linha. A melhor solução será definirmos uma função que encapsule esse código. É o que faremos a seguir.

O próximo passo será gerar um arquivo chamado `teste` — você pode nomear a sua função da mesma forma. Em seguida, definiremos a função `cria_conta`, reutilizando a linha de `conta2`.

```
def cria_conta():
    conta2 = {"numero": 321, "titular": "Marco", "saldo": 100.0, "limite": 1000.0}
```

No entanto, queremos criar contas com outros valores, para isto, eles serão substituídos por **variáveis**.

```
def cria_conta(numero, titular, saldo, limite):
   conta = {"numero": numero, "titular": titular, "saldo": saldo, "limite": limite}
   return conta
```

No fim, retornaremos a `conta`. Se executarmos esta função, já teremos um resultado final. Vamos testar se o código esta funcionando. Após reiniciarmos o Console, em seguida, iremos importar do arquivo `teste`:

```
>>> from teste import cria_conta
```

Importamos a função `cria_conta` do módulo `teste`.

Queremos criar a `conta` usando a função `cria_conta`, ela receberá os seguintes valores referentes a cada uma das chaves:

```
>>> conta = cria_conta(123, "Nico", 55.0, 1000.0)
```

Nós passamos os quatro parâmetros e a conta será criada. Agora, se acessarmos o valor de `numero` da `conta`, teremos o seguinte retorno:

```
>>> from teste import cria_conta
>>> conta = cria_conta(123, "Nico", 55.0, 1000.0)
>>> conta["numero"]
123
```

Nós conseguimos agrupar os dados e encapsular a criação dentro de uma função. Porém, ficou trabalhoso saber o nome das chaves. É simples lembrar os nomes delas no código porque elas foram recém-criadas. Mas imagine se a função tivesse sido escrita por outra pessoa. Como saberíamos o nome da chave?

Iremos considerar que, além de agrupar os dados (ou seja, representar as características), nós também queremos ter funcionalidades com a conta. Tente se lembrar quais são as funcionalidades da sua conta: você deposita e saca dinheiro, verifica qual é o saldo e transfere valores entre diferentes contas.

Mais adiante, trabalharemos com essas parte de criar outras funcionalidades.