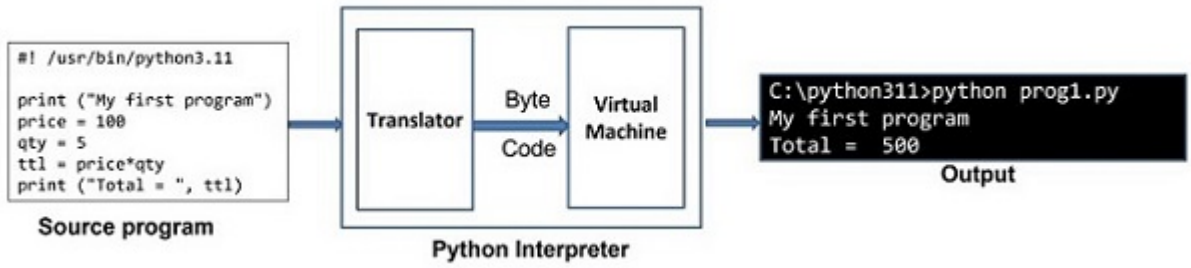
Intérprete Python e seus modos

Intérprete Python

Python é uma linguagem baseada em interpretador. Em um sistema Linux, o executável do Python é instalado no diretório **/usr/bin/** . Para Windows, o executável (python.exe) é encontrado na pasta de instalação (por exemplo **C:\python311**).

Este tutorial ensinará **como o interpretador Python funciona** no modo interativo e com script. O código Python é executado por um método de instrução por vez. O interpretador Python possui dois componentes. O tradutor verifica a sintaxe da instrução. Se for considerado correto, ele gera um código de byte intermediário. Existe uma máquina virtual Python que converte o código de bytes em binário nativo e o executa. O diagrama a seguir ilustra o mecanismo:



O interpretador Python possui um modo interativo e um modo de script.

Intérprete Python - Modo Interativo

Quando iniciado a partir de um terminal de linha de comando sem quaisquer opções adicionais, um prompt Python **>>>** aparece e o interpretador Python funciona com base no princípio de **REPL (Read, Evaluate, Print, Loop)** . Cada comando inserido na frente do prompt do Python é lido, traduzido e executado. Uma sessão interativa típica é a seguinte.

```
>>> price = 100
>>> qty = 5
>>> total = price*qty
>>> total
500
```

```
>>> print ("Total = ", total)
Total = 500
```

Para fechar a sessão interativa, insira o caractere de fim de linha (ctrl+D para Linux e ctrl+Z para Windows). Você também pode digitar **quit()** na frente do prompt do Python e pressionar Enter para retornar ao prompt do sistema operacional.

```
>>> quit()

$
```

O shell interativo disponível na distribuição padrão do Python não está equipado com recursos como edição de linha, pesquisa de histórico, preenchimento automático, etc. Você pode usar outro software interpretador interativo avançado, como **IPython** e **bpython** , para ter funcionalidades adicionais.

Intérprete Python - Modo de Script

Em vez de inserir e obter o resultado de uma instrução por vez como no ambiente interativo, é possível salvar um conjunto de instruções em um arquivo de texto, certificar-se de que possui extensão **.py** e usar o nome como linha de comando parâmetro para comando Python.

Salve as linhas a seguir como **prog.py** , usando qualquer editor de texto, como vim no Linux ou Notepad no Windows.

```
print ("My first program")
price = 100
qty = 5
total = price*qty
print ("Total = ", total)
```

Quando executamos o programa acima em uma máquina Windows, ele produzirá o seguinte resultado:

```
C:\Users\Acer>python prog.py
My first program
Total = 500
```

Observe que, embora o Python execute o script inteiro de uma só vez, internamente ele ainda é executado linha por linha.

No caso de qualquer linguagem baseada em compilador, como Java, o código-fonte não é convertido em código de bytes, a menos que todo o código esteja livre de erros. Em Python, por outro lado, as instruções são executadas até que a primeira ocorrência de erro seja encontrada.

Vamos introduzir um erro propositalmente no código acima.

```
print ("My first program")
price = 100
qty = 5
total = prive*qty #Error in this statement
print ("Total = ", total)
```

Observe a variável escrita incorretamente **prive** em vez de **price** . Tente executar o script novamente como antes -

```
C:\Users\Acer>python prog.py
My first program
Traceback (most recent call last):
  File "C:\Python311\prog.py", line 4, in <module>
   total = prive*qty
   ^^^^^
NameError: name 'prive' is not defined. Did you mean: 'price'?
```

Observe que as instruções anteriores à instrução errada são executadas e então a mensagem de erro aparece. Assim, agora está claro que o script Python é executado de maneira interpretada.

Intérprete Python - Usando Shebang #!

Além de executar o script Python como acima, o próprio script pode ser autoexecutável no Linux, como um script de shell. Você deve adicionar uma linha **shebang** no topo do script. O shebang indica qual executável é usado para interpretar as instruções Python no script. A primeira linha do script começa com **#!** E seguido pelo caminho para o executável Python.

Modifique o script prog.py da seguinte maneira -

```
#! /usr/bin/python3.11

print ("My first program")
price = 100
```

```
qty = 5
total = price*qty
print ("Total = ", total)
```

Para marcar o script como auto-executável, use o comando **chmod**

```
$ chmod +x prog.py
```

Agora você pode executar o script diretamente, sem usá-lo como argumento de linha de comando.

```
$ ./hello.py
```


Python interativo - IPython

IPython (significa **Interactive Python**) é um ambiente interativo aprimorado e poderoso para Python com muitas funcionalidades em comparação com o shell Python padrão. IPython foi originalmente desenvolvido por Fernando Perez em 2001.

IPython tem os seguintes recursos importantes -

- Capacidade de introspecção de objetos do IPython para verificar as propriedades de um objeto durante o tempo de execução.
- Seu destaque de sintaxe mostra-se útil na identificação de elementos da linguagem, como palavras-chave, variáveis , etc.
- O histórico das interações é armazenado internamente e pode ser reproduzido.
- O preenchimento de palavras-chave, variáveis e nomes de funções com tabulação é um dos recursos mais importantes.
- O sistema de comando Magic do IPython é útil para controlar o ambiente Python e executar tarefas do sistema operacional.
- É o kernel principal do notebook Jupyter e outras ferramentas front-end do Projeto Jupyter.

Instale o IPython com o utilitário instalador PIP.

```
pip3 install ipython
```

Inicie o IPython a partir da linha de comando

```
C:\Users\Acer>ipython
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934
64 bit (AMD64)] on win32
Type 'copyright', 'credits' or 'license' for more information
IPython 8.4.0 -- An enhanced Interactive Python. Type '?' for help.
In [1]:
```

Em vez do prompt regular >>> como no intérprete padrão, você notará dois prompts principais do IPython, conforme explicado abaixo -

- In[1] aparece antes de qualquer expressão de entrada.
- Out[1]aparece antes de Output aparecer.

```
In [1]: price = 100
In [2]: quantity = 5
In [3]: total = price*quantity
In [4]: total
Out[4]: 500
In [5]:
```

O preenchimento de guias é um dos aprimoramentos mais úteis fornecidos pelo IPython. O IPython exibe uma lista apropriada de métodos conforme você pressiona a tecla tab após o ponto na frente do objeto.

IPython fornece informações (introspecção) de qualquer objeto colocando **?** na frente dele. Inclui **docstring** , definições de função e detalhes do construtor da classe. Por exemplo, para explorar o objeto string var definido acima, no prompt de entrada, insira var?.

```
In [5]: var = "Hello World"
In [6]: var?
Type: str
String form: Hello World
Length: 11
Docstring:
str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str
```

```
Create a new string object from the given object. If encoding or
errors is specified, then the object must expose a data buffer
that will be decoded using the given encoding and error handler.
Otherwise, returns the result of object.__str__() (if defined)
or repr(object).
encoding defaults to sys.getdefaultencoding().
errors defaults to 'strict'.
```

As funções mágicas do IPython são extremamente poderosas. A magia de linha permite executar comandos DOS dentro do IPython. Vamos executar o comando **dir** no console IPython

```
In [8]: !dir *.exe
 Volume in drive F has no label.
 Volume Serial Number is E20D-C4B9

 Directory of F:\Python311

07-02-2023 16:55           103,192 python.exe
07-02-2023 16:55           101,656 pythonw.exe
               2 File(s)    204,848 bytes
               0 Dir(s)  105,260,306,432 bytes free
```

O notebook Jupyter é uma interface baseada na web para ambientes de programação de Python, Julia, R e muitos outros. Para Python, ele usa IPython como kernel principal.