Python - Sintaxe

Python - Sintaxe

A sintaxe Python define um conjunto de regras que são usadas para criar um programa Python. A sintaxe da linguagem de programação Python tem muitas semelhanças com as linguagens de programação Perl, C e Java. No entanto, existem algumas diferenças definidas entre os idiomas.

Primeiro programa Python

Vamos executar um programa Python para imprimir "Hello, World!" em dois modos diferentes de programação Python. (a) Programação em modo interativo (b) Programação em modo script.

Python - Programação em Modo Interativo

Podemos invocar um interpretador Python a partir da linha de comando digitando **python** no prompt de comando da seguinte forma -

```
$ python3
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Aqui **>>>** denota um prompt de comando Python onde você pode digitar seus comandos. Vamos digitar o seguinte texto no prompt do Python e pressionar Enter -

```
>>> print ("Hello, World!")
```

Se você estiver executando uma versão mais antiga do Python, como Python 2.4.x, precisará usar a instrução print sem parênteses, como em **print "Hello, World!"** . No entanto, na versão 3.x do Python, isso produz o seguinte resultado -

```
Hello, World!
```

Python - Programação em Modo Script

Podemos invocar o interpretador Python com um parâmetro de script que inicia a execução do script e continua até que o script seja concluído. Quando o script terminar, o intérprete não estará mais ativo.

Vamos escrever um programa Python simples em um script que é um arquivo de texto simples. Os arquivos Python têm extensão **.py** . Digite o seguinte código-fonte em um arquivo **test.py** -

```
print ("Hello, World!")
```

Presumimos que você tenha o caminho do interpretador Python definido na variável PATH . Agora, vamos tentar executar este programa da seguinte maneira -

```
$ python3 test.py
```

Isso produz o seguinte resultado -

```
Hello, World!
```

Vamos tentar outra maneira de executar um script Python. Aqui está o arquivo test.py modificado -

```
#!/usr/bin/python3

print ("Hello, World!")
```

Presumimos que você tenha um interpretador Python disponível no diretório /usr/bin. Agora, tente executar este programa da seguinte maneira -

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

Isso produz o seguinte resultado -

```
Hello, World!
```

Identificadores Python

Um identificador Python é um nome usado para identificar uma variável , função , classe , módulo ou outro objeto. Um identificador começa com uma letra de A a Z ou

de a a z ou um sublinhado (_) seguido por zero ou mais letras, sublinhados e dígitos (0 a 9).

Python não permite caracteres de pontuação como @, $ e% em identificadores.

> *Python é uma linguagem de programação que diferencia maiúsculas de minúsculas. Assim,* ***Manpower*** *e* ***manpower*** *são dois identificadores diferentes em Python.*

Aqui estão as convenções de nomenclatura para identificadores Python -

- Os nomes das classes Python começam com uma letra maiúscula. Todos os outros identificadores começam com uma letra minúscula.
- Iniciar um identificador com um único sublinhado indica que o identificador é um identificador **privado** .
- Começar um identificador com dois sublinhados iniciais indica um identificador fortemente **privado** .
- Se o identificador também terminar com dois sublinhados à direita, o identificador será um nome especial **definido pelo idioma .**

Palavras reservadas em Python

A lista a seguir mostra as palavras-chave Python. Estas são palavras reservadas e você não pode usá-las como constantes ou variáveis ou quaisquer outros nomes de identificadores. Todas as palavras-chave Python contêm apenas letras minúsculas.

e	como	afirmar
quebrar	aula	continuar
definição	del	Elif
outro	exceto	Falso
finalmente	para	de
global	se	importar
em	é	lambda

Nenhum	não local	não
ou	passar	elevação
retornar	Verdadeiro	tentar
enquanto	com	colheita

Linhas Python e recuo

A programação Python não fornece colchetes para indicar blocos de código para definições de classes e funções ou controle de fluxo. Os blocos de código são indicados por **recuo de linha** , que é rigidamente aplicado.

O número de espaços no recuo é variável, mas todas as instruções dentro do bloco devem ter o mesmo recuo. Por exemplo -

```
if True:
   print ("True")
else:
   print ("False")
```

No entanto, o bloco a seguir gera um erro -

```
if True:
   print ("Answer")
   print ("True")
else:
   print ("Answer")
   print ("False")
```

Assim, em Python todas as linhas contínuas recuadas com o mesmo número de espaços formariam um bloco. O exemplo a seguir possui vários blocos de instruções -

> *Não tente entender a lógica neste momento. Apenas certifique-se de*

entender os vários blocos, mesmo que eles não tenham colchetes.

```
import sys

try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```


Instruções multilinhas do Python

As instruções em Python normalmente terminam com uma nova linha. Python, entretanto, permite o uso do caractere de continuação de linha (\) para indicar que a linha deve continuar. Por exemplo -

```
total = item_one + \
        item_two + \
        item_three
```

As instruções contidas entre colchetes [], {} ou () não precisam usar o caractere de continuação de linha. Por exemplo, a seguinte instrução funciona bem em Python -

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```


Citações em Python

Python aceita aspas simples ('), duplas (") e triplas (''' ou """) para denotar literais de string, desde que o mesmo tipo de aspas inicie e termine a string.

As aspas triplas são usadas para abranger a string em várias linhas. Por exemplo, todos os itens a seguir são legais -

```
word = 'word'
print (word)

sentence = "This is a sentence."
print (sentence)

paragraph = """This is a paragraph. It is
 made up of multiple lines and sentences."""
print (paragraph)
```

Comentários em Python

Um comentário é uma explicação ou anotação legível pelo programador no código-fonte Python. Eles são adicionados com o propósito de tornar o código-fonte mais fácil de ser entendido pelos humanos e são ignorados pelo interpretador Python.

Assim como a maioria das linguagens modernas, Python suporta comentários de linha única (ou fim de linha) e de múltiplas linhas (bloco). Os comentários do Python são muito semelhantes aos comentários disponíveis nas linguagens de programação PHP, BASH e Perl.

Um sinal de hash (#) que não está dentro de uma string literal inicia um comentário. Todos os caracteres após # e até o final da linha física fazem parte do comentário e o interpretador Python os ignora.

```
# First comment
print ("Hello, World!") # Second comment
```

Isso produz o seguinte resultado -

```
Hello, World!
```

Você pode digitar um comentário na mesma linha após uma declaração ou expressão -

```
name = "Madisetti" # This is again comment
```

Você pode comentar várias linhas da seguinte maneira -

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

A sequência entre aspas triplas também é ignorada pelo interpretador Python e pode ser usada como comentários de múltiplas linhas:

```
'''
This is a multiline
comment.
'''
```

Usando linhas em branco em programas Python

Uma linha contendo apenas espaços em branco, possivelmente com um comentário, é conhecida como linha em branco e o Python a ignora totalmente.

Em uma sessão de intérprete interativo, você deve inserir uma linha física vazia para encerrar uma instrução multilinha.

Esperando pelo usuário

A linha seguinte do programa exibe o prompt, a instrução dizendo “Pressione a tecla Enter para sair”, e aguarda a ação do usuário -

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Aqui, "\n\n" é usado para criar duas novas linhas antes de exibir a linha real. Assim que o usuário pressiona a tecla, o programa termina. Este é um bom truque para manter uma janela do console aberta até que o usuário termine de usar o aplicativo.

Várias instruções em uma única linha

O ponto e vírgula (;) permite múltiplas instruções em uma única linha, visto que nenhuma das instruções inicia um novo bloco de código. Aqui está um exemplo de recorte usando ponto e vírgula -

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Vários grupos de instruções como suítes

Um grupo de instruções individuais, que formam um único bloco de código, é chamado de **suítes** em Python. Instruções compostas ou complexas, como if, while, def e class requerem uma linha de cabeçalho e um conjunto.

As linhas de cabeçalho iniciam a instrução (com a palavra-chave) e terminam com dois pontos (:) e são seguidas por uma ou mais linhas que compõem o conjunto. Por exemplo -

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Argumentos de linha de comando em Python

Muitos programas podem ser executados para fornecer algumas informações básicas sobre como devem ser executados. Python permite que você faça isso com -h -

```
$ python3 -h
usage: python3 [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit

[ etc. ]
```

Você também pode programar seu script de forma que ele aceite várias opções. Argumentos de linha de comando é um tópico avançado e deve ser estudado um pouco mais tarde, depois que você tiver passado pelo restante dos conceitos do Python.