Python - Syntax

Python - Syntax

The Python syntax defines a set of rules that are used to create a Python Program. The Python Programming Language Syntax has many similarities to Perl, C, and Java Programming Languages. However, there are some definite differences between the languages.

First Python Program

Let us execute a Python program to print "Hello, World!" in two different modes of Python Programming. (a) Interactive Mode Programming (b) Script Mode Programming.

Python - Interactive Mode Programming

We can invoke a Python interpreter from command line by typing **python** at the command prompt as following –

```
$ 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.
>>>
```

Here **>>>** denotes a Python Command Prompt where you can type your commands. Let's type the following text at the Python prompt and press the Enter –

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

If you are running older version of Python, like Python 2.4.x, then you would need to use print statement without parenthesis as in **print "Hello, World!"**. However in Python version 3.x, this produces the following result –

```
Hello, World!
```

Python - Script Mode Programming

We can invoke the Python interpreter with a script parameter which begins the execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script which is simple text file. Python files have extension **.py**. Type the following source code in a **test.py** file –

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

We assume that you have Python interpreter path set in PATH variable. Now, let's try to run this program as follows –

```
$ python3 test.py
```

This produces the following result –

```
Hello, World!
```

Let us try another way to execute a Python script. Here is the modified test.py file –

```
#!/usr/bin/python3

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

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows –

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

This produces the following result –

```
Hello, World!
```

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers.

> *Python is a case sensitive programming language. Thus,* ***Manpower*** *and* ***manpower*** *are two different identifiers in Python.*

Here are naming conventions for Python identifiers –

- Python Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is **private** identifier.
- Starting an identifier with two leading underscores indicates a strongly **private** identifier.
- If the identifier also ends with two trailing underscores, the identifier is a **language-defined** special name.

Python Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	as	assert
break	class	continue
def	del	elif
else	except	False
finally	for	from
global	if	import
in	is	lambda
None	nonlocal	not
or	pass	raise

return	True	try
while	with	yield

Python Lines and Indentation

Python programming provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by **line indentation**, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

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

However, the following block generates an error −

</> Open Compiler

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

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

> *Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.*

```
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
```

Python Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

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

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example following statement works well in Python −

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

Quotations in Python

Python accepts single ('), double (") and triple ('' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
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)
```

Comments in Python

A comment is a programmer-readable explanation or annotation in the Python source code. They are added with the purpose of making the source code easier for humans to understand, and are ignored by Python interpreter

Just like most modern languages, Python supports single-line (or end-of-line) and multi-line (block) comments. Python comments are very much similar to the comments available in PHP, BASH and Perl Programming languages.

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

</> Open Compiler

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

This produces the following result −

```
Hello, World!
```

You can type a comment on the same line after a statement or expression −

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

You can comment multiple lines as follows −

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

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

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

Using Blank Lines in Python Programs

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

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

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open

until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon −

</> Open Compiler

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

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called **suites** in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

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

Command Line Arguments in Python

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -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. ]
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.