Python Syntax:

Summary:

In this l, you’ll learn about the basic Python syntax so that you can get started with the Python language quickly.

Whitespace and indentation

Python uses whitespace and indentation to construct the code structure.

The following example shows of Python code with white space and indentation:

**add = 0  
for i in range(1,11):  
 # fetch even numbers  
 if i%2 == 0:  
 #print(i)  
 add += i  
print('Addition of even numbers is:’, add)**

The meaning of the code isn’t important to you now. Please pay attention to the code structure instead.

At the end of each line, you don’t see any semicolon to terminate the statement. And the code uses indentation to format the code.

By using indentation and whitespace to organize the code, Python code gains the following advantages:

* First, you’ll never miss the beginning or ending code of a block like in other programming languages such as Java or C#.
* Second, the coding style is essentially uniform. If you have to maintain another developer’s code, that code looks the same as yours.
* Third, the code is more readable and clear in comparison with other programming languages.

Comments

The comments are as important as the code because they describe why a piece of code was written.

When the Python interpreter executes the code, it ignores the comments.

In Python, a single-line comment begins with a hash (#) symbol followed by the comment. For example:

# This is a single comment in python

””” ””” this triple double quote is used for multiline comment

Continuation of statements

Python uses a newline character to separate statements. It places each statement on one line.

However, a long statement can span multiple lines by using the backslash (\) character.

The following example illustrates how to use the backslash (\) character to continue a statement in the second line:

if (a == True)and(b == False)and\

(c == True):

print (“Continuation of statements”)

Identifiers

Identifiers are names that identify variables, functions, modules, classes and other objects in Python.

The name of an identifier needs to begin with a letter or underscore (\_). The following characters can be alphanumeric or underscore.

Python identifiers are case-sensitive. For example, the counter and Counter are different identifiers.

In addition, you cannot use Python keywords for naming identifiers.

>>> name = 'python'  
>>> *# name is an identifier and 'python' is object*>>> b = 'b'  
>>> *# left side == identifier*>>> *# right side == object/value*>>> *# Rules to declare an identifier*>>> *# 1. a-z, A-Z are valid*>>> a = 10  
>>> a  
10  
>>> ab = 50  
>>> NUM = 34.6  
>>> NUM  
34.6  
>>> ab  
50  
>>> name = 'prashant prakash Patil'  
>>> name  
'prashant prakash Patil'  
>>> *# let’s check type of x and y*>>> type(x)  
<class 'int'>  
>>> type(y)  
<class 'str'>  
>>> *#------*>>> address = 'Satara Plot 23 pin-411203'  
>>> address  
'Satara Plot 23 pin-411203'>>> *#2. \_ is allowed*>>> bankname = 'SBI Pune'  
>>> bankname  
'SBI Pune'  
>>> bank\_name = 'BOI Kolhapur'  
>>> bank\_name  
'BOI Kolhapur'  
>>> pass 1 = 1234  
SyntaxError: invalid syntax  
>>> pass\_1 = 1234  
>>> pass\_1 *#allowed*1234  
>>> *#3. Space is not allowed in between*>>> a b = 700  
SyntaxError: invalid syntax  
>>> atm pin = 5555  
SyntaxError: invalid syntax  
>>> atm\_pin = 5555  
>>> atm\_pin  
5555>>> *#4. Special symbols and characters are not allowed*>>> *#!@$%^&\*~():"{}><?*>>> a@ = 55  
SyntaxError: invalid syntax  
>>> v% = 'py'  
SyntaxError: invalid syntax  
>>> pa$$word = 1234  
SyntaxError: invalid syntax>>> *#5. Number as a prefix is not allowed*>>> 4a = 400  
SyntaxError: invalid syntax  
>>> 5ab = 'rushikesh'  
SyntaxError: invalid syntax>>> *#5. number as a suffix is allowed*>>> a4 = 400  
>>> a4  
400  
>>> ab5 = 'rushikesh'  
>>> ab5  
'rushikesh'>>> *#6.Number are not a valid identifiers*>>> 4 = 400  
SyntaxError: can't assign to literal>>> *#7. number in between is allowed*>>> ab5c = 45  
>>> ab5c  
45  
>>> ab\_5\_c = 45 >>> ab\_5\_c  
45

Keywords

Some words have special meanings in Python. They are called keywords.

The following shows the list of keywords in Python:

*# from keyword give me an access to keyword list*['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Python is a growing and evolving language. So its keywords will keep increasing and changing.

Python provides a special module for listing its keywords called keyword*.*

To find the current keyword list, you use the following code:

>>> import keyword  
>>> keyword.kwlist

String literals

Python uses single quotes ('), double quotes ("), triple single quotes (''') and triple-double quotes (""") to denote a string literal.

The string literal need to be surrounded with the same type of quotes. For example, if you use a single quote to start a string literal, you need to use the same single quote to end it.

The following shows some examples of string literals:

s = ‘This is a string’

print(s)

s = “Another string using double quotes”

print(s)

s = ‘’’ string can span

multiple lines’’’

print(s)