Identifier

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).

Rules for writing identifiers

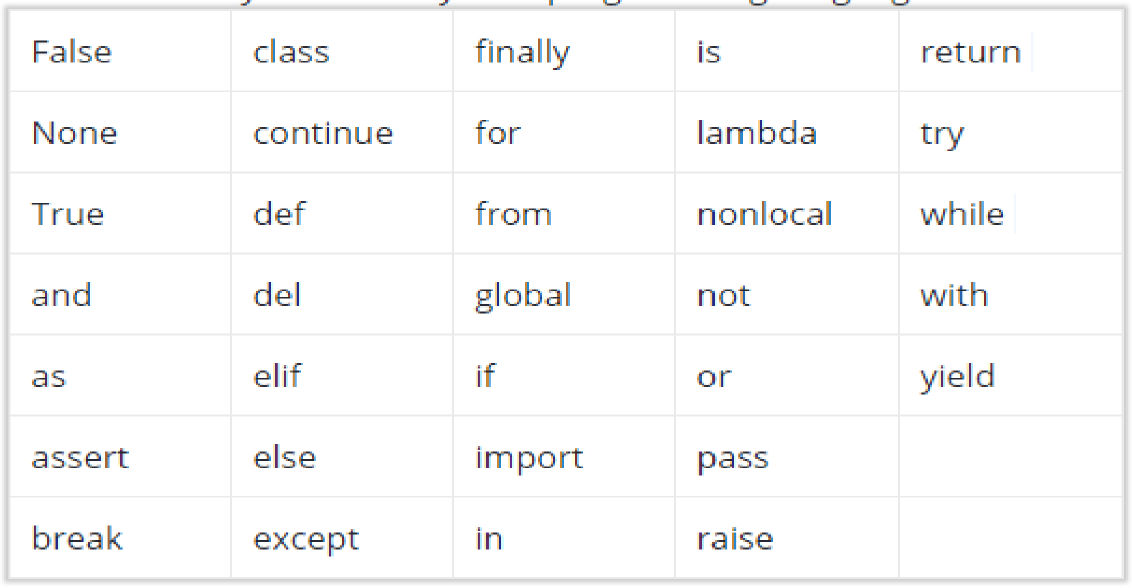
1. Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_). Names like `myClass`, `var_1` and `print_this_to_screen`, all are valid example.
2. An identifier cannot start with a digit. `1 variable` is invalid, but `variable1` is perfectly fine.
3. Keywords cannot be used as identifiers.

Python Keywords

Keywords are the reserved words in Python.
We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
In Python, keywords are case sensitive.

All the keywords except `True`, `False` and `None` are in lowercase and they must be written as it is.

Keywords in Python programming language



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

Literals

Objects are also called data structures. **Python** comes with some built-in objects. Some are used so often that **Python** has a quick way to make these objects, called **literals**. The **literals** include the string, Unicode string, integer, float, long, list, tuple and dictionary types.

Literals

- Numeric Literals
- String Literals
- Boolean Literals
- Special Literals
- Literal Collection

Literal is a raw data given in a variable or constant. In Python, there are various types of literals they are as follows:

Numeric Literals

Numeric Literals are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.

Numeric Type => Integer, Float, Complex

Problem =

1. Arithmetic Overflow Problem (3.455 * 4.677)
2. Arithmetic Underflow Problem (1/10000)
3. Loss of Precession Problem (10/3)

Eg=>

```
a = 0b1010          #Binary Literals
b = 100             #Decimal Literal
c = 0o310           #Octal Literal
d = 0x12c           #Hexadecimal Literal

#Float Literal
float_1 = 10.5
float_2 = 1.5e2
```

```
#Complex Literal
x = 3.14j
print(a, b, c, d)
print(float_1, float_2)
print(x, x.imag, x.real)
```

String literals

A string literal is a sequence of characters surrounded by quotes. We can use single, double or triple quotes for a string. And, a character literal is a single character surrounded by single or double quotes.

1. Single quotes
2. Double quotes
3. Triple quotes
4. Escape Sequences (\’, \\, \”, \n, \t)
5. Raw String(r or R)
6. String Formatting (Alignments left, right, center)

Boolean literals

A Boolean literal can have any of the two values: True or False.

Special literals

Python contains one special literal i.e. None. We use it to specify to that field that is not created.

Literal Collections

There are four different literal collections List literals, Tuple literals, Dict literals, and Set literals.

sys.exit

The **sys.exit()** function allows the developer to exit from Python. The **exit** function takes an optional argument, typically an integer, that gives an exit status. Zero is considered a “successful termination”. Be sure to check if your operating system has any special meanings for its exit statuses so that you can follow them in your own application. Note that when you call **exit**, it will raise the **SystemExit** exception, which allows cleanup functions to work in the **finally** clauses of **try / except** blocks.

Eg➔

```
import sys
sys.exit(0)
```