

### Lexical Analysis

Line Structure

Joining Lines

Blank Lines

Comments

Encoding Declaration

Indentation

### Input and Output

Printing output	<code>print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)</code>	<code>print("Hello world!", end=" ")</code>
Getting input	<code>input([prompt])</code>	<code>input("Enter your age")</code>

### Indentation

Leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements.

### Encoding Declaration

If a comment in the first or second line of the Python script matches the regular expression, this comment is processed as an encoding declaration; the first group of this expression names the encoding of the source code file.

```
#!/usr/
bi-
n/env
python
# -*-
coding:
utf-8 -
*_
```

### Comments

Hash	<code>#</code>	<code># some comment</code>
DocString	<code>"""</code>	<code>"""Demonstrates docstrings and does nothing really."""</code>

### Blank Lines

A logical line that contains only spaces, tabs, formfeeds and possibly a comment, is ignored.

During interactive input of statements, handling of a blank line may differ depending on the implementation of the read-eval-print loop.

In the standard interactive interpreter, an entirely blank logical line terminates a multi-line statement.

### Joining Lines

Explicit line joining	Two or more physical lines may be joined into logical lines using backslash characters ( <code>\</code> ), as follows: when a physical line ends in a backslash that is not part of a string literal or comment, it is joined with the following forming a single logical line, deleting the backslash and the following end-of-line character.	<code>if 1900 &lt; year &lt; 2100 and 1 &lt;= month &lt;= 12 \ and 1 &lt;= day &lt;= 31 and 0 &lt;= hour &lt; 24 \ and 0 &lt;= minute &lt; 60 and 0 &lt;= second &lt; 60: #</code> Looks like a valid date return 1
-----------------------	---	--



By **Venkatesh Jeyakumaresan** (codegutsy)  
[cheatography.com/codegutsy/](https://cheatography.com/codegutsy/)

Not published yet.  
Last updated 29th July, 2020.  
Page 1 of 3.

Sponsored by **ApolloPad.com**  
Everyone has a novel in them. Finish Yours!  
<https://apollopad.com>

### Joining Lines (cont)

Implicit line joining	Expressions in parentheses, square brackets or curly braces can be split over more than one physical line without using backslashes.	<pre>month_names = ['Januari', 'Februari', 'Maart', # These are the 'April', 'Mei', 'Juni', # Dutch names 'Juli', 'Augustus', 'September', # for the months 'Oktober', 'Novem- ber', 'December'] # of the year</pre>
-----------------------	--	--

### Remember:

- > A line ending in a backslash cannot carry a comment.
- > A backslash does not continue a comment.
- > A backslash does not continue a token except for string literals (i.e., tokens other than string literals cannot be split across physical lines using a backslash).
- > A backslash is illegal elsewhere on a line outside a string literal.

### Line Structure

Logical Lines	A logical line is what Python sees as a single statement	<p>Here is one logical on a physical line:</p> <pre>1) my_list = [1, 2, 3, 4] 2) print("- Hello world!")</pre>
---------------	--	--

Physical Lines	A physical line is what you see when you write the program. Two or more physical lines may be joined into logical lines using backslash characters.	<p>Here is one logical line on two physical lines:</p> <pre>1) my_list = [1, 2, 3, 4] 2) s = 'This is a string. \ This continues the string.'</pre>
----------------	---	---

### Introduction

Founder	<b>Guido van Rossum</b>
First release	February 20, 1991
Developers	<i>Python Software Foundation</i>
Paradigm	Multi-paradigm: functional, imperative, object-oriented, structured, reflective
Type Discipline	Duck, dynamic, gradual
Language Type	Interpreted

### Identifiers

Lu - uppercase letters	A..Z	UPPERCASENAME
LI - lowercase letters	a...z	LOWERCASENAME
Lt - titlecase letters	a..zA..z	titleCaseName
NI - letter numbers	a...zA...z-1...n	name2000   NAME999
Pc - connector punctuations	a_zA_Z	name_connector   NAME_C-ONNECTOR

A Python identifier is a name used to identify a variable, function, class, module or other object.

Identifier naming standards are mentioned above.

### Keywords and Reserved words

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

Keywords cannot be used as ordinary identifiers.



Variables	
Definition	Variable is a reserved memory location to store values
Syntax	IDENTIFIER_NAME = VALUE <sup>(1)</sup>
Assignment Operator	=
Walrus Operator/Assignment Expressions	:=
Example	number = 10   name = "Some name"   print(walrus := True)
Multiple Assignment	a = b = c = 1   a,b,c = 1,2,"john"
Deleting variable	del VARIABLE_NAME
(1) VALUE => any built-in type or user-defined class or function	

Built-in Types		
Boolean types	Truthy Falsy values	Scalar
Numeric types	Integer, Float, Complex	Scalar
Sequence types	List, Tuple, Range	Compound
Text Sequence types	String	Scalar
Mapping types	dictionary	Compound
Binary Sequence types	bytes, bytearray, memoryview	Compound
Set types	set, frozenset	Compound
Other types	functions, modules, class, methods, code objects, type objects, elipsis object, null object, NotImplemented objects	
Special attributes		

Boolean Type	
Reference	<a href="#">Link</a>

Numeric Type	
Integer	<a href="#">Link</a>
Float	<a href="#">Link</a>
Complex	<a href="#">Link</a>

Sequence Types	
Common sequence operations	<a href="#">Link</a>
List	<a href="#">Link</a>
Tuple	<a href="#">Link</a>
Range	<a href="#">Link</a>
String	<a href="#">Link</a>

Mapping type	
Dictionary	<a href="#">Link</a>

Binary Sequence Types	
Bytes	<a href="#">Link</a>
Bytearray	<a href="#">Link</a>
Memoryview	<a href="#">Link</a>

Sets types	
Set	<a href="#">Link</a>
Frozenset	<a href="#">Link</a>

Other Types	
Functions	<a href="#">Link</a>
Class	<a href="#">Link</a>
Modules	<a href="#">Link</a>
Elipsis	<a href="#">Link</a>
Null	<a href="#">Link</a>
NotImplemented	<a href="#">Link</a>
Special Attributes	<a href="#">Link</a>

