Multi-Line Strings

Physical lines of codes >>>> end with a physical line NEWLINE character

Logical lines of code >>>> end with a logical NEWLINE token

Tokenized

Physical newline vs logical newline

Sometimes, physical newlines are ignored in order to combine multiple physical lines into a single logical line of code, terminated by a logical NEWLINE token.

This happens because computers can read a cod in a single line. But for human readability we write our codes in multiple lines. Otherwise would be impossible for us to read a code in a single line and found a meaning for it.

The conversion can be implicit or explicit

Implicit: Doesn’t need to inform the code that you are breaking a line

The interpreter will

List literals: [] square brackets

Tuple literals: () parenthesis

Dictionary literals: {} curly braces

Set literals: {} curly braces

Function arguments / parameters

[1, # item 1

2, # item2

3, # item 3

]

Explicit: Need to inform the code that you are breaking the line. That can be done by using the “\” (backlash) character.

Multi-line statements are not implicitly converted to a single logical line.

Wrong: Right:

if a if a \

and b and b \

and c: and c:

Comments cannot be part of a statement, not even a multi-line statement.

Multi-Line String Literals.

Multi-line string literals can be created using triple delimiters (‘ single or “ double)

‘’’This is

a multi-line string’’’

“””this is

A multi-line string”””

Non-visible character such as newlines, tabs, spaces, etc. are actually part of the string.

You can use escaped characters (e.g. \n, \t), use the string formatting, etc.

A multi-line string is just a regular string.

Multi-line string are not comments, although they can be used as such, especially with special comments called docstrings. That means that a comment will be ignored by the compiler when reading the code. In other hand a multi-line string used for a comment, the compiler will read it.

Examples of coding

a = [1, 2, 3]

a = [1, #item 1

,2]

A = {‘key1’: 1 #value for key 1

,’key2’: 2 #value for key 2

}

If a > 5 \

and b > 10 \

and c > 20:

print(‘yes’)