

# MULTI-LINE STATEMENTS AND STRINGS



## Python Program

—————> physical lines of code

—————> logical lines of code

—————> tokenized

end with a physical **newline** character

end with a logical **NEWLINE** token

## 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

Conversion can be implicit or explicit



## Implicit

Expressions in:

list literals: [ ]

tuple literals: ( )

dictionary literals: { }

set literals: { }

function arguments / parameters

supports inline comments

```
[1,      [1, #item 1  
2,      2, #item 2  
3]      3  #item 3  
]
```

```
def my_func(a,  
            b, #comment  
            c):  
    print(a, b, c)
```

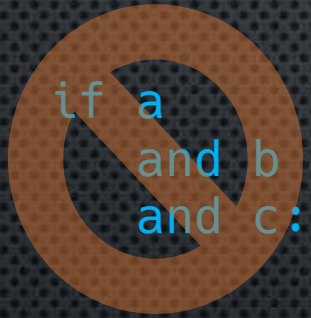
```
my_func(10, #comment  
        20, 30)
```



## Explicit

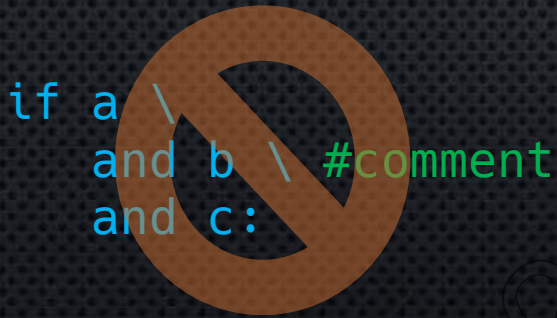
You can break up statements over multiple lines explicitly, by using the `\` (backslash ) character

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



```
if a \  
    and b \  
    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"""
```

Be aware that non-visible characters such as **newlines**, **tabs**, etc. are actually part of the string – basically anything you type.

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

A multi-line string is just a regular string.

Multi-line strings are not comments, although they can be used as such, especially with special comments called **docstrings**.



