



# PYTHON

---

# DATA TYPES

# Built-in Data Types



1. Numbers Types: `int`, `float`, `complex`

2. String Type: `str`

3. Boolean Type: `bool`

# Numbers Types

In Python, numeric data type represents the data that has a numeric value. The numeric value can be an integer, floating number, or even complex number.

## 1. Integers ( `int` )

The integer data type is a numeric data type that holds integers. For example the numbers 1,2,3 and so on. So that every variable that has an integer value, it will be categorized as an integer.

```
[ ] #Number
    #type data Integer
    length = 10
    print(length)
    type(length)

10
int
```

# Numbers Types

---

## 2. Float

Almost the same as the integer data type, it's just that the float data type is used for variables that have a fractional / decimal value.

Example :

```
[ ] #type data Complex  
width = 5.5  
print(width)  
type(width)
```

# Numbers Types



## 3. Complex

Complex numbers are represented in the form  $x+yj$ , where  $a$  is a real number and  $y$  is the imaginary one

Example :

```
[23] c = 1+2j
```


```
[24] type(c)
```

```
complex
```

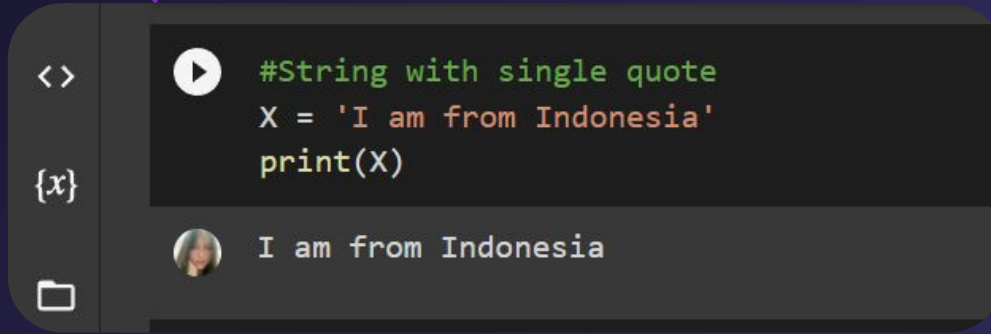
# String Types






Strings are sequences of **character** data. The string type in Python is called **"str"**. String literals may be delimited using either **single or double quotes**. All the characters between the opening delimiter and matching closing delimiter are part of the string.



# String with single quote

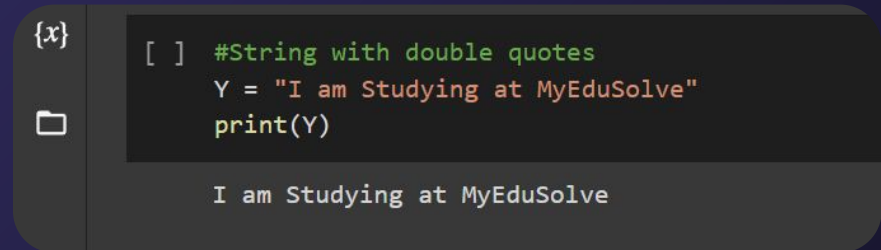


A screenshot of a Python IDE interface. On the left, there is a sidebar with icons for a code editor (two arrows), a variable explorer (a curly brace with 'x'), and a file explorer (a folder icon). The main area shows a code editor with the following text: `#String with single quote`, `X = 'I am from Indonesia'`, and `print(X)`. Below the code editor, there is a console area showing the output: `I am from Indonesia`. The background is dark with some decorative elements like a purple circle and wavy lines in the top right corner.


```
<> {x}  #String with single quote  
X = 'I am from Indonesia'  
print(X)  
  
 I am from Indonesia  

```

Using either  
single or double  
quotes, will  
show the same  
output

# String with double quote



A screenshot of a Python IDE interface. On the left, there is a sidebar with icons for a code editor (two arrows), a variable explorer (a curly brace with 'x'), and a file explorer (a folder icon). The main area shows a code editor with the following text: `[ ] #String with double quotes`, `Y = "I am Studying at MyEduSolve"`, and `print(Y)`. Below the code editor, there is a console area showing the output: `I am Studying at MyEduSolve`. The background is dark with some decorative elements like a purple circle and wavy lines in the top right corner.

```
{x} [ ] #String with double quotes  
Y = "I am Studying at MyEduSolve"  
print(Y)  
  
I am Studying at MyEduSolve  

```



# Notes :

---

```
[ ] Z = ''  
    print(Z)
```

A string in Python can contain as many characters, the only limit is your machine's memory resources. A string can also be empty.



# How to use a *quote character* as a part of string

You can't use a quote character as part of the string itself. If you open the string with a single quote, Python will assume the next single quote as the closing delimiter.



```
[ ] #Wrong example
    #String_1 = 'in this string I will use a single quote (') character'
    #print(String_1)

    #If we run this code, it can be syntax error
```

# How to use a *quote character* as a part of string

If you want to include either type of quote character within the string, the simplest way is to delimit the string with the other type. If you want *to use single quote in your string, use double quote as your delimiter.*

```
[ ] #Use single quote in string with double quote delimiter
string_2 = "I want to be a data scientist, so I'll start to learn Python"
print(string_2)
```

I want to be a data scientist, so I'll start to learn Python

# How to use a *quote character* as a part of string

If you want to include either type of quote character within the string, the simplest way is to delimit the string with the other type. If you want *to use double quote in your string, use single quote as your delimiter.*



```
#Use double quotes in string with single quote delimiter
string_3 = 'Andrew McAfee once said, "The world is one big data problem."'
print(string_3)
```



Andrew McAfee once said, "The world is one big data problem."

# Triple- Quoted String



```
[ ] #Triple-quoted string
mul_string = """This is a
multiline string.
You can create many lines
in one time"""

print(mul_string)
```

```
This is a
multiline string.
You can create many lines
in one time
```

Triple-quoted strings are delimited by matching groups of three single quotes or three double quotes. You can use triple-quoted string to ***create multiline string.***

# Escape Sequences in Strings

```
[ ] #multiline string with backslash+n  
mul_string2 = "This is \nanother multiline string. \nBut this time, \nI don't use triple-quoted string."  
print(mul_string2)
```

If you want to interpret a character or sequence of characters within a string a differently, you can accomplish that using backslash character.

For example, in string, if you want to put a sentence in **new line without using enter**, you can use **backslash+n**.

```
This is  
another multiline string.  
But this time,  
I don't use triple-quoted string.
```

# Boolean Types

---

**Boolean is a datatype that will return the value of "True" or "False". This datatype is also known as logical data type due to its capability to solve logical problems**



# Basic Example of Boolean

```
[ ] #Basic example of boolean

A = True #return the value of "True"
B = False #return the value of "False"
print(str(A) + " " + str(type(A))) #type() is used to determine the object's type
print(str(B) + " " + str(type(B)))

True <class 'bool'>
False <class 'bool'>
```

```
print(str(3>1)) #these line return the value of "True"
print(str(3<1)) #these line return the value of "False"
```

```
True
False
```

Boolean can also be used to determine the comparison between two values



# Boolean Capability

```
#Comparison for String  
print(str("Code"=="Code"))  
print(str("Tensorflow"!="Keras"))  
  
True  
True
```

Boolean datatype are capable to determine whether three is a bigger value than one. If we run the cell the debug will automatically determine the boolean value. Boolean can also be used to determine the comparison for string data type

# Boolean Capability

```
#logical operation  
print(str(2>0 or 1>3))  
print(str(2>0 and 1>3))  
print(str(not(2<0)))
```

```
True  
False  
True
```

Another capability of boolean data type is to determine logical problem

Boolean can also use "is" and "in" operator to solve logical problem

```
#"in" operator  
print(str("e" in "MyEduSolve"))
```

```
True
```

```
#is operator  
x = [2,3]  
y = x  
  
x is y
```

```
True
```

The background is a deep purple color. It features abstract, flowing wavy lines in a lighter shade of purple, primarily on the left and right sides. Scattered throughout the background are numerous small dots in various shades of purple, creating a starry or digital effect.

**Thank You!**