Python

Python is a popular programming language.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

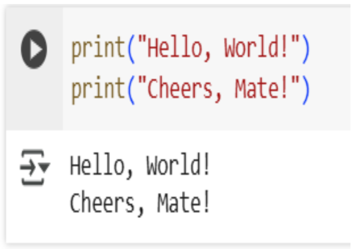
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

Why use Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

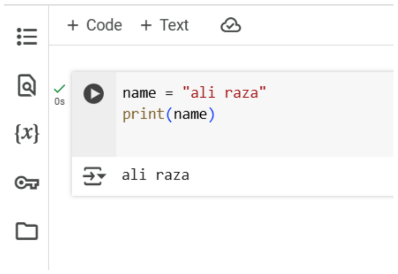
Coding in Python using Google Colab

The Python **print()** function is often used to output variables.



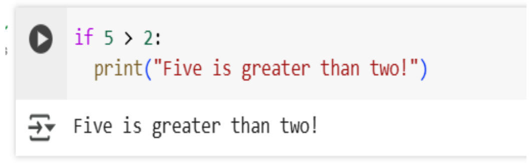
```
print("Hello, World!")
print("Cheers, Mate!")
```

```
Hello, World!
Cheers, Mate!
```



```
name = "ali raza"
print(name)
```

```
ali raza
```

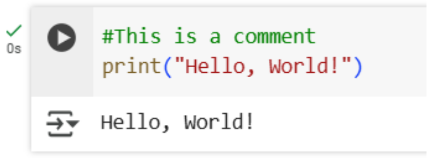


```
if 5 > 2:
  print("Five is greater than two!")
```

```
Five is greater than two!
```

Python Comments

Comments starts with a #, and Python will ignore them:



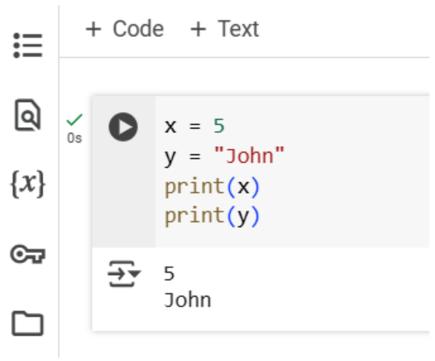
```
#This is a comment
print("Hello, World!")
```

```
Hello, World!
```

Python Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.



```
x = 5
y = "John"
print(x)
print(y)
```

```
5
John
```

Built-in Data Types

In programming, data type is an important concept.

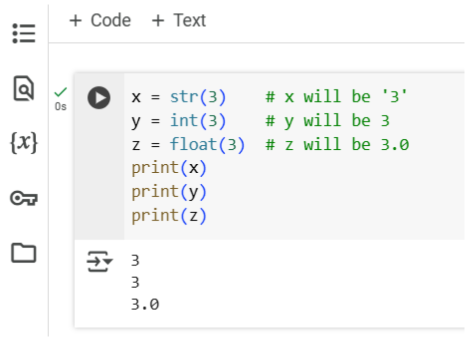
Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

Casting

If you want to specify the data type of a variable, this can be done with casting.

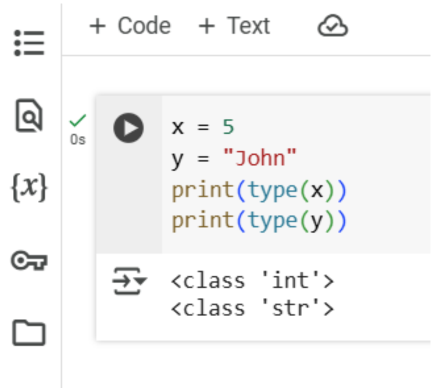


```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
print(x)
print(y)
print(z)
```

```
3
3
3.0
```

You can get the **data type** of a variable with the type() function.

Result defined you the type of x and type of y

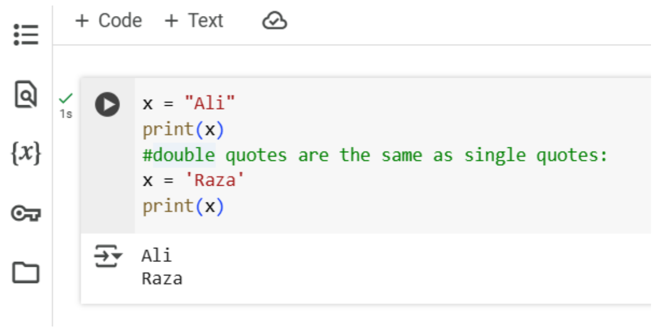


```
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

Single or Double Quotes?

String variables can be declared either by using single or double quotes:

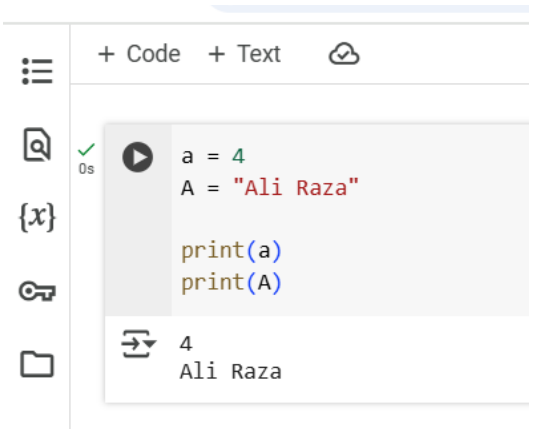


```
x = "Ali"
print(x)
#double quotes are the same as single quotes:
x = 'Raza'
print(x)
```

```
Ali
Raza
```

Case-Sensitive

Variable names are case-sensitive.



```python
a = 4
A = "Ali Raza"

print(a)
print(A)
```

```
4
Ali Raza
```

A variable name must start with a letter or the underscore character

A variable name cannot start with a number

A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive (age, Age and AGE are three different variables)

A variable name cannot be any of the Python keywords.

Multi Words Variable Names

Camel Case

Each word, except the first, starts with a capital letter:

my**V**ariable**N**ame = "John"

Pascal Case

Each word starts with a capital letter:

MyVariableName = "John"

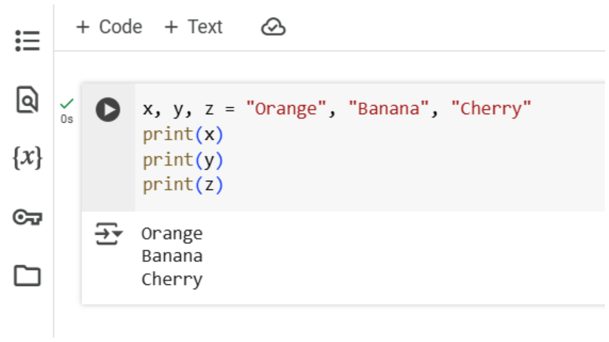
Snake Case

Each word is separated by an underscore character:

my_variable_name = "John"

Many Values to Multiple Variables

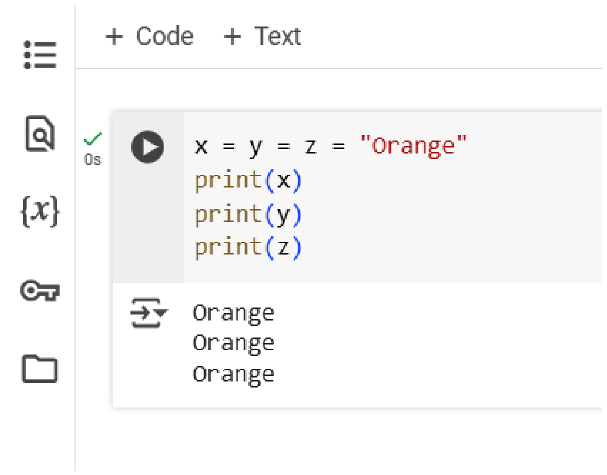
Python allows you to assign values to multiple variables in one line:



```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

```
Orange
Banana
Cherry
```

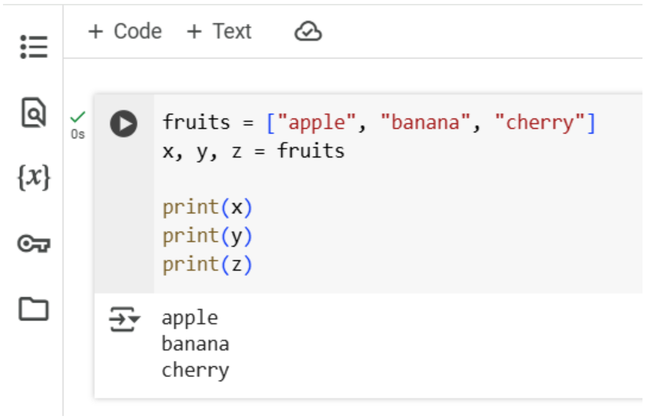
And you can assign the *same* value to multiple variables in one line:



```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

```
Orange
Orange
Orange
```

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called *unpacking*.

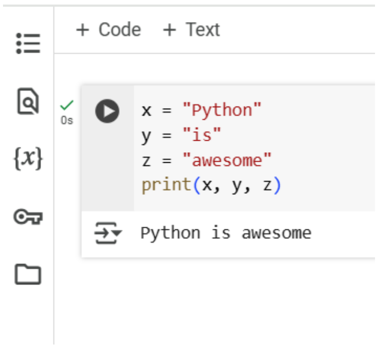


```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits

print(x)
print(y)
print(z)
```

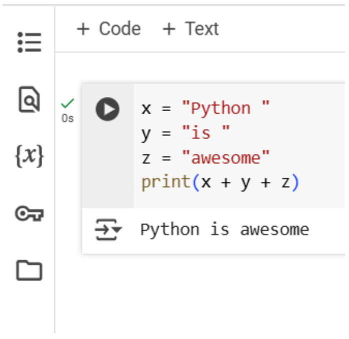
```
apple
banana
cherry
```

Output Variables



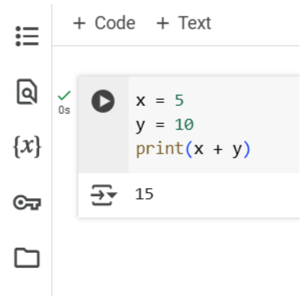
```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

```
Python is awesome
```



```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

```
Python is awesome
```



```
x = 5
y = 10
print(x + y)
```

```
15
```