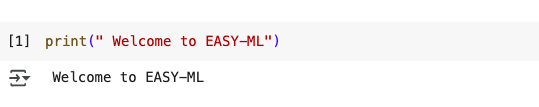
# PYTHON INTRODUCTION

Python was invented in the early 1990s by Guido van Rossum

It is an open-source project. Although often categorized as a scripting language, Python is much more powerful and versatile.

Designed with scalability in mind, it has supported object-oriented and functional programming paradigms since its inception, making it a robust and flexible language for a wide range of applications.

Example of a python code

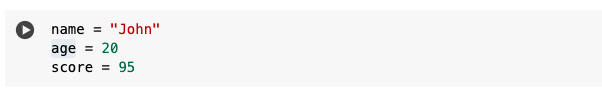


## Variables

In Python, a variable is a name that stores a value.

A variable stores data (like a number or word). To create a variable you choose a name and using the = sign to assign it a value.

**Example:**



**Rules for variable names:**

* Must start with a letter or underscore (\_)
* Cannot start with a number
* Can contain letters, numbers, and underscores
* Should not be a Python reserved word like if, while, class, etc.

There are some reserved words in Python:

and, assert, break, class, continue, def, del, elif, else, except, exec,

finally, for, from, global, if, import, in, is, lambda, not, or,

pass, print, raise, return, try, while

**Arithmetic Operations**

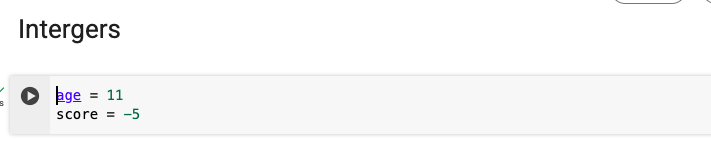
* Addition, subtraction, multiplication, division, modulus, exponentiation
* Only performed on numbers

**String Operations**

These include, creating strings, concatenating strings, slicing, splitting, trimming, change of case

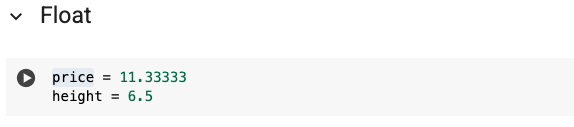
## Basic Data Types

1. **Integers (default for numbers)-**Used to store whole numbers (positive or negative, no decimals).



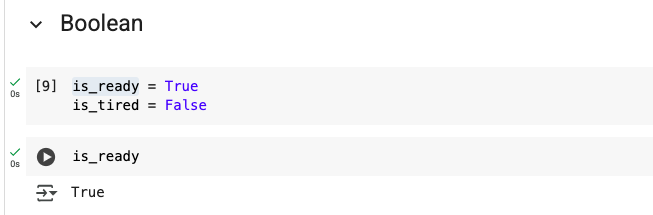
1. **Floats -** Used to store **decimal numbers** (numbers with a point).

**Example**



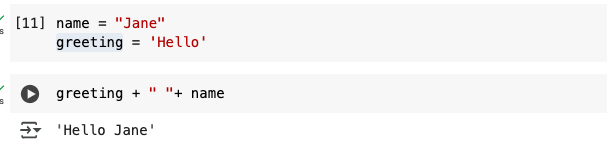
1. **Boolean (bool)**- Used to store True or False values (yes/no, on/off logic).

**Example**



1. **Strings**-Used to store text (anything in quotes).

Example



* Can use “” or ‘’ to specify with “abc” == ‘abc’
* Unmatched can occur within the string: “matt’s”
* Use triple double-quotes for multi-line strings or strings than contain both ‘ and “ inside of them: “““a‘b“c”””

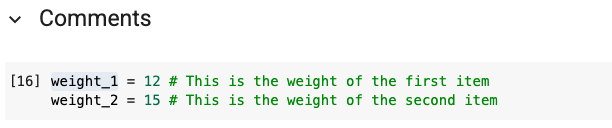
## Whitespaces

* Whitespace is meaningful in Python: especially indentation and placement of newlines
* Use a newline to end a line of code
* Use \ when must go to next line prematurely
* No braces {} to mark blocks of code, use *consistent* indentation instead
* First line with *less* indentation is outside of the block
* First line with *more* indentation starts a nested block
* Colons start of a new block in many constructs, e.g. function definitions, then clauses

## **Comments**

* Start comments with #, rest of line is ignored
* Can include a “documentation string” as the first line of a new function or class you define
* Development environments, debugger, and other tools use it: it’s good style to include one

Example

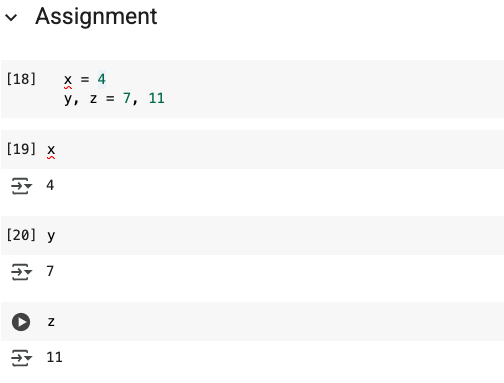


## Assignment

* *Binding a variable* in Python means setting a *name* to hold a *reference* to some *object*
* *Assignment creates references, not copies*
* Names in Python do not have an intrinsic type, objects have types
* Python determines the type of the reference automatically based on what data is assigned to it
* You create a name the first time it appears on the left side of an assignment expression: x= 3
* A reference is deleted via garbage collection after any names bound to it have passed out of scope

You can assign to multiple names at the same time

Example



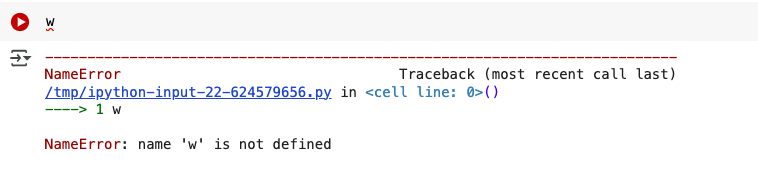
This makes it easy to swap values

x, y = y, x Assignments can be chained

a = b = x = 2

## Accessing Non-Existent Name

Accessing a name before it’s been properly created (by placing it on the left side of an assignment), raises an error



## Naming conventions

* The Python community has these recommended naming conventions
* **joined\_lower** for functions, methods and attributes
* **joined\_lower** or **ALL\_CAPS** for constants
* **TitleCaps** for classes
* **camelCase** only to conform to pre-existing conventions
* Attributes: interface, \_internal, \_\_private