**Variables**

Variables are containers for storing data values.

**Creating Variables**

Python has no command for declaring a variable.

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

Variables do not need to be declared with any particular type, and can even change type after they have been set.

x = 4

x = “Code”

Print(x)

Output :

Code

**Casting**

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

x = str(3)

y = int(3)

z = float(3)

Print(x)

Print(y)

Print(z)

Output :

3

3

3.0

**Get the Type**

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

x = 5

y = “John”

Print(type(x))

Print(type(y))

Output :

<class ‘int’>

<class ‘str’>

**Single or Double Quotes?**

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

x = “John”

Print(x)

x = ‘John’

Print(x)

Output :

John

John

**Case-Sensitive**

Variable names are case-sensitive.

a = 4

A = “Sally”

Print(a)

Print(A)

Output:

​4

Sally

**Variable Names**

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

**Rules for Python variables:**

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.

Legal variable names

myvar = “John”

My\_var = “John”

\_my\_var = “John”

myVar = “John”

MYVAR = “John”

Myvar2 = “John”

Print(myvar)

Print(my\_var)

Print(\_my\_var)

Print(myVar)

Print(MYVAR)

Print(myvar2)

Illegal variable names:

2myvar = “John”

My-var = “John”

My var = “John”

#This example will produce an error in the result

**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)

Output:

​Orange

Banana

Cherry

Note: Make sure the number of variables matches the number of values, or else you will get an error

**One Value to Multiple Variables**

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

Example

x = y = z = “Orange”

Print(x)

Print(y)

Print(z)

Output :

Orange

Orange

Orange

**Unpack a Collection**

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.

Example

Unpack a list:

Fruits = [“apple”, “banana”, “cherry”]

x, y, z = fruits

Print(x)

Print(y)

Print(z)

Output :

apple

Banana

Cherry

**Output Variables**

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

x = “Python is awesome”

Print(x)

Output :

Python is awesome

In the print() function, you output multiple variables, separated by a comma:

Example

x = “Python”

y = “is”

z = “awesome”

Print(x, y, z)

Output :

Python is awesome

You can also use the + operator to output multiple variables:

Example

x = “Python “

y = “is “

z = “awesome”

Print(x + y + z)

Output :

Python is awesome

Notice the space character after “Python “ and “is “, without them the result would be “Pythonisawesome”.

For numbers, the + character works as a mathematical operator:

Example

x = 5

y = 10

Print(x + y)

Output :

15

In the print() function, when you try to combine a string and a number with the + operator, Python will give you an error:

Example

x = 5

y = “John”

Print(x + y)

Output :

Error

The best way to output multiple variables in the print() function is to separate them with commas, which even support different data types:

Example

x = 5

y = “John”

Print(x, y)

Output :

5 john

**Global Variables**

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside

Example

Create a variable outside of a function, and use it inside the function

x = “awesome”

Def myfunc():

Print(“Python is “ + x)

Myfunc()

Output :

Python is awesome

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

Example

Create a variable inside a function, with the same name as the global variable

x = “awesome”

Def myfunc():

X = “fantastic”

Print(“Python is “ + x)

Myfunc()

Print(“Python is “ + x)

Output :

Python is fantastic

Python is awesome

**The global Keyword**

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

Example

If you use the global keyword, the variable belongs to the global scope:

Def myfunc():

Global x

x = “fantastic”

Myfunc()

Print(“Python is “ + x)

Output :

Python is fantastic

Also, use the global keyword if you want to change a global variable inside a function.

Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = “awesome”

Def myfunc():

Global x

x = “fantastic”

Myfunc()

Output :

x = “awesome”

​

Def myfunc():

Global x

x = “fantastic”

​

Myfunc()

​

Print(“Python is “ + x)

​

Python is fantastic

**Python Numbers**

There are three numeric types in Python:

Int

Float

Complex

Variables of numeric types are created when you assign a value to them:

Example

X = 1 # int

Y = 2.8 # float

Z = 1j # complex

To verify the type of any object in Python, use the type() function:

Example

Print(type(x))

Print(type(y))

Print(type(z))

Output :

<class ‘int’>

<class ‘float’>

<class ‘complex’>

**Int**

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example

Integers:

x = 1

y = 35656222554887711

z = -3255522

Print(type(x))

Print(type(y))

Print(type(z))

Output :

<class ‘int’>

<class ‘int’>

<class ‘int’>

**Float**

Float, or “floating point number” is a number, positive or negative, containing one or more decimals.

Example

Floats:

x = 1.10

y = 1.0

z = -35.59

Print(type(x))

Print(type(y))

Print(type(z))

Output :

<class ‘float’>

<class ‘float’>

<class ‘float’>

Float can also be scientific numbers with an “e” to indicate the power of 10.

Example

Floats:

x = 35e3

y = 12E4

z = -87.7e100

Print(type(x))

Print(type(y))

Print(type(z))

Output :

<class ‘float’>

<class ‘float’>

<class ‘float’>

**Complex**

Complex numbers are written with a “j” as the imaginary part:

Example

Complex:

x = 3+5j

y = 5j

z = -5j

Print(type(x))

Print(type(y))

Print(type(z))

Output :

<class ‘complex’>

<class ‘complex’>

<class ‘complex’>

**Type Conversion**

You can convert from one type to another with the int(), float(), and complex() methods:

Example

Convert from one type to another:

x = 1 # int

y = 2.8 # float

z = 1j # complex

#convert from int to float:

A = float(x)

#convert from float to int:

B = int(y)

#convert from int to complex:

C = complex(x)

Print(a)

Print(b)

Print(c)

Print(type(a))

Print(type(b))

Print(type(c))

Output :

1.0

2

(1+0j)

<class ‘float’>

<class ‘int’>

<class ‘complex’>

Note: You cannot convert complex numbers into another number type

**Random Number**

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

Example

Import the random module, and display a random number between 1 and 9:

Import random

Print(random.randrange(1, 10))

Output :

8