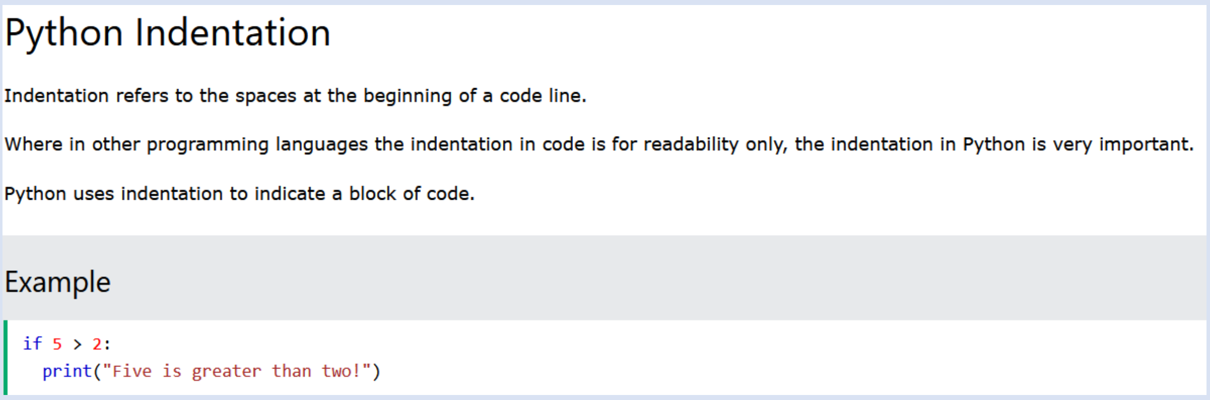
https://www.w3schools.com/python/python_syntax.asp



Python Indentation

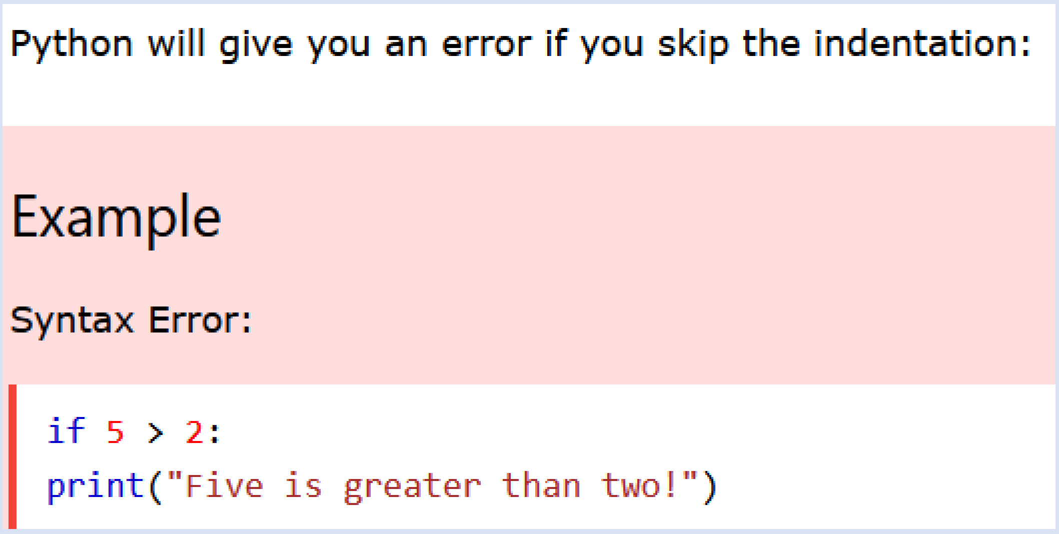
Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

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

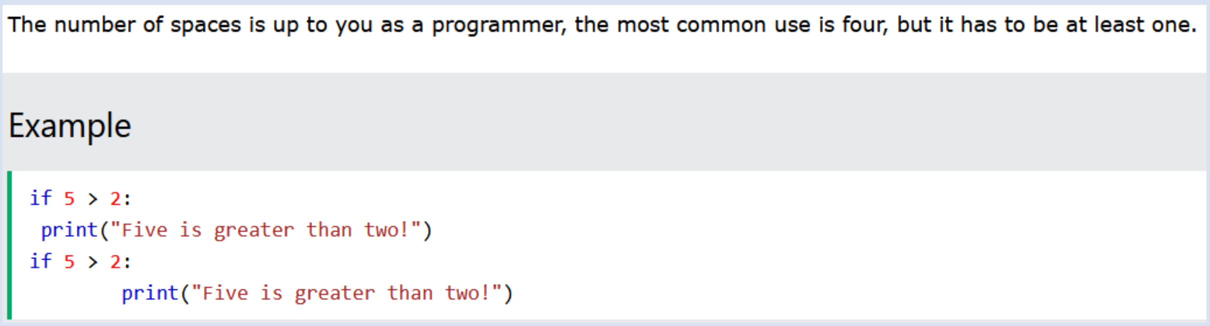


Python will give you an error if you skip the indentation:

Example

Syntax Error:

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



The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

Example

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

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

Example

Syntax Error:

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

Python Variables

In Python, variables are created when you assign a value to it:

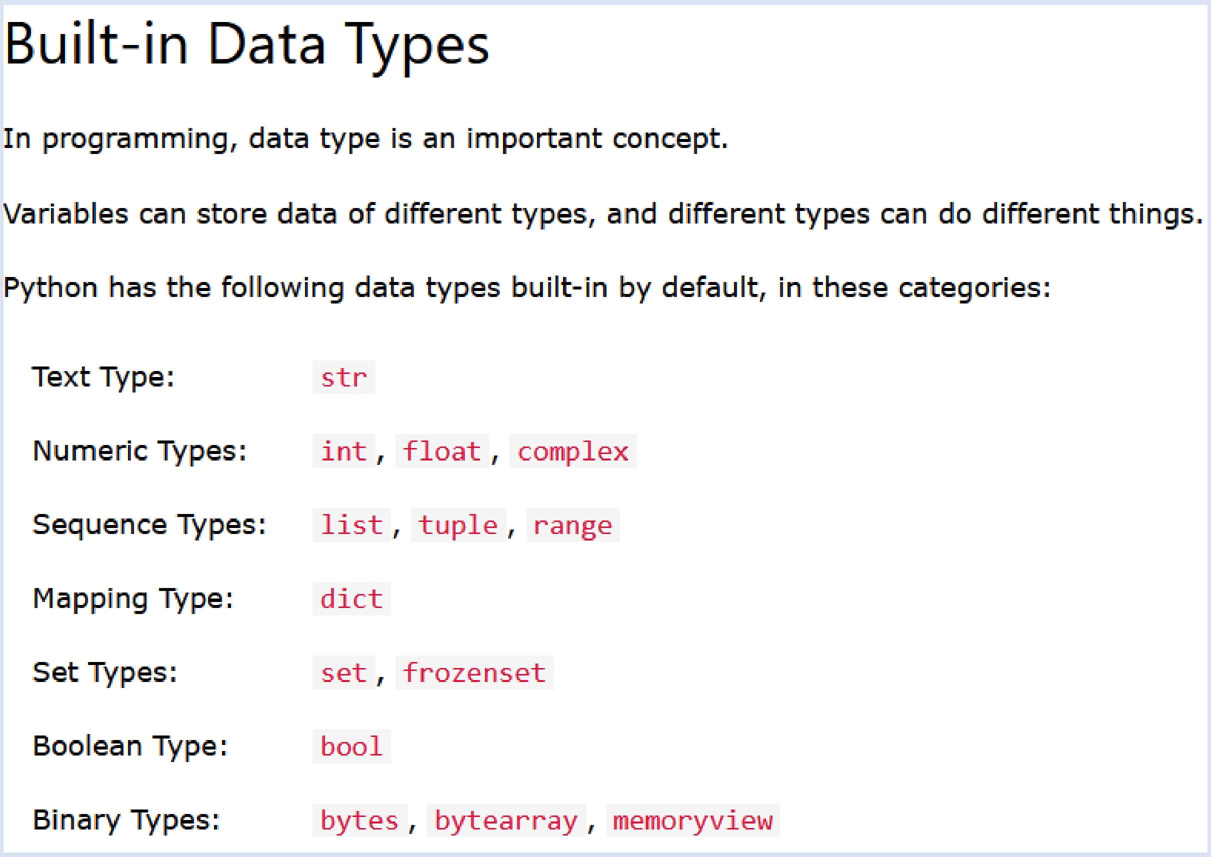
Example

Variables in Python:

```
x = 5
y = "Hello, World!"
```

Python has no command for declaring a variable.

https://www.w3schools.com/python/python_datatypes.asp



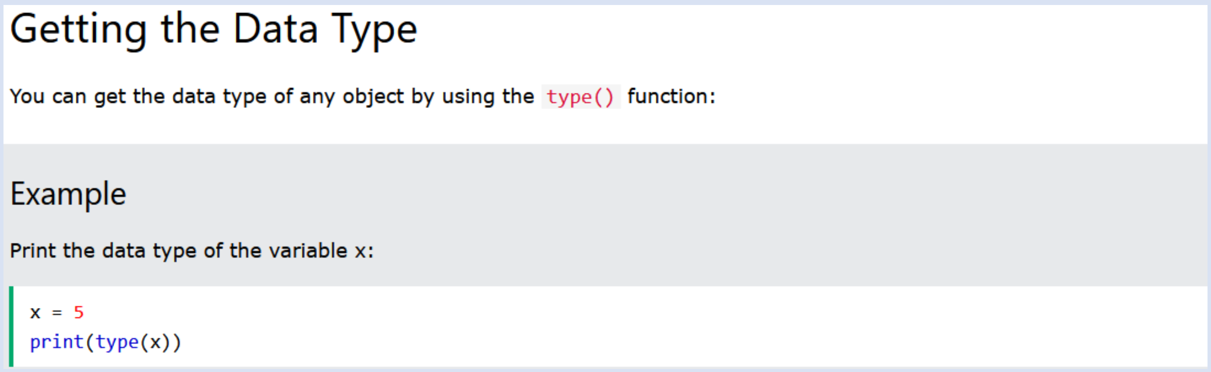
Built-in Data Types

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`



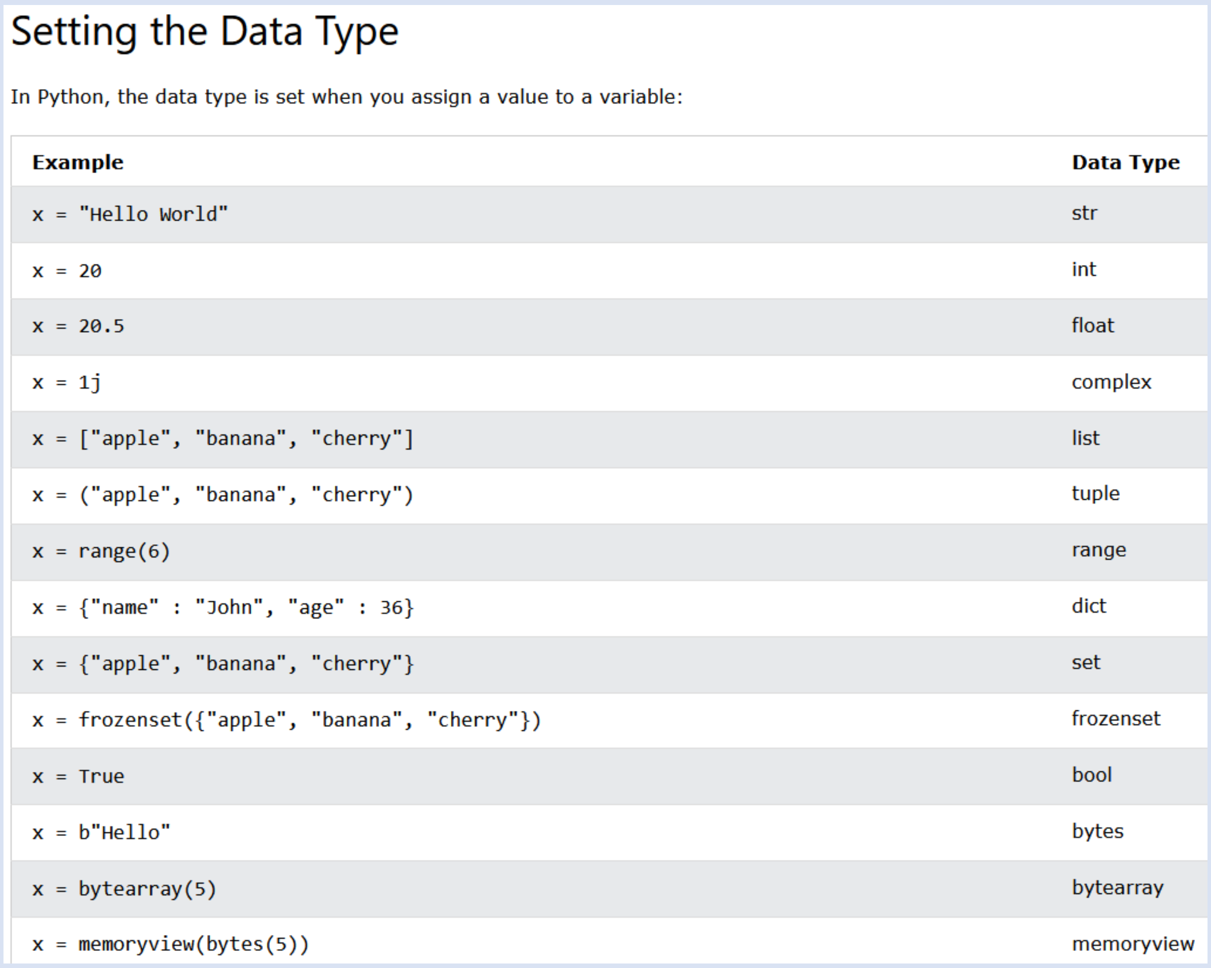
Getting the Data Type

You can get the data type of any object by using the `type()` function:

Example

Print the data type of the variable x:

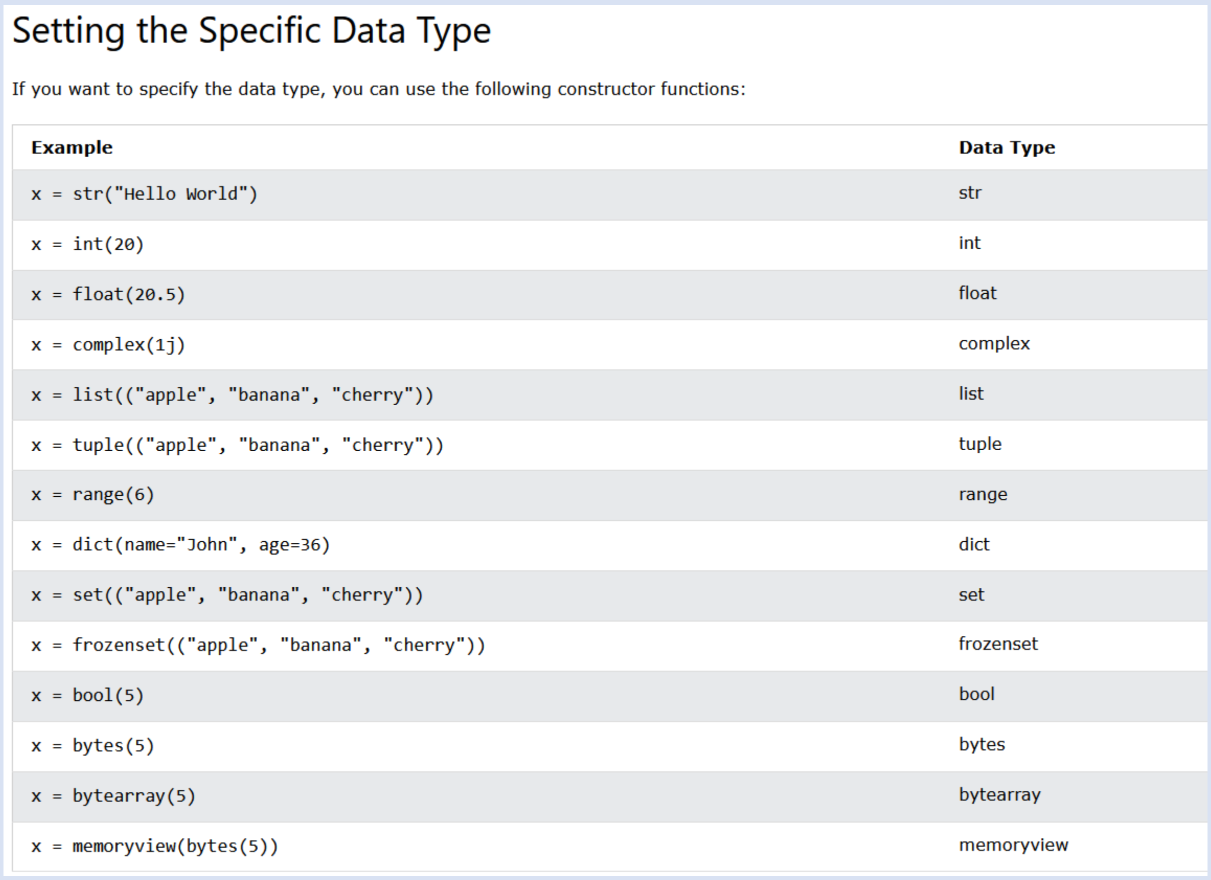
```
x = 5
print(type(x))
```



Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

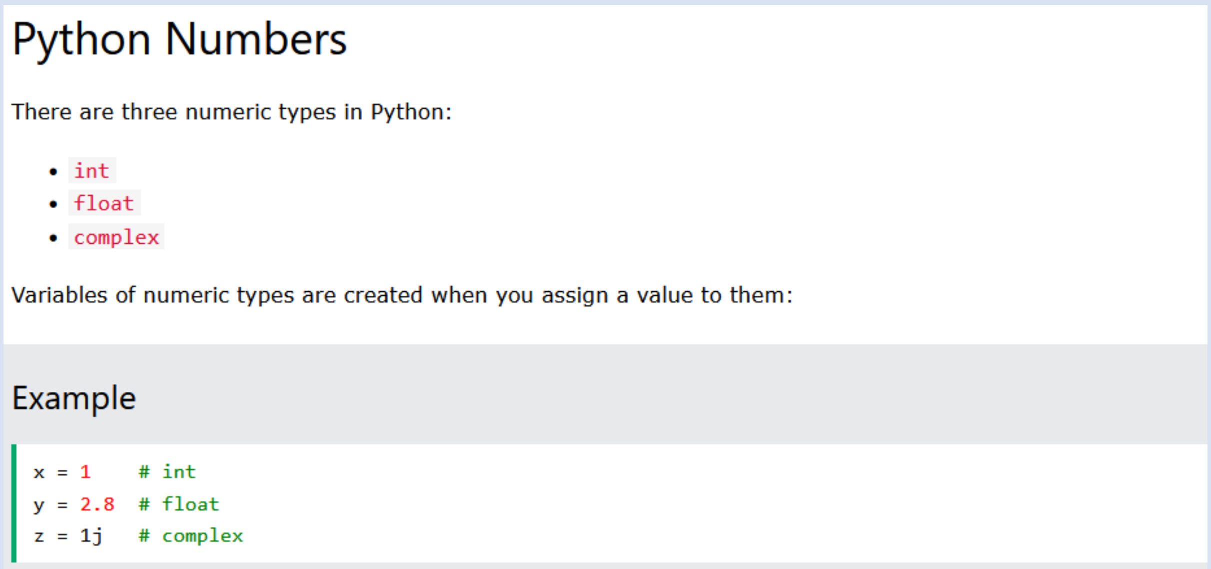


Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

Example	Data Type
`x = str("Hello World")`	str
`x = int(20)`	int
`x = float(20.5)`	float
`x = complex(1j)`	complex
`x = list(("apple", "banana", "cherry"))`	list
`x = tuple(("apple", "banana", "cherry"))`	tuple
`x = range(6)`	range
`x = dict(name="John", age=36)`	dict
`x = set(("apple", "banana", "cherry"))`	set
`x = frozenset(("apple", "banana", "cherry"))`	frozenset
`x = bool(5)`	bool
`x = bytes(5)`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

https://www.w3schools.com/python/python_numbers.asp



Python Numbers

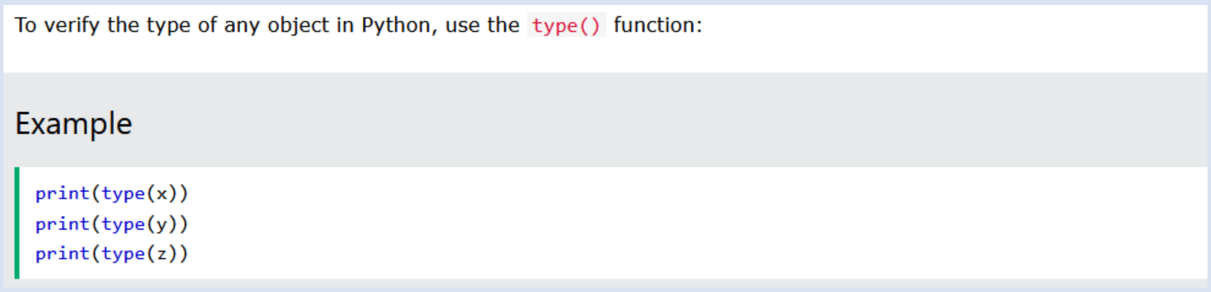
There are three numeric types in Python:

- `int`
- `float`
- `complex`

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

Example

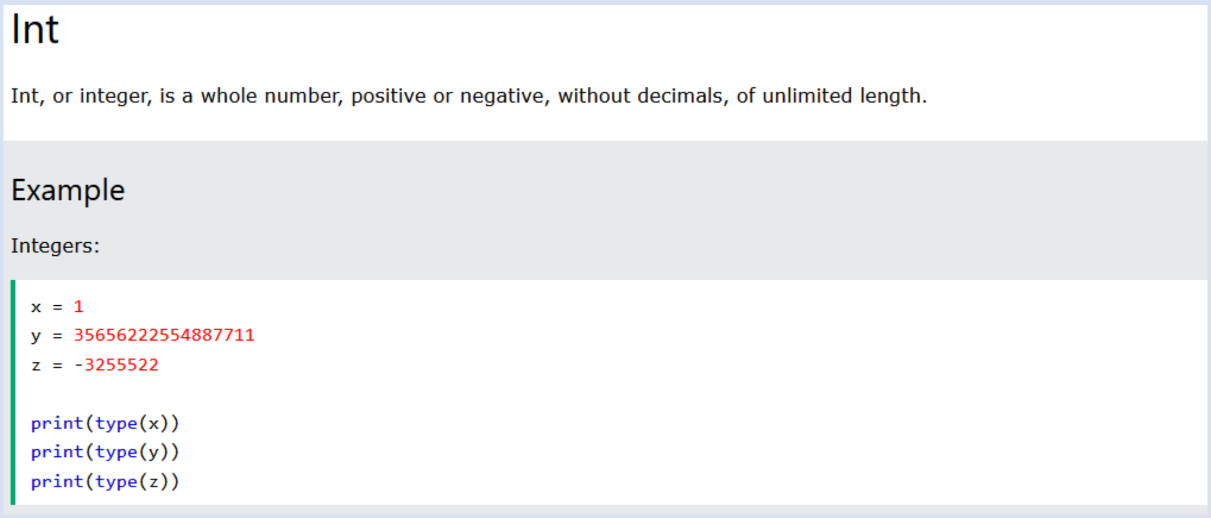
```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```



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

Example

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



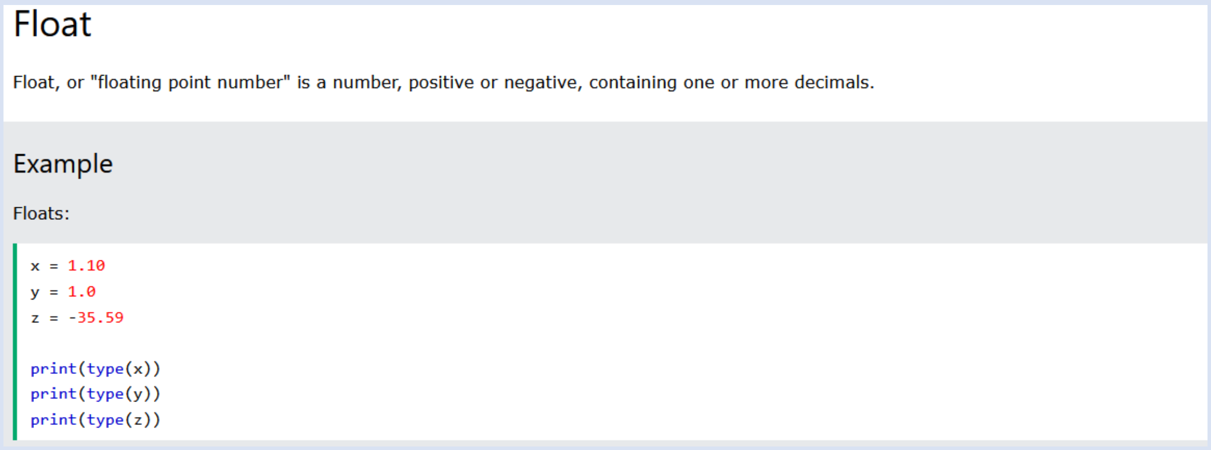
Int

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))
```



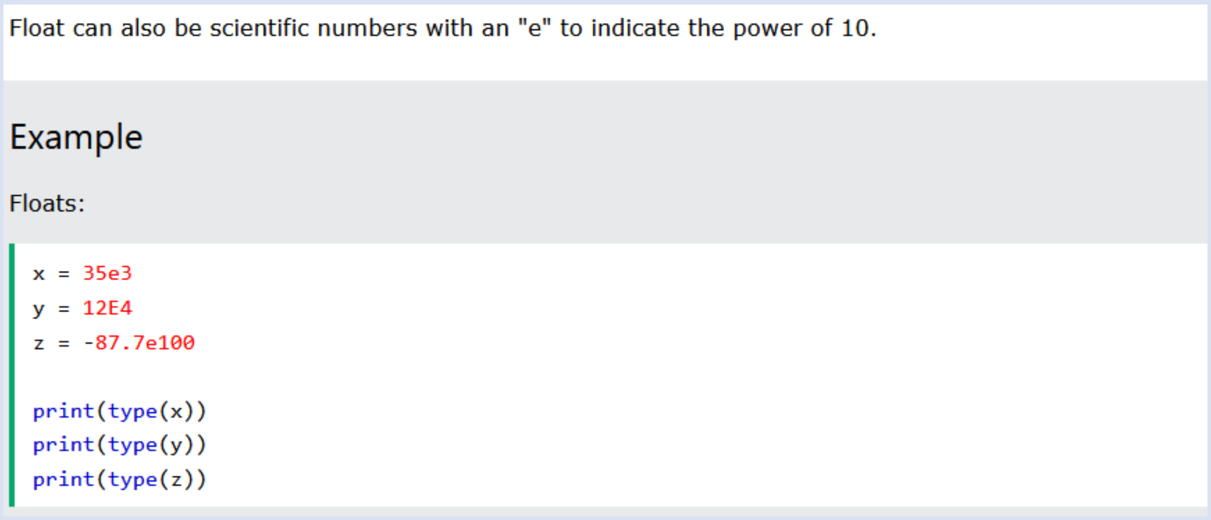
Float

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))
```

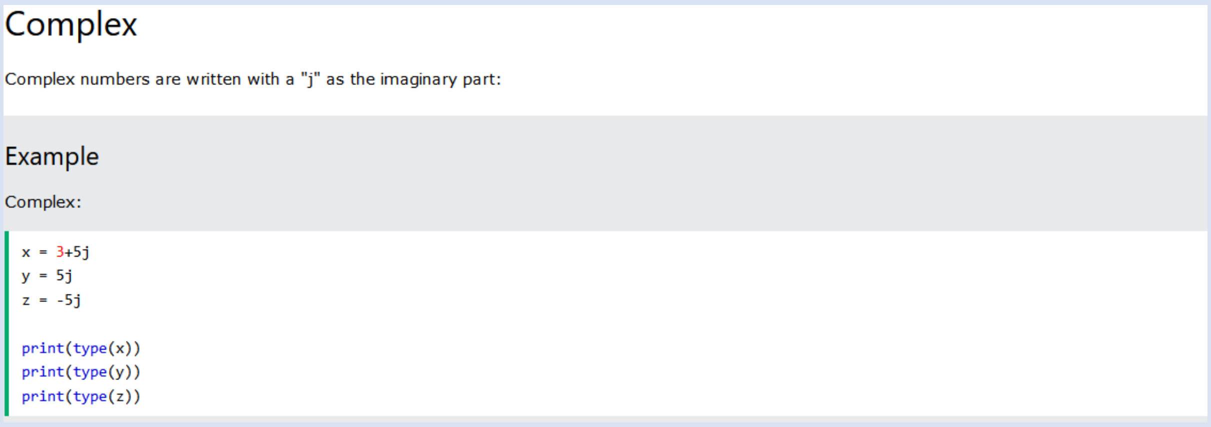


Float can also be scientific numbers with an "e" to indicate the power of 10.

Example

Floats:

```
x = 35e3
y = 12E4
z = -87.7e100

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



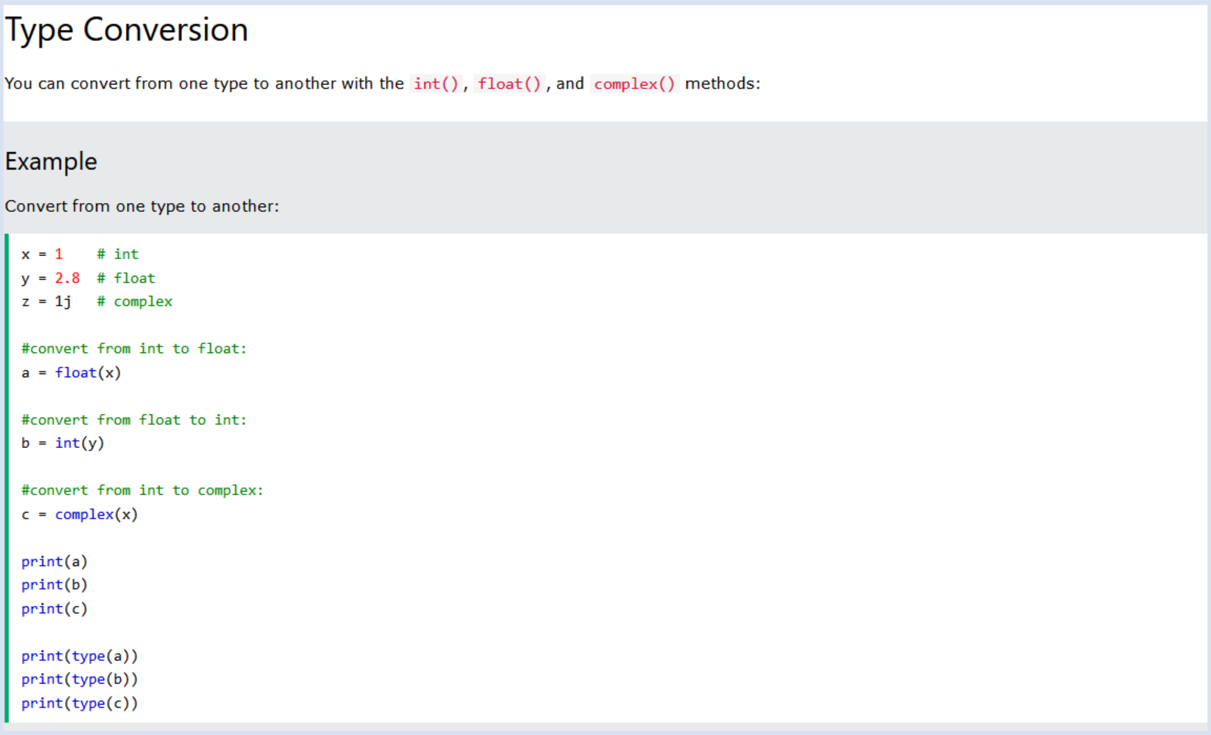
Complex

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))
```



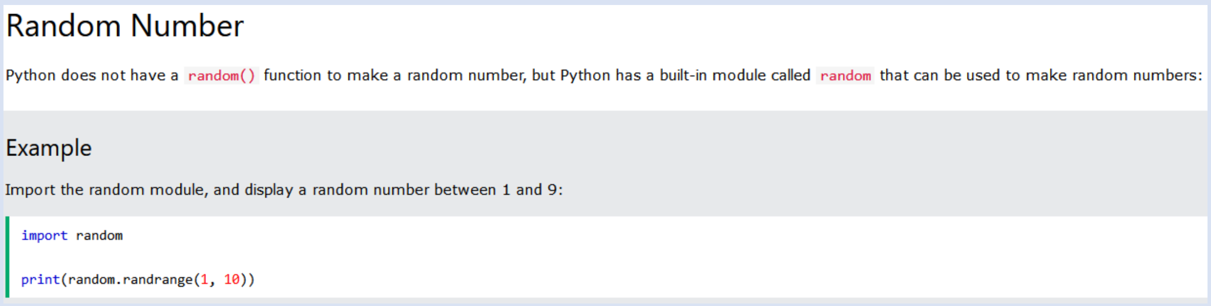
Type Conversion

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))
```



Random Number

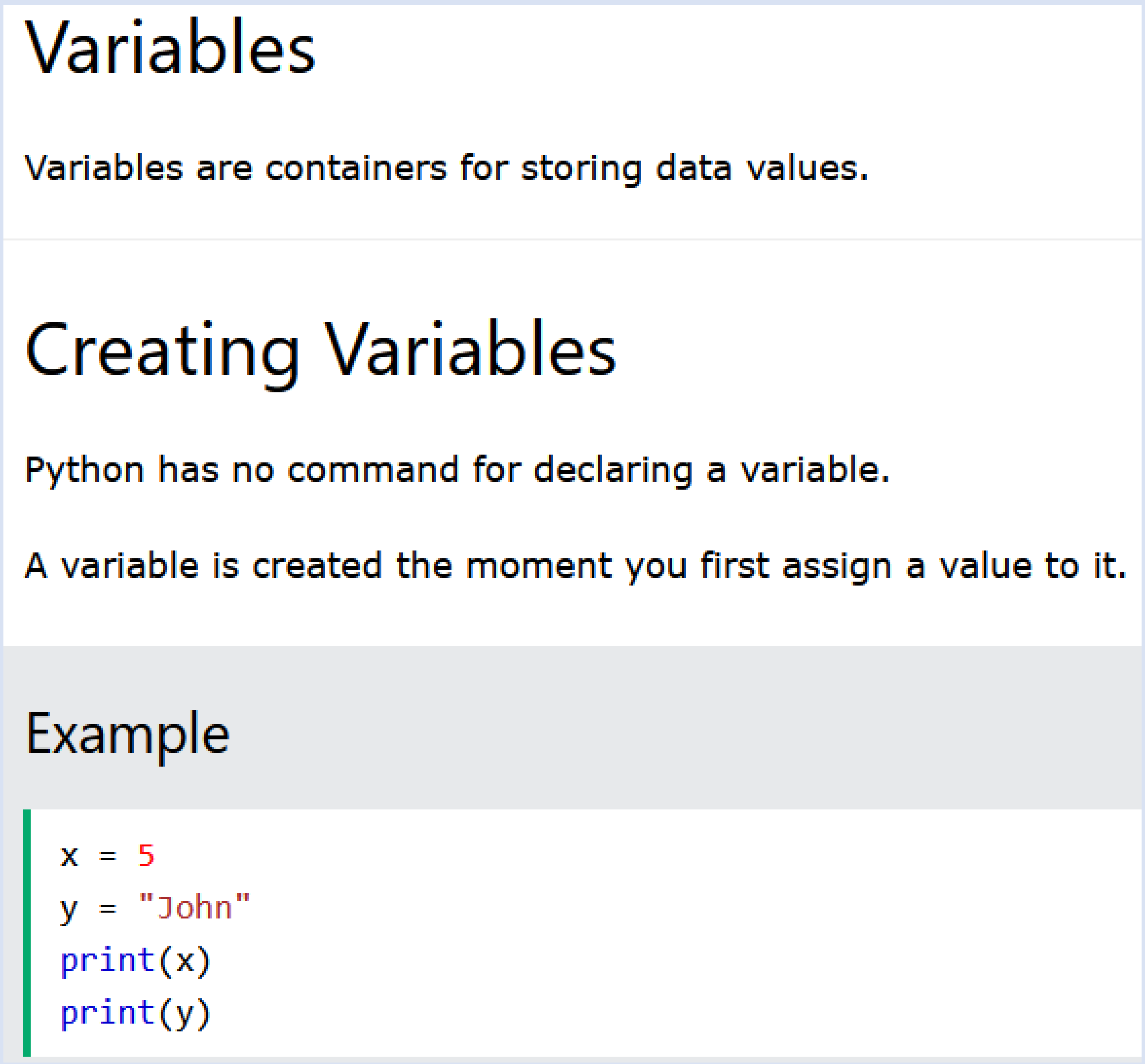
Python does not have a `random()` function to make a random number, but Python has a built-in module called `random` that can be used to make random numbers:

Example

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

```
import random

print(random.randrange(1, 10))
```

https://www.w3schools.com/python/python_variables.asp



Variables

Variables are containers for storing data values.

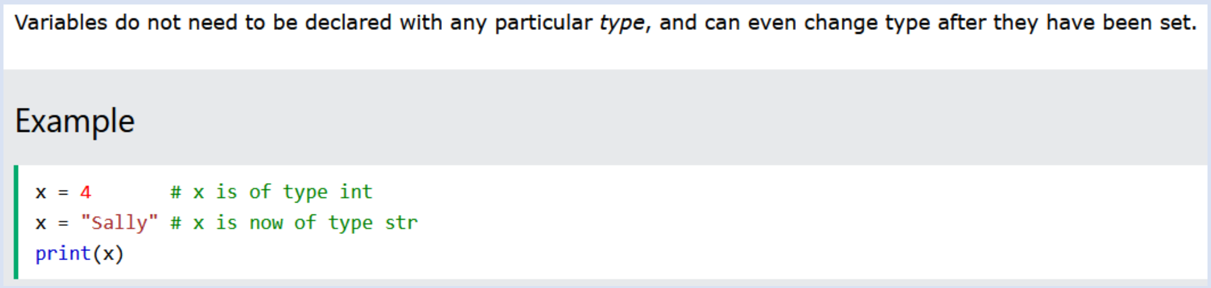
Creating Variables

Python has no command for declaring a variable.

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

Example

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



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

Example

```python
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Casting

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

Example

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

Get the Type

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

Example

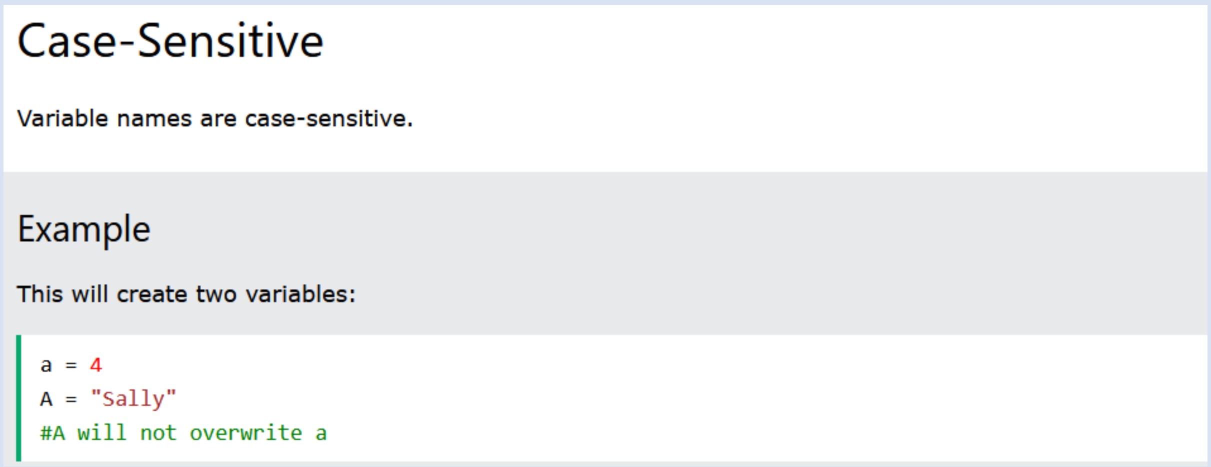
```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Single or Double Quotes?

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

Example

```
x = "John"
# is the same as
x = 'John'
```



Case-Sensitive

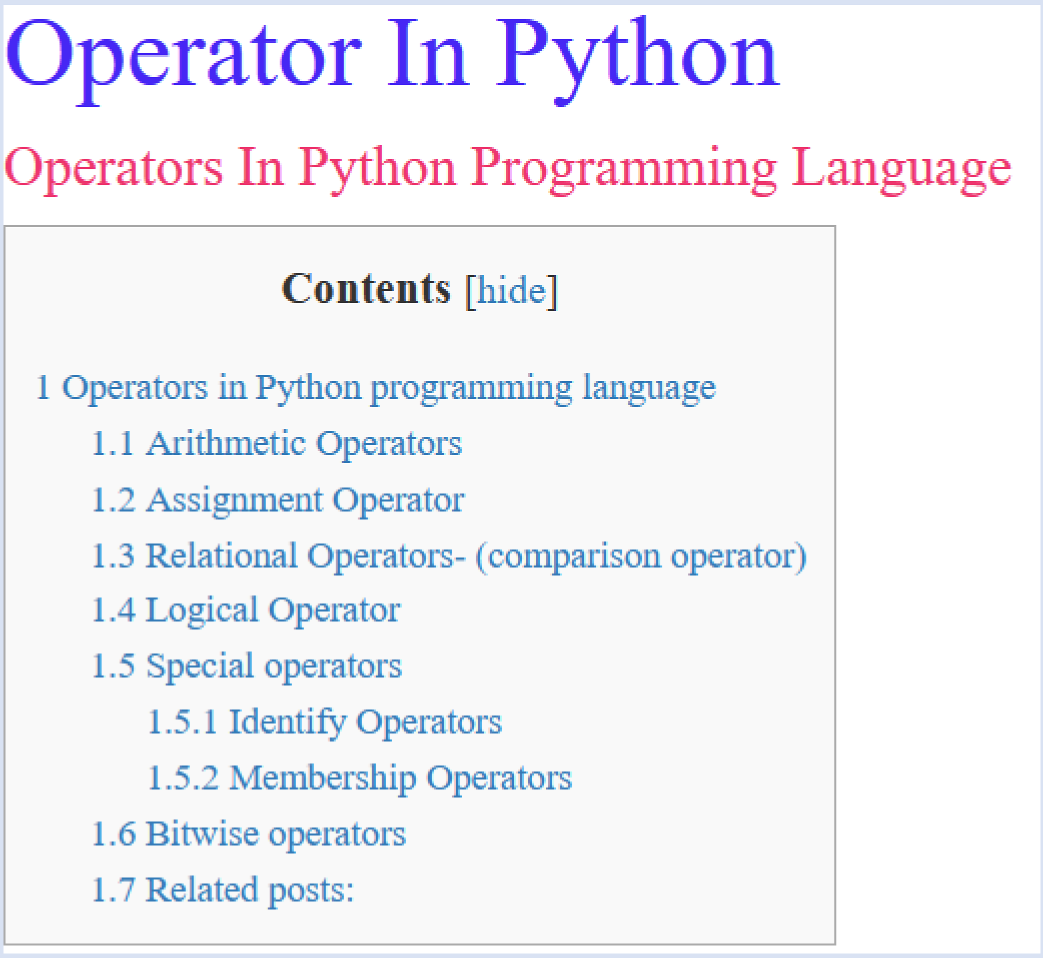
Variable names are case-sensitive.

Example

This will create two variables:

```
a = 4
A = "Sally"
#A will not overwrite a
```

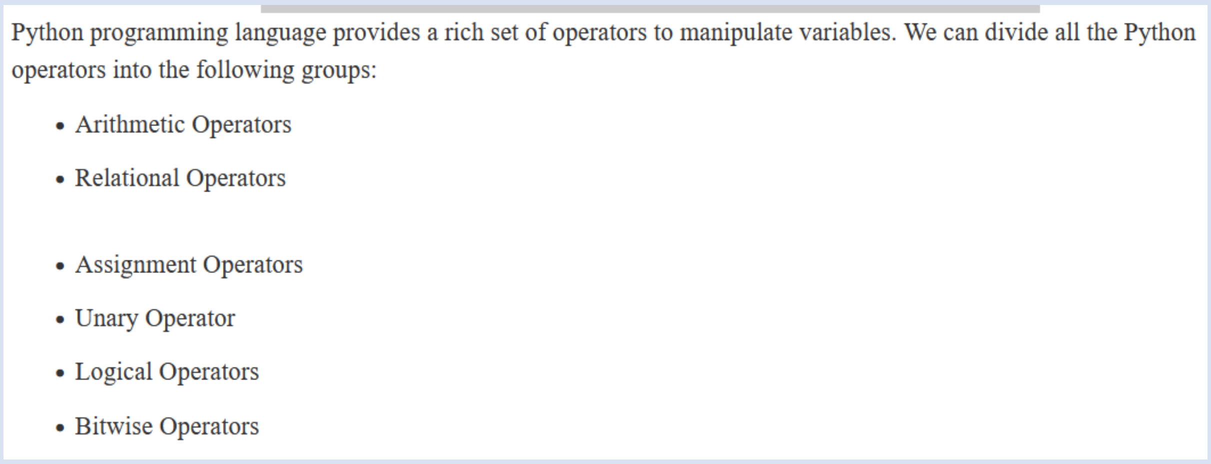
https://code4coding.com/operator-in-python-programming-language/



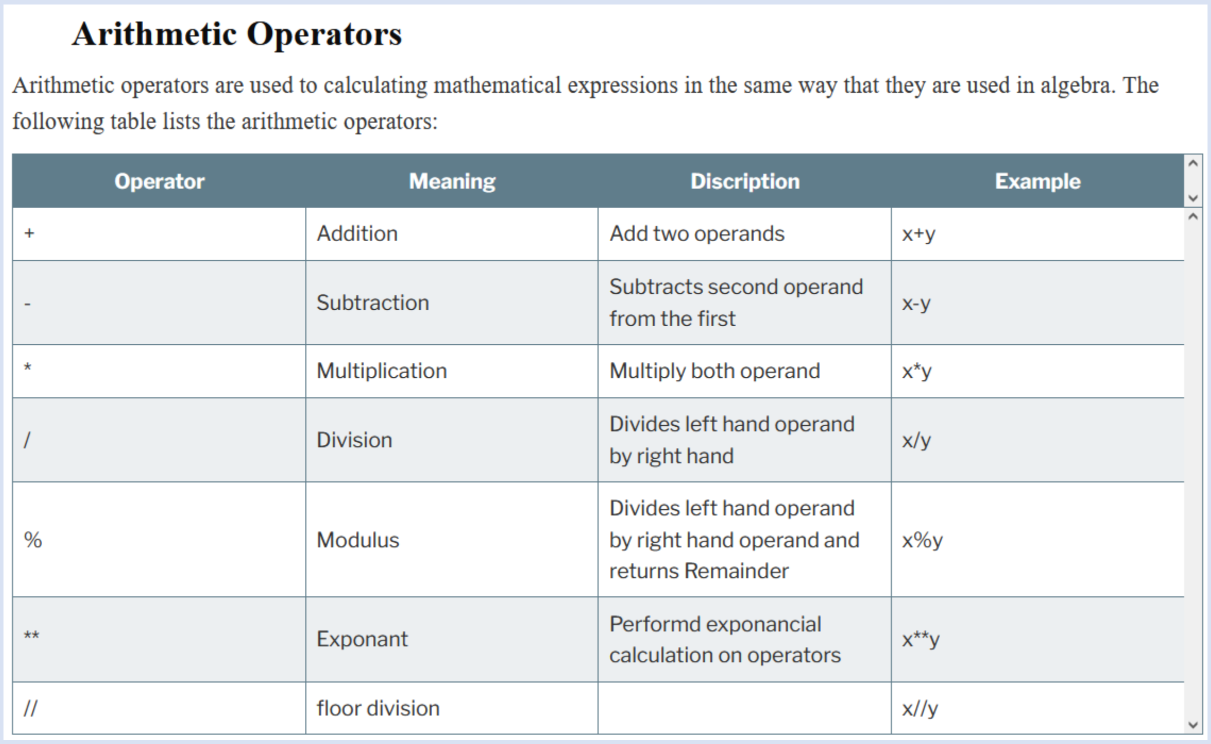
Operator In Python

Operators In Python Programming Language

Contents [hide]

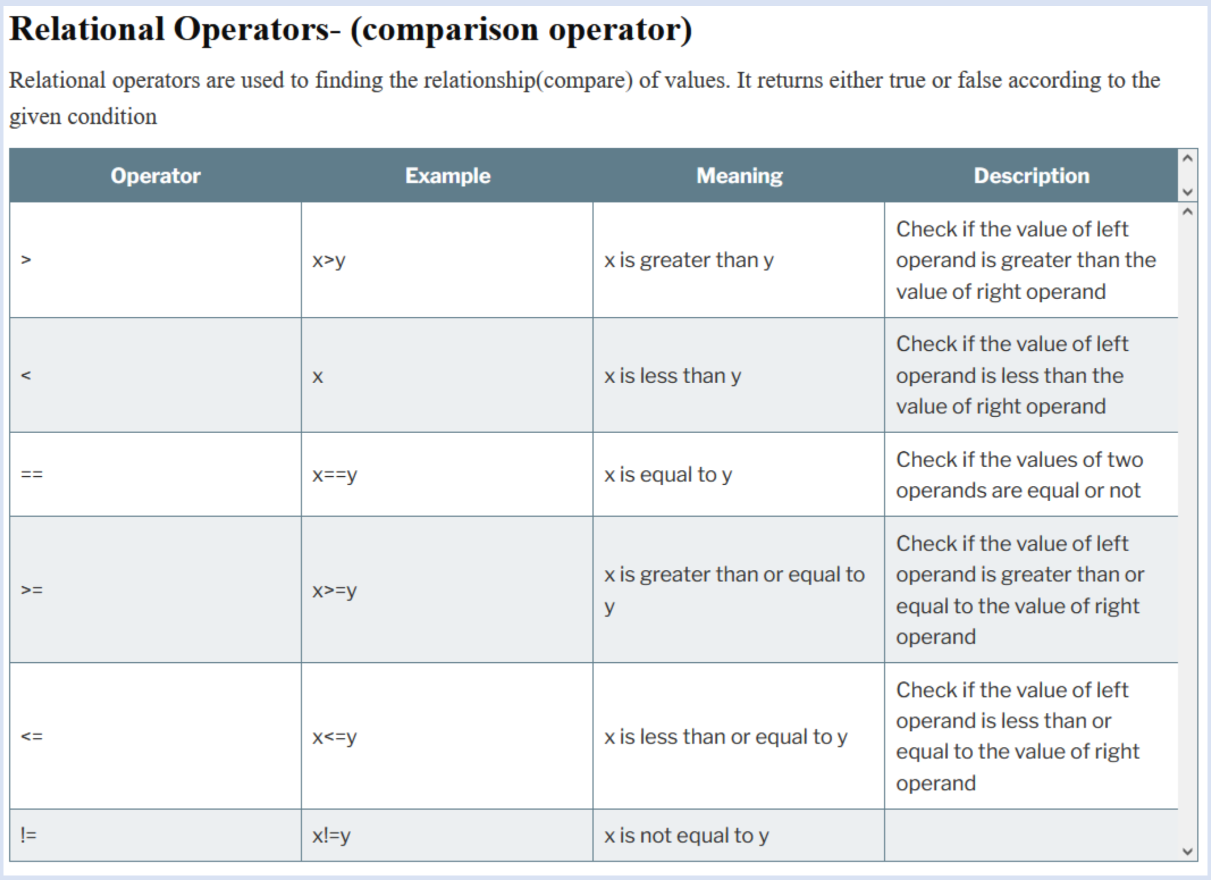

Python programming language provides a rich set of operators to manipulate variables. We can divide all the Python operators into the following groups:

- Arithmetic Operators
- Relational Operators
- Assignment Operators
- Unary Operator
- Logical Operators
- Bitwise Operators


Arithmetic Operators

Arithmetic operators are used to calculating mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

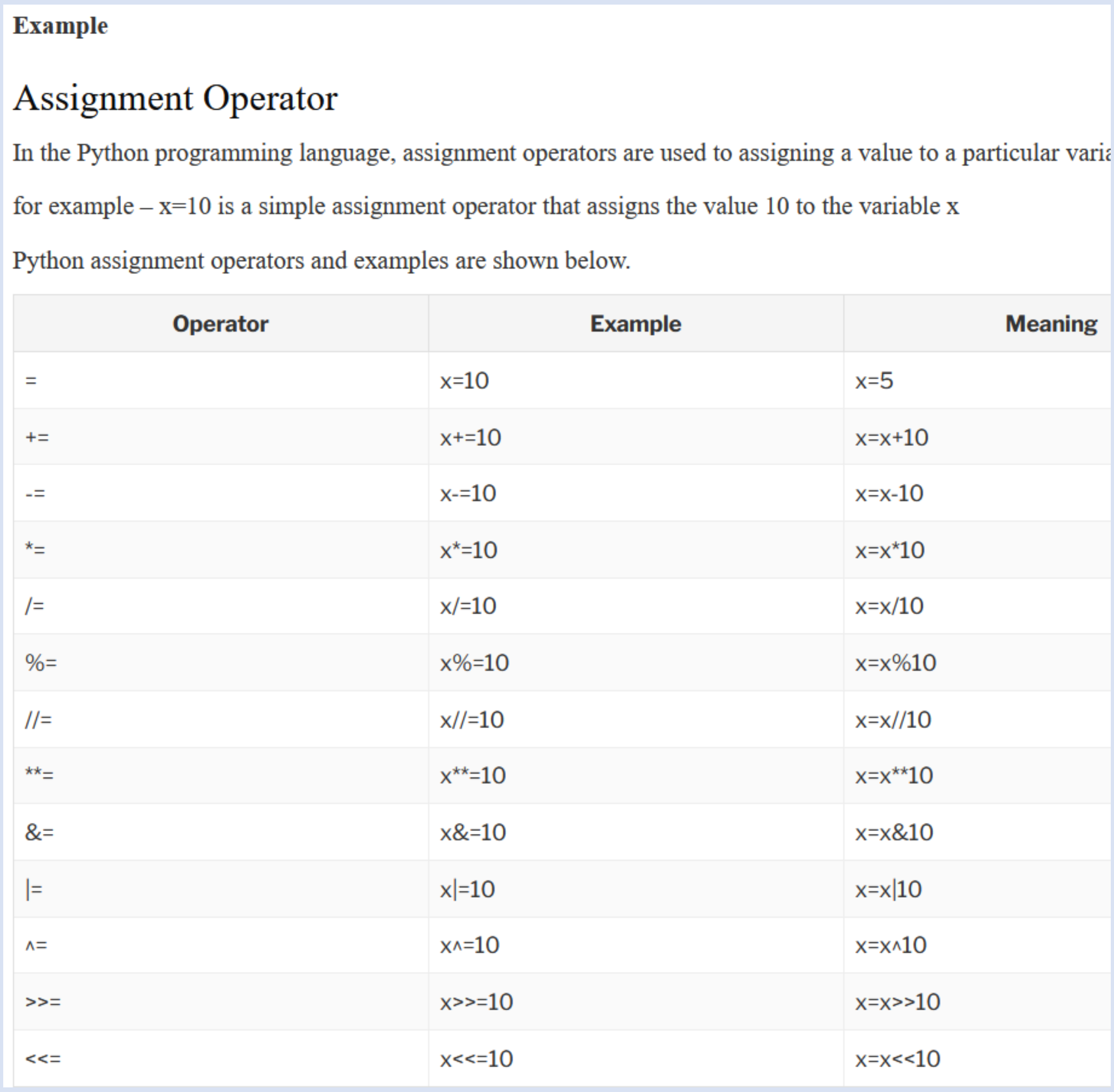
Operator	Meaning	Discription	Example
+	Addition	Add two operands	x+y
-	Subtraction	Subtracts second operand from the first	x-y
*	Multiplication	Multiply both operand	x*y
/	Division	Divides left hand operand by right hand	x/y
%	Modulus	Divides left hand operand by right hand operand and returns Remainder	x%y
**	Exponant	Performd exponancial calculation on operators	x**y
//	floor division		x//y



Relational Operators- (comparison operator)

Relational operators are used to finding the relationship(compare) of values. It returns either true or false according to the given condition

Operator	Example	Meaning	Description
>	x>y	x is greater than y	Check if the value of left operand is greater than the value of right operand
<	x	x is less than y	Check if the value of left operand is less than the value of right operand
==	x==y	x is equal to y	Check if the values of two operands are equal or not
>=	x>=y	x is greater than or equal to y	Check if the value of left operand is greater than or equal to the value of right operand
<=	x<=y	x is less than or equal to y	Check if the value of left operand is less than or equal to the value of right operand
!=	x!=y	x is not equal to y	



Example

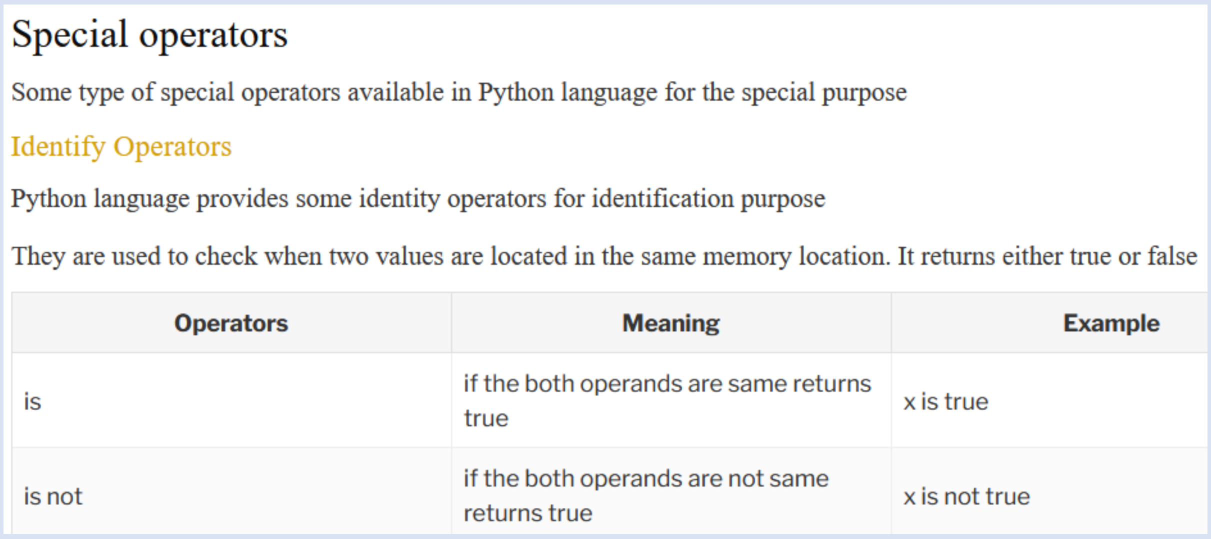
Assignment Operator

In the Python programming language, assignment operators are used to assigning a value to a particular varia

for example – x=10 is a simple assignment operator that assigns the value 10 to the variable x

Python assignment operators and examples are shown below.

Operator	Example	Meaning
=	x=10	x=5
+=	x+=10	x=x+10
-=	x-=10	x=x-10
*=	x*=10	x=x*10
/=	x/=10	x=x/10
%=	x%=10	x=x%10
//=	x//=10	x=x//10
**=	x**=10	x=x**10
&=	x&=10	x=x&10
|=	x|=10	x=x|10
^=	x^=10	x=x^10
>>=	x>>=10	x=x>>10
<<=	x<<=10	x=x<<10



Special operators

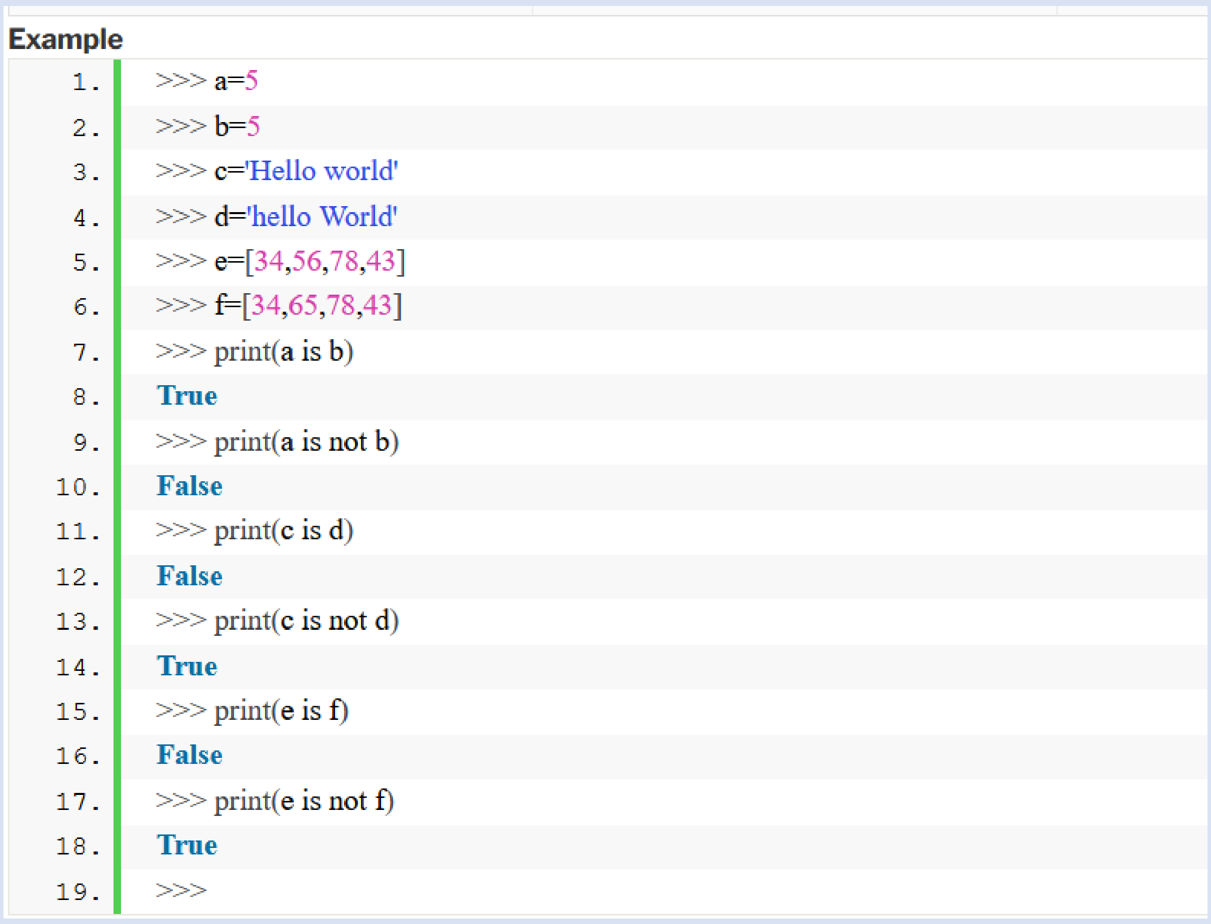
Some type of special operators available in Python language for the special purpose

Identify Operators

Python language provides some identity operators for identification purpose

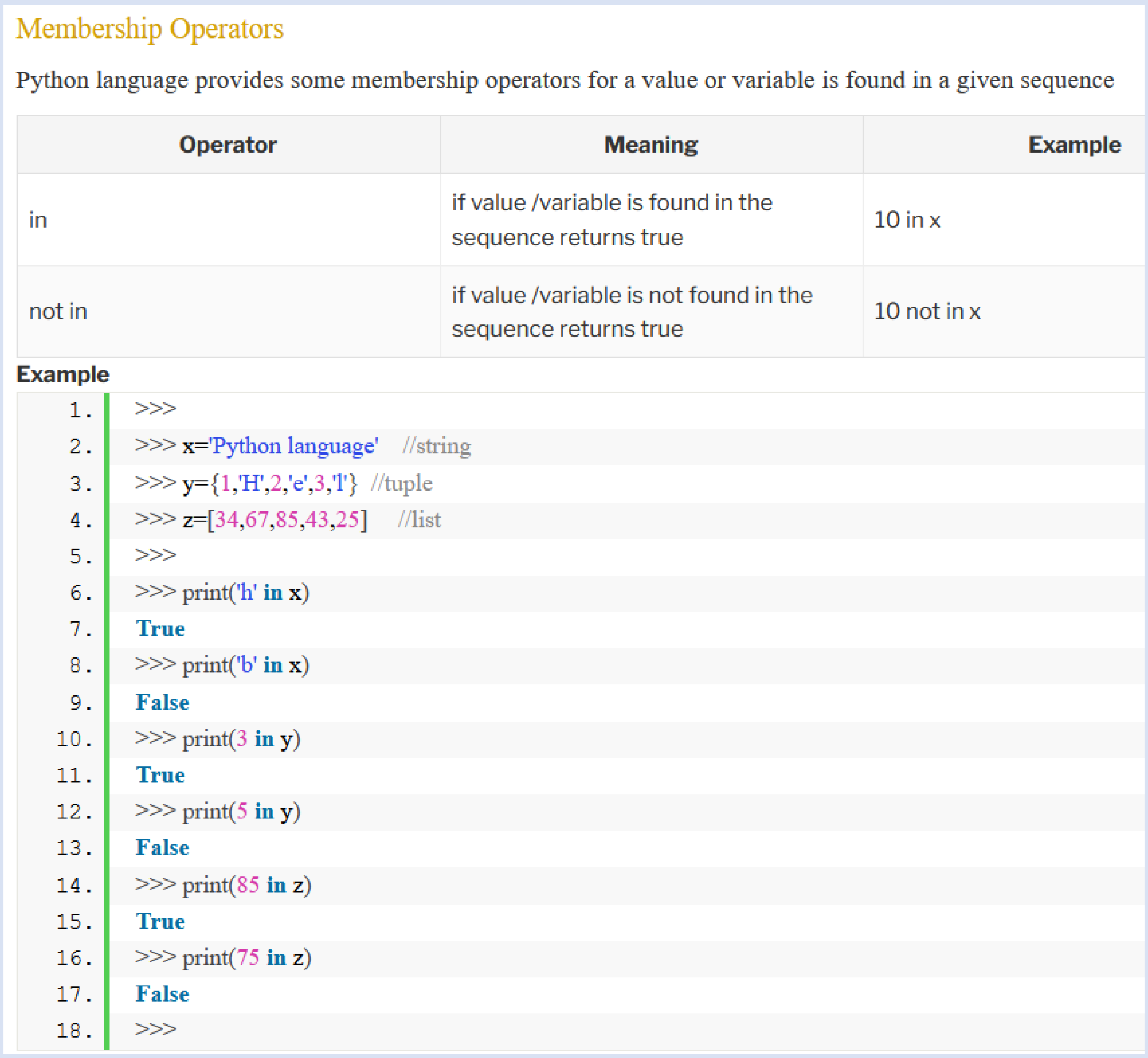
They are used to check when two values are located in the same memory location. It returns either true or false

Operators	Meaning	Example
is	if the both operands are same returns true	x is true
is not	if the both operands are not same returns true	x is not true



Example

```
>>> a=5
>>> b=5
>>> c='Hello world'
>>> d='hello World'
>>> e=[34,56,78,43]
>>> f=[34,65,78,43]
>>> print(a is b)
True
>>> print(a is not b)
False
>>> print(c is d)
False
>>> print(c is not d)
True
>>> print(e is f)
False
>>> print(e is not f)
True
>>>
```



Membership Operators

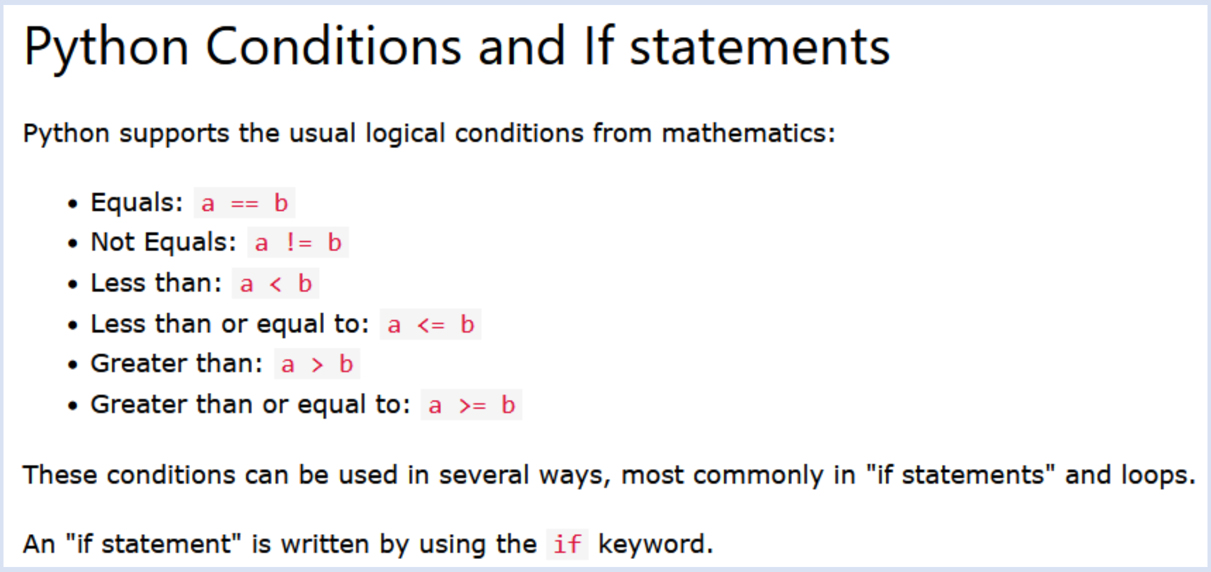
Python language provides some membership operators for a value or variable is found in a given sequence

Operator	Meaning	Example
in	if value /variable is found in the sequence returns true	10 in x
not in	if value /variable is not found in the sequence returns true	10 not in x

Example

```
>>>
>>> x='Python language'    //string
>>> y={1,'H',2,'e',3,'l'}  //tuple
>>> z=[34,67,85,43,25]     //list
>>>
>>> print('h' in x)
True
>>> print('b' in x)
False
>>> print(3 in y)
True
>>> print(5 in y)
False
>>> print(85 in z)
True
>>> print(75 in z)
False
>>>
```

https://www.w3schools.com/python/python_conditions.asp



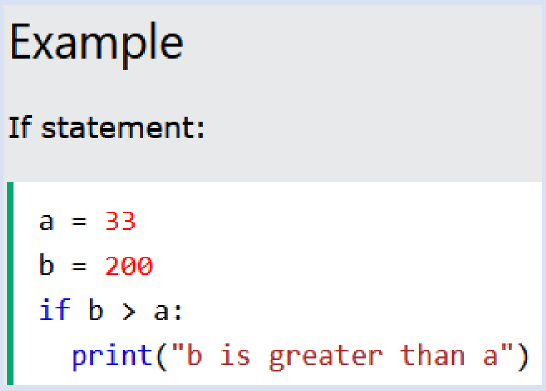
Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

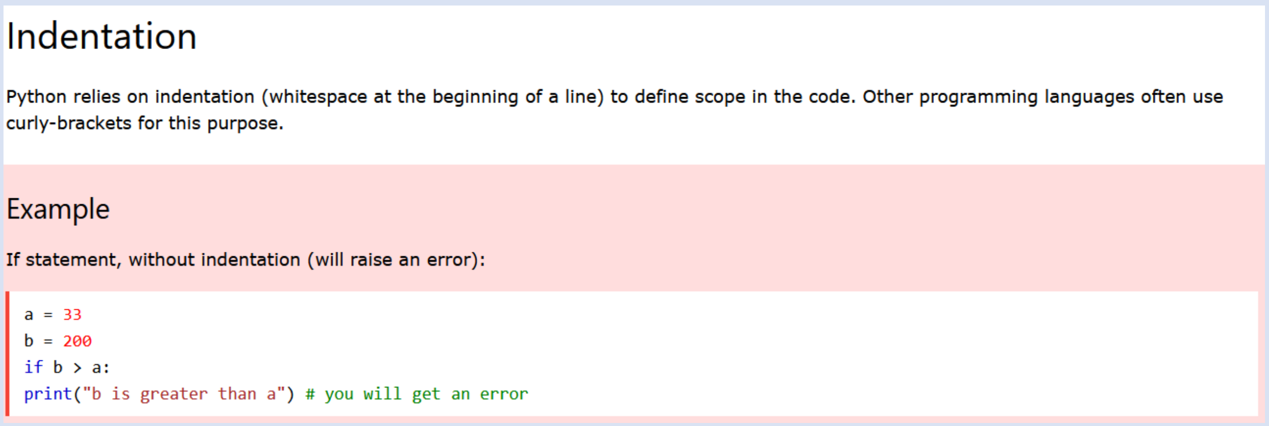
An "if statement" is written by using the `if` keyword.



Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```



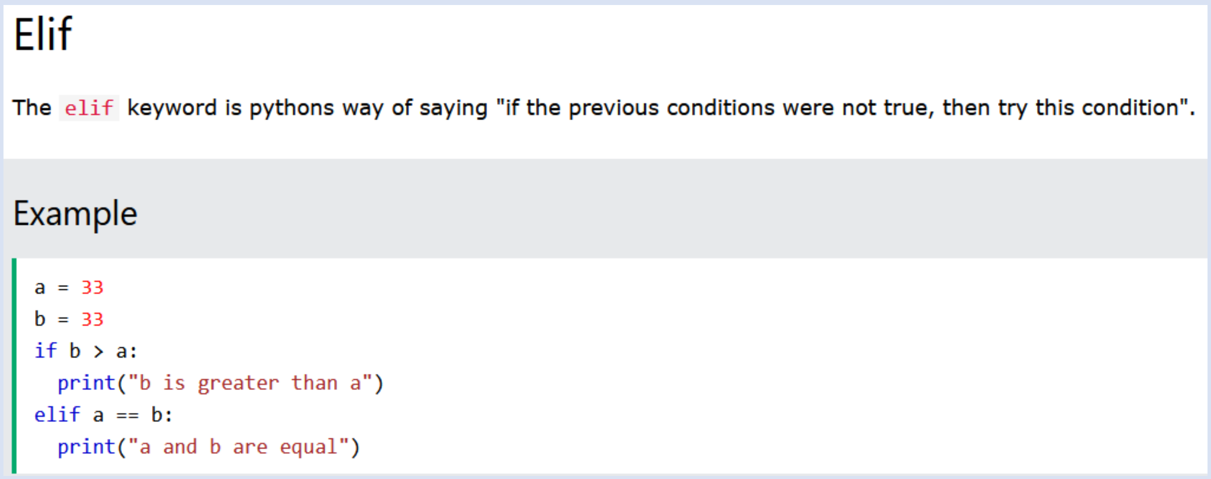
Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

Example

If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

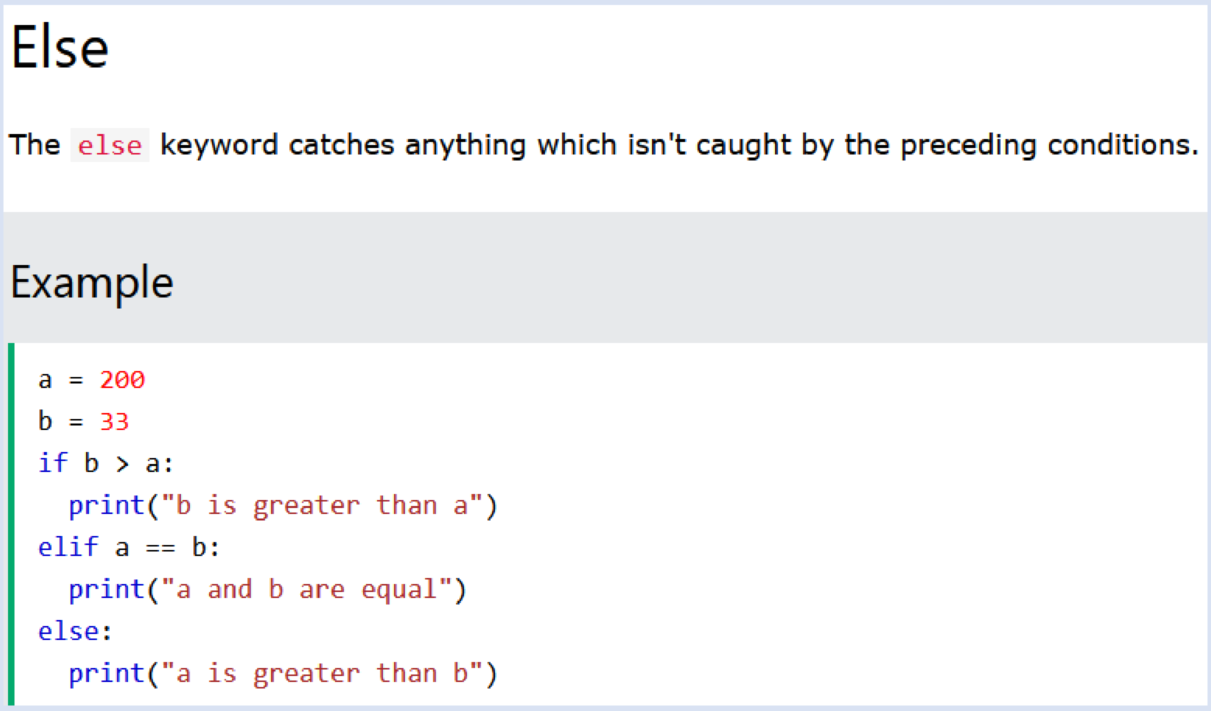


Elif

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

Example

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```



Else

The `else` keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

You can also have an `else` without the `elif`:

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

Example

One line if statement:

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

Example

One line if else statement:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

You can also have multiple else statements on the same line:

Example

One line if else statement, with 3 conditions:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

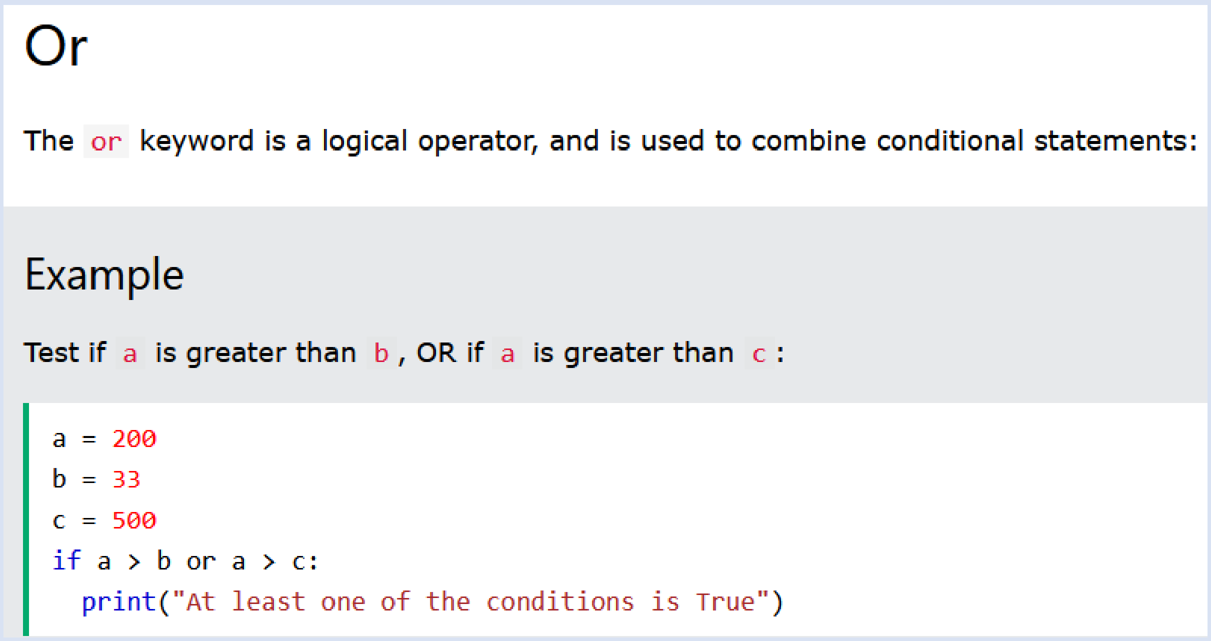
And

The `and` keyword is a logical operator, and is used to combine conditional statements:

Example

Test if `a` is greater than `b`, AND if `c` is greater than `a`:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```



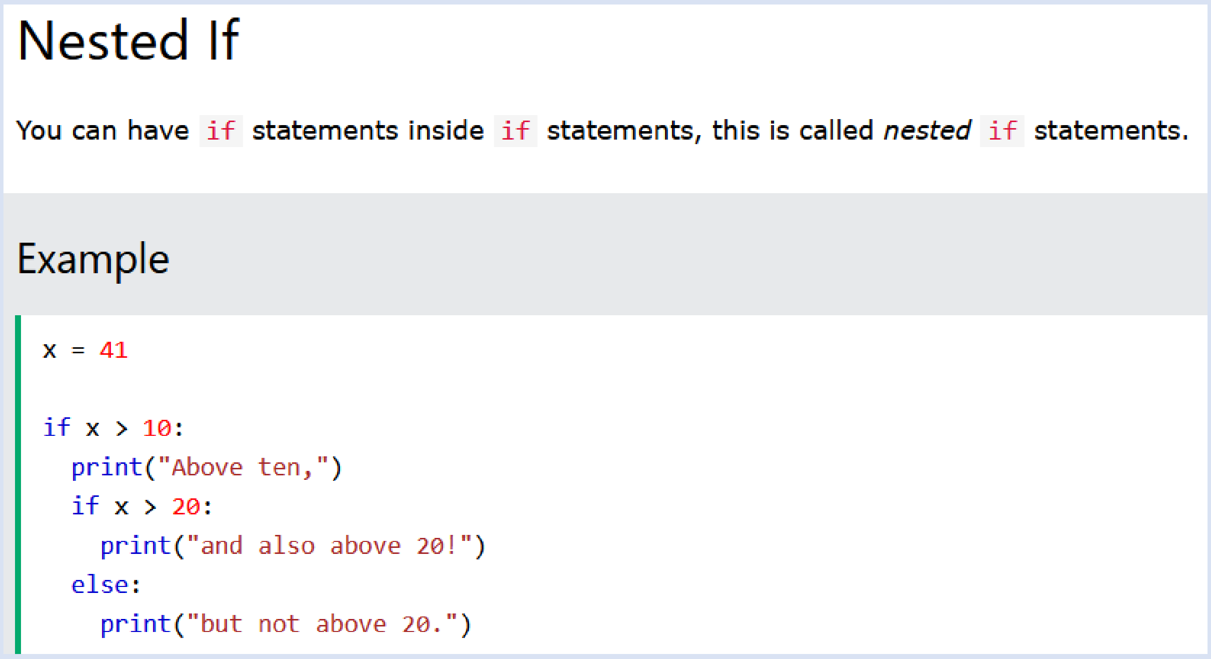
Or

The `or` keyword is a logical operator, and is used to combine conditional statements:

Example

Test if `a` is greater than `b`, OR if `a` is greater than `c`:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

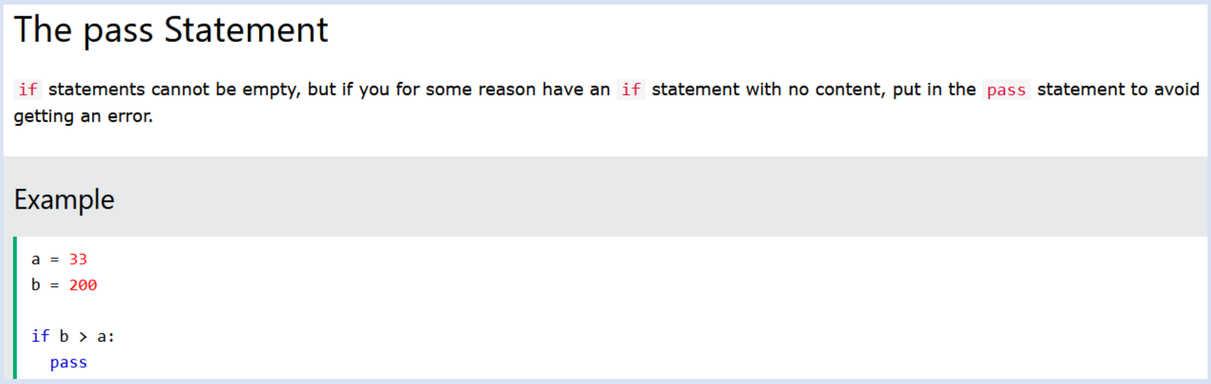


Nested If

You can have `if` statements inside `if` statements, this is called *nested* `if` statements.

Example

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```



The pass Statement

`if` statements cannot be empty, but if you for some reason have an `if` statement with no content, put in the `pass` statement to avoid getting an error.

Example

```
a = 33
b = 200

if b > a:
  pass
```

Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Note: remember to increment i, or else the loop will continue forever.

The `while` loop requires relevant variables to be ready, in this example we need to define an indexing variable, `i`, which we set to 1.

The break Statement

With the `break` statement we can stop the loop even if the while condition is true:

Example

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

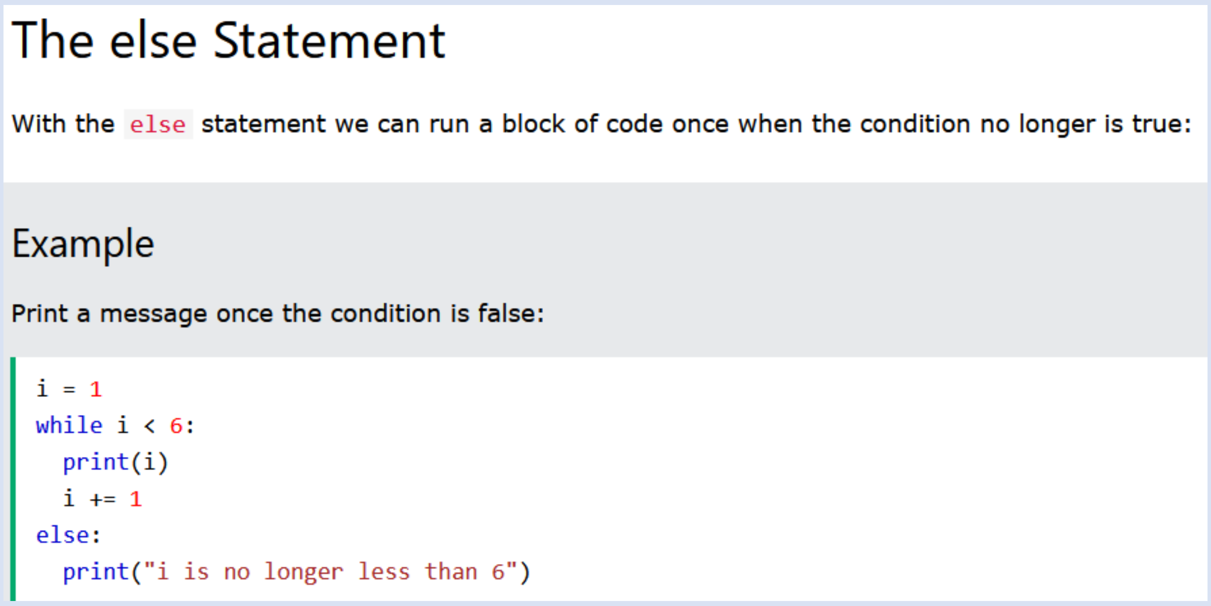
The continue Statement

With the `continue` statement we can stop the current iteration, and continue with the next:

Example

Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```



The else Statement

With the `else` statement we can run a block of code once when the condition no longer is true:

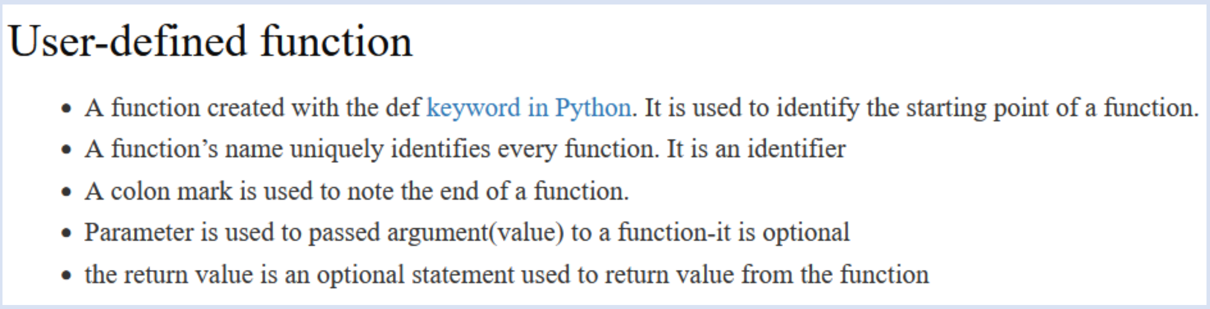
Example

Print a message once the condition is false:

```python
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

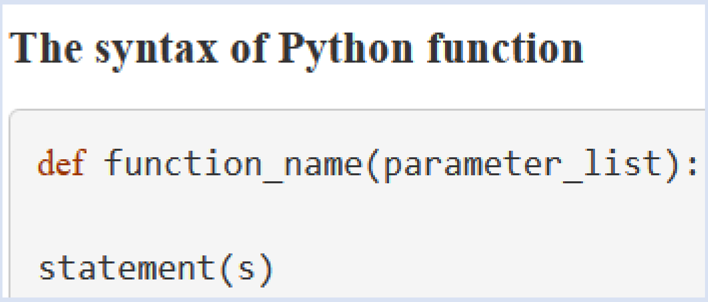
https://code4coding.com/function-in-python-programming-language/

Functions in Python



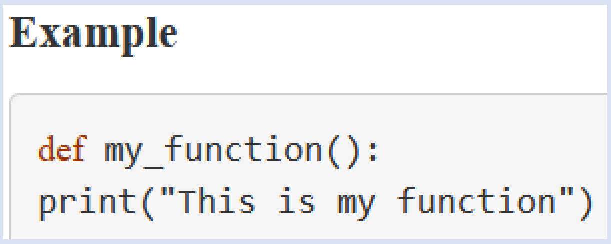
User-defined function

- A function created with the def keyword in Python. It is used to identify the starting point of a function.
- A function's name uniquely identifies every function. It is an identifier
- A colon mark is used to note the end of a function.
- Parameter is used to passed argument(value) to a function-it is optional
- the return value is an optional statement used to return value from the function



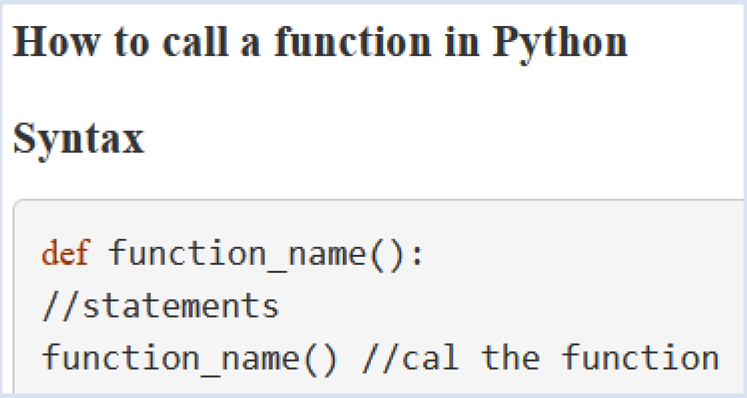
The syntax of Python function

```
def function_name(parameter_list):

statement(s)
```



Example

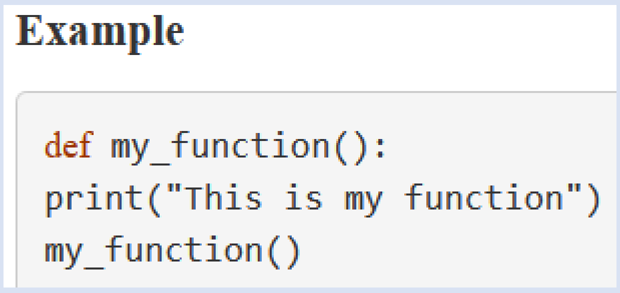
```
def my_function():
print("This is my function")
```



How to call a function in Python

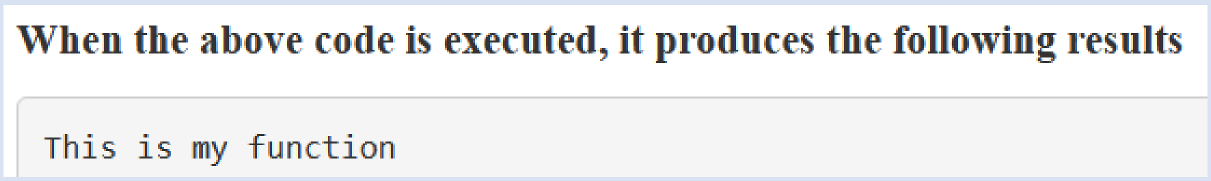
Syntax

```
def function_name():
//statements
function_name() //cal the function
```



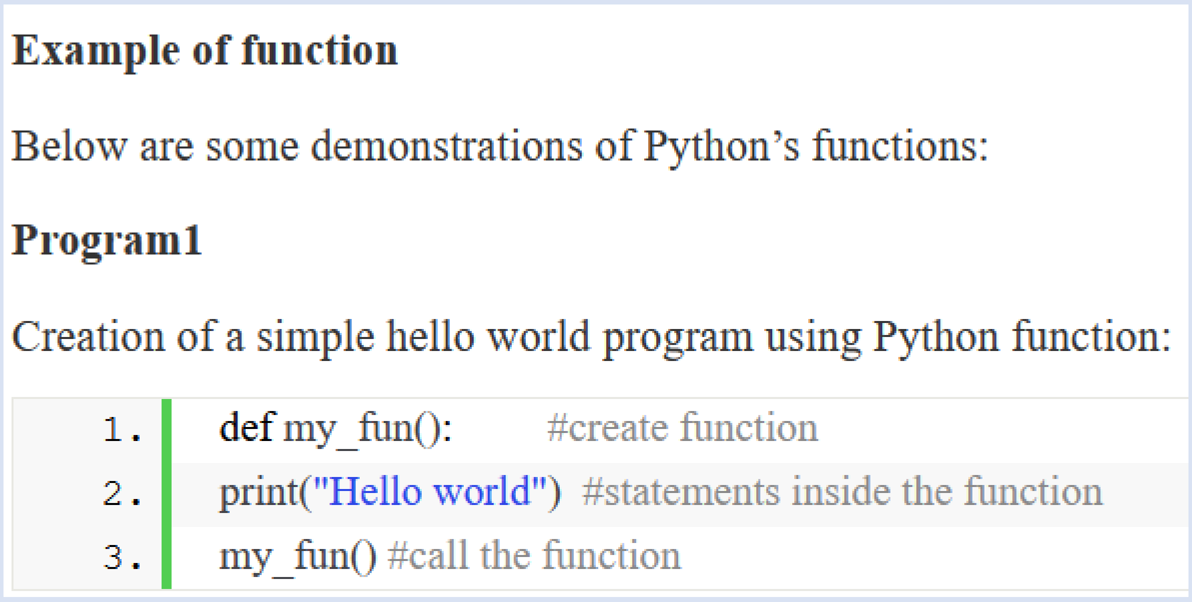
Example

```
def my_function():
print("This is my function")
my_function()
```



When the above code is executed, it produces the following results

```
This is my function
```



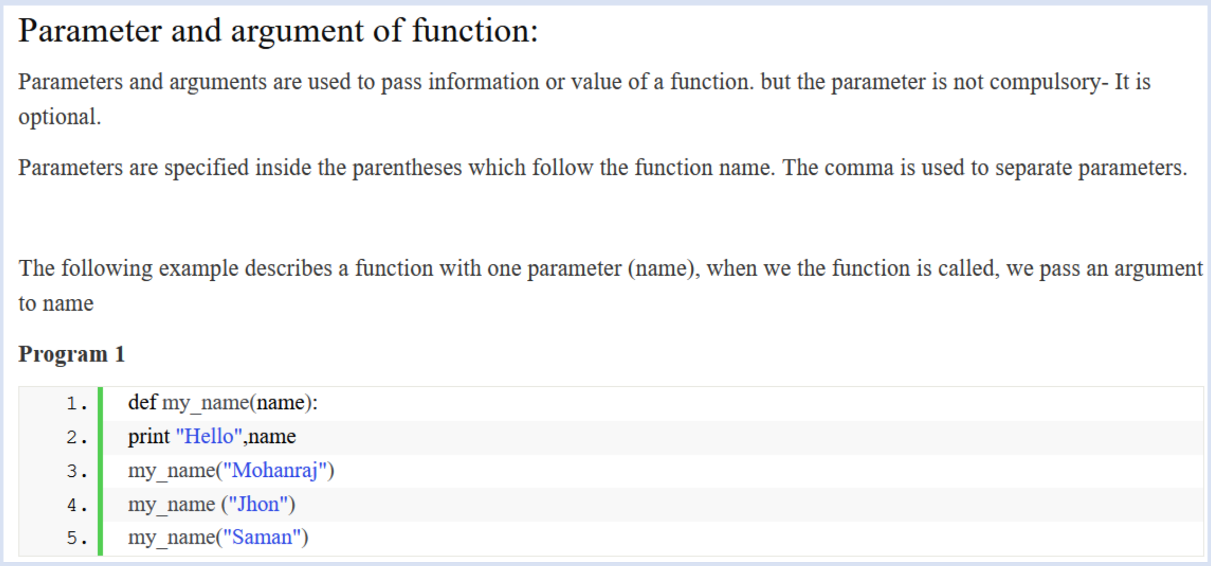
Example of function

Below are some demonstrations of Python’s functions:

Program1

Creation of a simple hello world program using Python function:

```
def my_fun():        #create function
print("Hello world")  #statements inside the function
my_fun() #call the function
```



Parameter and argument of function:

Parameters and arguments are used to pass information or value of a function. but the parameter is not compulsory- It is optional.

Parameters are specified inside the parentheses which follow the function name. The comma is used to separate parameters.

The following example describes a function with one parameter (name), when we the function is called, we pass an argument to name

Program 1

```
def my_name(name):
print "Hello",name
my_name("Mohanraj")
my_name ("Jhon")
my_name("Saman")
```

Here, name is a String value as a parameter.

In this program, we can pass a single parameter every time function is called.

When the above code is executed, it produces the following result

```
Hello Mohanaraj
Hello santhiran
Hello Kumar
```

Default parameter value

The following example describes how to use a default function in Python language

```
def my_Function(age=20):
    print("my age is :"+str(age))

my_Function(35)
my_Function(42)
my_Function()
my_Function(16)
```

When the above code is executed, it produces the following result

```
my age is :35
my age is :42
my age is :20
my age is :16
```

When we call the function with the parameter, it uses the argument value

When we call the function without parameter, it uses the default value

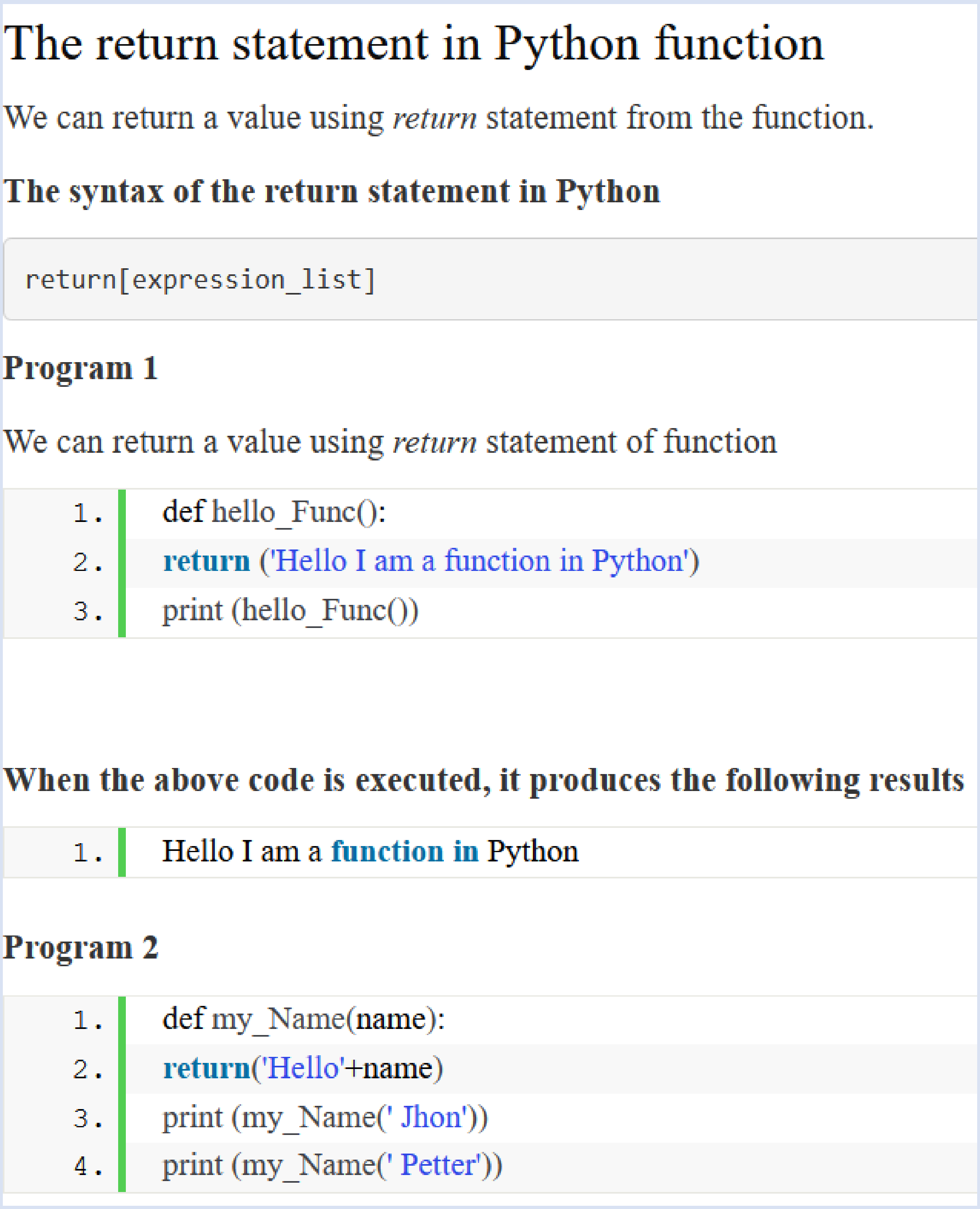
Program 3

This program passes string concatenations along with string arguments.

```
def display(fname,lname,age,gender):
    #create a function named display with parameter
    print ("My first name is :"+fname)
    print ("My last name is :"+lname)
    print ("My age is :"+str(age))
    print ("My sex is :"+gender)

display("Jhon","Petter",33,"Male")
display("Siyani","Smith",36,"Female")
#call the function with argument
```

When the above code is executed, it produced the following result

```
My first name is :Jhon
My last name is :Petter
My age is :33
My sex is :Male
My first name is :Siyani
My last name is :Smith
My age is :36
My sex is :Female
```



The return statement in Python function

We can return a value using *return* statement from the function.

The syntax of the return statement in Python

```
return[expression_list]
```

Program 1

We can return a value using *return* statement of function

```
def hello_Func():
return ('Hello I am a function in Python')
print (hello_Func())
```

When the above code is executed, it produces the following results

```
Hello I am a function in Python
```

Program 2

```
def my_Name(name):
return('Hello'+name)
print (my_Name(' Jhon'))
print (my_Name(' Petter'))
```

When the above code is executed, it produces the following results

```
Hello I am a function in Python
```

Program 2

```
def my_Name(name):
return('Hello'+name)
print (my_Name(' Jhon'))
print (my_Name(' Petter'))
```

When the above code is executed, it produced the following result

```
Hello Jhon
Hello petter
```

program 3

```
def rtn_Ex():
   return 10
print rtn_Ex()
```

When the above code was executed, it produced the following result

```
10
```

Program 5

```
def addthree_num(x,y,z):
   #define a function with 3 parameter
   c=x+y+z
   return(c)
  #return value
result=addthree_num(5,7,9)
result1=addthree_num(54,57,89)
result2=addthree_num(356,877,792)
print result
print result1
print result2
```

When the above code was executed, it produced the following result

```
21
200
2025
```

Program 6

```
def def_Func(num1,num2=5):
    print num1,num2
def_Func(1)
def_Func(2)
def_Func(6)
```

When the above code was executed, it produced the following result

```
1 5
2 5
6 5
```

In the above program, 5 is a default value

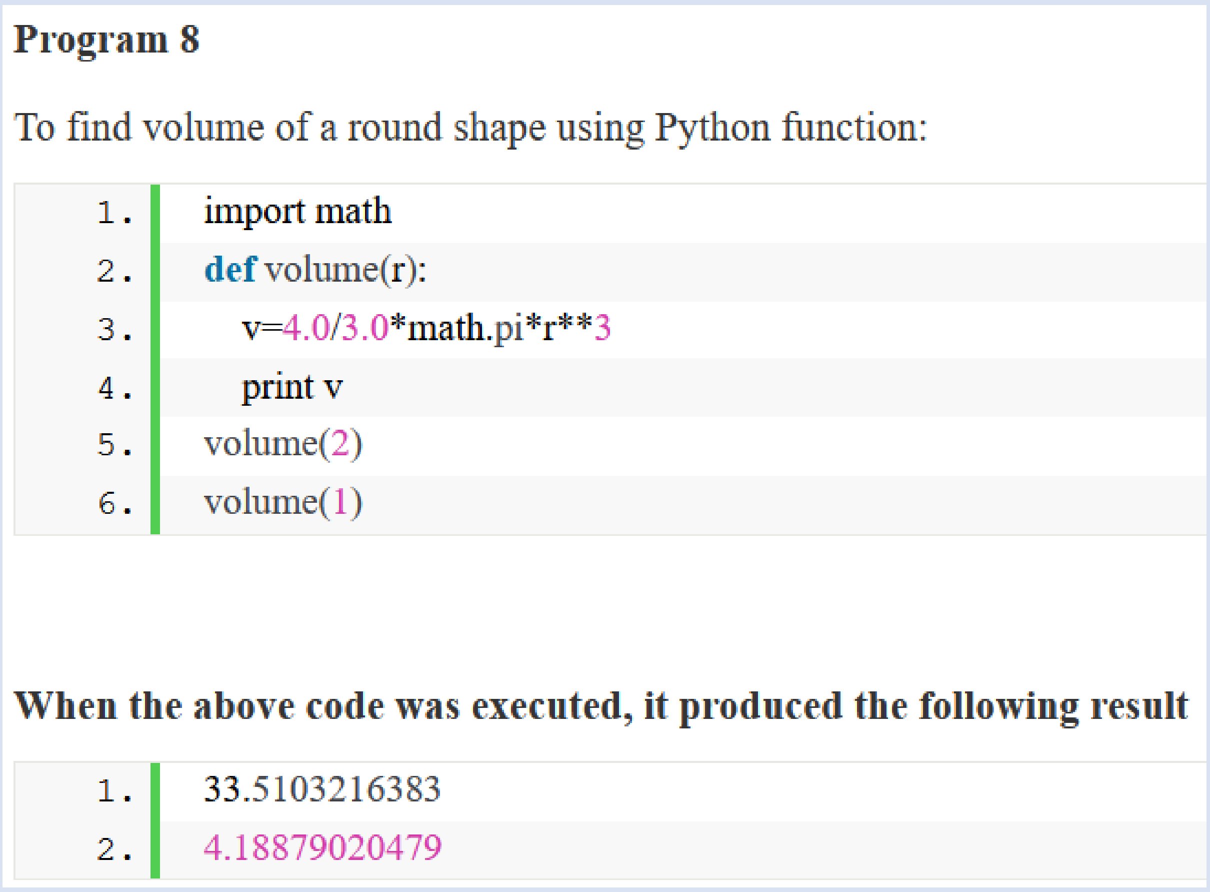
In the above program, 5 is a default value

Program 7

```
def student (name,marks, avg,age=16,school="Ananda college"):
    print name,marks,avg,age,school
student("Jhon",76,34.6)
student("Depak",48,23.3)
student("Saman",98,64.2)
```

When the above code is executed, it produced the following result

```
Jhon 76 34.6 16 Ananda college
Depak 48 23.3 16 Ananda college
Saman 98 64.2 16 Ananda college
```



Program 8

To find volume of a round shape using Python function:

```
import math
def volume(r):
   v=4.0/3.0*math.pi*r**3
   print v
volume(2)
volume(1)
```

When the above code was executed, it produced the following result

```
33.5103216383
4.18879020479
```