PYTHON NUMBERS

There are three built-in number types available in Python:

- integers (int)
- floating point numbers (float)
- complex numbers

integers (int)

- any number without the provision to store a fractional part is an integer;
- can be zero, positive or a negative whole number.

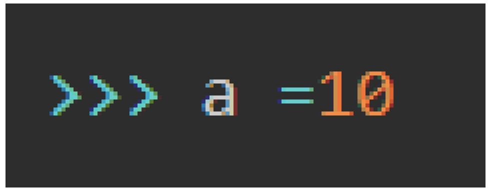
integers (int)

There are three ways to form an integer object:

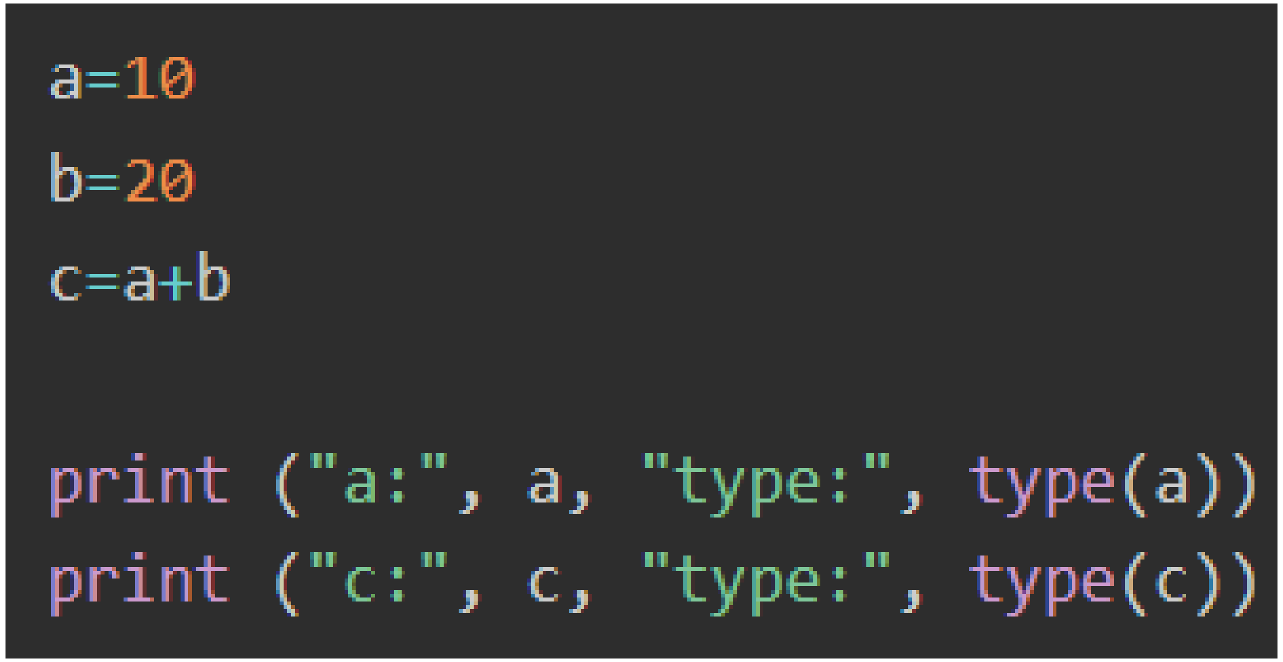
(a) literal representation

(b) any expression evaluating to an integer

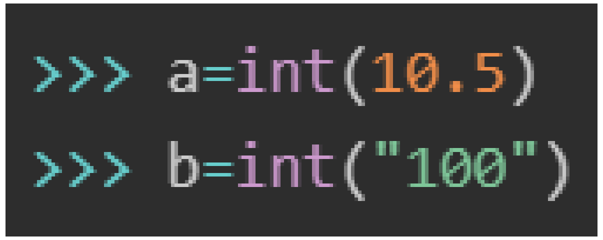
(c) using int() function



```
>>> a =10
```



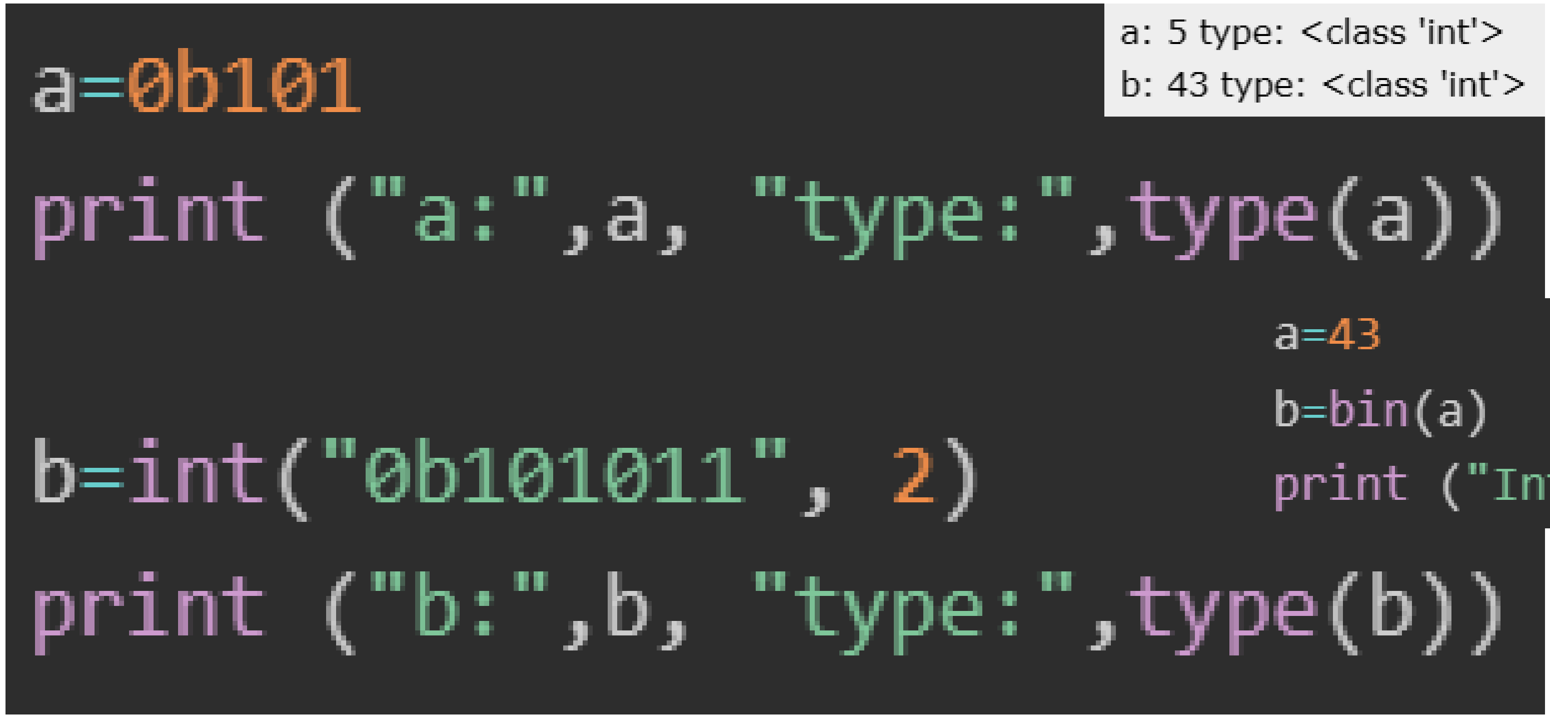
```
a=10
b=20
c=a+b

print ("a:", a, "type:", type(a))
print ("c:", c, "type:", type(c))
```



```
>>> a=int(10.5)
>>> b=int("100")
```

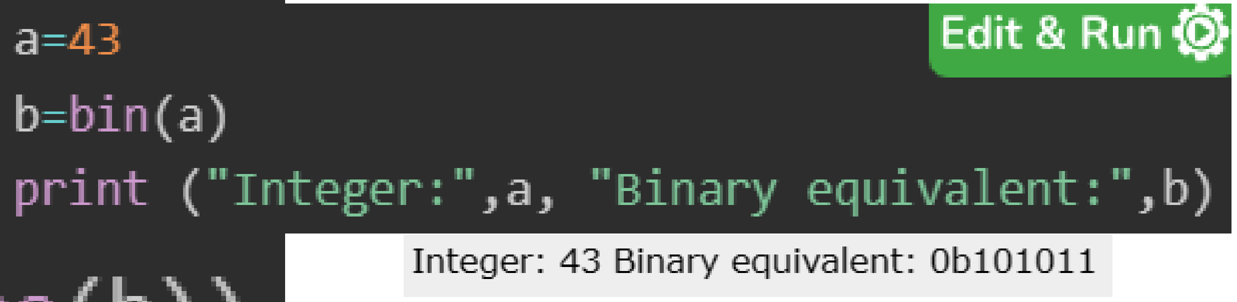
integers (int)

can represent as a binary, octal or Hexa-decimal number. However, internally the object is stored as an integer. binary digits (1 and 0) and prefixed with **0b** is a binary number.



```
a=0b101
print ("a:",a, "type:",type(a))

b=int("0b101011", 2)
print ("b:",b, "type:",type(b))
```

```
a: 5 type: <class 'int'>
b: 43 type: <class 'int'>
```

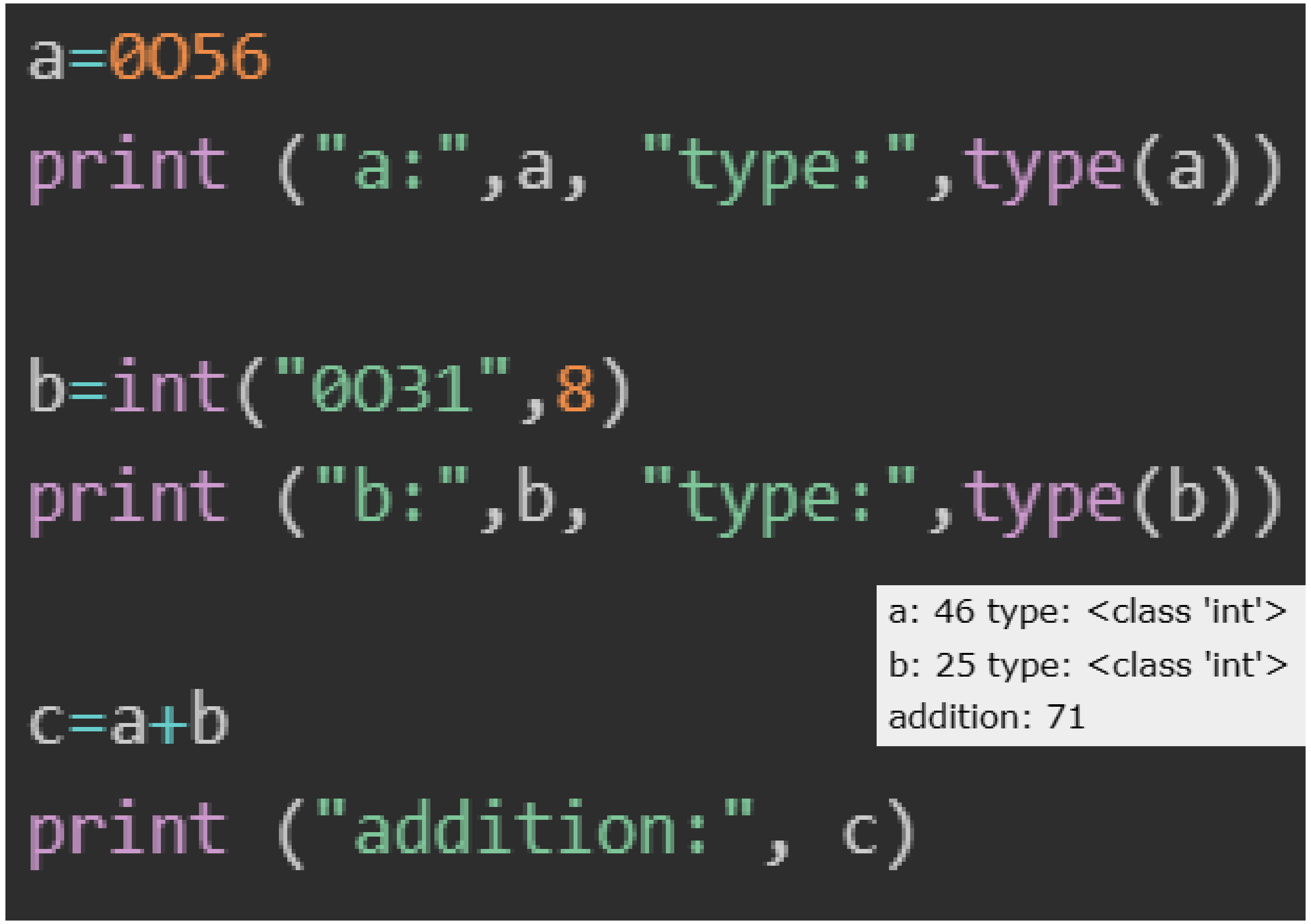


```
a=43
b=bin(a)
print ("Integer:",a, "Binary equivalent:",b)
```

```
Integer: 43 Binary equivalent: 0b101011
```

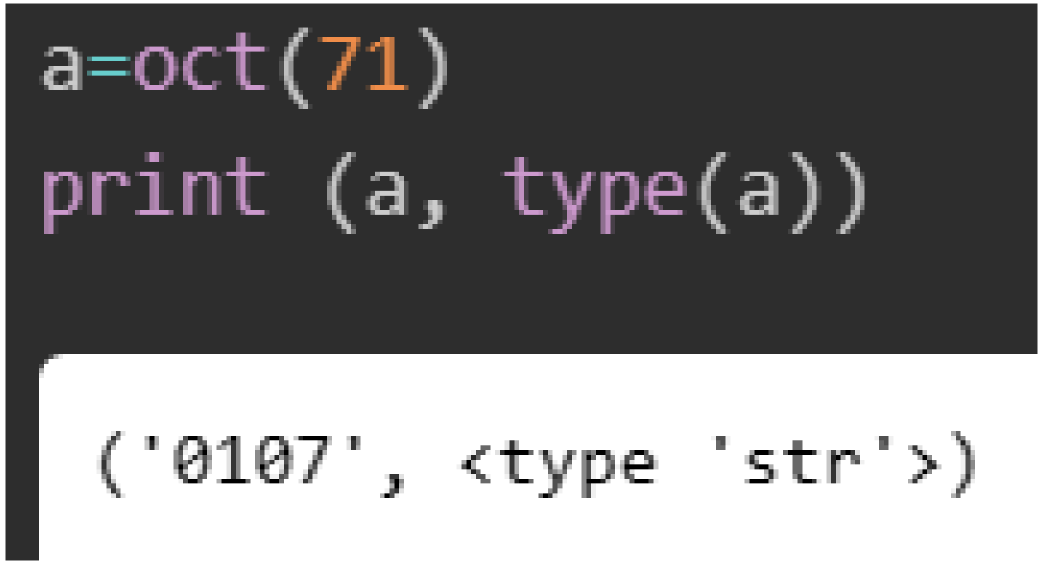
integers (int)

OCTAL - it needs to be prefixed by 0o (lowercase O) or 0O (uppercase O).



```
a=0O56
print ("a:",a, "type:",type(a))

b=int("0O31",8)
print ("b:",b, "type:",type(b))

c=a+b
print ("addition:", c)
```

```
a: 46 type: <class 'int'>
b: 25 type: <class 'int'>
addition: 71
```

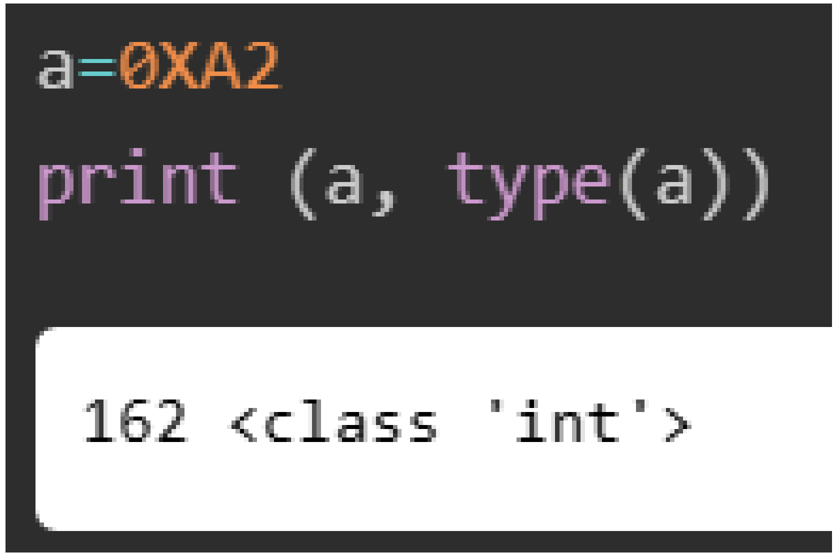


```
a=oct(71)
print (a, type(a))
```

```
('0107', <type 'str'>)
```

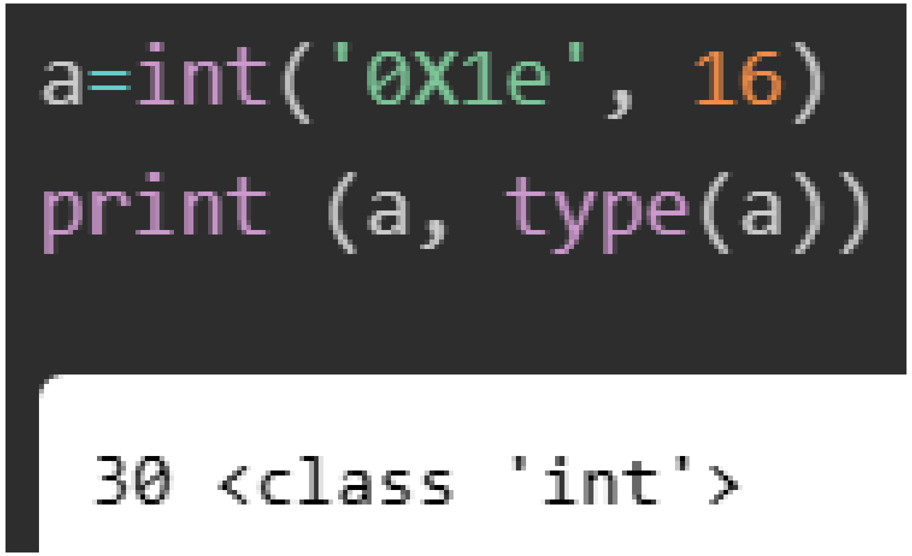
integers (int)

Hexa-decimal - prefix it by 0x or 0X.



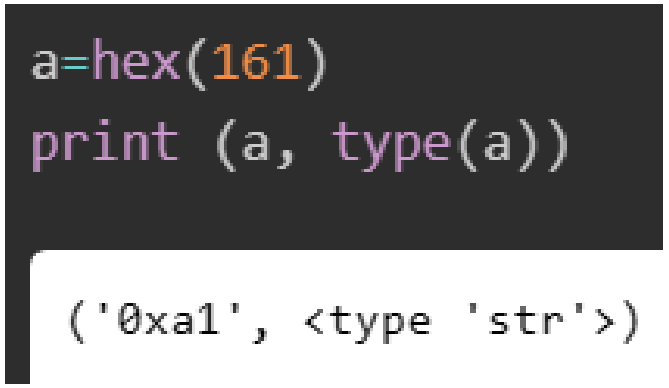
```
a=0XA2
print (a, type(a))
```

```
162 <class 'int'>
```



```
a=int('0X1e', 16)
print (a, type(a))
```

```
30 <class 'int'>
```



```
a=hex(161)
print (a, type(a))
```

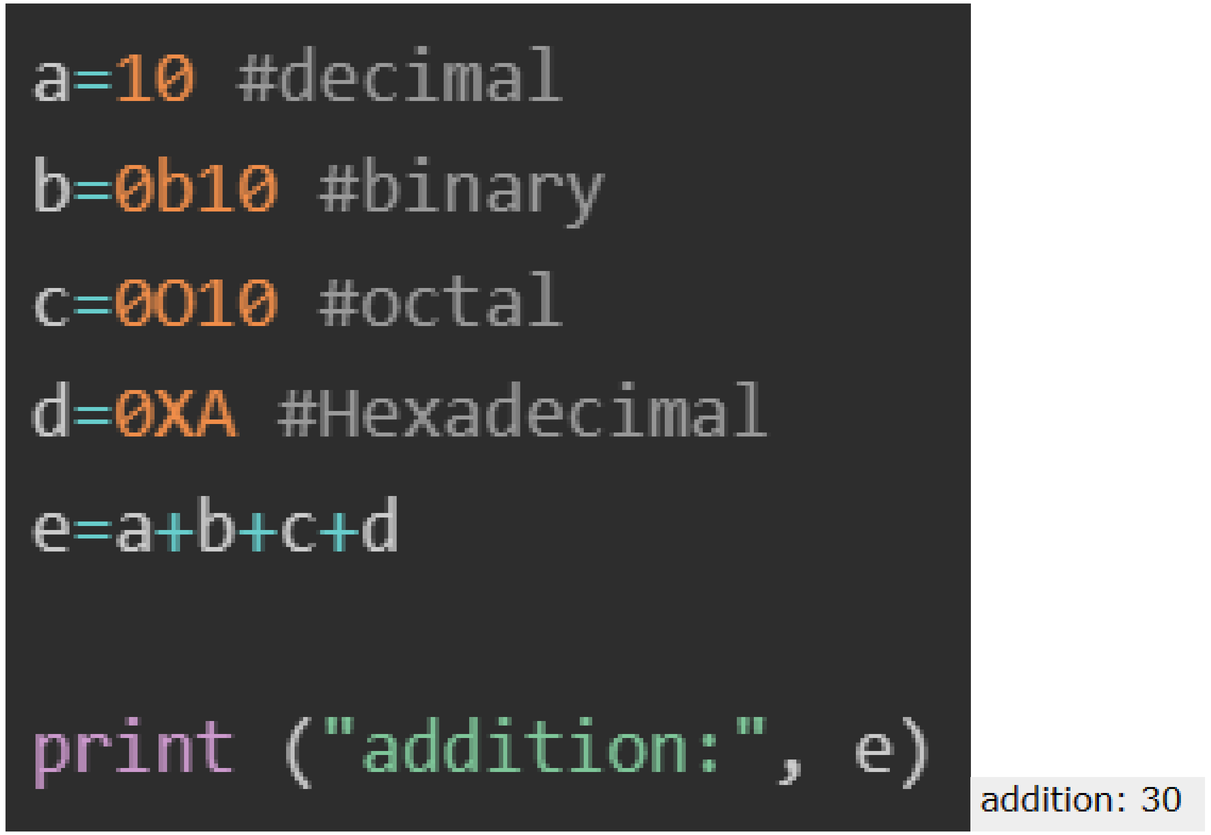
```
('0xa1', <type 'str'>)
```

```
num_string = "A1"
number = int(num_string, 16)
print ("Hexadecimal:", num_string, "Integer:",number)
```

```
Hexadecimal: A1 Integer: 161
```

integers (int)

Though an integer can be represented as binary or octal or hexadecimal, internally it is still integer. So, when performing arithmetic operation, the representation doesn't matter.



```
a=10 #decimal
b=0b10 #binary
c=0O10 #octal
d=0XA #Hexadecimal
e=a+b+c+d

print ("addition:", e)
```

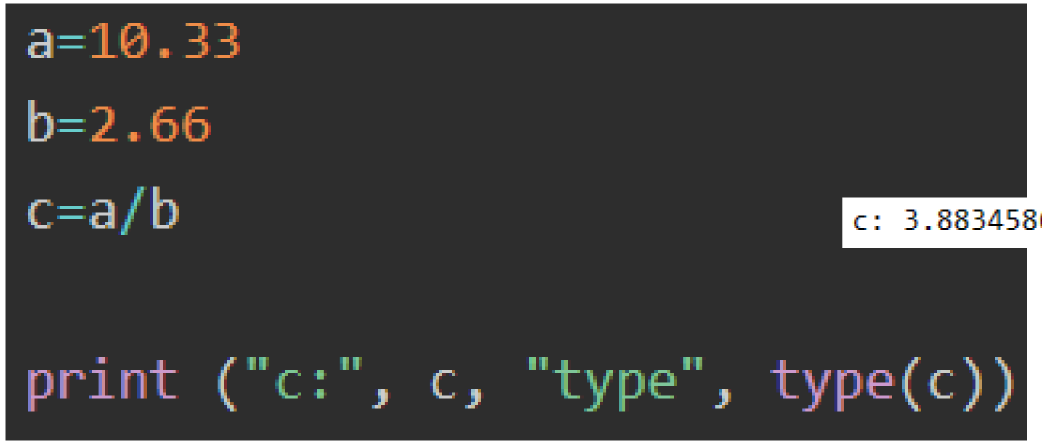
```
addition: 30
```

Floating point (float)

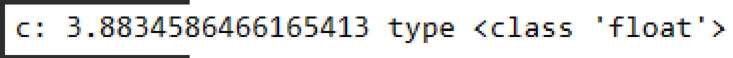
- a number, positive or negative, containing one or more decimals;
- can also be scientific numbers with an "e" to indicate the power of 10.

float

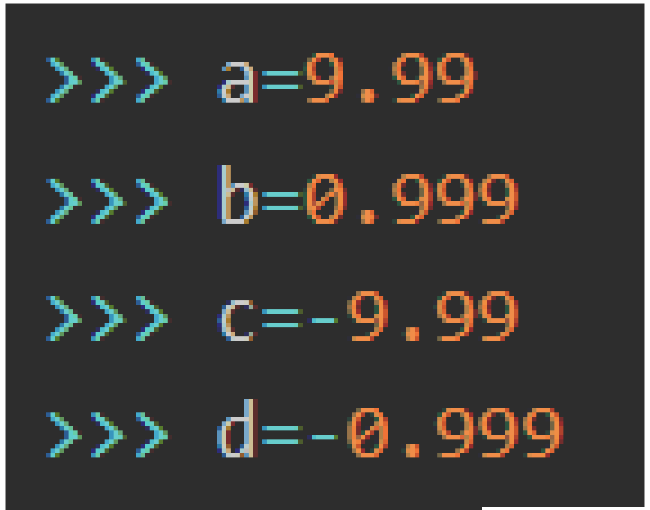
In Python, there is no restriction on how many digits after the decimal point can a floating point number have. However, to shorten the representation, the E or e symbol is used. E stands for Ten raised to. For example, E4 is 10 raised to 4 (or 4th power of 10), e-3 is 10 raised to -3.



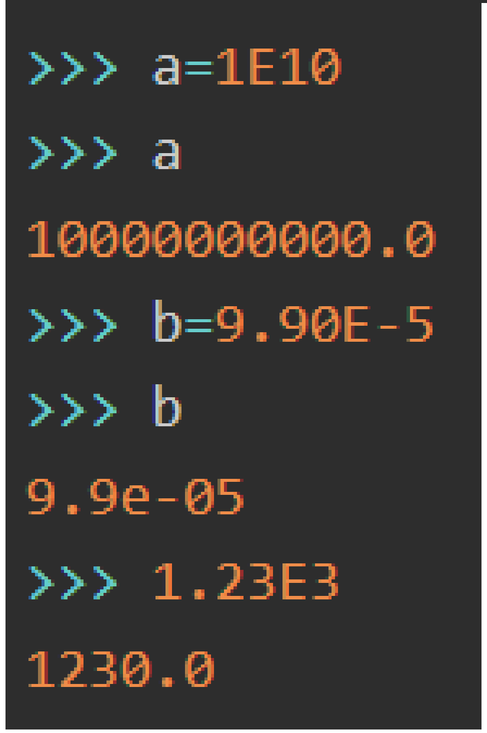
```
a=10.33
b=2.66
c=a/b

print ("c:", c, "type", type(c))
```



```
c: 3.8834586466165413 type <class 'float'>
```



```
>>> a=9.99
>>> b=0.999
>>> c=-9.99
>>> d=-0.999
```

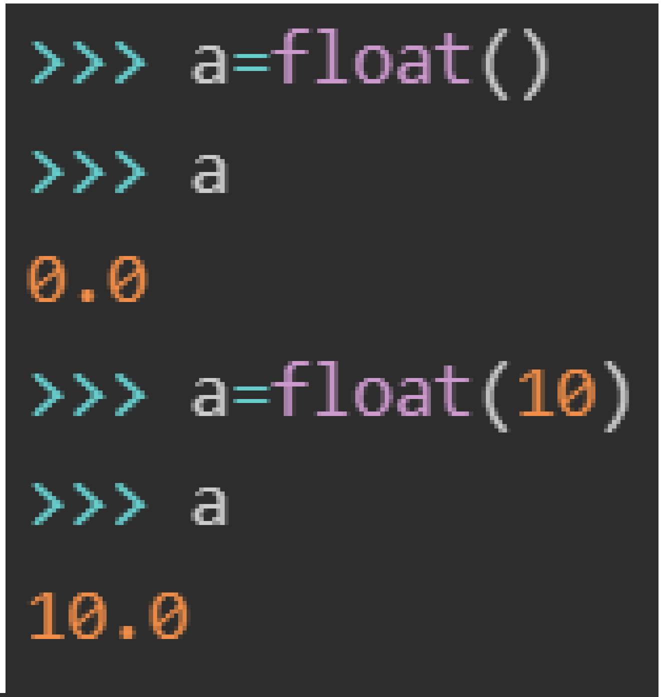


```
>>> a=1E10
>>> a
10000000000.0
>>> b=9.90E-5
>>> b
9.9e-05
>>> 1.23E3
1230.0
```

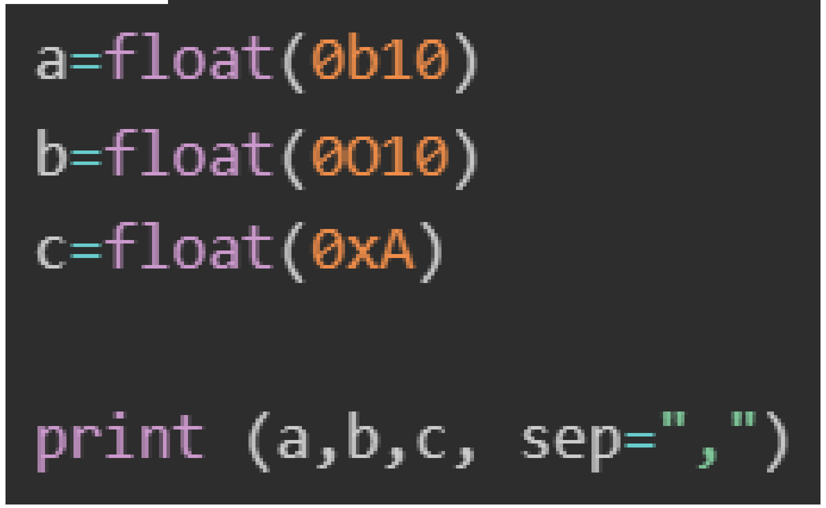
float

Python's float() function returns a float object, parsing a number or a string if it has the appropriate contents. If no arguments are given in the parenthesis, it returns 0.0, and for an int argument, fractional part with 0 is added.

Even if the integer is expressed in binary, octal or hexadecimal, the float() function returns a float with fractional part as 0.



```
>>> a=float()
>>> a
0.0
>>> a=float(10)
>>> a
10.0
```



```
a=float(0b10)
b=float(0O10)
c=float(0xA)

print (a,b,c, sep=",")
```

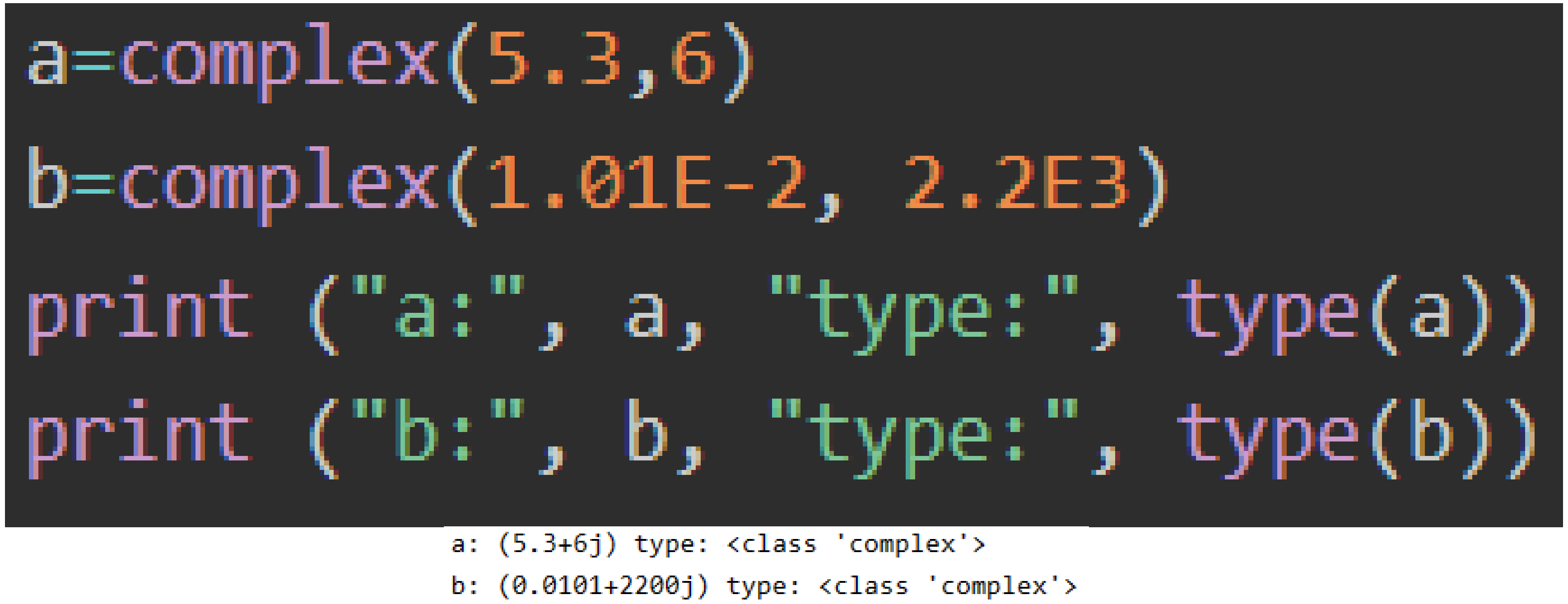
```
2.0,8.0,10.0
```

complex numbers

- written with a "j" as the imaginary part
- find their applications in mathematical equations and laws in electromagnetism, electronics, optics, and quantum theory.

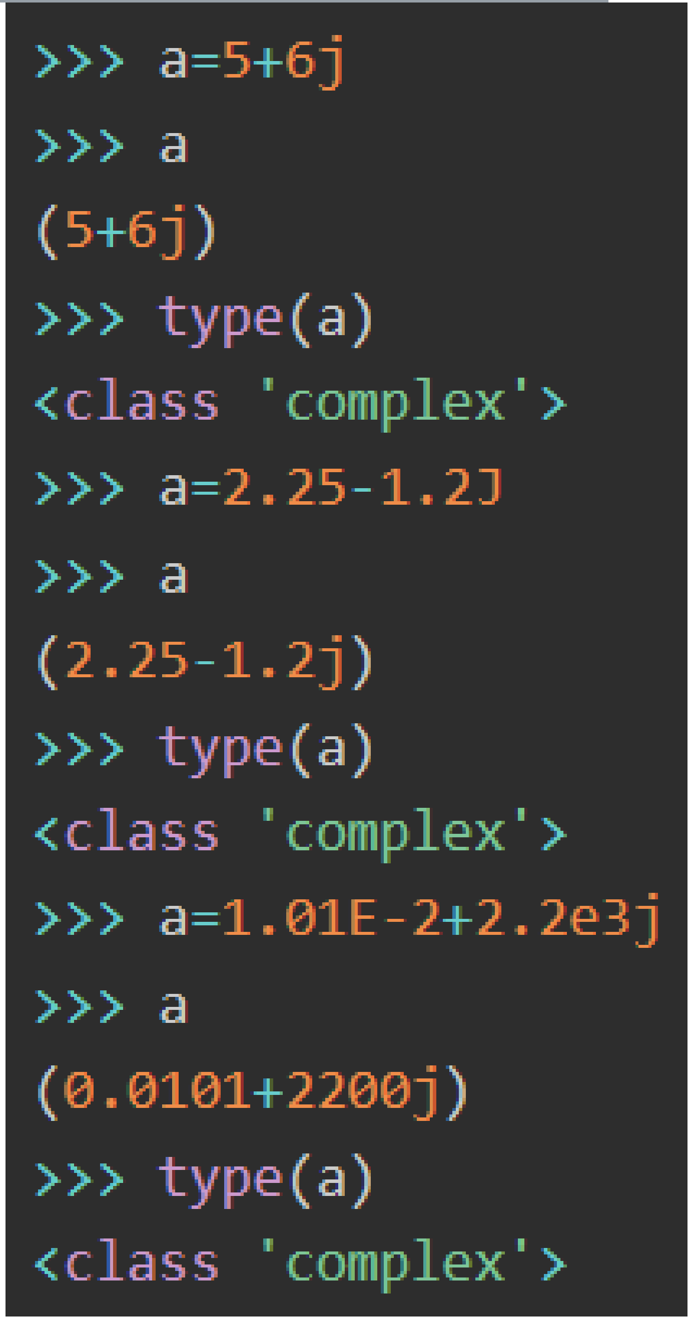
complex

Python's complex() function helps in forming an object of complex type. The function receives arguments for real and imaginary part, and returns the complex number.

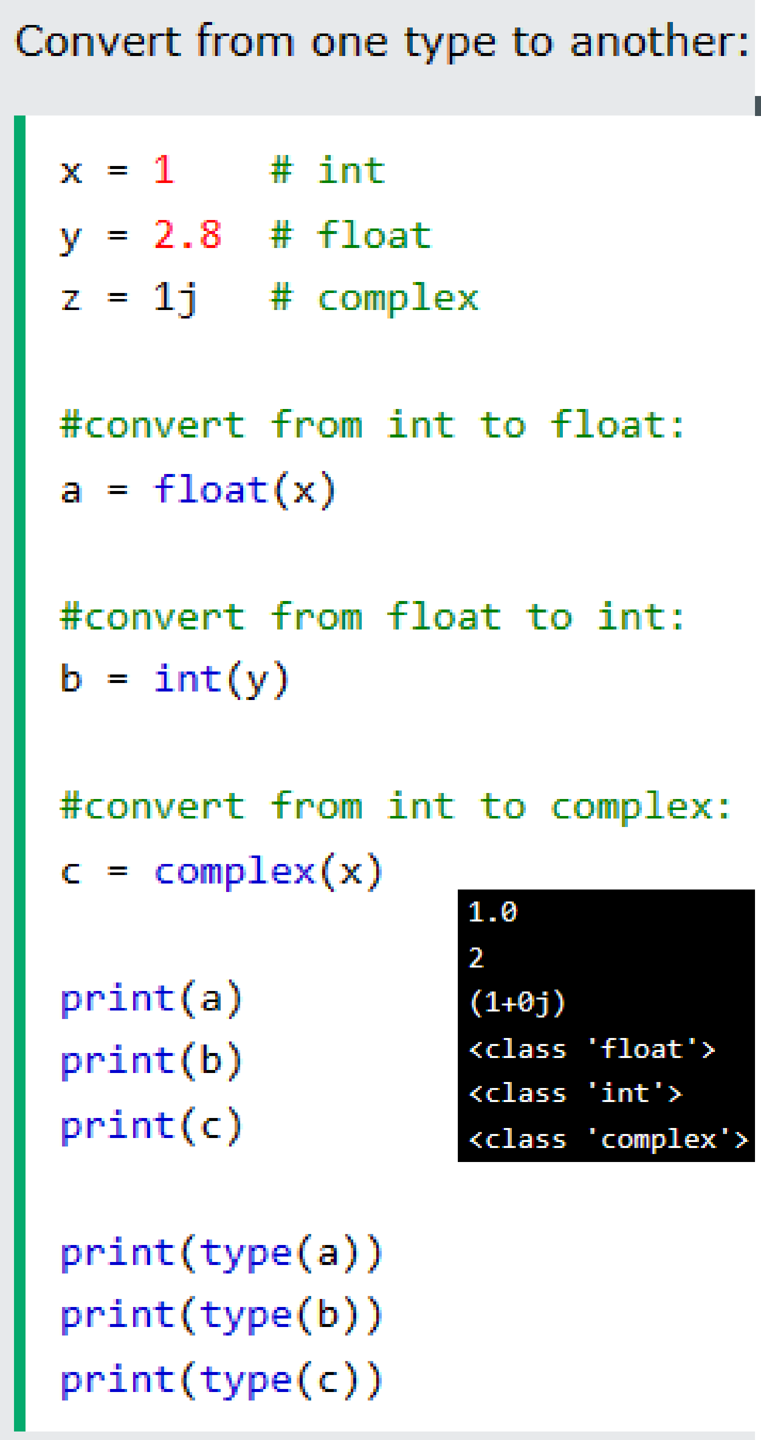


```
a=complex(5.3,6)
b=complex(1.01E-2, 2.2E3)
print ("a:", a, "type:", type(a))
print ("b:", b, "type:", type(b))
```

```
a: (5.3+6j) type: <class 'complex'>
b: (0.0101+2200j) type: <class 'complex'>
```



```
>>> a=5+6j
>>> a
(5+6j)
>>> type(a)
<class 'complex'>
>>> a=2.25-1.2J
>>> a
(2.25-1.2j)
>>> type(a)
<class 'complex'>
>>> a=1.01E-2+2.2e3j
>>> a
(0.0101+2200j)
>>> type(a)
<class 'complex'>
```



Convert from one type to another:

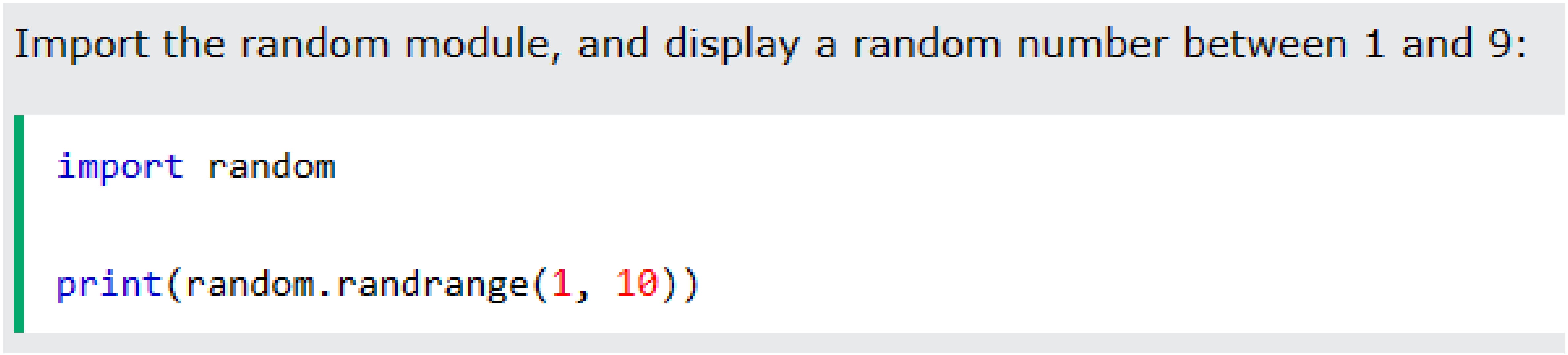
```python
x = 1    # int
y = 2.8  # float
z = 1j   # complex

#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))
```

```
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



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

```python
import random

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

Python Random Module (w3schools.com)

Applications???

Any questions???

REFERENCES:

- Learn Python Programming. (2023). https://www.tutorialsteacher.com/python
- Python Tutorial. (2022). https://www.w3resource.com/python/python-tutorial.php
- Python Tutorial. (n.d.). https://www.tutorialspoint.com/python/index.htm
- Python Tutorial. (n.d.). https://www.w3schools.com/python/default.asp