Digital Career Institute

Python Course - Introduction

Digital Career Institute

Operators and basic math functions

Digital Career Institute

Topics

- Math operators
- Basic math functions
- Assignment operators

Digital Career Institute

Operators and operands

- **Operators** are special symbols that represent computations like addition and multiplication.

- The values the operator uses are called **operands**.

Python math operators

- Arithmetic operators are used with numeric values to perform common mathematical operations:
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Modulus
  - Exponentiation
  - Floor division

Python math operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Python math operators - examples

```
>>> 2 + 3
5
>>> 10 - 4
6
>>> 3 * 4
12
>>> 12 / 2
6.0
>>> 12 / 5
2.4
```

```
>>> 10 % 2
0
>>> 10 % 3
1
>>> 2 ** 3
8
>>> 10 // 3
3
>>> 10 // 4
2
```

Python math operators - examples

- Modulus % returns the rest from division, for example:
  - 10 % 3 = 1, because 10 ÷ 3 = 3 + 1 (remainder) or 3 * 3 + 1 = 10

- Floor division returns integer part of the result, for example:
  - 10 // 4 = 2, because 10 ÷ 4 = 2.5 and the integer part of 2.5 is just number 2.

Built-in math functions

min() and max()

- The min() and max() functions can be used to find the lowest or highest value in a set of values:

```
>>> x = max(5, 10, 15)
>>> print("Max. value is", x)
Max. value is 15
>>> y = min(5, 10, 15)
>>> print("Min. value is", y)
Min. value is 5
```

abs()

- The abs() function returns the **absolute** (positive) value of the specified number:

```
>>> abs(12.34)
12.34
>>> abs(-12.34)
12.34
>>> abs(3 + 4j)
5.0
>>> abs(True)
1
>>> abs(False)
0
```

pow()

- The pow(*x*, *y*) function returns the value of x to the **power** of y (x^y):

```
>>> pow(2, 3)
8
>>> pow(2, 4)
16
>>> pow(2, 5)
32
>>> pow(-2, 3)
-8
>>> pow(2.5, 2)
6.25
```

round()

- The round() function returns a floating point number that is a **rounded version** of the specified number, with the specified number of decimals.

- The default number of decimals is 0, meaning that the function **without** second argument will return the **nearest** integer:

```
>>> round(1.23)
1
>>> round(1.56)
2
```

round()

Syntax

round(*number, digits*)

Parameter Values

Parameter	Description
number	Required. The number to be rounded
digits	Optional. The number of decimals to use when rounding the number. Default is 0

round() with second argument

```
>>> round(1.23456, 2)
1.23
>>> round(1.23456, 4)
1.2346
```

math module

- Python has also a built-in module called **math**, which extends the list of mathematical functions.
- To use it, you must **import** the math module:

```
>>> import math
```

- Importing and modules will be covered **later** in detail!

math.sqrt()

- When you have imported the math module, you can start using methods and constants of the module.

- The math.sqrt() method for example, returns the square root of a number:

```
>>> w = math.sqrt(2)
>>> s = math.sqrt(49)
>>> print("Square root of 2 is", w)
Square root of 2 is 1.4142135623730951
>>> print("Square root of 49 is", s)
Square root of 49 is 7.0
```

math.ceil() and math.floor()

- The math.**ceil()** method rounds a number **upwards** to its nearest integer, and the math.**floor()** method rounds a number **downwards** to its nearest integer, and returns the result:

```
>>> c = math.ceil(3.14)
>>> print(c)
4
>>> f = math.floor(3.14)
>>> print(f)
3
```

- More functions in documentation!

Assignment operators

Python assignment operators

- Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Python assignment operators

- Examples:

```
>>> x = 3
>>> y = 5
>>> x += 3
>>> print("x =", x)
x = 6
>>> y += 4
>>> print("y =", y)
y = 9
```

Number systems

- There are an infinite number of ways to represent numbers. Most modern civilizations use positional notation, which is efficient, flexible, and well suited for doing arithmetic.
- A notable feature of any positional system is its **base**, which represents the number of digits available. People naturally favor the **base-ten** numeral system, also known as the **decimal system**, because it plays nicely with counting on fingers.

Number systems

- Computers, on the other hand, treat data as a bunch of numbers expressed in the **base-two** numeral system, more commonly known as the **binary** system. Such numbers are composed of only two digits, zero and one.

- For example, the binary number 10011100_2 is equivalent to 156_{10} in the base-ten system. Because there are ten numerals in the decimal system (zero through nine) it usually takes fewer digits to write the same number in base ten than in base two.

What is base in numeral systems?

- A notable feature of any positional system is its **base**, which represents the number of digits available.

- People naturally favor the **base-ten** numeral system, also known as the **decimal system** (we have 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in this system - 10 digits) , because it plays nicely with counting on fingers.

- Computers, on the other hand, treat data as a bunch of numbers expressed in the **base-two** numeral system, more commonly known as the **binary** system. Such numbers are composed of only **two** digits - zero and one.

Octal and hexadecimal systems

- The **octal** numeral system, or **oct** for short, is the base-8 number system, and uses the digits 0 to 7,

- In mathematics and computing, the **hexadecimal** (also **base 16** or **hex**) numeral system is a positional numeral system that represents numbers using a radix (base) of 16.

- Unlike the common way of representing numbers using 10 symbols, hexadecimal uses **16** distinct symbols, most often the symbols "0" – "9" to represent values **0 to 9**, and "A" – "F" (or alternatively "a" – "f") to represent values **10 to 15**.

Integers with base other than 10

Prefix	Interpretation	Base
`0b` (zero + lowercase letter `'b'`) `0B` (zero + uppercase letter `'B'`)	Binary	2
`0o` (zero + lowercase letter `'o'`) `0O` (zero + uppercase letter `'O'`)	Octal	8
`0x` (zero + lowercase letter `'x'`) `0X` (zero + uppercase letter `'X'`)	Hexadecimal	16

Integers with base other than 10

- Binary numeral system
- Octal numeral system
- Hexadecimal numeral system

- Just use the right prefix!

```
>>> print(0b10)
2
>>> print(0o10)
8
>>> print(0x10)
16
```

Binary numbers

- **Binary number** is a number expressed in the base-2 numeral system or binary numeral system, a method of mathematical expression which uses **only two** symbols: typically "0" (zero) and "1" (one).

- How to count in binary?

Binary		
0		We start at 0
1		Then 1
???		But then there is no symbol for 2 ... what do we do?

Binary counting

Well how do we count in Decimal?	
0	Start at 0
…	Count 1, 2, 3, 4, 5, 6, 7, 8, and then…
9	This is the **last digit** in Decimal
10	So we start back at 0 again, but add 1 on the left

Binary counting

The same thing is done in binary:		
	0	Start at 0
.	1	Then 1
..	10	Now start back at 0 again, but **add 1 on the left**
...	11	1 more
....	???	But NOW what ... ?

Binary vs Decimal

Decimal vs Binary

Here are some **equivalent** values:

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

Binary numbers

- When you **say** a binary number, pronounce each digit (example, the binary number "101" is spoken as *"one zero one"*, or sometimes *"one-oh-one"*). This way people don't get confused with the decimal number.

- Detailed explanation of binary number system: here

- Binary numbers on Wikipedia

Integer to binary

- Built-in function **bin()** converts an integer number to a binary string prefixed with “0b” (zero and b). The result is a valid Python expression:

```
>>> bin(7)
'0b111'
>>> bin(8)
'0b1000'
>>> bin(1000)
'0b1111101000'
```

Binary to integer

- Built-in function **int()** returns an integer object constructed from a number or string.

- To convert from binary to integer we must set second argument of int() called **base**. For binary numbers the base is **2**:

```
>>> int('111', base=2)
7
>>> int('111', 2)
7
>>> int('101010101', 2)
341
```

At the core of the lesson

Lesson learned:

- We know math operators and how to use them
- We know basic math functions and how to use them
- We know assignment operators and how to use them

Digital Career Institute