

Chapter 1 — Python — Arithmetic, Numbers, and Variables

Introduction

This introductory Chapter will get you started with your first programs in Python. It will introduce the two most common types of numbers (integer, and floating-point), operations with numbers, strings, and variables.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Use the print statement to display one or more values.
- Explain the difference between floating-point and integer values.
- Set up mathematical expressions in Python.
- Produce output using string literals.
- Create variables and assign the results of an expression to them.
- Use variables in expressions and display the values of variables.

Prerequisites

This chapter requires no previous material. It is recommended that Bonus Chapter 1 and Bonus Chapter 2 be considered before tackling the concept of algorithmic thinking.

The Print Statement with Literals


Open up Python and try typing the following statement into the editor window on the screen, then click the Execute button and see what is displayed in the result window.

**** need to work on this for users of the playground, spyder, idle... ****

```
1| print("hello")
```

The `print` statement will evaluate the Python expression within the parentheses and then display the result onto screen.



<code>print(expression)</code>	Statement
<p>The print statement displays the result of the expression on the screen. Print has many options and a few of them will be introduced in future sections and chapters.</p> <p>https://docs.python.org/3/library/functions.html#print http://bit.ly/py3print</p>	

Examples include:


```
1| print(9)
```

```
1| print(8.456)
```

```
1| print("I said barf.")
```

```
1| print('one plus one is two')
```


The first two examples simply show a number on the console. The third and fourth statements show the display of a string. A string is a group of letters, numbers and other symbols surrounded by either single quotes 'string' or double quotes "string". We will learn more about strings in the next chapter.


<code>literal</code>	Concept
<p>Literals are constant integers, floating-point numbers, strings and may other types of values that may be used in an expression, a statement, or in your program.</p> <p>Example literals include <code>1</code>, <code>67</code>, <code>89.45</code>, <code>-345</code>, <code>"hello"</code>, <code>"words and stuff"</code>, <code>'text'</code>.</p> <p>https://docs.python.org/3/reference/lexical_analysis.html#literals http://bit.ly/js3literal</p>	

NOTE: Statements and functions in Python are case-sensitive. This means that the statement `PRINT()` is not the same as `print()`. Be careful to type statements exactly as they appear.

Simple Math and Number Types

The two most common types of numbers in Python are Integer and Float. There are additional types that store complex and rational numbers, but they are beyond the scope of this introduction. You may enter integer or floating-point numbers directly into your programs.

<i>integer literal</i>	Concept
<p>An integer is a whole or counting number. In Python integers can range from $-\infty$ to ∞.</p> <p>Sample integers include 57834576384576384578, 67, 1, 0, -34, and -123456789.</p> <p>https://docs.python.org/3/library/functions.html#int http://bit.ly/js3int</p>	

<i>float literal</i>	Concept
<p>A float is a real (decimal) number from $-\infty$ to ∞. Floating-point numbers are entered with a decimal point. They may also be entered in scientific notation using "e" or "E". floating-point numbers are not exact values, like integers. Floats are stored in the computer's memory as an exponent and mantissa with a limited number of digits.</p> <p>Sample floats include: 1.2, 0.0, -23.456, 2.67e5, and 9.456E-6.</p> <p>https://docs.python.org/3/library/functions.html#float http://bit.ly/py3float</p>	

Simple Numeric Operators

This chapter will introduce seven numeric operators and how they are used. Most of them you know already but a couple of them may be new to you. They are:


- **+** — Addition — Add two numbers together.
 - 6 + 9 evaluates to 15

- `9 + 27.7` evaluates to `36.7`
- `405.5 + 123.3` evaluates to `528.8`
- `-` — Subtraction — Subtracts the second number from the first.
 - `9 - 3` evaluates to `6`
 - `3 - 10.9` evaluates to `-7.9`
- `-` — Negation — Changes the sign of a number.
 - `-4`
 - `-3.567`
- `*` — Multiplication — Multiply two numbers together.
 - `3 * 5` evaluates to `15`
 - `65.7 * 1.2` evaluates to `78.84`
- `/` — Division — Divide two numbers. Will always return a float value even if the result is a whole number.
 - `9 / 4` evaluates to `2.25`
 - `16 / 4` evaluates to `4.0`
 - `8 / 4.5` evaluates to `1.7777777777777777`
- `**` — Exponentiation
 - `2**3` evaluates to `8`
 - `3.9**0.5` evaluates to `1.9748417658131499`

Python also allows you to use parenthesis () to change the order of evaluation of operators.

Operators evaluate the same way that they do in common arithmetic:

- parenthesis;
- exponentiation;
- division, multiplication;
- then addition, subtraction.

<code>+ - * / ** ()</code>	Numeric Operators
<p>The five basic operators of addition, subtraction, multiplication, division and exponentiation are included in Python. You may also use parenthesis as you define your expressions to change the order of operations or clarify.</p> <p>These operators (except division) will return an integer value if both operands are integer. If either or both are floating-point, a float will be returned by the operation.</p> <p>Example expressions with operators include: <code>(-4 + 6) * 3</code>, <code>7 / (1 + 0.03)</code>, <code>** 3</code>, and <code>4 + 5 * 9</code></p>	

https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex http://bit.ly/py3types	
--	--

Now that you know the `print()` statement and mathematical operators you can use Python like a calculator.

```
1| print(9*56)
```

```
1| print(7/8 + 9)
```

```
1| print((80+1) **.5)
```

Strings

A string is a group of letters, numbers, and symbols surrounded by quotation marks. The quotation marks come in 4 varieties: 1) single quotes `'xxx'`, 2) double quotes `"xxx"`, 3) triple single quotes `'''xx'''`, and 4) triple double quotes `"""xx"""`. The triple quote is used for long strings that may contain quotes. You may put anything you want in the quotes and Python will treat it as a value that you can use in your program.

Examples include:

- `'Hola'`
- `"Bon Jour"`
- `"BR549"`
- `'5+6='`
- `'''A string with "quotes" of 'both' types.'''`
- `"""He said 'Barf!'"`
- `"""President Linclon said, "Four score and seven years ago".""`

It is suggested that you be consistent with what type of quote used in your program.

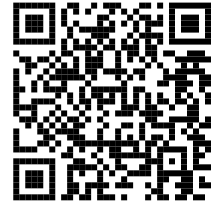
<i>string literals</i>	
------------------------	--

Concept



Strings allow us to embed non numeric text and other symbols into our program. Strings are delimited by one of four types of quotes. A string should not contain the type of quote that is used to surround the string, as the compiler will become confused.

https://docs.python.org/3/reference/lexical_analysis.html#index-16
<http://bit.ly/py2litstr>



Chapter 3 is all about strings and how we manipulate them in our programs.

Sending Multiple Values to Print

The print statement was shown in an earlier topic to display the result of a calculation or some other expression to the user. If you want multiple values to be displayed on a single line, you may pass multiple values to print and it will display them with a space between them.

```
print(expression, expression...)
```

Statement

Print with multiple comma separated values.

<https://docs.python.org/3/library/functions.html#print>
<http://bit.ly/py3print>



This version of the print statement will display each value with a separator between them on the screen. Usually a space ' ' will be used.

```
1| print(1, '/', 2, '-', 1/2)
2| print(3, 9)
```

```
1/2=0.5
3 9
```

Programs with Multiple Lines

It would be very difficult to do a complex process with only a single line program. Typical Python



programs may be several hundred lines of code. When you have a multi line program, the statements are executed in the sequence that you entered them.

Lets look at the example:

```
1| print("Approximations for PI")
2| print(22/7, 25/8, 255/81)
```

which displays:

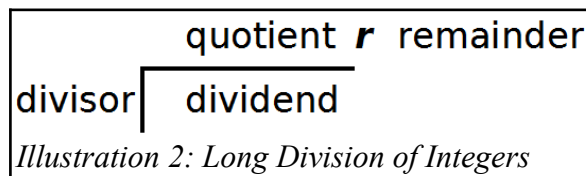
```
Approximations for PI
3.142857142857143 3.125 3.1481481481481484
```

It can be seen that the statements are take one at a time and executed "sequentially" to build a larger and more complex program.


Special Integer Operators

In many programming languages, Python included, there are a couple of special integer division operators. They are called floor division and remainder.

Remember back to doing "long division" with integers, your answer was two whole numbers, the quotient and the remainder. The floor division operator gives the quotient and the remainder operator gives the remainder.



- `//` — Floor Division — Returns the closest whole number less than the result of a division.
 - `9//4` evaluates to `2`
 - `-9//4` evaluates to `-3`
- `%` — Remainder — Return the remainder after floor division (also known as modulo).
 - `1%3` evaluates to `1`
 - `4%3` evaluates to `1`
 - `9%3` evaluates to `0`

// %	Numeric Operators
<p>Special integer operators of floor divide and modulo. If two integers are divided using floor divide <code>//</code> an integer is returned and the remainder is discarded. The modulo operator <code>%</code> returns the remainder of long division of two numbers.</p>	
<p>https://docs.python.org/3/reference/expressions.html?highlight=operators#binary-arithmetic-operations http://bit.ly/py3binop</p>	


An example of doing integer division showing the quotient and remainder may be seen by finding the integer result of 17 divided by 3:

```
1| q = 17 // 3
2| r = 17 % 3
3| print(q, r)
```

5 2

Comment Statements

Sometimes to make a program easier to read or to tell Python to ignore a small bit of code we will use the pound sign (`#`). Anything on a line after the `#` will be skipped by Python, we call these statements comments.

# # some ueser text statement # more user notes	Statement
<p>The pound sign, not in a string, tells Python to ignore the text following it. Programmers will add messages to themselves reminding them what a tricky bit of code is doing, or to put messages for other programmers to see.</p>	
<p>Comment statements are also sometimes used to tell python to skip lines of code.</p> <p>https://docs.python.org/3/reference/lexical_analysis.html?highlight=comment#comments http://bit.ly/py3comment</p>	


```
1| # a sample program by J.M.Reneau
2| print(5*9) # How many square feet?
3| # print(5/9)
```

45

Variables

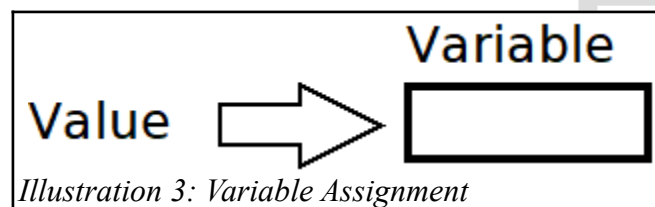
In computer programming a variable is a label assigned to a block of the computer's memory. In Python, a variable can be used to store and retrieve a value, a list of values, and many other types of values. In this chapter we are going to keep it simple and look at only at variables containing numeric values. The variable also includes a "scope" where it can be seen in a program, but this is a topic for future chapters.

There are two basic operations we do with variables: 1) assignment, and 2) retrieval.

Variable Assignment

You can assign a value to a variable and it will remember the value until it is changed or until the program terminates.

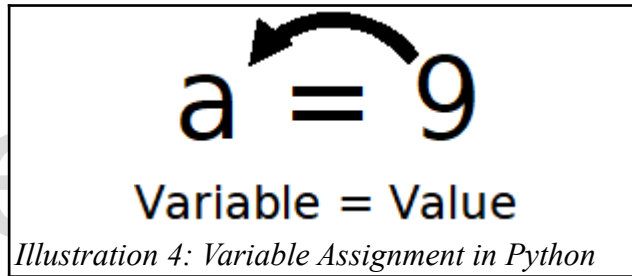
Technically the process in Python is called binding, where a name is bound to a value. To make this introduction more generic, the more common description of assignment will be used. In Chapter 6 you will see much more details about names (variables) and bindings.



As you can see in the illustration the value is written into a memory location that has a name assigned to it. In reality, you the programmer assign the name. For example:

```
1| a = 9
```

Can be thought of as: 1) allocate a place in memory; 2) name the place 'a'; and 3) store the value 9 into it.



You may assign the result of an expression easily to a variable by placing the expression after the equal sign.

```
1 | Area = 8.5 * 9.3
```

`variable = expression`

Statement

The = sign performs assignment. It evaluates the expression on the right of the equal sign and stores the result into the variable.

https://docs.python.org/3/reference/simple_stmts.html?highlight=assignment#assignment-statements

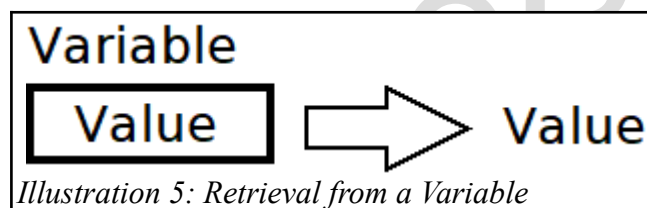
<http://bit.ly/py3assign>



NOTE: Variable names are case-sensitive. Also, variables must start with a letter and contain only letters, numbers, and underscores. You should avoid creating a variable with the name of a builtin function in Python, unless you know what you are doing.

Variable Retrieval

Once a value has been assigned a value, the variable name may be used in most places where a constant value may be used. You may retrieve the value several times, but the value stored will not change until you explicitly change it.



```
1 | a = 9
2 | b = 6
```

```
3| print(a+b)
4| c = a ** b
5| print(c)
```

```
15
531441
```

Summary

Summary goes here

Important Terms

- assignment
- comment
- expression
- float
- floor division
- integer
- literal
- number
- print
- quotes
- remainder
- statement
- string
- variable

Exercises

1. Write a one line Python program to calculate the difference of 8 times 3 subtracted from 32.
2. Create program to calculate the volume of a rectangular solid. Assign the variable **h** to 4, **w** to 9, **d** to 5, and **v** to the volume. Display the volume with a print statement.
3. Write a program that will display the quotient and remainder of the division of two integers. Assign the variable **dividend** to 99 and the variable **divisor** to 8. Display the results using a single print statement, like shown below. Try your program with several dividends and divisors.

```
99 divided by 8 is 12 remainder 3
```

4. Create a Python program with two variables: **r** set to 8 and **pi** set to 3.14. Calculate and display the surface area of a circle and the volume of a sphere with the radius **r**.

$$a = \pi r^2$$

$$v = 4\pi \frac{r^3}{3}$$



5. The Body Mass Index (BMI) is a calculation that returns a value that is used in health screening. A person with a BMI of less than 18.5 can be considered underweight. A BMI of 18.5 to 25 to be an optimal weight for an adult. A BMI from 25 to 30 is overweight, and a score higher than 30 is considered obese. In a Python program assign the variable **h** to the height of 5.5 feet and the variable **w** to the weight of 154 pounds. Use the formula below to calculate and print BMI.

$$BMI = \frac{w}{h^2} \times 702$$

6. Create a Python program to calculate the present value of an amount of money in the future. In your program assign the variable **i** to the interest rate of 8%, the variable **fv** to the future value of \$1000.00, and the variable **n** to represent 5 years. Save your calculation into a variable named **pv** and then print it.

$$pv = \frac{fv}{(1+i)^n}$$

Word Search

e a n u v s t r i n g s k g t
x . r . g f w l i t e r a l a
f a f n n l g d h . c s r f s
r e m a i n d e r f d x y s s
s f . i f l o a t h n c r p i
w r e x p r e s s i o n h j g
f m y j k g z j l e n x z a n
f l o o r . d i v i s i o n m
h . k q f s h o e j n d s z e
y b c j z q u o t e s p j h n
v a r i a b l e h p w r i . t
i n t e g e r y y x z i m k r
j z n u m b e r x k w n k n .
o c c u r r e n c e t w h . t s l f
l v m d s t a t e m e n t d .

assignment, comment, expression, float, floor division, integer, literal, number, print, quotes, remainder, statement, string, variable

References

<https://docs.python.org/3/library/stdtypes.html>



https://en.wikibooks.org/wiki/Python_Beginner_to_Expert/Native_Types

Free eBook Edition

Please support this work at
<http://syw2l.org>

Free eBook Edition

