**Arithmetic Operators**

Arithmetic operators are often used for personal and business needs, such as how much money to deposit in your checking account before making a purchase.

Here are Python arithmetic operators that you can use in an arithmetic operation.

|  |  |
| --- | --- |
| **Name** | **Symbol** |
| Addition | + |
| Subtraction | - |
| Multiplication | \* |
| Division | / |
| Modulus (Remainder) | % |
| Exponentiation | \*\* |
| Floor Division (Quotient) | // |

**Arithmetic Operations**   
Along with the assignment operator, =, the table below has examples of each operator being used in an arithmetic operation.

|  |  |
| --- | --- |
| **Operator** | **Example** |
| + | 10 + 2 = 12 |
| - | 10 – 2 = 8 |
| \* | 10 \* 2 = 20 |
| / | 10 ÷ 2 = 10 / 2 = 5.0  11/5 = 2.2 |
| % | 11 % 2 = 1; the result of this operation is 1, because modulus, %, returns the remainder of an operation |
| The result of 14 % 3 = 2 because 2 is the remainder |
| \*\* | 10 \*\* 2 = 10 \* 10 = 100; this raises 10 to the power of 2, and |
| 2 \*\* 3 = 2\* 2 \* 2 = 8; this raises 2 to the power of 3 |
| // | 11 // 2; the result of this operation is 5, because floor division, //, returns the quotient of an operation, and the result of this, 13 // 3 operation, is 4 |

When your program executes, such as adding two numbers, 3 + 4 = 7, instructions and data for the operation, stored are at different memory addresses, and even though you can use these values in other arithmetic operation, working with numbers may quickly become cumbersome and difficult to follow, but assigning each value to an identifier, makes your programs manageable and easier to follow.

**Names**

In Python, you can use a name, also called a variable, or identifier, to make an assignment; to do so, you need a variable, the Python assignment operator, and a value: width = 3. From this assignment, Python stores the value, 3, in memory and associates the variable with the memory location where 3 is stored, and if you wanted to assignment other values to width, you could do it like this, width = 6.

You can also write this code at the Python primary prompt:

>>>width = 3

>>>length = 4

>>>area = length \* width

Program in memory:

|  |  |  |
| --- | --- | --- |
| Memory Location | Variable Name | Stored Value |
| A | length | 4 |
| B | width | 5 |
| C | area | 12 |

Note: These addresses, memory locations, are for demonstration only.

You could keep width the same value, but to change length you would write:

>>>length = 8

>>>area = length \* width

Program in memory:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Location |  | Variable Name | Stored Value |
| D |  | length | 8 |
| B |  | width | 5 |
| E |  | area | 24 |

Note that length was assigned 8, length = 8; as a result, Python stores 8, not in the same location as 4 but in a different one, and since area has changed, 24 is at a new location, but width stays in place since it was not changed.

There are specific rules and suggested conventions for naming Python variables.

**Naming Rules and Conventions**   
When naming variables, you must follow Python rules, but convention is your responsibility, and, therefore, it should be consistent.

Variable names should be lowercase, such as eggs; avoid using single letters, but if you do, use ones that are easily recognized when displayed on different screens.

Identifiers must have a consistent case; for instance, the variable age is not the same as Age.

The first character in a name must be an underscore or alphabetic; for example, age, Age, \_age, although different identifiers, they are valid in Python.

You can include numeric characters in variable names, but not at the beginning, student1 is valid, but 1student is not.

Variables must not have spaces, so my sons age = 10, is not valid, but mysonsage = 10 and the Python preferred, my\_sons\_age = 10 are.

Although, the underscore is the Python preferred, when variables are more than one word, such as my daughters age, you can use mixed case, with the first letter of the first word in lowercase and the first letter of remaining words, uppercase: myDaughtersAge.

Since my\_daughters\_age is a variable, it can be assigned different values, such as my\_daughters\_age = 17, or my\_daughters\_age = 7.

**Operators and Variables**

If you are planning a trip to the grocery store to purchase a number of items, such as apple, bread, papaya, orange, and banana, but on the shopping list you wrote, a, b, c, d, e, and how many of each, 5, 1, 2, 6, 13, when you get to the store you might be puzzled on what each letter is for and how many of each letter to get.

An alternate approach would be a list with a meaningful name for each, and how many of each, and maybe the total number of items, such as this:

|  |  |
| --- | --- |
| Item | quantity |
| apple | 5 |
| bread | 1 |
| papaya | 2 |
| orange | 6 |
| banana | 13 |
|  |  |
| Total items | 27 |

At the Python primary prompt, >>>, you could create a shopping list program with each item, how many, and total:  
  
>>> apple = 5  
>>> bread = 1

>>> papaya = 2  
>>> orange = 6

>>> banana = 13

>>> total = apple + bread + papaya + orange + banana  
>>> total

27

>>>

In RAM, each name is associated with the memory location where the corresponding value is stored; here is an example of variables in your program, the memory location where each is stored, and value stored in each location; once you make an assignment, such as apple = 5, Python stores 5 at memory location A and associate apple with it.

Example:

These do not represent memory locations in a real computer; they are for demonstration purposes only.

|  |  |  |
| --- | --- | --- |
| Variable Name | Memory Location | Stored Value |
| apple | A | 5 |
| bread | B | 1 |
| papaya | C | 2 |
| orange | D | 6 |
| banana | E | 13 |
|  |  |  |
| total | F | 27 |

In an arithmetic computation, you are likely to have several different operators, not only addition, and in such cases, you must be aware of the order in which operators are executed. Python calls this operator precedence.

**Operator Precedence**

When you work with operators from different precedence levels, you must pay close attention to the rules Python apply during computation; these rules are called operator precedence levels that go from highest to lowest, with the highest being done first, and those at the same level are applied left to right. Python also refers to this as most binding, highest, to least binding, lowest.

Here are Python arithmetic precedence levels going from highest, top, to lowest and same row, left to right:

|  |  |
| --- | --- |
| Operator | Description |
| () | Parentheses (grouping) |
| \*\* | Exponentiation |
| \*, /, % | Multiplication, division, remainder |
| +, - | Addition, subtraction |

The parentheses although not an arithmetic operator, is listed because it is often used in arithmetic operations; it will be discussed later.   
In the shopping list example, you saw the addition operator being applied for totaling the number of grocery items, but you can apply others as well.   
apple = 5  
bread = 1  
papaya 2  
orange = 6  
banana = 13

total = apple + bread + papaya + orange + banana

By going left to right, Python arrives at 27 for total by adding, apple + bread, and continues the addition until computation completes.

If you decided to buy 2 fewer apples, you would apply the subtraction operator to the calculation:

total = apple - 2 + bread + papaya + orange + banana

total = 25

Python arrives at 25 for total by subtracting and adding, apple -2 + bread, which continues until computation completes, by going left to right, and even though two operators are used in this calculation, no need to be concerned about precedence order since they are at the same precedence level.

Now, if you decided to buy twice as many oranges, you could double what you already have by adding 6:

total = apple + bread + papaya + orange + 6 + banana

total = 5 + 1 + 2 + 6 + 6 + 13   
Giving total = 33

Instead of adding 6, you could multiply the number of oranges you already have by 2, but in this case, careful attention must be paid to the order in which Python executes the statement.

total = apple + bread + papaya + orange \* 2 + banana

total = 5 + 1 + 2 + 6 \* 2 + 13

Giving total = 33

To arrive at 33, Python, computed the multiplication first followed by the addition; this is because multiplication is at a higher precedence level.

However, if the precedence were ignored and instead going left to right, the answer would have been different, 41, that is 14 \* 2 + 13.

If you decided to divide the bananas among you and four of your friends, you could use the division operator to see how many each of you will get:  
how\_many\_each = bananas / your friends + you

how\_many\_each = 13 / 4 + 1  
It is important to not only focus on precedence, but also the computation sequence needed to arrive at the correct result; for example, if you computed this left to right, because division has a higher precedence than addition, you would each get 4.25 bananas, but if you first added 4 + 1, your result would be 2.6 bananas each.

You can tell Python which operation to do first by using the parentheses; although it is not an arithmetic operator, it is often used in arithmetic operations, and since parenthesis has the highest precedence in Python, you can use it to change the order of operation: how\_many\_each = 13 / (4 + 1) = 13/5 = 2.6

Now what if you wanted to share only whole bananas, and keep the rest for another day, you would use the floor division operator this way,

how\_many\_each = banana // 5 = 13//5

You would get 2 bananas each, because the floor division returns the quotient only.

On the other hand, if you wanted to share the remainder instead of the quotient, you would use the modulus operator, %, to tell how many are left after the division, how\_many\_left = banana % 5 = 13 % 5; would result in a total of 3 bananas, because the modulus operator returns the remainder only; this does not mean each will get three bananas; instead there are three bananas to share.

If you are planning to start a vegetable garden with each side measured the same, and you wanted to calculate the area, you would apply the exponentiation, \*\*, operator; for instance, if each side were 5 feet, area = side\*\*2 = 5\*\*2 = 25.

Now consider including a flower garden 5 feet long and 3 feet wide, and you want to add the area to the area of the vegetable garden, to get the total square feet, you would write, length \* width + side\*\*2 = 5 \* 3 + 5\*\*2.

But before you begin your calculation, you checked the Python precedence table and discovered that exponentiation has the highest precedence, followed by multiplication, then addition; as a result, you cannot go left to right, but instead you must follow the order of precedence, which is 5 \* 3 + 25 = 15 + 25 = 40, but left to right is 15+5\*\*2 = 20\*\*2 = 400, incorrect.

**Assignment**

Some equations are not in Python syntax, which could cause errors; make necessary corrections before executing.

1.

Use the Python primary prompt, >>>, to assign values to a and b; after each assignment press Enter, and after you type each arithmetic operation, press Enter to display the result.

>>> a = 13

>>> b = 3

>>> a + b

>>> a - b

>>> a \* b

>>> a / b

>>> a % b

>>> a \*\* b

>>> a // b

2.

Use the Python primary prompt, >>>, to assign values to x and y; after each assignment press Enter, and after you enter each arithmetic operation, type z and press Enter**,** to display the result.

>>> x = 4

>>> y = 3

>>> z = x + y

>>> **z**

>>> z = x \* y

>>> **z**

>>> z = x % y

>>> **z**

>>> z = x // y

>>> **z**

>>> z = x \*\*2

>>> **z**

3.   
Use the Python primary prompt to compute the following

3A.  
l = 8, w = 5

a = l x w

B.  
b1 = 5, b2 = 7, h = 2

a = 1/2×(b1 + b2)×h

3C.  
d = 5, h = 6  
v = 1/3 x 3.14 (d/2)\*\*2 x h

4.   
The original price for a container of coffee is $6.00; what is the final price if it is on sale for 30% off? The tax on coffee after any discount is 1%.

5.   
Compute the following temperature conversions:  
Given:   
Fahrenheit = 85 degrees

Celsius = 25 degrees  
  
A.  
Fahrenheit into Celsius

C = 5/9 × (F − 32)

B.  
Celsius to Fahrenheit

F = (9/5 + 32) x C

6.   
Compute the following using quotients only

Z = (7/2 - 10/3 + 9/2) \*\* 2

7.   
Compute the following using remainders only

Z = (7/2 - 10/3 + 9/2) \*\* 2

8. The cost to ride the train is $3.00 for adults and half price for children; what will it cost for four adults and 7 children to ride the train

9. Josh has a basket of peaches from which he plans to make peach pies; each pie needs four peaches

A.  
The operator would you recommend using that will tell how many he will have left from the basket of peaches?

B.  
If the basket has 33 peaches, how many pies could Josh make using whole peaches?

10.  
Compute I given:  
r= .04

t = 5

n = 1

p = 25000

i = p \* [(1 + r/n)t\*n – 1]