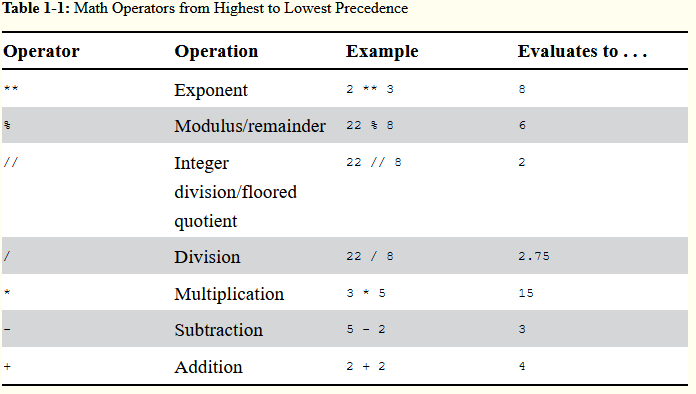
Section 1: Notes

Expression = Values + Operators (Ex. 2 + 2, 2 being the value and + being the operator); Expressions always evaluate/simplify down to a single value.

Follow these rules in order, next from left to right

* Please (Parentheses)
* My Dear (Multiply or Divide)
* Aunt Sally (Add or Subtract)

-----------------------------------------------------------------

Data type is a category of values and every value belongs to a data type. These types are:

* int (integer, such as -2 and 30)
* float (Decimal numbers, such as 3.14 or even 42.0 aka floating point numbers “floats”)
* string (Data value for text values)
  + string concatenation - when the "+" operator is used between strings, it combines the strings together
  + string replication - useful for making a string repeat itself by using "\*" instead of "+".

Order of operation can be utilized when printing strings. Ex. 'Alice' + '!' \* 10 outputs 'Alice!!!!!!!!!!'

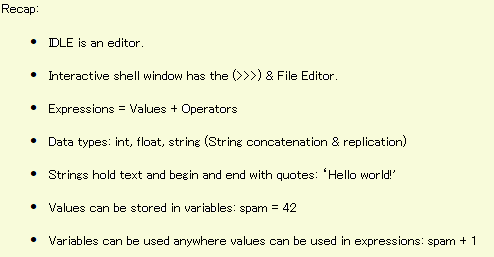
-----------------------------------------------------------------------------------------------------------------------------------------------------

A variable is an assignment statement that can be thought of as a container that has been assigned a value (ex. spam = 42); spam = container and 42 = value.

When declaring variables for projects, try to think of defining names as to what it pertains to as to avoid bad naming for you and others.

Declaring a variable: spam = 'Hello' (you can redeclare by replacing ‘Hello’ with whatever you want)

So if a Python instruction evaluates to a single value, it's an expression. Otherwise it's a statement. But mostly we just generically called them instructions or code.

Evaluates = Expression | Doesn't evaluation = Statement

Section 1: First Program

# - comments, python ignores comments

blank lines are skipped, use them to group relevant code and tidy up your code.

-----------------------------------------------------------------------------------------------------------------------------------------------------

Functions = mini-programs in the program.

print() is a function that displays the values passed, such as str, int, etc. Ex. print(‘Hello world!’)

input() is a function that scans for user inputs. Ex. myName = input() or myAge = input()

len() function allows you to print the length of characters for a string. Ex. print(len(myName))

-----------------------------------------------------------------------------------------------------------------------------------------------------

In code, a function is its name, followed by parentheses and optionally sometimes there are values passed to the function inside the parentheses. In this context these values are called arguments. But really, values and arguments are the exact same thing.

-----------------------------------------------------------------------------------------------------------------------------------------------------

The str() and int() functions return string and integer values of whatever you pass them. This is really handy if you need to convert between data types. (Context: print('You will be ' + str(int(myAge) + 1) + ' in a year.'))

str(26)

'26' - dictated that it's a string by quotes ''

>>> int('1234')

1234

>>> float('3.14')

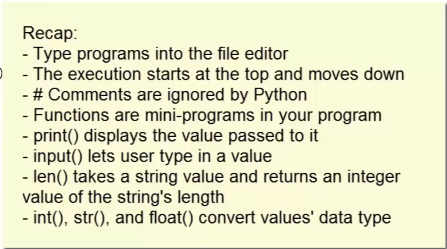
3.14

>>> float(1)

1.0 - turns an integer if input into a float

You can't do math on a string, so inputs such as, '26' + 1 returns with an error and the same goes for concatenation on integers, 27 + ' years old'. This is when converting between data types is useful:

* >>> int('26') + 1 - this allows the string '26' to be converted to an int.
* >>> str(int(myAge) + 1) - The input function of myAge helps convert int to str to allow concatenation.

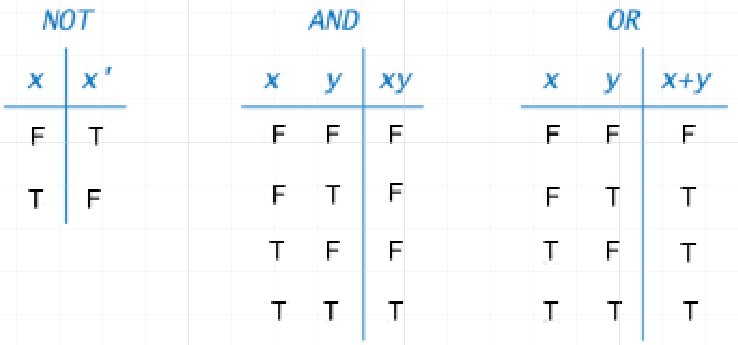


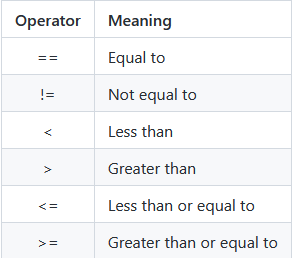
Section 2: Flow Chart & Control Concepts

Instructions called flow control statements can decide which python instructions to execute under which conditions.

Boolean data types have two values, true and false, it's binary. If assigning the value true or false, always type True or False, can't be TRUE, false, 'True'. Case sensitive or else you'll return errors or in the last instance, a string which doesn't constitute a boolean value.

Integers and strings are never equal to each other. But, integer values and float values can equal to each other.



Table 1: Comparison Operators

The simplest flow control statements are IF and Else. Lines of code indented at the same level are known as blocks.

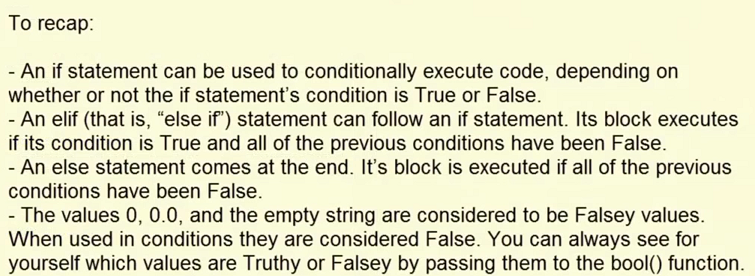
Blocks are useful, because it tells what code will run as part of a loop, class, or function and usually groups statements which would be treated as if they were one statement.

>>> if myName == 'Alice' or myName == 'alice': # Incorporates an or statement within an IF statement

Syntax: You must put a colon (:) at the end of if or else statements.

Elif statements: else-if statement provides as many as conditions to check as you need. You can have as many elif statements follow an if statement as you need but the order of the statements does matter. The execution enters the first block that has a true condition. The rest of the conditions won't even be checked.

Typically when you take users input, it’s stored as a string and attempting to use that input in an if or else statement checks for boolean, this is known as truthy and false values for strings. Non-blank names (actually typing something) like Alice is truthy, whereas if you leave something blank (not typing anything or 0) it’s false.

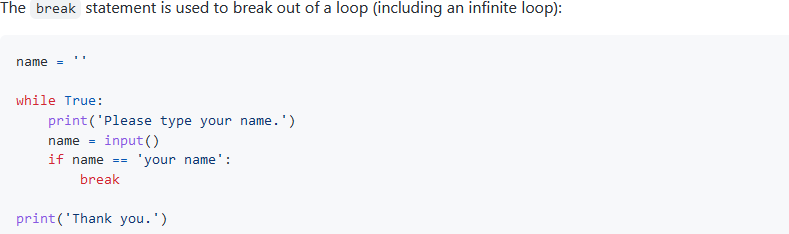


Section 2: While & For Loops

While statements act differently than if statements, as long as the condition remains true it keeps looping or itterates continuously, the first time that the condition returns as false the program/loop stops.

Input validation is when the input function is called the user can type in anything If you ask the user their age instead of a number they could type in Abraham Lincoln or type in a negative number. So loops are a good way to ensure that the program keeps asking the user until they've entered some valid input for your program.

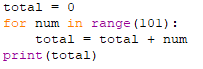
A certain kind of bug that you can have with loops called an infinite loop. If you ever have this issue/bug where it won't quit printing press ctrl + c, this is known as a keyboard interrupt.



Continue statement will cause the execution to immediately jump back to the start of a loop and recheck the condition.

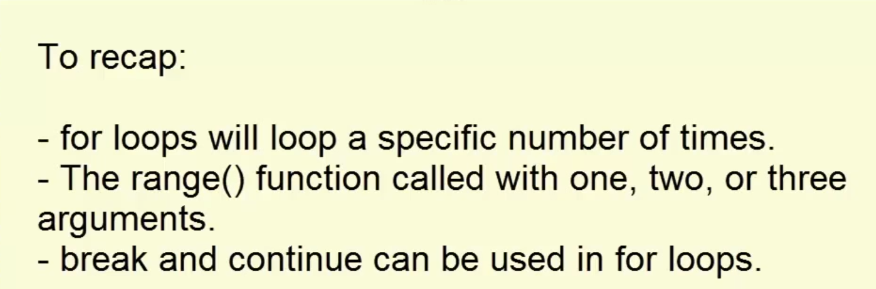
For Loop - Instead of looping as long as a certain condition is true like a while loop, a foor loop iterates a specific number of times.

When the iteration begins, the variable i is set to zero and each time it iterates the i value changes by 1 until it reaches 5 iterations.

The figure to the left, this iterates 100 times from 0-100 adding up the values. If you need to iterate X amount of times, always add 1 additional to the number if it needs to be included, because it will always go up to that number but does not include that number.

Range is a date type that allows you to change integers to range to follow any sequence of integers, allowing you to start at numbers other than zero.

For i in range(0, 10, 2) - 0 dictates it'll start at 0, it'll go up to but not include the value of 10, and 2 dictates that it'll iterate by adding 2 instead of 1. (Ex. It'd print 0, 2, 4, 6, 8) Just as you can increment up, you can also decrement.



Section 3: Functions

(When installing, make sure to add python to path or you'll have to add in system variables and if it isn't added also add scripts to enable pip: C:\Users\Hangm\AppData\Local\Programs\Python\Python39\Scripts)

The import statement allows you to import different modules (math, random, etc). Ex. Random isn't a built in function, it exists only inside the random module. To which you need to import it, then if you use random.randint(1, 10) it requires the module name infront of it when you call it.

Import multiple modules by simply separating them with a comma (ex. Random, sys, os, math)

from <module name> import \* - The star allows you to import everything, allows you to call the module without using the period in random.randint. While convienant, the longer name provides more readable code by what modules are also used.

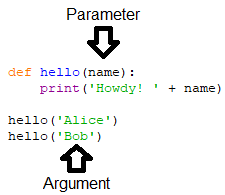
Import sys - sys.exit() - Allows you to terminate a program early.

Pyperclip.copy() | pyperclip.paste() - functions for reading/writing text to the clipboard.

-----------------------------------------------------------------------------------------------------------------------------------------------------

Functions = mini-programs in the program, code that executes when the function is called. Function calls can be part of expressions because they evaluate to a value returned by the function

Def hello(): - replace hello with any name, it's a placeholder, this is the beginning of defining a function.

Deduplicating code - Getting rid of duplicated or copied and pasted code, making it shorter, easier to read, and update.

Arguement = the value passed in the function call

Parameter = the variable inside the function

Data convert (len to str) - str(len('hello'))

None value - a value that represents a lack of a value.

Every function call has return value, even the print function but you don't have to have a return statement in all of your functions. If your function statement doesn’t have a return statement the return value defaults to a none value.

 This allows it to print "Hello World" on the same line, adding the comma and end='' used to place a space after the displayed string instead of a newline.

Sep=’’ - keyword argument that specifies what character should be used to separate multiple arguments (an empty space by default)

-----------------------------------------------------------------------------------------------------------------------------------------------------