**Python basic building blocks 101**

Similar to Tableau and SQL, there are different types of statements; for example, x=10, if, for, while.

There are multiple line continuation characters to indicate that the code continues on the next line; for example, :, \, (), [], {}.

Additionally, various data types use different brackets; for example, [] defines lists and indexing, () identify parameters, and {} are used for dictionaries and sets. There are more examples, but let’s keep it brief for now.

There are some general rules for writing Python statements, which are:

* Python allows for commenting within code blocks to help keep track of why the code has been written or what it is trying to achieve (Starting the line with a '#' character will instruct the interpreter to treat the line as a comment and not code to be executed.).
* For data analysis, libraries, packages, modules, and classes (essentially pre-built code blocks [more about these later]) must be imported at the start of each working session. It is good practice to import the libraries close to the top of the notebook for maintenance purposes, but it is not essential.
* Python is case-sensitive, and spelling or typing errors will result in code not running.
* Indentation (the spaces at the start of a line of code) is required – use four spaces as a standard.
* It is good practice to be consistent with naming conventions.
* Operators can be used for mathematical, statistical, and logical operations.
* To view (also known as call or pass) the output (result, return), the print() function is used. You’ll see this a lot. Simply listing the variable name will also produce the output, but it is good practice to control and format the output using the print statement. More on this later.
* Single quotation marks (straight – '') are used to indicate a quote within a quote, while double quotation marks (straight – "") indicate word-for-word quotes. For example:
  + single quotes: used for identifiers and string literals (e.g. name = 'Sophia')
  + double quotes: used for text, string interpolation, and quotations (e.g. text = "Python is fun!")
* PEP 8 is a handbook on reading, speaking, and writing Python.

In special situations, you can use triple quotations (""" *...add a comment here…* """) to write comments that span more than one line. These are called [**docstrings**](https://fourthrev.instructure.com/courses/895/modules/items/67567).

(comments are for **why** it is performing a specific task). A good rule is to always add comments for any line of code where the purpose is not obvious.

The values that an operator acts on are called operands. Consider the expression 3 + 4 = 7. Here, 4 and 3 are called *operands* and the + symbol is the *operator*. An operand can be either a literal value or a variable that references an object. Therefore, if you assign a = 3 and b = 4 the expression would instead read a + b = 7. A sequence of operands and operators, like a + b - 2, is called an *expression*.

Operators in Python include:

* **assignment operators:** allocate different values to variables
* **arithmetic operators:**perform mathematical operations on variables or values
* **comparison operators:** compare values
* **logical operators:** merge conditional statements
* **membership operators:** check if an element exists within another element or not
* **bitwise operators:** compare binary numbers.

You use assignment operators to assign values to variables, and you use arithmetic operators in complete mathematical equations.

|  |  |
| --- | --- |
| **Precedence** | **Operator and description** |
| 1 | ( )Brackets (the operator or section in brackets will always run first) |
| 2 | \*\*: Exponentiation (raise to the power) |
| 3 | ~ + -: Complement, unary plus, and minus (method names for the last two are +@ and -@) |
| 4 | \* / % //: Multiply, divide, modulus, and floor division |
| 5 | + -: Addition and subtraction |
| 6 | >> <<:Right and left bitwise shift |
| 7 | &: Bitwise AND |
| 8 | ^ |: Bitwise exclusive OR and regular OR |
| 9 | <= < > >=: Comparison operators |
| 10 | <> == !=: Equality operators |
| 11 | = %= /= //= -= += \*= \*\*=: Assignment operators |
| 12 | in not in: Membership operators |
| 13 | not or and: Logical operators |