1. **Exponential**

* 3 \*\* 4 =

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

1. **Integeral division**

* 7 // 2 = 3 && -7 // 2 = -4

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

1. **Vaariables**

* x = 3
* y = 4

or

* x, y = 3, 4

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

1. **Python Operators**

* <https://www.programiz.com/python-programming/operators>

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Type of Variable**

* x =int( 4.7)
* Print(type(x))
* Int

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

1. **Floating Poing Arithmetic: Limitations and issues**

* <https://docs.python.org/3/tutorial/floatingpoint.html> (0.1 is 0.1000000001 in python)

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Style Code for Python**

* <https://www.python.org/dev/peps/pep-0008/>

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Errors and Exceptions**

* <https://docs.python.org/3/tutorial/errors.html>

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Strings**

* this\_string = 'Simon\'s skateboard is in the garage.' **\** is used to add **‘** in between the string
* mystring[2:6:2]
* print('Dark {1} {0} {2}'.format(‘brown’, ‘quick’, ‘fox’))
* == Dark quick brown fox
* quick = "SHailja"
* >>> print(f'Dark {quick}')
* Dark SHailja
* result = 100/777
* >>> result
* 0.1287001287001287
* >>> print("The result was {r:10.5f}".format(r=result))
* The result was 0.12870

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **String Method**

* maria\_string = "Maria loves {} and {}"

print(maria\_string.format("math", "statistics"))

* islower()
* find()
* format()
* count()
* rindex //finds from last

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **List**
   * my\_list = [1,2,3,"String"]
   * print(len(my\_list))
   * ==4
   * my\_list.append("six") //**Add item to last of the list**
   * print(len(my\_list))
   * ==5
   * my\_list.pop() **//removes the last element or at a given index and return the deleted item**
   * my\_list.sort() **//sort the items in ascending order**

* **Slicing List**
  + print(list\_of\_random\_things[1::3])
  + [3.4 , 'a string']
* len() returns how many elements are in a list.
* max() returns the greatest element of the list. How the greatest element is determined depends on what type objects are in the list. The maximum element in a list of numbers is the largest number. The maximum elements in a list of strings is element that would occur last if the list were sorted alphabetically. This works because the max function is defined in terms of the greater than comparison operator. The max function is undefined for lists that contain elements from different, incomparable types.
* min() returns the smallest element in a list. min is the opposite of max, which returns the largest element in a list.
* sorted() returns a copy of a list in order from smallest to largest, leaving the list unchanged.
* new\_str = "\n".join(["fore", "aft", "starboard", "port"])
* A helpful method called append adds an element to the end of a list.
* letters = ['a', 'b', 'c', 'd']
* letters.append('z')

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

1. **in** or  **not in**
   * 'this' **in** 'this is a string'
   * True

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

1. **Mutability**

* **Mutability** is about whether or not we can change an object once it has been created. If an object (like a list or string) can be changed (like a list can), then it is called **mutable**. However, if an object cannot be changed with creating a completely new object (like strings), then the object is considered **immutable**
* lists are **mutable**. While strings are **immutable.**

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Order**

* **Order** is about whether the position of an element in the object can be used to access the element. **Both strings and lists are ordered.** We can use the order to access parts of a list and string.

**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Tuples**

* A tuple is another useful container. It's a data type for **immutable ordered** sequences of elements. They are often used to store related pieces of information. Consider this example involving latitude and longitude:
* dimensions = 52, 40, 100
* length, width, height = dimensions
* print("The dimensions are {} x {} x {}".format(length, width, height))

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

1. **Sets**

* A **set** is a data type for **mutable** **unordered** collections of **unique elements**. One application of a set is to quickly remove duplicates from a list.
* numbers = [1, 2, 6, 3, 1, 1, 6]
* unique\_nums = set(numbers)
* print(unique\_nums)
* **{1, 2, 3, 6}**



**--------------------------------------------------------------------------------------------------------------------------------------**

1. **Dictionaries**

* A **dictionary** is a **mutable** data type that stores mappings of unique keys to values. Here's a dictionary that stores elements and their atomic numbers.
* Dictionaries can have keys of any immutable type, like integers or tuples, not just strings. It's not even necessary for every key to have the same type! We can look up values or insert new values in the dictionary using square brackets that enclose the key.
* elements = {"hydrogen": 1, "helium": 2, "carbon": 6}
* print(elements["helium"]) # print the value mapped to "helium"
* elements["lithium"] = 3 # insert "lithium" with a value of 3 into the dictionary
* print("carbon" in elements)
* print(elements.get("dilithium"))
* n = elements.get("dilithium")
* print(n is None) True
* print(n is not None) False

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

**Compound Data Structure**

