## String Reference Cheat Sheet

### String operations

* **len(string) -** Returns the length of the string
* **for character in string** -Iterates over each character in the string
* **if substring in string** - Checks whether the substring is part of the string
* **string[i]** - Accesses the character at index **i** of the string, starting at zero
* **string[i:j]** - Accesses the substring starting at index **i**, ending at index **j** minus 1. If **i** is omitted, its value defaults to **0**. If **j** is omitted, the value will default to **len(string)**.

### String methods

* **string.lower()** - Returns a copy of the string with all lowercase characters
* **string.upper()** - Returns a copy of the string with all uppercase characters
* **string.lstrip()** -Returns a copy of the string with the left-side whitespace removed
* **string.rstrip()** -Returns a copy of the string with the right-side whitespace removed
* **string.strip()** - Returns a copy of the string with both the left and right-side whitespace removed
* **string.count(substring)** - Returns the number of times substring is present in the string
* **string.isnumeric()** - Returns True if there are only numeric characters in the string. If not, returns False.
* **string.isalpha()** - Returns True if there are only alphabetic characters in the string. If not, returns False.
* **string.split()** - Returns a list of substrings that were separated by whitespace (whitespace can be a space, tab, or new line)
* **string.split(delimiter)** - Returns a list of substrings that were separated by whitespace or a delimiter
* **string.replace(old, new)** - Returns a new string where all occurrences of old have been replaced by new.
* **delimiter.join(list of strings)** - Returns a new string with all the strings joined by the delimiter

## Lists and Tuples Operations Cheat Sheet

Lists and tuples are both sequences, so they share a number of sequence operations. But, because lists are mutable, there are also a number of methods specific just to lists. This cheat sheet gives you a run down of the common operations first, and the list-specific operations second.

### Common sequence operations

* **len(sequence)** - Returns the length of the sequence
* **for element in sequence** - Iterates over each element in the sequence
* **if element in sequence** - Checks whether the element is part of the sequence
* **sequence[i]** - Accesses the element at index i of the sequence, starting at zero
* **sequence[i:j]** - Accesses a slice starting at index i, ending at index j-1. If i is omitted, it's 0 by default. If j is omitted, it's len(sequence) by default.
* **for index, element in enumerate(sequence)** - Iterates over both the indexes and the elements in the sequence at the same time

### List-specific operations and methods

* **list[i] = x** - Replaces the element at index i with x
* **list.append(x)** - Inserts x at the end of the list
* **list.insert(i, x)** - Inserts x at index i
* **list.pop(i)** - Returns the element a index i, also removing it from the list. If i is omitted, the last element is returned and removed.
* **list.remove(x)** - Removes the first occurrence of x in the list
* **list.sort()** - Sorts the items in the list
* **list.reverse()** - Reverses the order of items of the list
* **list.clear()** - Removes all the items of the list
* **list.copy()** - Creates a copy of the list
* **list.extend(other\_list)** - Appends all the elements of other\_list at the end of list

### List comprehension

* **[expression for variable in sequence]** - Creates a new list based on the given sequence. Each element is the result of the given expression.
* **[expression for variable in sequence if condition]** - Creates a new list based on the given sequence. Each element is the result of the given expression; elements only get added if the condition is true.

**Dictionary Methods Cheat Sheet**

**Syntax**

**x = {key1:value1, key2:value2}**

**Operations**

* **len(dictionary)** - Returns the number of items in the dictionary
* **for key in dictionary** - Iterates over each key in the dictionary
* **for key, value in dictionary.items()** - Iterates over each key,value pair in the dictionary
* **if key in dictionary** - Checks whether the key is in the dictionary
* **dictionary[key]** - Accesses the item with key key of the dictionary
* **dictionary[key]** = value - Sets the value associated with key
* **del dictionary[key]** - Removes the item with key key from the dictionary

**Methods**

* **dict.get(key, default)** - Returns the element corresponding to key, or default if it's not present
* **dict.keys()** - Returns a sequence containing the keys in the dictionary
* **dict.values()** - Returns a sequence containing the values in the dictionary
* **dict.update(other\_dictionary)** - Updates the dictionary with the items coming from the other dictionary. Existing entries will be replaced; new entries will be added.
* **dict.clear()** - Removes all the items of the dictionary

## Conditionals Cheat Sheet

### Comparison operators

* a == b: a is equal to b
* a != b: a is different than b
* a < b: a is smaller than b
* a <= b: a is smaller or equal to b
* a > b: a is bigger than b
* a >= b: a is bigger or equal to b

### Logical operators

* a and b: True if both a and b are True. False otherwise.
* a or b: True if either a or b or both are True. False if both are False.
* not a: True if a is False, False if a is True.