**Notes**

**Python Fundamentals**

**Python Syntax**

What is a code block?

A code block is a set of lines of code that belong together. For example, the first line of an if statement gives the condition, but the line(s) that follow explain what we want to happen *if* the condition is true. Examples of code block keywords include:

* def (functions)
* if, elif, else (conditional statements)
* for, while (loops)
* Class (classes)

**Numbers**

A screenshot of a computer

Description automatically generated with low confidence

There are 3 basics types of numbers in Python.

* int - whole numbers, positive or negative.  ex. 35
* float - decimal numbers, positive or negative.  ex. 4.2
* complex - are a part of the real number system and are often referenced with the letter j.  ex. 1 + 3j.  \*\*Note\*\* If you're not sure if you need to use them, it's safe to say you can ignore this data type.

**Strings** - https://docs.python.org/3/library/stdtypes.html

The following is a list of commonly used string methods:

* string.upper(): returns a copy of the string with all the characters in uppercase.
* string.lower(): returns a copy of the string with all the characters in lowercase.
* string.count(substring): returns number of occurrences of substring in string.
* string.split(char): returns a list of values where string is split at the given character. Without a parameter the default split is at every space.
* string.find(substring): returns the index of the start of the first occurrence of substring within string.
* string.isalnum(): returns boolean depending on whether the string's length is > 0 and all characters are alphanumeric (letters and numbers only). Strings that include spaces and punctuation will return False for this method. Similar methods include .isalpha(), .isdigit(), .islower(), .isupper(), and so on. All return booleans.
* string.join(list): returns a string that is all strings within our set (in this case a list) concatenated.
* string.endswith(substring): returns a boolean based upon whether the last characters of string match substring.

**List** - <https://docs.python.org/2/tutorial/datastructures.html>

Some built-in functions for sequences:

* enumerate(sequence) used in a for loop context to return two-item-tuple for each item in the list indicating the index followed by the value at that index.
* map(function, sequence) applies the function to every item in the sequence you pass in. Returns a list of the results.
* min(sequence) returns the lowest value in a sequence.
* sorted(sequence) returns a sorted sequence

Additional link: <https://docs.python.org/2/library/functions.html>

The following are some commonly used list methods:

* list.extend(list2) adds all values from a second sequence to the end of the original sequence.
* list.pop(index) remove a value at given position. if no parameter is passed, defaults to final value in the list.
* list.index(value) returns the index position in a list for the given parameter.

Additional link: <http://www.linuxtopia.org/online_books/programming_books/python_programming/python_ch14s07.html>

**Tuples**

You may recognize some of these built-in functions for sequences:

* max(sequence) returns the largest value in the sequence
* sum(sequence) return the sum of all values in sequence
* enumerate(sequence) used in a for-loop context to return two-item-tuple for each item in the sequence indicating the index followed by the value at that index.
* map(function, sequence) applies the function to every item in the sequence you pass in. Returns a list of the results.
* min(sequence) returns the lowest value in a sequence.
* sorted(sequence) returns a sorted sequence

Additional Link: <https://docs.python.org/2/tutorial/datastructures.html>

**Dictionaries**

Built-in Functions and Methods

Python includes the following standalone functions for dictionaries:

* cmp(dict1, dict2) - Compares two dictionaries. The comparison process starts with the length of each dictionary, followed by key names, followed by values. The function returns 0 if the two dicts are equal, -1 if dict1 > dict2, 1 if dict1 < dict2.
* len() - give the total length of the dictionary.
* str() - produces a string representation of a dictionary.
* type() - returns the type of the passed variable. If passed variable is a dictionary, it will then return a *dict*type.

Python includes the following dictionary methods:  
(either dict.method(yourDictionary) or yourDictionary.method() will work)

* .clear() - removes all elements from the dictionary
* .copy() - returns a shallow copy dictionary
* .fromkeys(sequence, [value] ) - create a new dictionary with keys from sequence and values set to value.
* .get(key, default=None) - For key *key*, returns value or default if key is not in dictionary.
* .has\_key(key) - returns true if a given key is available in the dictionary, otherwise it returns false.
* .items() - returns a list of dictionary's (key, value) tuple pairs.
* .keys() - return a list of dictionary keys.
* .setdefault(key, default=None) - similar to get(), but will set dict[key]=default if key is not already in dictionary.
* .update(dict2) = adds dictionary dict2's key-values pairs to an existing dictionary.
* .values() - returns list of dictionary values.

**Conditionals**

**Text

Description automatically generated with low confidence**

**For Loops**

Breaks

The break statement exits the current loop prematurely, resuming execution at the first post-loop statement. The break statement can be used in both while and for loops.

The most common use for the break is when some external condition is triggered, requiring a hasty exit from a loop.

When loops are nested, a break will only exit from the innermost loop.

Diagram

Description automatically generated

Continue

The continue statement immediately returns control to the beginning of the loop. In other words, the continue statement rejects, or skips, all the remaining statements in the current iteration of the loop, and continues normal execution at the top of the loop.

The continue statement is very useful when you want to skip specific iteration(s), but still keep looping to the end.

Diagram

Description automatically generated

**Functions**

Parameters vs Arguments

Wait, but what's the difference between a parameter and an argument? These two words get mixed up a lot in programming. In this example 'name' is a parameter while "Michael", "Anna", and "Eli", are arguments. We define parameters. We pass in arguments into functions.