* Condition and Branching
  + Comparison Operators
    - Booleans (True or False)
    - == is used for comparison.
    - < and >, <= and >=
    - != (Not Equal)
    - Comparison for STR are case sensitive.
  + Branching
    - If statements
      * If true
    - Else statements
      * If false
    - Elif Statement
      * Additional if statement.
  + Logic Operators: OR
    - Will run if either condition is true. It wont run only when both conditions are false.
* Loops
  + Range Function
    - Range(N)
      * Generates a list
      * Output is 0,…,N-1
    - Range(A, B)
      * Generates a list
      * Output A,…,B-1
  + For Loops
    - For I in range(0,5)
      * List[i] = whatever.
    - Sometimes no need for index
      * For squares in squares:
        + Square \*This will update square to each element in squares until the last one.
    - A picture containing text

      Description automatically generatedEnumerate
  + While Loops
    - Graphical user interface, text, application, email

      Description automatically generated
* Functions
  + Len()
    - Takes input of str and returns len
  + Sum()
    - Takes a list or tuple and returns the sum.
  + Sorted()
    - Takes a list and returns a sorted list (Remember to set it to a new list)
    - The original list remains the same.
  + Sort()
    - Album\_rating.sort()
    - This sorts the original list
  + User created functions
    - Its important to add a documentation string to illustrate what the function does. (This is called a Docstring)
    - If a function does noting after defining the function just insert “pass” to by pass some functions.
    - Graphical user interface

      Description automatically generated with medium confidenceScope
      * Local
        + Local variables only exist within a function and not part of the entire class
        + But if a variable is mentioned in a function and is not local. Python will check globally if the variable exists and use it.
        + Note you can change global variables within functions, and this will cause the variable outside of the function to change.
      * Global
        + Global variables are variables in the class rather than the method.
        + Within functions you can define a variable as global by putting global in front of it.
  + Exception Handling
    - Try and Except and Else Statement
    - Purpose is mostly for debugging and catching errors within a program.
  + Objects and Classes

Text

Description automatically generatedText

Description automatically generated

* Methods change objects.
* Text

  Description automatically generatedDiagram, schematic, timeline

  Description automatically generatedGraphical user interface, application

  Description automatically generated

Diagram

Description automatically generatedGraphical user interface

Description automatically generated with low confidenceText

Description automatically generated

Text

Description automatically generated

In some cases, a function will only have self as the function thus not passing any attributes into the function. Note in this case r is being passed. And self.radius is the previously defined attribute of the \_\_init\_\_ function.

Arrow

Description automatically generated

**class** analysedText(object):

**def** \_\_init\_\_ (self, text):

*# remove punctuation*

formattedText **=** text.replace('.','').replace('!','').replace('?','').replace(',','')

*# make text lowercase*

formattedText **=** formattedText.lower()

self.fmtText **=** formattedText

**def** freqAll(self):

*# split text into words*

wordList **=** self.fmtText.split(' ')

*# Create dictionary*

freqMap **=** {}

**for** word **in** set(wordList): *# use set to remove duplicates in list*

freqMap[word] **=** wordList.count(word)

**return** freqMap

**def** freqOf(self,word):

*# get frequency map*

freqDict **=** self.freqAll()

**if** word **in** freqDict:

**return** freqDict[word]

**else**:

**return** 0

class analysedText(object):

def \_\_init\_\_ (self, text):

unwantedChar = ".!,?"

for char in unwantedChar:

newText = text.replace(char, ' ')

newText = newText.lower()

self.fmtText=newText

def freqAll(self):

wordList = self.fmtText.split

worldLib = {}

for word in set(wordList):

worldLib[word] = wordList.count(word) #setting the frequency of the word : count

return worldLib

def freqOf(self,word):

wordList = self.fmtText.split

worldLib = {}

for word in set(wordList):

worldLib[word] = wordList.count(word) #setting the frequency of the word : count

return worldLib[word]