1. Fetching keystroke using pyhook and pythoncom
2. Passing each event to separate thread with lock
   1. Key up and key down event passes separately
   2. Key press and release time (epoch time in millisecond accuracy - 1455091739.100591), status, and key name are the arguments
3. Initial variable declaration
   1. Counter = 0
   2. Dictionary – dict\_order – {name:counter} – {‘space’:0}
   3. Dictionary – dict\_value – {counter:[key down time, key up time]} – {0:[0.0, 0.0]}
   4. Key list which is storing words and will be emptied when words pass to dictionary creation function
   5. Main tree dictionary
      1. Main tree dictionary is consist of root node as ‘Space’
      2. Each node will have following attributes
         1. Event name
         2. Hold time – duration between key press and release
         3. Release down time – duration between previous key down and current key down (not considering the duration between last key up and current key down is because last key can be held pressed when new key is pressing which throws error)
   6. Class for creating the object of keystrokes
      1. Self.hold
      2. Self.release
      3. Self.name
4. Thread listener accept the arguments
   1. \_\_init\_\_ start the thread, no explicit calls to start() required
   2. Printing event name and time
   3. Checking the event type using status
   4. If key down
      1. Finding epoch time using time stamp of event generation
      2. Release = epoch\_time - dict\_values[count][0] #counter not incremented, so count pointing to last value
      3. If release > 3.0 #we don’t want to wait for too long
         1. Give current word to store
         2. Dictionary – dict\_order – {name:counter} – {‘space’:0}
         3. Dictionary – dict\_value – {counter:[key down time, key up time]} – {0:[0.0, 0.0]}
         4. Counter = 0
      4. Incrementing counter
      5. Storing {name:counter} to dict\_order
      6. Storing {counter:[key down time]} to dict\_value
   5. If key up
      1. Finding epoch time using time stamp of event generation
      2. Storing current value
         1. Count = dict\_order[name] – name is from key up event, both key up and key down name should be same for a particular key
         2. Appending value to dict\_Value’s list – dict\_value[count].append(key up time)
      3. Retrieving older value and creating the parameters
         1. Hold = dict\_values[count][1] – dict\_values[count][0]
         2. Release = we already calculated
      4. If it is first keystroke, previous value will be zero:
         1. Release = 0.0